Implementations of Service Oriented Architecture and Agile Software Development: What Works and What Are the Challenges?

Milan Schramm, Maya Daneva
University of Twente,The Netherlands

Abstract

Today many organizations use service-oriented architecture and agile software development as their software paradigms. While both certainly have their advantages, in the fields of Empirical Software Engineering and Information Systems these have been treated in relative isolation and their impact on each other is not well understood. This paper performs a grounded theory research by empirically analyzing professional blog posts published in IBM's Developerworks platform, to find good practices and common pitfalls of using a service-oriented architecture and agile software development together. The perspective taken in this study is the one of service-oriented architecture practitioners involved in agile projects. We found that continuous integration, collaboration, governance and continuous improvement are good practices that result out of merging the two paradigms. We found that the challenges of the joint use of service-oriented architecture and agile lie in the engineering of non-functional requirements, compliance requirements as well as up-front architecture evaluation.

1. Introduction

The past decade witnessed many companies' efforts to achieve a transition from waterfall to agile system delivery models. More and more organizations of various sizes realized that waterfall development and project management was either not sustainable or not realistic in their contexts. In turn, they switched to the agile software development and project management paradigm, e.g. by embracing specific agile method e.g. SCRUM and XP. Much empirical research has focused on this transition and its outcomes, e.g. the 2016 systematic literature review of Bissi et al. [1] comparing test-driven development (one of the most prominent agile methods) and test-last as it is in waterfall, concluded that using agile approaches leads to "a significant increase in internal software quality and a meaningful increase in external software quality".

In a nutshell, all existing agile methods adhere to the agile manifesto [2] stating the four principles of 'being agile': (1) Individuals and interactions over processes and tools, (2) Working software over comprehensive documentation, (3) Customer collaboration over contract negotiation and (4) Responding to change over following a plan. At the same time, in the service-oriented architecture (SOA) community, practitioners came up with their own SOA manifesto that concerns SOA initiatives. This manifesto seems to leverage agile concepts in SOA context. SOA is defined as "a set of components which can be invoked, and whose interface descriptions can be published and discovered" [3]. In the style of the agile manifesto, the SOA manifesto includes [4]: (1) Business value over technical strategy, (2) Strategic goals over project-specific benefits, (3) Intrinsic interoperability over custom integration, (4) Shared services over specific-purpose implementations, (5) Flexibility over optimization, (6) Evolutionary refinement over pursuit of initial perfection.

The principles of both agile software development and SOA are rather about being evolutionary than revolutionary (incremental over big-bang) and rather being effective than efficient. But how does this work in practice? Just imagine the following example: A short-term request by the customer makes a change in the next sprint of the software development necessary. This change implies a change of the respective service, which is used by other services. Obviously, these other services may not necessarily be aware of the change, so that the misaligned services cannot perform. While the question of how to cope in such situations is relevant to practitioners, to the best of our knowledge very little is published in agile literature and in SOA literature on how SOA and agile fit together, how synergies are optimized and how side effects as the one in our example are controlled. As we will see in the next section, while there is much academic work done with a focus on agile and SOA, no study until now took practitioners' perspective and

978-1-4799-8710-8/16/$31.00 ©2016 IEEE

presented a realistic SOA context in terms of organizational and technical complexity.

This paper makes a step to narrow this knowledge gap. Based on qualitative information from practitioners, we analyzed the ways in which they reason about the synergies of agile and SOA paradigms and the challenges and pitfalls they explored.

The rest of the paper is structured as follows: Section 2 presents related work. Section 3 lists our research question and describes our research process. Section 4 is on our results, Section 5 provides a discussion and Section 6 analyzes validity threats. Section 7 summarizes our contributions.

2. Related Work

As part of preparing this paper, we searched the Scopus digital library (www.scopus.com) for related work, by using the following search string: "((agile OR agility) AND (SOA OR service-oriented architecture OR service system delivery OR services")). Below, we summarize the publications we found relevant to our study, in reverse chronological order.

Carvalho and Azevedo (2013) are among the very few who studied the combination of SOA and agile principles, for the purpose of designing a method in the future, that leverages the best of SOA and XP [5]. They delve into the possible merits of the combination of SOA and agile and the possible challenges. The particular focus of [5] is on the service construction phase in a service development lifecycle. Unfortunately, there are no follow-up publications of these authors concerning the design of the method that they stated [5] they would develop.

Shahrbanoo et al. (2012) [24] proposed five quality attributes to help architects create a core architecture before any SOA-based development starts. By prioritizing organizations' business goals, mapping them to business processes, extracting quality attributes, determining core business processes, and selecting features before every iteration, these authors managed to add agility and agile software development to the SOA methodology. The authors add a small portion of agility to SOA rather than proposing a way to achieve a planned step-by-step process for applying agile at the level of a service system delivery project. In this sense, the authors' proposal goes at the expense of the agile principle "Responding to change over following a plan" [3].

Next, the 2010 paper of Roy and Debnath [19] presents a way to develop services using XP practices in a SOA context. Specifically, the authors' goal is to leverage the agile practice of team collaboration to the context of SOA implementations. The narrative however is on a too high level leaving out details on how exactly the proposed approach would be applied to the various SOA contexts.

Karsten and Cannizzo [16] investigated the functioning of a distributed team using XP and Scrum in a SOA project. As their key focus is on the distributedness of the SOA project organization, no information is reported on the principles of SOA and agile that were combined (nor on how to do so in a pragmatic way).

The study of Callahan [11] looked into the applicability of XP practices to SOA projects, while focusing on the relationship between those practices and SOA principles and complexities. The study concludes that using small releases, on-site customer and pair programming practices is advantageous to SOA projects. However, the study deemed the lack of an up-front design and the minimalistic approach to documentation harmful in SOA context. Although the paper calls for adapting XP core practices to make them fit the complex realities of SOA initiatives, it offers no specific adaptation of the XP core practices to SOA contexts.

Schatz [23] investigates the complementary use of agile and SOA from systems evolution standpoint. This author frames the discussion on agile and SOA in terms of how XP adds value to businesses (that are also interested in SOA). While this study adds to the conversation on the benefits (e.g. speed and quality) that one can materialize from XP practices used in the delivery of SOA-compliant systems, there are no details on the relationship between agile and SOA principles and on the implications of XP for a complex SOA context.

Krogdahl et al. [17] state that a big up-front analysis contradicts with agile principles and recommend an agile approach to SOA projects that employs the concept of "just enough" architecture. These authors discuss two synergetic factors between agile and SOA paradigms – namely, continuous refactoring and response to change, which are advantages of both. After defining seven principles of agile software development, the authors investigate how each principle may be used to implement a SOA. Although the authors [17] present an organizational context of complementary use of agile software development and SOA, they miss on explaining how their practices apply in scaled-up and more complex projects, where all arising challenges are more significant.

Ivanyukovich et al. [14] map SOA concepts to XP concepts and make a call to adapt XP practices to work with service development. These authors hypothesize that not all XP practices would be relevant to SOA projects. Their paper doesn't discuss agile and SOA in terms of principles and steps that would work, nor provide specific instances of benefits and challenges.

Other approaches such as those in [10,12,13] don't discuss agile software development but rather talk about (business) agility in SOA implementations. E.g. [13] uses a definition of agile SOA which is more dynamic, flexible and responsive to change than traditional SOA.

The references in the above paragraphs indicate the interest in the synergies of SOA and agile from the perspective of both the agile and the SOA communities. The work of these authors certainly informs our research

and motivates it, as so far we could find no study that looks into the practices that practitioners used and found to work and the challenges encountered along the execution of SOA projects by using the agile paradigm.

3. Research Questions and Research Process

Following the Goal-Question-Metric paradigm for structuring empirical software engineering research [8], we set the following goal of this study: it is to understand how agile and SOA work together, from the perspective of SOA practitioners in the field. To this end, we ask the following three research questions (RQs): **RQ1:** *What aspects do SOA practitioners consider important in SOA implementations using the agile principles?* **RQ2:** *What ways of combining SOA and agile principles work, according to practitioners involved in agile projects of SOA?* and **RQ3:** *Which faults should be avoided when combing SOA and agile principles, according to practitioners involved in agile projects of SOA?*

We developed a qualitative research design that used publically available qualitative data in a well-known professional blogging platform pertaining to the field of interest, namely SOA. We treated this platforms as a repository of information – perceptions, evaluations, opinions and articles, provided by practitioners in the SOA sector. The analysis of such information is expected to provide a deeper understanding about the practice of agile delivery of SOA projects, from the perspective of those in the field, namely the practitioners writing blog posts and those commenting them.

Our data collection approach was inspired by Hookway [30] and Verner et al. [29] who suggest this data collection strategy be used in situations when a researcher would like to balance the cost for executing the study against breadth and depth of the study and where publically available qualitative data are easily available for researchers to analyse. As Hookway [30] states, "Blogs offer substantial benefits for social scientific research providing similar, but far more extensive opportunities than their 'offline' parallel of qualitative research." First, blogs provide a publicly available, low-cost and instantaneous technique for collecting substantial amounts of data [30]. Further, blogs are data in textual form, allowing for analysing qualitative data immediately without the resource intensiveness of voice recording and transcription.

We chose the IBM Developerworks blogging platform (http://www.ibm.com/developerworks/blogs/) because of the high variety of involved practitioners, the existence of historical data and the professional quality of the posted documents and comments. The web site consists of blog posts (including their comments), white papers, forum discussions, and news articles. Following a search process similar to the search process in systematic reviews [31], we searched for posts in this blog, that relate to the central topic of SOA and agile. The details of the textual data collection are described in Section 3.1.

Once the data was ready for analysis, we applied the coding practices of the grounded theory approach, which seeks to develop theories based on systematically gathered and analyzed data [28]. In this study, our empirical data was analyzed according to the framework of Corbin and Strauss [7] in combination with the guidelines by Urquhart et al. [29] and the coding techniques by Saldaña [21]. We processed the data in three steps: open, axial, and selective coding, which used the practices of constant comparison (comparing text from one blog post with text from another post on the same topic), iterative conceptualization, theoretical sampling, and theoretical integration. For the purpose of our analysis, we used a computer-assisted qualitative data analysis software, Atlas.ti 7 (see http://atlasti.com/). The details of our data analysis are presented in Section 3.2.

3.1. Empirical Data Collection

As the focus of our work lies on agile and SOA principles, we searched our selected blogging platform for blog posts that were tagged with words such as "agile", "agility", "SOA", "service-oriented architecture", "service systems". Our search resulted in 319 documents. To prepare the documents for the coding process, we screened these 319 hits by using the following exclusion criteria:

- Duplicates: if a post of the same author appears multiple times, one post is only included.
- Educational offers (e.g. Webinars): a post that represents an invitation to an event dealing with SOA and agile, is excluded.
- Job openings: a post advertising a job vacancy is excluded.
- Product manuals: a post that publishes a link to a technical documentation or a manual related to SOA and agile, is excluded.
- Overview pages: a post that presents a summary of other posts, is excluded
- Search results: a post that reports on search results of a practitioner in the area of SOA/agile, is excluded.

After applying these criteria, we excluded 174 documents. The remaining 145 were imported into Atlas.ti 7 for the coding and analysis process. The analysis of these documents is discussed in the following sections.

3.2. Data Analysis

As already indicated, the first step is known as *open coding* and includes the reading of all documents, the creation of the codes and the linking of the codes to phrases, paragraphs or images of the documents. Our open coding approach implemented the so-called

structural, descriptive, initial, and holistic techniques proposed by Saldaña [21]. This means, researchers analyse their textual data by considering the research questions (structural coding), while still remaining open to the content and the new nuances of it (initial coding). The process is presented in Figure 1 below.

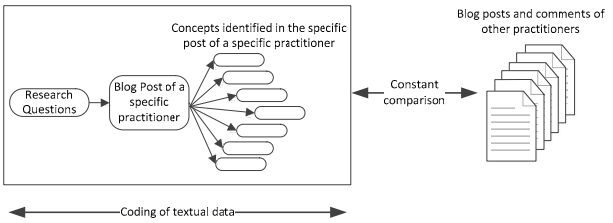

Figure 1: The open coding process.

Descriptive coding then takes place, which is the process in which a meaning is distilled from a practitioner's paragraphs in a blog post and is summarized by using a word (or a label) that is called 'code'. An example of this type of coding of textual data is shown in Figure 2.

We used holistic coding any time when we needed to describe one or more paragraphs of text at once instead of analyzing each paragraph line-by-line.

In line with [21], the codes resulting from the open coding were revised, reorganized and re-analyzed. This was done with the coding techniques of *focused, axial, and theoretical coding*. This means that during this second iteration, categories are built based on the existing codes and analytic memos are written. The purpose of these categories is to "reassemble" the data that was "fractured" in the previously used coding process and to help the researcher to uncover the relevant context types [25]. This reassembling was achieved by applying techniques of focused and axial coding [21], which means that given the existing code structures, categories are built based on code similarity. Axial coding is now used to relate categories to their subcategories and codes. These relations are used in the theoretical coding process, which links all categories, subcategories, and codes to one core category [21], which explains what "this research is all about" [25]. During the second step, we wrote analytic memos. These memos keep record of the analysis. In the time of the second step, the memos helped explain and reflect on the work in progress [25]. Since neither the memos nor the coding processes can be perfect on the first attempt, the categories, subcategories, codes, and memos are created in an iterative process, which means that the "depth and quality of conceptualization" improves with time [25].

In the third step, we used the memos to link the concepts to each other by establishing possible relationships that are traceable to the data. The coding steps are not necessarily separate but rather interrelated and iterative; if some analyzed categories are not well supported, we checked for the internal validity of our grouping of codes into categories and re-analyzed the respective documents.

Text fragment	ID	Code name
As an example, one current client was endeavouring to redesign a database-driven business process solution. IT believed the key objective was to design a more user-friendly interface for the business users, but the business users simply wanted to reduce the current 6-monthn delay between new solution requirements and change management. This appears to be simply an issue of miscommunication, but it is one that is challenging the relationship between business users and IT, and causes downstream problems. If this client's business and IT stakeholders were to collaborate together on process and rules discovery, they would realize outcomes that would be beneficial for both groups. What is needed is an iterative approach to allow business and IT to work together on design in terms of process flow and solution definition aspects.	1.1 1.2 1.3 1.4 1.5 1.6	Time reduction of project delivery Lack of alignment between business and IT Alignment through collaboration Agile approach Process flow Solution design
The devil is in the detail. An agile approach often requires a high level of cultural readjustment because it effectively redesigns the software development process as it shifts development from an IT-centric waterfall approach to one that is founded on an iterative approach to solution design. In many organizations, implementing agile runs counter to the traditional roles of both business and IT. As a result, it is important to have executive sponsorship of these types of initiatives [SOA] to establish a mandate for change.	3.1 3.2 3.3 3.4	Tricky aspect Culture shift Unintuitive Executive commitment

Figure 2: Descriptive coding from textual data.

4. Results

Our results of the qualitative data analysis are summarized in Figure 3. It shows the conceptual categories (or themes) that belong to the SOA implementations using agile principles, as per the practitioners' experiences shared on the IBM Developerworks blogging platform.

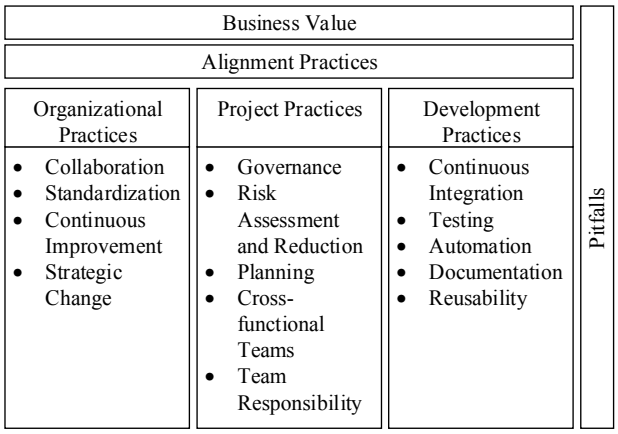

Figure 3: Conjoined SOA and agile software development

As Figure 3 indicates, we found that most of the conceptual categories that the practitioners from IBM's Developerworks blog talk about, could be grouped in a theme called ***Practices.*** This theme includes operational activities (or practices) that appear to be at three levels:

- Organizational
- Project
- Development

In addition to those, two other categories emerged: ***Alignment*** and ***Business value.*** From the analysis of the blog posts, we observed that these don't concern directly the operational level but rather the strategic level of SOA and are therefore placed in Figure 3, above the organizational, project, and development practices. We make the note that generating business value is not meant as a set of practices but rather a desirable state of the SOA and is achievable by following the operational practices at the three levels. For example, one blog post indicated that business value is realized when an organization establishes *"a more effective linkage of business issues with IT, especially through reuse and asset-based industry solution design and development"*. Similarly, the category ***Pitfalls*** doesn't mean a set of practices that would result in wrong output, but rather an organizational or project state that should be avoided. As one blog post names it, it is *"the lack of business and IT collaboration in solution design, the limited or discontinuous requirement definition and process modeling"*.

In the following subsections, we explain each of the categories that in the words of the practitioners writing the blog posts, were all justified by and supporting the business value generation. As it's usual in qualitative studies, we provide quotations from the qualitative data to illustrate the concepts.

4.1. Organizational Practices

We found that according to the IBM's blog authors, the organizational practices are perceived as applicable organization-wide. They are not tied to a particular SOA solution. Instead, they apply independently from the phase of a SOA project or the type of work being done. Our analysis revealed four practices:

1) Collaboration: The practitioners from the IBM blog deemed important that in order to realize synergies between business and IT in a SOA-based strategy, appropriate collaboration processes must coordinate plans and solutions without hampering effectiveness by requiring overly detailed synchronization. Authors of three blog posts deemed a common understanding of the enterprise applications landscape *"key"* to achieving synergies. As one put it, *"understanding the application landscape is a prerequisite for knowledge sharing and consensus-building in the organization"*.

In large-scale SOA initiatives, the blog posts made statements of practitioners that observed agile software development approaches to have well-defined communication processes such as daily scrum meetings and scrum-of-scrums so that all stakeholders are usually engaged and aligned. These blog posts elaborated that *"this frequent stakeholder interaction and agile planning based on client demands"* ensures that SOA project team *"gained their sponsorship and in the end implemented a well-aligned solution"*. Rapid communication in the team and across teams *"created an organizational culture of rapid responses"*, where business requirements are met immediately instead of being buried in approval processes. Three posts elaborated that collaboration relied on the availability of information and therefore on transparency – which was ensured throughout the organizational processes, e.g. *"If your client is unavailable, you will notice limited or incomplete requirements definitions and process models and if this continues, you'd see a misaligned, suboptimal solution and non-informed decisions"*.

Applying sophisticated collaboration processes are key for any flexible and seamless integrated SOA initiative and they help to spread the enterprise SOA vision and roadmap through all stakeholders. Examples of such processes, cited in the IBM blogging platform, are: (1) collaborative brainstorm sessions to help achieve a common understanding, agreement and language, plus to help lead to a faster time-to-market e.g. by having aligned requirements and early prototypes; (2) cross-functional teams – which together with integrated applications and project dedication enabled each team member to focus on the current project and thereby eliminating process and communication bottlenecks.

Those blog posts talking about cross-functional teams, however, indicated that using such teams effectively is a challenge when attempting to maintain the same level of collaboration in scaled or distributed projects, since most of the collaboration happens face-to-face. One way to implement a scaled agile organization, according to the authors of these posts, is to build agile communities of diverse groups of people across the organization, where

people share ideas and establish shared principles aligned with their needs to increase to the cross-organization communication and information dissemination: *"It pays off to foster idea sharing in agile communities"*.

2) Standardization: Our qualitative analysis indicated that a standard-based, governed delivery process was necessary for a successful adoption of SOA in the organization. Standardization clarified the process for the customer and ensures their engagement. Depending on the business needs, one may choose to implement either more ad hoc or more formal approaches. While formalized approaches may prevent change, ad hoc approaches may result in chaos by limiting the collaboration.

On the technical side, the SOA approach should provide a unified interface through a standardized access protocol. These technical standards allow a flexible infrastructure which eases maintenance, enhancements and replacements. By ensuring that a consistent set of architectural patterns and a unified data model is used, one makes sure to be able to compose new applications without the need to re-architect the services.

3) Continuous Improvement: In order to stay competitive, the IBM practitioners from the blog perceived that organizations must apply industry-standard best practices and more important continually learn from their own mistakes and successes. This should lead to continuous improvement of the business processes with regards to effectiveness and efficiency. According to the observations on the blog posts, to follow this kaizen principle in a dynamic environment, the organization needs to be interconnected, committed and smart.

Last, the focus of SOA on the reuse of assets enables the IT to continually improve the reused services instead of re-programming the same over and over again.

4) Strategic Change: Those IBM blog posts that spoke of strategic change were the opinion that a fully-integrated SOA environment supported by agile principles is achievable if the organization has *"the right vision"*, executive and senior management commitment, and *"a sound understanding"* of the existing portfolio. As one post put it: *"The organization is better off starting small by including critical business processes, and then gradually expand to create a robust platform supporting integrity, reliability and scalability of business processes across the organization"*.

'Starting small' meant that one should start with one small, strategic, and business critical project in which each team member is fully committed to the project. After a successful project, the team member should spread their knowledge in new project. This ensured a steady knowledge dissemination through the complete organization. An implication of this was that SOA fostered the break from silo thinking to enterprise transformation thinking which allowed improved portfolio and technology management with resulting cost and efficiency savings.

4.2. Project Practices

The project practices apply in each (project) team of the organization. By implementing them in teams, the way of working standardizes and is in the end adjusted to achieve alignment and generation of business value. We found five practices in this category:

1) Governance: Governance processes were deemed important according to the blog postings to help coordinate and monitor plans and solutions without hampering effectiveness by requiring overly detailed synchronization. This is done by a collaborative decision on how the project has to be governed, what has to be measured, how it would be measured and how to respond to those measurements. The practitioners found it is important not to govern the easy and wrong things but the those KPIs that significantly correlate with the business goals, then measure them continuously.

To maintain the advantages of small and autonomous agile teams, the governance process should not add *"wait time"* (in the wording of a practitioner's post) but rather provide the so-called *"self-services"*. This requires the team to be aware of enterprise (governance) processes such as reuse, common policies, guidelines and conventions. The awareness of the whole enterprise was deemed important in SOA as well, since SOA crosses department lines within business as well as IT and focuses on shared services, reuse and agility. This requires collaboration and end-to-end thinking.

In what specific ways does governance help SOA implementations? Based on the information from the IBM Developerworks platform, we assembled this list of specific examples of how governance *"helps"*: (1) by providing precise and clear information about the business goals, (2) by defining roles and responsibilities in the organization, and (3) by applying these roles end-to-end from initial design to production. Furthermore, according to those blog posts, governance was found to help achieve higher service reuse, and overcome the tendency of isolated, localized decisions and solution deliveries about service architecture and design.

***2) Risk Assessment and Reduction*:** In our qualitative data, there was a general agreement that in agile teams, the team themselves should monitor the technical risks continuously and mitigate those risks proactively e.g. communicate change to the external consuming services in SOA. This was deemed important because a change within any layer of a service implementation may cause problems. In these situations, the problem should be isolated quickly and the necessary adjustments should be made in time.

To reduce the risks of poor performance and a non-aligned solution, the blog authors used early prototypes and traceability links from the business problem to the technical requirement are necessary. The prototypes were to address (project) risks as early as possible and to

remove uncertainties. Traceability along with 1) ***Continuous Integration*** was to avoid late-phase integration and thereby enabling the team to explore requirement misinterpretations early.

However, the blog authors also indicated a challenge due to the inherent risk involved in using agile with unexperienced teams, e.g. in terms of being predictable with regards to manageability, costs, quality, and time. Therefore they thought, agile approaches were not advisable in non-changing environments.

3) Planning: Those blog posts that discussed cross-functional teams, were explicit about their assumption that having such a team and removing too much of the early planning would result in poor integration and enterprise improvement. To prevent this, checkpoints of validation, continuous business collaboration, integrated planning and delivery process across the enterprise, and architectural thinking in the team were deemed necessary. As one blog post put it, architectural thinking means an explicit *"awareness of architectural principles, patterns and reusable services"*.

4) Cross-functional Teams: those blog posts putting forward the notion of cross-functional teams, seemed to agree that with diverse and balanced backgrounds with respect to experience and specializations, the team were perceived as more likely to leverage their synergies. For example, cross-functional teams prevented extreme group thinking, mitigated risks and built robustness by avoiding extreme tendencies. An example for a way to achieve a cross-functional team is to introduce a Service Architect in the development team and make this individual responsible for interacting with various technical and non-technical roles within the SOA initiatives. This role can be used in the team to ensure a correct implemented architecture and immediate feedback can be used to improve the architecture continuously.

5) Team Responsibility: For the agile practices to work in SOA, a few blog posts elaborated that they must be implemented on *"an individual team level"*. This is possible by designing an aligned and transparent team process and incorporating the depending services into the development (the so called *"automated building and testing"*).

Letting the team being self-organized means that the team members prioritize the work items in alignment with the stakeholders needs. By having the team roles decided by the team members themselves based on their interests and knowledge, the team motivation increases. Hence no traditional leadership in a directing manner should exist; the manager should rather coach each team member.

This *"pull model"* is challenged in larger teams since people may lose focus and think that other team members would take care of the work items. This risk is reduced with the collaborative approach of ***4) Cross-functional Teams***. Another challenge is the experience of the team; inexperienced teams are likely to fail being self-organizing, which is mitigated with a mixed team consisting of junior and senior employees.

According to IBM's blog authors, in their SOA projects, the team responsibility with regards to data and functionality was defined in a service description. Each organization must allow each project team a level of flexibility and autonomy to implement the solution and their own internal (development) processes in way which fits best to their needs.

4.3. Development Practices

Development practices apply on a project level and describe the technical way of working in a software development project. Our data analysis resulted in four categories. We found five practices in this category:

1) Continuous Integration: Those blog posts that talked about continuous integration, were in favor for agile teams having at their disposal an up-to-date and flexible codebase to be responsive to ever-changing business and customer needs. In these posts, continuous integration meant to rebuild the software automatically with every change, so that bugs are exploited early, fixed immediately and an up-to-date codebase exists. One post put forward that *"This immediate feedback discourages developers to submit poor code quality since automated testing would detect it within minutes or hours instead of days or weeks or in worst case just before due date. If you manage to reduce time gap, it brings a faster resolution itself since the developer would still remember more details of the code."* Therefore, continuous integration was thought to result over time in higher code quality, fewer integration problems and rapid release cycles.

2) Testing: A particular challenge found in two blog posts concerning agile software development and SOA is that *"testers may find bugs after the developers already moved on"*, which may mean that the discovered issues may be resolved after deadline. One of the posts provided solution practices to this issue. E.g. by implementing all tests in an automated fashion, it could be possible to test the system continuously during the iterations after each integration. If test failure occurs, the developer could fix it since one only need to investigate the changes that were made since the last build. In the recommendation of this practitioner (who authored the post), this would enable SOA applications to be *"releasable"* all times. Implementing automated unit tests immediately after writing a few lines of code results in a more robust code base, which increases the quality of the product in the end. Furthermore, this practitioner observed that *"mapping business requirements to test cases, acceptance testing helps to keep record the progress and that neither too less nor too much functionality is implemented*".

3) Automation: While there were blog posts praising the role of process automation, they also indicated that to automate processes, these have to be *"repeatable and*

traceable". Only then automated processes would add specific value: e.g. *"reduce the chance of human error, increase the team's efficiency and enables the team to focus on the business needs"*. It also reduces *"the time developers spend on repetitive tasks and increases the team's ability to respond faster to change"*.

Despite all advantages, the migration of those processes surrounding SOA projects to an automated environment was deemed costly and lengthy which may not reach break-even in simpler project. According to these IBM posts, the more complex the SOA project, environment and deployment is, the more value automation provides.

In agile software development, one may use automation to manage versions, to manage changes in the backlog or to visualize requirement or dependency models. Scripted migration environments, and deployments have the advantages of being able to run from a single command, being in a known state, enable self-service, standardization, being predictable and repeatable. Regular deployment to a clean, near-production environment, where automated tests are executed, would let the team identify environment, integration and performance issues immediately.

4) Documentation: The practitioners in the IBM blog acknowledge that agile development often avoids documentation, which led especially in large and complex projects to *"forgotten details"*. In SOA projects, the IBM blog authors included documentation as part of the build in an automated fashion, which helped that the documentation was always up-to-date.

Furthermore, these blog authors focused on the high-level key processes, business design, and requirements and shared that they documented those *"just enough to proceed"*. This, in turn, allowed their teams to start immediately with implementation, which resulted in a shorter timeliness and a higher responsiveness to change.

Last, in the experiences of these blog authors, when documenting something, one should always check whether this specification supports the business goals in an effective way. E.g. user stories were used to trace the customer's requirements through the whole process from coding, testing to deployment.

5) Reusability: There seemed to be a consensus among IBM blog posts that SOA projects *"rely on reusability"* and that this should be exploited. By focusing from the start on reusability of service, the blog authors made sure that the service and their data are exposed in acceptable quality. This was done by encapsulating functional capabilities separate from their implementations.

Another observation shared in the posts was that while the services are reusable in general, it doesn't always mean that they are actually reused. According to the practitioners, services only provide additional business value *"if they are actually reused"*. E.g. asset libraries were deemed helpful in finding reusable services by classifying and describing the services in a standardized way and store them in a central repository, but these libraries should be actively used so that reuse – and business value, is leveraged.

Last, a few blog authors wrote about their experiences that separating concerns on different layers helped their SOA project organizations to implement the business requirement in a flexible way. E.g., by separating the data management and its processing, one remains flexible in the use of the data. By separating the interface from business logic, the designer is able to align the interface with the customer's expectations. By minimizing the coupling between services, the reusability is maximized.

4.4. Alignment

This section describes on a higher-level how to achieve alignment of business and IT in SOA initiatives that use agile development approaches. In what follows we enumerate the practices that IBM blog authors used to achieve business-IT alignment in their SOA projects:

1. Leveraging the planning and the prioritization of the upcoming work items at the time between agile iterations. By involving business and customer stakeholders in the agile prioritization process, it is ensured that the development provides value, that the process itself is performed without the necessity of ad hoc changes and that the needs are met in time.

2. Leveraging traces from the services back to the business goals. Those practitioners who used this tracing mechanism experienced that if tracing a service back to goals is impossible, then the service is not an ideal candidate for exposure and the need of it should be questioned. By providing end-to-end visibility in line with accountability, measurement, and traceability, teams found a common and transparent understanding of the organization and are therefore more likely to deliver aligned solutions.

3. Treating the SOA initiative as a business initiative. A technology-driven SOA implementation is likely to fail since the benefits would not necessarily provide business value. To avoid this, one can understand SOA primarily as a business initiative: the value of SOA is delivered by an agile business and IT design and delivery process and by strengthening the organization's goals and strategy. If a SOA team fosters the alignment between IT and business, then this establishes a culture of change and cooperation, increases the ability to respond to change and the flexibility, reduces costs, reduces risks, and an improved portfolio management is achieved. According to the blog authors, this synergetic relationship, where one adapts to the needs of the other, helps to secure growth for business.

Those practitioners who wrote blog posts on alignment and who used one of more of the above mechanisms experienced that the collaborative and aligned approach

of SOA and agile software development resulted in a higher productivity since the scope and objectives of the desired solution are defined jointly. E.g. SOA enabled customers to model, compose and manage services independently from its underlying layers, therefore make the business processes more flexible and getting the IT environment more closely aligned and responsive.

4.5. Business Value

Our data analysis brought us to the finding that all the aforementioned practices that relate to the concepts in this study (e.g. collaboration, cross-functional teams), are designed in such a way that they increase the productivity and enable the teams to focus on their competencies and to be responsive to change. For example, as one posts says, *"To generate business value, you need to do a lot of things first. Cross-functional teams being the most important one, also reuse oriented culture that rewards reuse…"*. The increase in efficiency and effectivity results ultimately in an increase of business capability and business profitability and therefore *"provides business value"*.

We found, agile software development is implemented in organizations because of its underlying premise to deliver business value. By using a matured agile approach including traditional elements such as governance, one reduces the risks of it *"being chaotic"*, leverages its business value and achieves a sustainable competitive advantage.

Business changes such as mergers and acquisitions or shifting market conditions happen rapidly. In changing market conditions, the first mover's advantage enables companies to reach customers with technological innovations first. The flexible technological leadership is achieved - according to the IBM practitioners, by a software development process, which is able to reply to these market conditions, and reusable assets to adapt services to new and changing demands.

4.6. Pitfalls

This section presents the most prominent pitfalls of complementarily using SOA and agile development. In the sections above, we mentioned some pitfalls and risks, therefore in this section, we focus on some additional aspects that would impede the process of delivering business value.

First, the IBM blog authors mentioned that SOA implementations *"are not necessary if an organization depends on a homogeneous IT environment, where processes are coordinated and executed centrally"*. SOA relies heavily on distributed computing and loosely coupled services; however, this is not always a desired scenario. By implementing a centralized, tightly coupled applications, the efficiency and (real-time) performance is higher than with SOA equivalents. This is especially important in scalable and non-changing scenarios. In a context where SOA is necessary but context is changing at rapid pace, agile would add much value. However, it was pointless to implement agile in SOA contexts that are relatively stable and are not disrupted by unpredictable events.

Second, external regulatory requirements e.g. by laws, are likely to require additional documentation, which normally would not have been specified. Especially in mission-critical, safety-critical or health-care-critical solutions, businesses demand the up-front analysis be extended to ensure that the application is implemented correctly. Both requirements contribute to dissolving benefits of agile approaches.

Third, from the posts of the IBM blog authors, we found the opinion that in simple environments, pure agile software development works. With increasing complexity – and most SOA initiatives are overly complex, one needs *"to introduce more sophisticated, disciplined techniques such as additional non-functional requirements, matured agile approaches, or prototyping instead of user stories"*.

5. Limitations of the study

We evaluated the possible validity threats of this paper based on the methodological suggestions in [20, 22, 33]. Regarding external validity, two questions are of paramount importance. First, to what extent the observations of this study could be also observed by other practitioners in other projects within the large IBM organization? And second, to what extent the results are observable in projects happening in clients' organizations of other IT consulting companies, e.g. GE Capital, Hewlett Packard and CGI?

Our participants are practitioners employed by IBM. The use of further data sources extending the IBM Developerworks would have increased the external validity. However, we make the note that the analyzed textual information in the posts in IBM Developersworks is written by a broad variety of practitioners in the field of software development and software architecture. Most of them have multi-year experience as IBM consultants and have worked with several clients' companies in several business sectors. Therefore, extending the scope of our data collection by including other blogging platforms (e.g. one hosted by Hewlett Packard) would have meant to increase the amount of data but not significantly the amount of different backgrounds, experiences and types of project organizations.

Following the analytical generalization guidelines by Seddon & Scheepers [22], we think it is possible to have similar observations in communities of large companies that share contextual factors similar to the settings in which the IBM practitioners work. We expect that companies that are large and experienced in both SOA and agile, that have a knowledge sharing infrastructure,

who are mature in process-oriented thinking, and require a certain complexity, to have practitioners that would observe similar benefits and pitfalls as those in IBM. Following Seddon and Scheepers [22], we could think that the themes discussed on this blog regarding SOA and agile might be also the ones observable in discussion blogs supported by other large companies, such as Hewlett Packard, CGI and GE Capital that engage in consulting on SOA and in introducing agile approaches in clients' organizations. We could think that our results might be observable in those companies because of similarity in the project contexts. As indicated in [22], similar contexts could create similar circumstances and similar mechanisms at play. So, practitioners from those companies that are working on SOA initiatives similar to those of IBM might well observe similar benefits and pitfalls of using agile in SOA projects. Since the authors of the IBM Developerworks' documents are experienced practitioners with work experience in many settings, it is safe to assume that their statements also appear in other contexts. Hence, the propositions made by this paper could possibly be also observable in other contexts.

An inherent weakness of qualitative studies is whether the participants always acted as assumed in their projects. We acknowledge that this risk is elevated in our study as the Developerworks blog is sponsored by IBM and some publicity and commercial interests may well be present behind the authors' papers. Plus, despite the willingness of the IBM community to share insights into their practices, there is no way for us (the researchers) to ensure that each practitioner acted in their projects in a way completely consistent with IBM's practices. We cannot know whether the contributors are representative and whether their contributions are up-to-date nor whether IBM covers all industry practices.

Regarding construct validity, we used multiple sources of evidence and a chain of evidence in order to ensure the construct validity of our study [33]. We analyzed multiple documents provided by multiple practitioners within the IBM community, however, although the posts are of high quality, it is clear that the purpose of many is to sell products and consultancy services. This may be represented in the degree of automation it is proposed in this paper since IBM's products support semi-automated processes. Another effect of the selling attitude is that the pitfalls are underrepresented in the posts. To eliminate this threat, interviews with the practitioners would have been necessary. Since we focused on the coding of the blog posts, no sufficient resources were available to conduct interviews.

Furthermore, to overcome internal validity threats [33], the coding process is checked randomly with 10% of selected data sources by the two researchers. The input of the first step of coding (the open coding step) was done independently by the two researchers who later compared their own codes to ensure that the understanding of posts is as much rigor and repeatable as possible. The differences between both were discussed and consolidated during the iterative approach of this research. To mitigate further internal validity threats, the complete coding process could have been done in parallel with at least one more researcher involved, who could have done partial peer reviewing [21].

6. Discussion

Agile software development approaches have not been conceived with SOA in mind and their immediate focus is business value generation as part of project delivery, not applications' initial conceptualization [6]. One might expect that SOA project might demand special adaptations of agile paradigms regarding at least two specific aspects of SOA initiatives: (1) SOA projects imply architecture be upfront, and (2) SOA projects plan very little for the actual change of the services once they are built. Our study indicates that the fact that agile paradigms are not SOA-minded doesn't preclude their deployment in SOA initiatives. Our results suggest however that in order for agile to be deployed successfully, SOA organizations took care of a number of aspects that in the opinion of the IBM blog authors were perceived as conductive to SOA success. Two aspects that were deemed most important in this study, are the attention to business-IT alignment and the attention to managing risks (or the pitfalls) imposed due to the use of agile methods. Our results (Figure 3) also suggest that for agile to work in SOA, agile practices need to be thought in terms of at three levels: organizational, project and development level.

As part of carrying out this research, we compared our findings with previously published ones. Our results confirm the observations of Christou et al. [6], that the agile paradigm fosters business value generation in SOA initiatives even in organization known as 'conservative' (e.g. such as banking). Similarly to these authors, the IBM blog authors shared observations that agile approaches to SOA in their settings resulted to keeping projects in time and within budget. This comparison between our results and those in [6] allows us therefore to say that our study adds up to the body of evidence suggestion that agile and SOA mutually support each other and their combination adds value for organizations.

Our results have been compared also with the literature sources listed in Section 2. Unlike the empirical studies mentioned there, our work provides a more detailed description of how agile and SOA initiatives work together. We however note that we were surprised that organizational distributedness of teams (which is the subject of [16]) was not included as a topic of discussion on the IBM blog. Would it mean that no one of the practitioners considered it an issue in agile and SOA? Or was it not an issue solely in client organizations

implementing SOA with the involvement of IBM? As we could not find any hint in our textual data to those questions, we think that they are good candidates for being researched in the future.

Furthermore, when comparing the papers of Section 2, it becomes evident that the SOA organization environments treated in literature (e.g. [17, 23, 24]) and those in this study, differ. While this paper includes several statements about the scalability, complexity, and continuity of the good practices, prior papers such as [17] or [24] miss any discussion about it. Reflecting on the research, the research gap in terms of scalability, complexity, and continuity implies that reality is by far more complex than previous studies considered and that further research could explore these factors further.

We also note that the blog authors did not present a process view of agile, e.g. by using a detailed descriptions of steps, each one defined by its inputs and outputs. Instead, they talked of 'practices'. No practitioner has put forward agile methodological approaches for large-scale projects, such as [33]. This was a surprise, knowing that the Disciplined Agile Delivery approach [33] has been invented at IBM. One might say that practitioners seemed to follow a cherry-picking strategy to selecting and matching agile practices that fit particular client organization's context and SOA initiative. This might well be the case, as the strive for maintaining business-IT alignment and business value generation was well-understood by the audience of the blog and was in fact a driver that determined which practices help SOA projects at organization, project and development level. However, to know for sure, some more exploration is needed and therefore we consider this a line for future research.

7. Conclusion and Implications

This paper presented an empirical study based on the experience of practitioners in the IBM community who publish online on the Developerworks blogging platform. This blog's authors discussed their working practices and pitfalls with regards to SOA and agile software development. The key findings of this research from a practitioner point of view are the following:

To retain the benefits of SOA and agile software development, governance processes should be implemented with focus on traceability and standardization of processes and services as well as the reusability of the services. In this way, one makes sure that assets are actually reused, which saves costs and increases the overall quality.

The second result is the collaboration and communication in cross-functional teams. Integrated teams consisting of developers, architects, testers and business analysts increase the inter-team collaboration and different backgrounds provide synergies between the roles and avoid extreme tendencies. Autonomous teams can decide on their own who fulfills the team roles and they prioritize the work items together with the business. By continuously improving the own processes following industry best practices and feedback sessions involving all stakeholders, the processes become more efficient, effective and aligned with the stakeholders.

This study has some implications for SOA practitioners and for researchers. First, to those SOA practitioners who want to embark on agile, our study could possibly suggest that they might be better off considering choosing as their drivers the generation of business value and the strive for business-IT alignment. Knowing agile is a development and project management paradigm, one needs to "lift" it and reconsider it from a business-IT alignment perspective if one wants to put it at work on a SOA initiative.

Second, our study suggests that practitioners should maintain a business perspective on their SOA initiative if they wish to adopt agile and make is a successful approach to delivering their SOA systems. We found no discussion on technical agile practices (e.g. code-sharing). While this doesn't necessarily mean that such practices are less important, it is indicative that complementing SOA and agile takes a top down perspective. We found that aspects of risk reduction (Section 4.2), alignment (Section 4.4) and business value (Section 4.5) were determining the allocation of the agile practices to various levels, organization's level, project's level and development level.

Furthermore, the academic implications of this paper are the following: First, we used only the IBM Developerworks as data source. Although we think that other organizations are in similar contextual factors and therefore would provide similar insights, extending the scope to other sources could bring further insights which are not experienced by IBM practitioners yet.

Even though the good practices are derived from an industry source, our conceptual model (Figure 3) is not tested empirically. We therefore think that future case study research would be worthwhile to collect evidence of the practical value of the practices in organizations.

Second, we made a step towards more systematic research on joint SOA and agile development. We made first insights on this phenomenon explicit. A logical next step would be to make the connections between the working practices explicit. This would improve the practical implication of this paper. For example, the two practices ***Planning*** and ***Team Responsibility***: While *Planning* focuses on creating an agile overall plan, *Team Responsibility* describes the prioritization of the work items in this plan. However, the connection between both is not elaborated in detail. In our opinion, elaborating the relationships between the concepts and dependencies between the practices would leverage the ability of using SOA and agile software development for the purpose of achieving better alignment and optimizing business value

in a more repeatable manner. This could form a line of future research.

Third, this grounded theory study found pitfalls when using SOA and agile software development conjointly. To the best of our knowledge, no research on this has been done on these "side-effects". Further research on the pitfalls would not only increase the academic knowledge but also help practitioners to mitigate the risks associated with a combined SOA and agile software development approach.

Fourth, this paper described the good practices of joint SOA and agile software development. We created a descriptive theory, which "goes beyond basic description in analyzing or summarizing salient attributes of phenomena and relationships among phenomena" [9]. By doing that, the paper makes a first step to of explicating the knowledge in this area. This knowledge is helpful in the sense of enabling future researchers to ask new and relevant research questions.

8. References

1. Bissi, W., Serra Seca Neto, A. G., Figueiredo Pereira Emer, M.C., The effects of test driven development on internal quality, external quality and productivity: A systematic review, *Info & Soft Technology*, 74, 2016, 45-54
2. Beck, K., Beedle, M., Bennekum, A. van, Cockburn, A., Cunningham, W., Fowler, M., Thomas, D. (2001). Manifesto for Agile Software Development. Retrieved February 11, 2013, from http://agilemanifesto.org/
3. Haas, H., & Brown, A. (2004). Web Services Glossary. *W3C Working Group Note*. http://www.w3.org/TR/ws-gloss/
4. Arsanjani, A., Booch, G., Boubez, T., Brown, P. C., Happell, D., Erl, J. deVadoss T., Wilhelmsen, H. (2009). SOA Manifesto. http://www.soa-manifesto.org/
5. Carvalho, F., Azevedo, L.G., Service Agile Development Using XP, In: *, 2013 SOSE,* 254-259.
6. Christou, I.; Ponis, S.; Palaiologou, E., Using the Agile Unified Process in Banking, IEEE Software, 27(3), 2010, 72-79.
7. Corbin, J. M., Strauss, A. Grounded theory research: Procedures, canons, and evaluative criteria. *Qualitative Sociology*, *13*(1), 3–21.
8. Boehm, B., Rombach, H.D., Zelkowitz, M.V. (eds.), Foundations of Empirical Software Engineering, Springer, 2005.
9. Gregor, S. (2006). The nature of theory in information systems. *MIS Quarterly*, *30*(3), 611–642.
10. Gronli, T.-M., Bygstad, B., A Successful Implementation of Service Oriented Architecture. *26thICAINA Workshops*, 2012, 41–46
11. Callahan, A. G., Suitability of Extreme Programming and RUP Software Information Systems: Service-Oriented Architecture and Software Engineering, Helsinki University of Technology, Finland 2006.
12. Haines, M. N., & Rothenberger, M. A. (2010). How a service-oriented architecture may change the software development process. *Comm. of the ACM*, *53*(8), 135.
13. Ibbotson, J., Chapman, S., & Szymanski, B. K. (2007). The Case for an Agile SOA. *First Annual Conf. of the Information Technology Alliance*, 1–7.
14. Ivanyukovich, A., Gangadharan, G. R., D'Andrea, V., Marchese, M., (2005), Towards a Service-Oriented Development Methodology, J of Integrated Design and Process Science, IOS Press.
15. Jones, Toward an acceptable definition of service, IEEE Software 22(3), 2005, 87-93
16. Karsten, P., Cannizzo, F., (2007), The Creation of a Distributed Agile Team. In: Agile Processes in Software Engineering and Extreme Programming, 235-239.
17. Krogdahl, P., Luef, G., Steindl, C., Service-oriented agility: An initial analysis for the use of agile methods for SOA development. *SCC 2005*, *2*, 93–100 vol.2.
18. Papazoglou, M.P.; Traverso, P.; Dustdar, S.; Leymann, F., Service-Oriented Computing: State of the Art and Research Challenges, Computer 40(11), 2007, 38-45.
19. Roy, S., Debnath, M. K., (2010) "Designing SOA based e-governance system using eXtreme Programming methodology for developing countries", ICSTE, vol. 2, V2-277 -V2-282.
20. Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Softw. Eng.*, *14*(2), 131–164
21. Saldaña, J., *The coding manual for qualitative researchers*. 2009, Sage.
22. Seddon, P. B., & Scheepers, R. (2011). Towards the improved treatment of generalization of knowledge claims in IS research: drawing general conclusions from samples. *European Journal of Information Systems*, *21*(1), 6–21.
23. Schatz, B., (2006) SOA and agile development achieving success through continuous integration and testing, http://www.agileinfusion.com/pdf/agilejournal_article_July2006_SOA.pdf.
24. Shahrbanoo, M., Ali, M., & Mehran, M., An Approach for Agile SOA Development using Agile Pricipals. *IJCSIT 2012*, *4*(1), 237–244.
25. Strauss, A. L., & Corbin, J. M. *Basis of Qualitative Research: Techniques and Procedures for Developing Grounded Theory* Sage, 1998, 312.
26. Sumrell, M. (2007). From Waterfall to Agile - How does a QA Team Transition? *Agile 2007*, 291–295.
27. Sureshchandra, K., & Shrinivasavadhani, J. (2008). Moving from Waterfall to Agile. *Agile 2008*, 97–101.
28. Urquhart, C., Lehmann, H., & Myers, M. D. (2009). Putting the "theory" back into grounded theory: guidelines for grounded theory studies in information systems. *Information Systems Journal*, *20*(4), 357–381.
29. Verner, J.M., Sampson, J., Tosic,V., Bakar, N., Kitchenham, B.,Guidelines for Industrially-Based Multiple Case Studies in Software Engineering. RCIS 2009, 313-324.
30. Hookway, N., 'Entering the blogosphere': some strategies for using blogs in social research, Qualitative Research, 8(1) 91–113
31. Inayat, I., Salim, S.S., Marczak, S., Daneva, M., Shamshirband, S. A systematic literature review on agile requirements engineering practices and challenges, Computers in Human Behavior, 2014, 1-15
32. Ambler, S., Lines, M., Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise, IBM Press, 2012.
33. Yin, R. K., *Case study research,* 2008, Sage.