Balkans JETSS (2020) 2: 148-162
DOI: https://doi.org/10.31410/Balkans.JETSS.2020.3.2.148-162
UDEKOM Balkan
www.udekom.org.rs

THE ROLE OF AGILE SOFTWARE ARCHITECT IN THE AGE OF DIGITAL DISRUPTION AND TRANSFORMATION

Zoran Dragičević[1]
Saša Bošnjak[2]

Received: April 1, 2020 / Revised: June 1, 2020 / Accepted: June 12, 2020

Abstract: *The consequence of the increasing development and use of digital technologies, in every segment of society, is the emergence of digital disruption - a powerful external pressure that is changing the way business is done in all industries. Businesses are responding to digital disruption by digital transformation, which involves organizational change, redefining and aligning digital and business strategies, new business models, increased agility of software development and delivery processes, migration and/or integration of legacy systems using cloud-based platforms and ecosystems. In such a context, one of the key responsibilities of a software architect is to maintain the agility of the organization by defending the flexibility of digital strategy and IT resources so that the enterprise is able to transform and respond adequately and rapidly to the effects of digital disruption. In this regard, the question arises as to how digital disruption and business transformation affect the change in the role, importance, competence and agility of a software architect, especially in the context of the development of complex business software systems. This paper aims to present the role of an agile software architect in the era of digital disruption and transformation, by integrating the results of theoretical and empirical research. A systematic literature review identifies the role, importance, and competencies of a software architect in implementing agile architecture. In other hand, empirical research, based on a case study in a large enterprise, provides a better understanding of the importance of software architect for aligning business and digital strategy, as well as its contribution to increasing the agility of the process of developing, delivering and integrating complex business software systems.*

Keywords: *Agility, Agile software architect, Digital disruption, Digital transformation.*

JEL Classification L86

✉ zoran.dragicevic021@gmail.com

[1] Kompanija Boksit a.d. Milići, RS, Bosnia & Herzegovina
[2] University of Novi Sad, Faculty of Economics in Subotica, Serbia

1. INTRODUCTION

Digital innovations based on digital technologies have led to the emergence of digital disruption - a powerful external pressure on organizations in all industries, which threatens their competitive positions by systematically influencing value creation, disruption, and recombination of the value chain and links between resources, but at the same time facilitates direct interaction and transactions (Skog *et al.*, 2018). Highly innovative technology organizations, referred to as digital disruptors, identify customer needs, create and deliver value to customers at lower costs, using open source tools and platforms, with faster development times and greater impact on user experience than anything seen before (McQuivey, 2013). Accordingly, Baiyere & Hukal (2020) define digital disruption as "*the alteration of a domain-specific paradigm due to the digital attributes of an innovation*".

Faced with such challenges, organizations are undertaking digital transformation initiatives by questioning the wishes and needs of clients, and operational models, where they use the new possibilities to increase competitiveness in order to transform the value proposition to the customers and/or reconfigure the operational models (Berman, 2012). Digital transformation is the response of organizations to the effects of digital disruption, which involves redefining and aligning digital and business strategies, organizational changes, new business models, increasing the agility of development and delivery of software solutions, migration and/or integration of existing systems using cloud-based platforms and ecosystems. Thus, Westerman *et al.* (2011) define digital transformation as "*the use of technology to radically improve performance or reach of enterprises*", while Liere-Netheler *et al.* (2018) define digital transformation in a broader context as „*metamorphosis that is based on the intensive combination of present and future technologies that will change the paradigm of how value-generating processes in and between enterprises as well as with customers take place. Digital transformation will affect business models and corporate strategies*".

In order to survive in a contemporary ever-changing environment, organizations must be adaptable, collaborative, agile, innovative, and without fear of failure. While more and more organizations are embracing new agile principles such as creating incredible conditions, security as a prerequisite, experimenting and learning quickly and constantly gaining value, there is a lack of flexibility, especially in preserving the long-term ability of organizations to adapt and change in response to digital disruption (Prikladnicki *et al.*, 2017). Although the agile movement has had a tremendous impact on changing modes, some agile aspects have become dogmatic, which has slowed the spread of agile principles to some more complex software development contexts. On the other hand, the increasing need for speed and availability has led to technological expansion, characterized by the development of clouds, DevOps / CD movements and software ecosystems, which also creates new needs for process and project management (Kruchten, 2019), whereby technology alone cannot eliminate the root causes of failure to adopt agile principles and processes, such as misunderstanding of contexts, communication barriers, or poor collaboration, as these activities cannot be automated (Ozkaya, 2019).

With all this in mind, it seems that an agile software architect has an increasing potential to become a key factor in enhancing agility and defending the flexibility of digital strategy and IT resources so that an organization is able to transform and adequately respond to the effects of digital disruption launching digital transformation initiatives or becoming a digital disruptor on its own through digital innovation. In this connection, the question is how digital disruption and business transformation affect the change in the role, importance, competencies and agility of a software architect, especially in the context of agile development of complex business software systems. This paper aims to determine the importance and role, responsibilities and competencies of an agile software architect in the era of digital disruption and transformation.

The rest of the paper is structured as follows: Chapter 2 describes the methodology of theoretical and empirical research. Chapter 3 presents the results of a systematic literature review and a case study of a large enterprise. Chapter 4 discusses the results, answers the research question, and compares the results of similar research, with the limitations of the research outlined. Chapter 5 identifies emerging trends and future research directions, while Chapter 6 provides concluding considerations, implications, and suggestions for further research.

2. METHODOLOGY

The theoretical part of the research starts from the results of research into the trends, challenges, success factors and practices of agile architecture in the digital era (Dragičević & Bošnjak, 2019a) and represents its logical extension. It was implemented using the Systematic Literature Review (SLR) methods and guidelines of Kitchenham (2007). The SLR protocol aims to identify the key roles, responsibilities and competencies of the software architect in implementing agile architecture. A next query was applied to the automatic search process: *(agile OR lean OR evolutionary OR continuous) AND architect* AND software AND development*, for database search IEEE Xplore, ArXiv, ACM (*title* or *abstract*), that is, a query *allintitle:"agile architecture" OR „continuous architecture" OR „lean architecture" OR „evolutionary software architecture"* for Google Scholar. Based on the inclusion criteria, papers were selected that define or discuss the role of the software architect in implementing Agile, Continuous, Lean, or Evolutionary architecture, 2009-2019. Non-primary studies, not published in journals or conferences, not related to the research objective, or merely mentioning terms in the query are excluded. Selected papers, after eliminating duplicates, undergo quality assessment criteria, defined by the recommendations of Kitchenham (2007) and Dybå & Dingsøyr (2008). Snowballing was used for manual search (Webster & Watson, 2002; Jalali & Wohlin, 2012). The results of the selection and quality assessment of the papers are given in Table 1. Coding and thematic analysis techniques were used for the extraction of data, qualitative analysis and data synthesis.

Table 1. Search and selection results

Criteria	IEEE Xplore	ACM	ArXiv	Google Scholar
After a search by queries	915	973	47	173
Selected on the basis of title and/or abstract	31	12	3	10
After removing duplicates and quality assessment	19	4	1	4
Added after manual search (snowballing)	5			
Total:	**33**			

Source: Authors

The empirical part of the research was realized at Kompanija Boksit a.d. Milići (hereafter Boksit), based on the case of the development of a complex software system, as a logical continuation of previously realized research in a given context (Dragičević & Bošnjak, 2019b, 2019c). The case study aimed to evaluate the results of the theoretical part of the research with a better understanding of the importance of software architect for aligning business and digital strategy, as well as its contribution to increasing the agility of the process of development and integration of complex business software systems. In the data collection process, a semi-structured interview with one of the authors was used, who in various roles, including the role of the software architect, was in charge of defining and operationalizing the business and digital strategy (Dragičević & Bošnjak, 2019b, 2019c). Key case study research questions were based on SLR results to empirically val-

idate them through exploring the role of software architect in Boksit, given the broader context, business vision and strategy, as well as organizational, architectural, methodological and infrastructure aspects of the software development and delivery process. A qualitative approach was used to analyze the data because it supports a more detailed description of the observed phenomenon and better insight into the complex processes (Eisenhardt & Graebner, 2007). To reduce the risk of bias, the study was conducted by two authors.

3. FINDINGS

This part of the paper first presents the results of the theoretical part of the research (SLR) and then the results of the case study.

A. SLR RESULTS

The first part of the SLR identified 5 key roles of an agile software architect: *Leader of change* (Business perspective), *Linking Element* (Organizational perspective), *Pragmatic Architect* (Architectural perspective), *Servant Leader & Facilitator* (Development perspective) and *Workflow & Traffic Enabler* (Operational perspective). Each role, and its associated perspective, is the result of coding, thematic analysis, and synthesis that defines the key responsibilities of an agile software architect (hereafter architect):

Leader of Change (Business perspective)

Understanding context, vision, and strategy: The architect strives to achieve measurable and realistic goals (Buschmann, 2012); focuses on stakeholder needs rather than plan or budget (Erder & Pureur, 2016); has a broad view, see other industries as a source of ideas (Erder & Pureur, 2016; Hohpe *et al*., 2016); takes care of business, social and cultural aspects (Hohpe *et al*., 2016); enables fundamental improvements in the organization's capabilities (Bass, 2017); sees architectural agility as a comparative advantage (Sturtevant, 2017); understands business goals, context & enable customization to context (Ozkaya, 2019).

Business and IT alignment: The architect continuously exchanges ideas with a wider group of people, from the very beginning (Blair *et al*., 2010; Buschmann, 2012; Hadar & Sherman, 2012); think beyond structure and technology (Buschmann & Henney, 2013; Nord *et al*., 2014); understands how the system will be used (Mirakhorli & Cleland-Huang, 2013); raises systemically important issues and balances opposing views (Woods, 2015); assesses architectural impact in economic terms: cost & business value (Martini & Bosch, 2016; Poort, 2016); acts quickly and facilitates decision making in an uncertain environment in collaboration with stakeholders (Erder & Pureur, 2016; Hohpe *et al*., 2016).

Risk and time dimension management: The architect takes into account the temporal dimension of architecturally important events (Poort, 2016); assesses the architectural impact on risk (Martini & Bosch, 2016; Poort, 2016); assesses the technical risk of developing complex and critical systems (Waterman, 2018b); demonstrates the benefits of new technologies by designing an executable prototype (Erder & Pureur, 2016); understands that erosion of architecture can lead to technical bankruptcy (Sturtevant, 2017); understands and maps the essencial characteristics of development process selection to the context (Ozkaya, 2019).

Linking Element (Organizational perspective)

Horizontal and vertical organization structuring: The architect deals with the organization structure (Nord *et al*., 2014; Shahin *et al*., 2019); connects parts of the organization horizontally and vertically (Hohpe *et al*., 2016); builds bridges between and across different levels of organization (Hohpe *et al*., 2016); considering highly skilled teams (Shahin *et al*., 2019); architects in the organization must work better together, e.g. working as an "expert council" (Martensson *et al*., 2019).

Organizing and coordinating teams: The Architect provides communication and avoids the "ivory-tower" trap (Blair *et al*., 2010; Faber, 2010; Erder & Pureur, 2016; Holmes & Nicolaescu, 2017); can be a team member as the first among equals (Blair *et al*., 2010; Britto *et al*., 2016); aligns multi-team coordination requirements and opportunities (Nord *et al*., 2014); builds bridges between teams (Hohpe *et al*., 2016); influences team morale (Bass, 2017); considers highly skilled teams (Shahin *et al*., 2019); supports the team members conventionally more business-oriented (Bass, 2019).

Pragmatic Architect (Architectural perspective)

Architectural vision and decomposition strategy: The architect delivers architecture as a service (Blair *et al*., 2010; Faber, 2010); defines the architectural vision and gives good examples (Buschmann, 2012; Martensson *et al*., 2019); provides a coherent and sustainable product architecture (Erder & Pureur, 2016); understands the system, its parts, and how they communicate (Bass, 2017); drives decomposing strategies (Shahin *et al*., 2019); designs loosely coupled architectures (Shahin *et al*., 2019); plans for integration in steps (Martensson *et al*., 2019).

Focus on QA & ASR: The architect focuses on quality attributes (QA) and architecturally significant requirements (ASR) (Faber, 2010; Woods, 2016; Larrucea *et al*., 2018; Waterman, 2018a); identifies defects that are key to sustainability and evolution (Britto *et al*., 2016); manages architecture and monitors the current state of architecture (Martini & Bosch, 2016; Holmes & Nicolaescu, 2017); defines the most important characteristics (Martensson *et al*., 2019); communicates clearly and continuously the importance of the main ASR (Martensson *et al*., 2019); has feedback from the team or analysis tools on the status of the system, to understand if the architectural requirements are at risk (Martensson *et al*., 2019).

Simple high-level design: The architect defines boundaries and work frames, selects templates and components (Madison, 2010; Durdik, 2011); documents top-level macro architecture (Gerdes *et al*., 2016); creates a high-level design, in particular, interactions among components/services (Shahin *et al*., 2019); uses architecture standards, reference architectures and well-accepted architectural principles to gain control of complexity (Sturtevant, 2017; Bass, 2019); establishes and refines conventions for structuring large scale software systems (Bass, 2019); creates a software structure that enables the autonomy and effectiveness of the developers (Martensson *et al*., 2019); manages architectural health as a codebase grows (Sturtevant, 2017); uses good practices to keep designs simple (Waterman, 2018a).

Planning for options and delay of decisions: The architect's main result of work is decisions, not documentation (Poort, 2014); delays decisions, plans for options, and thinks in the direction of the Minimum Viable Architecture (MVA) (Blair *et al*., 2010; Madison, 2010; Hadar & Sherman, 2012; Buschmann & Henney, 2013; Erder & Pureur, 2016; Holmes & Nicolaescu, 2017; Waterman, 2018a; Shahin *et al*., 2019); takes responsibility for decisions that are risky, expensive, and difficult to change (Woods, 2016; Bass, 2017), uses pragmatic modeling (Hohpe *et al*., 2016; Zimmermann, 2016).

Fast delivery before reuse: The architect designs systems for rapid delivery in different environments (Erder & Pureur, 2016); avoids overthinking about reusability, but understand that a lack of explicit control on reuse makes CD harder (Shahin *et al.*, 2019); uses a balanced approach where system design and architecture effort is focused on the system's most important characteristics (Martensson *et al.*, 2019).

Servant Leader & Facilitator (Development perspective)

Team support: The architect understands source code and codes as needed (Babar, 2009; Madison, 2010; Buschmann, 2012; Mirakhorli & Cleland-Huang, 2013; Erder & Pureur, 2016); transfers technical knowledge as a consultant (Babar, 2009; Martini & Bosch, 2016; Martensson *et al.*, 2019); assists the team in "breaking the rules" (Faber, 2010); removes barriers that block team agility and frustrate stakeholders (Buschmann & Henney, 2013; Mirakhorli & Cleland-Huang, 2013; Erder & Pureur, 2016; Woods, 2016); translates complex concepts into understandable concepts (Hohpe *et al.*, 2016); works in a decentralized style (Erder & Pureur, 2016); helps the team understand and implement the chosen development process and best practices (Bass, 2019).

Architectural agility: The architect is involved in all stages of the development process (Hadar & Sherman, 2012); evaluates a design, code, and functionality - uses tools (Madison, 2010; Mirakhorli & Cleland-Huang, 2013); rules the delivery process (Erder & Pureur, 2016); maximizes the team's architectural agility (Waterman, 2018a); places greater emphasis on evolutionary changes (Shahin *et al.*, 2019); helps build architectures that are responsive to needs over time (Shahin *et al.*, 2019); balances focus on agile process and agile architecture (Sturtevant, 2017); uses an agile process to create a change-tolerant architecture (Waterman, 2018a); proves the architecture with code iteratively (Madison, 2010; Waterman, 2018a).

Communication: The architect prioritizes mentoring and learning over documentation (Mirakhorli & Cleland-Huang, 2013; Hohpe *et al.*, 2016); uses code as a form of documentation and a means of communication (Prause & Durdik, 2012); spends the most time with people living with his decisions (Buschmann, 2012; Mirakhorli & Cleland-Huang, 2013); minimizes multitasking on simultaneous projects (Erder & Pureur, 2016); implements the prescribed processes (Bass, 2017); communicates iteratively with the team (Martensson *et al.*, 2019); sees communication as a key challenge and encourage frequent and on-task communication among all stakeholders and team members (Ozkaya, 2019).

Preserving system integrity: The architect ensures that team's decisions are consistent throughout the system (Madison, 2010; Buschmann, 2012); influences design decisions that affect the whole system (Buschmann, 2012); protects the conceptual integrity of product architecture and design (Buschmann, 2012; Poort, 2014; Erder & Pureur, 2016); manages technical debt (Zimmermann, 2016); reduces risk to a level that's satisfactory to the team and stakeholders (Waterman, 2018b); balances between risk and agility (Waterman, 2018b); designs for failure in which, instead of preventing failures (reliability), learn how to deal with failures (resilience) (Shahin *et al.*, 2019).

Collaboration with users: The architect has empathy for users and works closely with users - the „follow-me-home" technique (Mirakhorli & Cleland-Huang, 2013); receives real requests from users feedback (Mirakhorli & Cleland-Huang, 2013); assesses team's and customer's risk tolerance (Waterman, 2018b); balances customer demands, with a focus on their delivery (Erder & Pureur, 2016); removes artificial barriers to delivering better, faster, cheaper software to the users (Ozkaya, 2019).

Workflow & Traffic Enabler (Operational perspective)

DevOps integration and automation: The architect engages in sophisticated DevOps development and production infrastructure (Nord *et al.*, 2014; O'Connor *et al.*, 2016; Larrucea *et al.*, 2018); takes into account the complete life cycle of the software product (Erder & Pureur, 2016); provides product delivery resources (Erder & Pureur, 2016); works with the team to ensure their familiarity with DevOps tools (Bass, 2017); ensures that the development, build, staging, and production environments are as identical as possible (Bass, 2017); takes care of the scaling, complexity, and distribution of the software (Britto *et al.*, 2016); makes tailored DevOps strategy (Larrucea *et al.*, 2018).

Monitoring & tracing: The architect in collaboration with the team defines the subject of monitoring and reporting (Bass, 2017); deals with errors when the system is in production (Bass, 2017); enables traceability and determination of the sequence of events that led to the error (Bass, 2017); extensively uses monitoring tools (Shahin *et al.*, 2019); exposes the different types of log and monitoring data with a standard format (Shahin *et al.*, 2019); makes architectural design decisions to improve the testability (Shahin *et al.*, 2019).

The second part of the SLR identifies the necessary competencies of an agile software architect, which include knowledge, skills, and experience, as well as personality:

Competencies

Knowledge, skills and experience: The architect is a generalist (inversion of specialization) (Zimmermann, 2016; Pautasso *et al.*, 2017a); deeply knows and understands the business domain (Martini & Bosch, 2016; Poort, 2016; Holmes & Nicolaescu, 2017); owns technical knowledge and experience (Buschmann & Henney, 2013; Erder & Pureur, 2016; Hohpe *et al.*, 2016; Martini & Bosch, 2016; Bass, 2017); has business, financial, educational, management and other non-technical skills (Poort, 2014; Hohpe *et al.*, 2016); he is a good and passionate communicator (Faber, 2010; Erder & Pureur, 2016; Hohpe *et al.*, 2016); knows technology and hardware infrastructure (Buschmann & Henney, 2013; Holmes & Nicolaescu, 2017); combines and transfers knowledge from isolated domains (Hohpe *et al.*, 2016); has experience in design (Buschmann & Henney, 2013); has the skills needed to communicate and collaborate (Erder & Pureur, 2016; Martini & Bosch, 2016); has experience working with different teams (Erder & Pureur, 2016); has a broad knowledge of the application of architectural and agile practices (Holmes & Nicolaescu, 2017); recognizes and nurtures talents (Bass, 2017); has technical expertise to coordinate agile teams (Bass, 2019).

Personality: The architect possesses leadership qualities (Buschmann, 2012; Woods, 2015; Hohpe *et al.*, 2016; Martensson *et al.*, 2019); charisma (Martini & Bosch, 2016); bases authority on knowledge and a willingness to help (Faber, 2010); has wide horizons (Erder & Pureur, 2016); can work in uncertain conditions (Erder & Pureur, 2016); assume responsibility (Hadar & Sherman, 2012); he is pragmatic (Buschmann & Henney, 2013).

B. CASE STUDY RESULTS

The results of the case study, realized by interviewing an agile software architect in Boksit, generally confirm the SLR results, with discrepancies and/or some doubt as to individual views, as well as the need to highlight the specific role of the agile software architect in the development of a complex business software system:

Understanding context, vision, and strategy: All 7 SLR positions were confirmed, with the further emphasized the need for „*an architect to view data as a strategic resource*".

Business and IT alignment: There is full agreement with 4/6 SLR positions. There is no full agreement with the position that the architect can always understand how the system will be used (Mirakhorli & Cleland-Huang, 2013) because even the users themselves do not usually know it in advance, but rather „*needs intuition, lean thinking and moving forward in small steps*". Also, it's not a shared position that it is easy to evaluate the impact of architecture on cost and business value (Martini & Bosch, 2016; Poort, 2016). On the other hand, the architect is seen as „*a key factor in aligning business and digital strategy*", which requires that „*the architect be focused on the effective operationalization of the business strategy, as well as understand the importance of linking strategic and operational goals with the data*".

Risk and time dimension management: 4/6 SLR positions were confirmed. There is doubt about attitudes regarding the ability of an architect to pre-determine, both architectural impact on risk (Martini & Bosch, 2016; Poort, 2016) and technical risk in the development of complex and critical systems (Waterman, 2018b). In this connection, the need for „*the architect to keep the software system as open as possible for expansion and connection*" is particularly emphasized.

Horizontal and vertical organization structuring: 3/5 SLR positions were confirmed. In a dynamic environment, organizational change is very common, with the organizational structure being the result of the influence of context, vision and strategy. Therefore, there is a general agreement that the architect must deal with the structure of the organization (Nord *et al.*, 2014; Shahin *et al.*, 2019), but not in terms of defining it, but its influence on the development team, architecture and design of the software system according to Conway's law. There is an agreement in attitude that more architects need to work closely together (Martensson *et al.*, 2019), but this has not been empirically verifiable. On the other hand, it is further emphasized that „*the architect must bear in mind the complex connection between the organizational structure and the culture of the organization*".

Organizing and coordinating teams: 5/7 SLR positions were confirmed. Although there is general agreement on the positions regarding the organization and coordination of multiple teams (Nord *et al.*, 2014; Hohpe *et al.*, 2016), this could not be empirically confirmed. On the other hand, it is further emphasized that „*the architect must promote professionalism and discipline for both team members and key users*".

Architectural vision and decomposition strategy: 6/7 SLR positions were confirmed. There is no agreement with the position that the architect should plan in the steps (Martensson *et al.*, 2019), because the integration must be in the architect's focus from the beginning - through the development of the prototype, as in every subsequent iteration. On the other hand, it further emphasized that „*the architect should use the business architecture as a link between business goals and IT resources, as well as to better understand the interaction between the software system and the user*".

Focus on QA & ASR: 5/6 SLR positions were confirmed. There is no agreement that the architect's insight into the real state of the software system and understanding of architectural risks on sustainability and evolutivity may be primarily based on feedback from team members and analytical tools (Martensson *et al.*, 2019), but on the feedback from key users, a complete understanding the source code and design details to the lowest level, also. On the other hand, it is emphasized that „*the architect must consider safety aspects and requirements from the outset, as security significantly affects the overall design*".

Simple high-level design: 7/8 SLR positions were confirmed. Although there is a general agreement to define and implement conventions when it comes to structuring large scale software systems (Bass, 2019), this could not be empirically confirmed. On the other hand, it is emphasized that „*the architect should use the Convention Over Configuration design paradigm to keep under control the number of design decisions, especially when it comes to sharing common information between parts of a software system*".

Planning for options and delay of decisions: All 5 SLR positions were confirmed, with the further emphasized the need that „*the architect must pay particular attention to the possible negative impact of design decisions on the rights, preferences, customization, and personalization of users*".

Fast delivery before reuse: All 3 SLR positions were confirmed, with the further emphasized the need that „*the architect must take care of the delivery of both, current and future values to the user*".

Team support: 6/7 SLR positions were confirmed. While there is a general agreement with the position that an architect should translate complex concepts into understandable terms (Hohpe *et al*., 2016), it is emphasized that before that, "*in communication with team members and users, the architect must fully clarify and define all complex concepts in order to software system be user-friendly* ".

Architectural agility: 7/9 SLR positions were confirmed. The dogmatic application of agile processes is not sufficient for the continuous rapid, independent delivery of autonomous parts of a complex software system to the production environment. Therefore, there is no full agreement on the positions of balanced focus on agile process and agile architecture (Sturtevant, 2017), and use an agile process to create a modifiable, change-tolerant architecture (Waterman, 2018a). Instead of the dogmatic application of agile processes, it is much more important that „*the architect continuously creates, adapts and advances the conditions for increasing the agility of the development team, the development process, and architecture, especially in a context that requires the integration of existing complex business software systems*". On the other hand, it is emphasized that „*the architect should use design principles that support architectural agility*".

Communication: 7/8 SLR positions were confirmed. There is no full agreement with the position that the architect and the team should communicate only iteratively (Martensson *et al*., 2019), but „*at any time when the need arises, the sooner the better*".

Preserving system integrity: All 3 SLR positions were confirmed. Besides, it is emphasized that „*the architect must have a special responsibility for the application of systemic business rules and the preservation of the integrity of shared resources*."

Collaboration with users: 4/5 SLR positions were confirmed. There is no full agreement with the position that the architect receives the right requests only from user feedback (Mirakhorli & Cleland-Huang, 2013), but emphasizes that the user requirements should be the result of a proactive approach in which „*the architect, by relying on his knowledge and experience, through continuous collaboration and communication helps from the outset key users to better articulate requirements for the development of a minimum viable product*".

DevOps integration and automation: 6/7 SLR positions were confirmed. While there is an agreement that the architect must deal with DevOps production infrastructure (Nord *et al*., 2014;

O'Connor *et al*., 2016; Larrucea *et al*., 2018), it is noted that infrastructure does not have to be highly sophisticated or fully automated. On the other hand, it was emphasized that „*an architect must be able to implement a secure development, test and production environment, usually in the form of a private cloud, using technologies that have reached a certain level of maturity*".

Monitoring & tracing: 5/6 SLR positions were confirmed. Although there is an agreement with the position that the architect must use monitoring tools (Shahin *et al*., 2019), it is emphasized that this not need be so intense, but that „*the architect should adjust the degree of monitoring and logging to the needs*".

Knowledge, skills and experience: 12/13 SLR positions were confirmed. While there is general agreement on the position that an architect must possess the technical expertise to coordinate agile teams (Bass, 2019), this could not be empirically confirmed. Besides, it was emphasized that „*the architect must be able to solve very complex problems and situations, as well as support the adoption of new, innovative technologies and their proper application*". Also, specialization inversion is necessary „*not only for the architect but for other team members too".*

Personality: All 7 SLR positions were confirmed. Besides, it is emphasized that „*the architect must have a good intuition and an open mind to solve non-standard, complex problems, in which he must show exceptional commitment and perseverance*".

4. DISCUSSION

In this section, general considerations are given, the results obtained are discussed and the research question is answered, the results are compared to other similar studies, and the limitations of the research are presented:

General considerations: Regarding the theoretical part of the research (SLR), out of a total of 33 selected studies, 23 (70%) were published in a journal and 10 (30%) at conferences. Most of the papers were based on expert opinion 15 (46%), followed by empirical studies 10 (30%) and papers based on experience 8 (24%). Bearing in mind that papers based on the opinion and/or experience of experts are prevalent, the results of theoretical research have been further validated through a case study. The results of the case study indicate a high percentage of complete agreement with SLR positions (109/128 or 85%), with the lowest agreement regarding organizational (8/12 or 69%) and business (15/19 or 79%) perspectives, while the highest agreement is with the positions from an architectural perspective (26/29 or 90%). On the other hand, the results of the case study further point to certain specifics when it comes to the role and competence of an agile software architect in the development of a complex business software system, especially from a business and organizational perspective.

The answer to the research question: The role of an agile software architect has changed dramatically compared to that of a traditional software architect. It has evolved from a specialist in the traditional architectural domain, focusing on making costly changes in the early stages of the project, planning functionalities and defining a detailed structure and behavior of the system, to a generalist in the digital-architectural world, with evolution, change and time in focus. On the long side, the role of a software architect has changed in the agile movement as well. It has evolved from an initial stance of „no software architect at all", through „team as an architect", to a software architect as a member of an agile team responsible for the top-level architecture design. The intensive development of Internet-based systems, cloud technologies, CD and DevOps practices, and the growing

use of microservices, which enable rapid, autonomous software delivery in a distributed production environment and rapid user feedback, have contributed in particular to this. Digital technology development, digital innovation and digital disruption create a business environment where new business models are increasingly relying on digital strategies, which are changing traditional IT strategies, with the tendency to become business strategies. In this context, instead of the traditional "ivory tower" (top-down) role, on the one hand, and „no software architect at all", through „team as an architect" (bottom-up), on the other hand, the software architect should take the middle-out role. Such a role means greater responsibility, extended beyond the software architecture and software development process, to the business vision and strategy, organizational aspects and DevOps infrastructure. It requires much more knowledge, skills and experience, as well as appropriate personality traits. An agile software architect is a leader of change, not only a pragmatic architect who leads a team and serve the team, but a key element that connects business vision, strategy, business goals, IT resources, parts of the organization, teams, and stakeholders, in a way that promotes and enables evolutionary development and continuous delivery of software to the production environment, to ensure a continuous flow of value for the user, in the short and long term.

Related work: To the best of our knowledge, the only paper exploring the role of software architect in an agile development process using SLR methods and empirical research has been published by Marić & Tumbas (2016). The authors note that the software architect is formally a member of an agile team, usually a senior developer and/or team leader, involved in the entire development process. Design decisions are made by the software architect in collaboration with the client while coordinating the team's work on the detailed design. In addition to the software architect role, the authors identify the following roles: system architect - who has a managerial position in the organization and is responsible for the architecture of the entire system; and solution architect - who is part of the software delivery team and work closely with the client organization operations team. The authors conclude that because of the great architectural challenges, agile team seeks to shift responsibility for architecture to a software architect, who must be an experienced individual, with superior technical knowledge. Although the results indicate the importance of a software architect in an agile team, research is limited to the software development process, while the business, organizational, and infrastructure dimensions are partially addressed through a brief description of the roles of the system and solution architect. If we compare the results of Marić & Tumbas (2016) with the described role of agile software architect in the era of digital disruption and transformation, we can see a trend of integrating the roles of software, system and solution architect.

Limitations: To reduce the risk of bias more researchers are involved in all phases of SLR and case study research. To reduce the risk of omission of relevant studies, the query is tailored to each database with an additional manual search. However, the risk of bias of primary study authors should also be taken into account, as papers based on the opinion or experience of the author predominate. On the other hand, the case study is based on the example of the development of complex business software in one company and qualitative data, where a semi-structured interview technique was applied for qualitative analysis.

5. FUTURE RESEARCH DIRECTIONS

Several trends can be observed. The trend of inversion of specialization is spreading to all members of the development team, which is related to the decrease in the size of the development team and the appearance of the so-called "two-pizza" teams, whose members are extremely qualified, disciplined and motivated. Thus, the development of microservices requires full-stack developers who combine database design, integration, domain business logic and user interface skills. Also,

there is a trend of decreasing the number of architectural decisions by encapsulating them into the technological DevOps environment, middleware platforms, software tools and collaborative development environment. This trend is especially supported by the growing of serverless computing infrastructure for services deployment and scaling, which is hidden from developers (Hohpe *et al.*, 2016; Pautasso *et al.*, 2017b). On the other hand, despite expectations that the role of the software architect becomes virtual or the responsibility of the team, it seems that the agile software architect takes a key place in the software development and delivery process, with an exciting future ahead with the new challenges of developing the next generation of intelligent-connected systems, systems made up of external services, as well as the use of ML (Machine Learning) and analytics in system design, the integration of many IoTs, and the use of dynamic run-time. It is expected that architecture from the static structure and quantified data will gradually evolve into something that is more statistical characteristics and trends, with the increasing importance of algorithms and data, software-defined architectural design and dynamic service composition. The role of an agile software architect will also evolve towards increased monitoring of information flow and decision making just-in-time, with increasing AI (Artificial Intelligence) support, increased focus on politics, policies, algorithms and probability (Hohpe *et al.*, 2016; Woods, 2016). Therefore, future research could more closely address the identified trends.

6. CONCLUSION

Digital disruptors are organizations that take advantage of the favorable conditions of intensive digital technology development to expand digital innovation that is changing the paradigm of business across all industries. This puts a great deal of external pressure on other organizations, which must increase agility while preserving the flexibility of digital strategy and IT resources to transform and adequately respond to the challenges of digital disruption. In this regard, this paper presents the results of a combined theoretical (SLR) and empirical (case study) research of the impact of digital disruption and transformation on changing the importance, role, responsibility, competencies, and agility of a software architect, with particular reference to the development of complex business software systems. Within the SLR, 33 papers from journals and conferences were selected and analyzed, while a case study of a complex business software system was conducted.

The research findings indicate the increased importance, and changed the role and responsibilities of an agile software architect in the era of digital disruption and transformation. The SLR identified and confirmed through a case study 5 key agile software architect roles in 5 different perspectives: 1) *Leader of Change* role from a business perspective - assumes responsibilities related to understanding context, business vision, and strategy, aligning business and IT, and risk & time management; 2) *Linking Element* role from an organizational perspective - implies responsibilities related to horizontal and vertical structuring of the organization, team's organization and coordination; 3) *Pragmatic Architect* role from an architectural perspective - implies responsibilities related to architectural vision and decomposition strategy, focus on quality attributes and architecturally significant requirements, simple top-level design, planning for options and delay of decisions, and a greater focus on fast delivery, relative to reuse; 4) *Servant Leader & Facilitator* role from a development perspective - implies responsibilities related to team support, architectural agility, communication, preservation of product integrity and collaboration with customers; and 5) *Workflow & Traffic Enabler* role from an operational perspective - assumes responsibilities related to DevOps integration, automation, monitoring, and tracing. To successfully realize these roles and responsibilities, agile software architect must have exceptional competencies, which imply a very broad knowledge, skills and experience, as well as appropriate personality traits to achieve a synergistic effect in working with stakeholders, especially with other team members and key users.

The results have implications for both, practitioners and the scientific community. It can be stated that the traditional "ivory tower" role of the software architect has been overcome, as well as the initial expectations of the agile approach proponents that the software architect has no place in the agile team and/or that the architecture should be the responsibility of the agile team. Instead, findings suggest that an agile software architect needs to take a middle-out position on the shoulders of the team to fully realize all different roles in related perspectives - the scope of which extends beyond the software product life cycle. In doing so, organizations can use the identified competencies as additional recruitment guidelines, for both, agile software architects and beginners who have the potential to take on such a challenging role over time. On the other hand, the obtained results enable a better understanding of the importance of an agile software architect for aligning business and digital strategy, as well as its contribution to increasing the agility of the process of development and integration of complex business software systems.

Given the limitations of the research, the results obtained should be subject of additional empirical research, in different contexts of the development of complex software systems, with a particular focus on the business and organizational perspective. Further research could address in more detail the observed trends related to inversion of specialization, the impact of technology on reducing the number of architectural decisions, the challenges of developing the next generation of intelligent-connected systems, and the further evolution of the role of agile software architect in such an environment.

REFERENCES

Babar, M. A. (2009) 'An exploratory study of architectural practices and challenges in using agile software development approaches', in *2009 Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture*, pp. 81–90.

Baiyere, A. & Hukal, P. (2020) 'Digital Disruption: A Conceptual Clarification', *Proceedings of the 53rd Hawaii International Conference on System Sciences*, 3, pp. 5482–5491.

Bass, J. M. (2019) 'Agile on a Large Scale', *Itnow*, 61(1), pp. 56–57.

Bass, L. (2017) 'The Software Architect and DevOps', *IEEE Software*, 35(1), pp. 8–10.

Berman, S. J. (2012) 'Digital transformation: Opportunities to create new business models', *Strategy and Leadership*, 40(2), pp. 16–24.

Blair, S., Watt, R. & Cull, T. (2010) 'Responsibility-driven architecture', *IEEE Software*, 27(2), pp. 26–32.

Britto, R., Šmite, D. & Damm, L. O. (2016) 'Software Architects in Large-Scale Distributed Projects: An Ericsson Case Study', *IEEE Software*, 33(6), pp. 48–55.

Buschmann, F. (2012) 'A week in the life of an architect', *IEEE Software*, 29(3), pp. 94–96.

Buschmann, F. & Henney, K. (2013) 'Architecture and agility: Married, divorced, or just good friends?', *IEEE Software*, 30(2), pp. 80–82.

Dragičević, Z. & Bošnjak, S. (2019a) 'Agile architecture in the digital era: Trends and practices', *Strategic Management*, 24(2), pp. 12–33.

Dragičević, Z. & Bošnjak, S. (2019b) 'Digital transformation in the mining enterprise: The empirical study', *Mining and Metallurgy Engineering Bor*, (1–2), pp. 73–90.

Dragičević, Z. & Bošnjak, S. (2019c) 'Harmonizing business and digital enterprise strategy using SOA middle-out and service-based approach', *Journal of Engineering Management and Competitiveness (JEMC)*, 9(2), pp. 97–112.

Durdik, Z. (2011) 'Towards a process for architectural modelling in agile software development', in *Proceedings of the joint ACM SIGSOFT conference -- QoSA and ACM SIGSOFT symposium -- ISARCS on Quality of software architectures -- QoSA and architecting critical systems -- ISARCS - QoSA-ISARCS '11*, pp. 183–192.

Dybå, T. & Dingsøyr, T. (2008) 'Empirical studies of agile software development: A systematic review', *Information and Software Technology*, 50(9–10), pp. 833–859.

Eisenhardt, K. M. & Graebner, M. E. (2007) 'Theory Building from Cases: Opportunities and Challenges', *The Academy of Management Journal*, 50(1), pp. 25–32.

Erder, M. & Pureur, P. (2016) 'What's the Architect's Role in an Agile, Cloud-Centric World?', *IEEE Software*, 33(5), pp. 30–33.

Faber, R. (2010) 'Architects as service providers', *IEEE Software*, 27(2), pp. 33–40.

Gerdes, S., Jasser, S., Riebisch, M., Schröder, S., Soliman, M., *et al.* (2016) 'Towards the essentials of architecture documentation for avoiding architecture erosion', *Proceedings of the 10th European Conference on Software Architecture Workshops - ECSAW '16*, pp. 1–4.

Hadar, I. & Sherman, S. (2012) 'Agile vs. plan-driven perceptions of software architecture', *2012 5th International Workshop on Co-operative and Human Aspects of Software Engineering, CHASE 2012 - Proceedings*, pp. 50–55.

Hohpe, G., Ozkaya, I., Zdun, U. & Zimmermann, O. (2016) 'The Software Architect's Role in the Digital Age', *IEEE Software*, 33(6), pp. 30–39.

Holmes, B. & Nicolaescu, A. (2017) 'Continuous Architecting: Just another buzzword?', in *Full-scale Software Engineering/The Art of Software Testing*, pp. 1–6.

Jalali, S. & Wohlin, C. (2012) 'Systematic Literature Studies: Database Searches vs. Backward Snowballing', in *ESEM'12: Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, pp. 29–38.

Kitchenham, B. (2007) 'Guidelines for performing Systematic Literature Reviews in Software Engineering'.

Kruchten, P. (2019) 'The End of Agile as We Know It', in *IEEE/ACM International Conference on Software and System Processes (ICSSP)*. Institute of Electrical and Electronics Engineers (IEEE), pp. 104–104.

Larrucea, X., Santamaria, I., Colomo-Palacios, R. & Ebert, C. (2018) 'Microservices', *IEEE Software*, 35(3), pp. 96–100.

Liere-Netheler, K., Vogelsang, K., Packmohr, S. & Hoppe, U. (2018) 'Towards a Framework for Digital Transformation Success in Manufacturing', *26th European Conference on Information Systems (ECIS 2018)*, (Schwab 2017), pp. 1–19.

Madison, J. (2010) 'Agile–Architecture Interactions', *IEEE Software*, 27(2), pp. 41–48.

Marić, M. & Tumbas, P. (2016) 'The role of the software architect in agile development processes', *Strategic Management*, 21(1), pp. 16–22.

Martensson, T., Stahl, D., Martini, A. & Bosch, J. (2019) 'Continuous architecture: Towards the goldilocks zone and away from vicious circles', *Proceedings - 2019 IEEE International Conference on Software Architecture, ICSA 2019*. IEEE, pp. 131–140.

Martini, A. & Bosch, J. (2016) 'A multiple case study of continuous architecting in large agile companies: current gaps and the CAFFEA framework', *Proceedings - 2016 13th Working IEEE/IFIP Conference on Software Architecture, WICSA 2016*, pp. 1–10.

McQuivey, J. (2013) *Digital disruption: Unleashing the next wave of innovation*. Forrester Research.

Mirakhorli, M. & Cleland-Huang, J. (2013) 'Traversing the twin peaks', *IEEE Software*, 30(2), pp. 30–36.

Nord, R. L., Ozkaya, I. & Kruchten, P. (2014) 'Agile in Distress: Architecture to the Rescue', in *International Conference on Agile Software Development*, pp. 43–57.

O'Connor, R. V. O., Elger, P. & Clarke, P. M. (2016) 'Exploring the impact of situational context – A case study of a software development process for a microservices architecture', in *2016 IEEE/ACM International Conference on Software and System Processes (ICSSP)*, pp. 6–10.

Ozkaya, I. (2019) 'Are DevOps and Automation Our Next Silver Bullet?', *IEEE Software*. IEEE, 36(4), pp. 3–95.

Pautasso, C., Zimmermann, O., Amundsen, M., Lewis, J. & Josuttis, N. (2017a) 'Microservices in Practice, Part 1: Reality Check and Service Design', *IEEE Software*, 34(1), pp. 91–98.

Pautasso, C., Zimmermann, O., Amundsen, M., Lewis, J. & Josuttis, N. (2017b) 'Microservices in Practice, Part 2: Service Integration and Sustainability', *IEEE Software*, 34(2), pp. 97–104.

Poort, E. (2014) 'Driving agile architecting with cost and risk', *IEEE Software*, 31(5), pp. 20–23.

Poort, E. (2016) 'Just Enough Anticipation: Architect Your Time Dimension', *IEEE Software*, (December), pp. 11–15.

Prause, C. R. & Durdik, Z. (2012) 'Architectural design and documentation: Waste in agile development?', *2012 International Conference on Software and System Process, ICSSP 2012 - Proceedings*, pp. 130–134.

Prikladnicki, R., Lassenius, C. & Carver, J. C. (2017) 'Trends in Agile Updated: Perspectives from the Practitioners', *IEEE Software*, 35(1), pp. 109–111.

Shahin, M., Zahedi, M., Babar, M. A. & Zhu, L. (2019) 'An empirical study of architecting for continuous delivery and deployment', *Empirical Software Engineering*, 24(3), pp. 1061–1108.

Skog, D. A., Wimelius, H. & Sandberg, J. (2018) 'Digital Disruption', *Business and Information Systems Engineering*. Springer Fachmedien Wiesbaden, 60(5), pp. 431–437. Available at: https://doi.org/10.1007/s12599-018-0550-4.

Sturtevant, D. (2017) 'Modular Architectures Make You Agile in the Long Run', *IEEE Software*, 35(1), pp. 104–108.

Waterman, M. (2018a) 'Agility, risk, and uncertainty, part 1: Designing an agile architecture', *IEEE Software*, 35(2), pp. 99–101.

Waterman, M. (2018b) 'Agility, Risk, and Uncertainty, Part 2: How Risk Impacts Agile Architecture', *IEEE Software*, 35(3), pp. 18–19.

Webster, J. & Watson, R. T. (2002) 'Analyzing the Past to Prepare for the Future: Writing a Literature Review', *Source: MIS Quarterly*, 26(2).

Westerman, G., Calméjane, C., Bonnet, D., Ferraris, P. & McAfee, A. (2011) 'Digital transformation: a roadmap for billion-dollar organizations', *MIT Center for Digital Business and Capgemini Consulting*, pp. 1–68.

Woods, E. (2015) 'Aligning Architecture Work with Agile Teams', *IEEE Software*, 32(5), pp. 24–26.

Woods, E. (2016) 'Software Architecture in a Changing World', *IEEE Software*, 33(6), pp. 94–97.

Zimmermann, O. (2016) 'Designed and Delivered Today, Eroded Tomorrow? Towards an Open and Lean Architecting Framework Balancing Agility and Sustainability', in *Proceedings of the 10th European Conference on Software Architecture Workshops*, p. 7.