Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 237 (2024) 667–674

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Industry Sciences and Computer Science Innovation

An Architectural View Model for Designing and Implementing Microservices-based Systems: Use Case in FinTech

Vu Nguyen Huynh Anh [a, b, *]

[a] Ho Chi Minh University of Banking, 36 Ton That Dam Street, District 1, 70000 Ho Chi Minh City, Vietnam
[b] LouRIM-CEMIS, Université catholique de Louvain, Place des Doyens 1, 1348 Ottignies-Louvain-la-Neuve, Belgium

Abstract

Describing the architecture of microservices systems allows for the illustration of various aspects of design and implementation with information tailored to different stakeholders. On the one hand, the microservices architecture provides flexibility, scalability, faster deployment times, and the potential to design solutions incorporating multiple technologies. An architecture-centered, scenario-driven, iterative development method, on the other hand, is used to explain the system from the perspective of many stakeholders, such as project managers, system architects, developers, and end users. To complement this point of view, this paper proposes an architectural view model that highlights numerous viewpoints and disciplines in microservices-based software development. Such methods are typically well suited for developing and implementing microservices-based software processes to satisfy stakeholders' demands and expectations.

Keywords: Architectural View Model; Microservices; FinTech.

1. Introduction

Software architecture is concerned with the planning and execution of the high-level structure of the software. It is the outcome of combining a specific number of architectural components in a few well-thought-out configurations to satisfy the system's primary functional and performance requirements as well as some additional, non-functional

* Corresponding author.
E-mail address: vunha@hub.edu.vn / vu.nguyenhuynh@uclouvain.be

10.1016/j.procs.2024.05.152

requirements like reliability, scalability, portability, and availability [9]. It used to be challenging to quickly deploy new features to end users because the bulk of applications were created using large monolithic structures and models that were inspired by waterfall processes [8]. The rivalry has changed recently, moving away from the waterfall paradigm and toward the agile style [6]. Furthermore, microservice design encourages the use of agile methods and empowers developers to independently implement and deploy services [14]. It makes it possible for major players to convert their monolithic systems into microservice-based ones or to include brand-new microservices into an existing monolithic system.

This study aims to simulate several perspectives on software processes based on microservices. In order to do this, we provide an architectural view model for developing and utilizing microservices-based applications. The organizational setting and management hierarchies are represented in this model using strategic modeling techniques. The application of this approach within specific perspectives on the creation of microservices systems will next be discussed.

The structure of this paper is as follows. Section 2 provides an overview of the research context, while Section 3 introduces the study technique and methods. The architectural view model named ArcMS is put forth in Section 4 for use in building and deploying microservices-based systems. A use case in FinTech is demonstrated in Section 5 while the evaluation of this model is shown in Section 6. Section 7 presents the paper's conclusion and suggests areas for future research.

2. Research Context

2.1. Architectural View Model

An Architectural View Model, such as the 4+1 View Model [9], is used to describe the architecture (design) of software-intensive systems by utilizing several, concurrent views. End-users, developers, system architects, and project managers all have different perspectives on the system, hence viewpoints are used to describe it from their points of view. The 4+1 View Model defines software design by employing five concurrent views, each of which solves a distinct set of issues: the logical view describes the object model of the design, the process view describes the concurrency and synchronization aspects of the design, the physical view describes the mapping of the software onto the hardware and shows the distributed aspects of the system, and the development view describes the software's static organization in the development environment.

2.2. Microservice Architecture

Lately, the popularity of microservice architecture, which was inspired by service-oriented architecture, has begun to grow [5]. Microservices are described as follows: "*A microservice is an independently deployable component of bounded scope that supports interoperability through message-based communication*"[12] and "*Microservice architecture is a style of engineering highly automated, evolvable software systems made up of capability-aligned microservices*"[12]. Several microservices make up the business logic in a microservice architecture. One microservice handles some of the business logic's external calls. Multiple microservices handle the more sophisticated ones.

3. Research Approach and Methodology

3.1. Research Question

This section discusses the research question in sufficient detail while maintaining the focus on our study. Following the preliminary investigation, we will carry out the following research inquiry in accordance with the context and goals of the study: ***How could an architectural view model for designing and implementing microservices-based systems be represented?*** By using an architectural view model, systems architects can recommend rules for development processes that take stakeholder requirements and expectations into account. The stages involved in conducting research are illustrated in the section that follows.

3.2. Design Science

The research process described in this paper, which is based on the Design Science Research (DSR) methodology for information systems [7, 13], begins with an explanation of the solution's goals: an architectural view model for designing and implementing microservices-based systems in alignment with stakeholders' long-term strategy. Table 1 shows the steps of acquisition architecture used in our study.

Table 1. Design Science Research Methodology for Information Systems: Acquisition architecture's steps.

Steps	Description
1. Problem identification and motivation	The fundamental issue is the inefficiency of the monolithic solution (see Section 1), along with the emergence of emergent technologies. Indeed, with the prevalence of agile development and microservice design, as well as the competitive pressures on enterprises, open and adaptable software systems are necessary for process optimization and proper IT integration;
2. Definition of the objectives for a solution	The objectives are to propose an architectural view model for designing and implementing microservices-based systems;
3. Design and development	Fundamentally, we extend the 4+1 Views Model of Philippe Kruchten and use strategic modeling techniques to represent the organizational setting and management structures of microservices-based systems. This work is depicted in detail in Sections 5;
4. Demonstration	This stage represents the ability and efficiency of architectural view model to support the microservices-based development processes. This demonstration is illustrated in detail in Sections 5;
5. Evaluation	The architectural view model will be evaluated. The evaluation is realized in Section 6;
6. Communication	It has been completed in this paper.

3.3. Research Approach and Method

The architectural view model is built up and validated in terms of software architecture and organizational setting and management structures of microservice systems in this paper, which outlines the research methodology. Basically, ArcMS broadens the 4+1 Views Model of Philippe Kruchten by providing various views for microservices-based software development. Those views are aligned with the entire agile software development process and characterized by using strategic modeling approaches to express the organizational setting and management structures as advice for practitioners.

4. Model

The generic model for describing ArcMS is shown in this section. The architecture of this model is inspired by the 4+1 Views Model of Philippe Kruchten [9]. The model consists of:

- ***The Scenario View***. The central perspective for capturing scenes is this one. In line with this perspective, there are strategic models that describe functional and non-functional requirements in terms of services provided by actors at the strategic level and describe the services in terms of organizational concepts (tasks, goals, qualities, resources, and their dependent actors) at the tactical level, and realize the services through atomic tasks at the operational level. It features a depiction of the complete system. With the support of the i* models [4, 16, 17] (Strategic Service Diagram (SSD), Strategic Dependency Diagram (SDD), Strategic Rationale Diagram (SRD)), and BPMN workflow, this perspective is meant to facilitate the development of microservices-based applications;
- ***The Logical View***. This viewpoint focuses on implementing the functional requirements of the system in terms of structural parts, key abstractions, and mechanisms. It depicts the system's objects as well as their relationship. This perspective is intended to aid in the creation of microservices-based software using Structural Diagrams;
- ***The Process View***. This viewpoint looks at the system's non-functional requirements (such as concurrency, performance, scalability, and usability). It depicts the system's processes (workflow rules) and how those

processes interact with one another. This approach also includes reusable interaction patterns for resolving reoccurring issues and meeting non-functional service levels. Through the use of Communication Diagrams and Dynamic Diagrams, this perspective is intended to aid in the creation of microservices-based software;

- ***The Component View***. This is an architectural picture of a system that includes the components necessary to assemble and release a physical system. It considers the organization of software modules (layer hierarchy, software administration, reuse, and tool restrictions). This view depicts the system's building blocks. This view is intended to aid in the development of microservices-based software by utilizing Component Diagrams and Package Diagrams;
- ***The Deployment View***. This view covers the nodes that make up the hardware structure of the system on which the system runs. It displays the execution environment of the system. This view is intended to aid in the creation of microservices-based software using Deployment Diagrams.

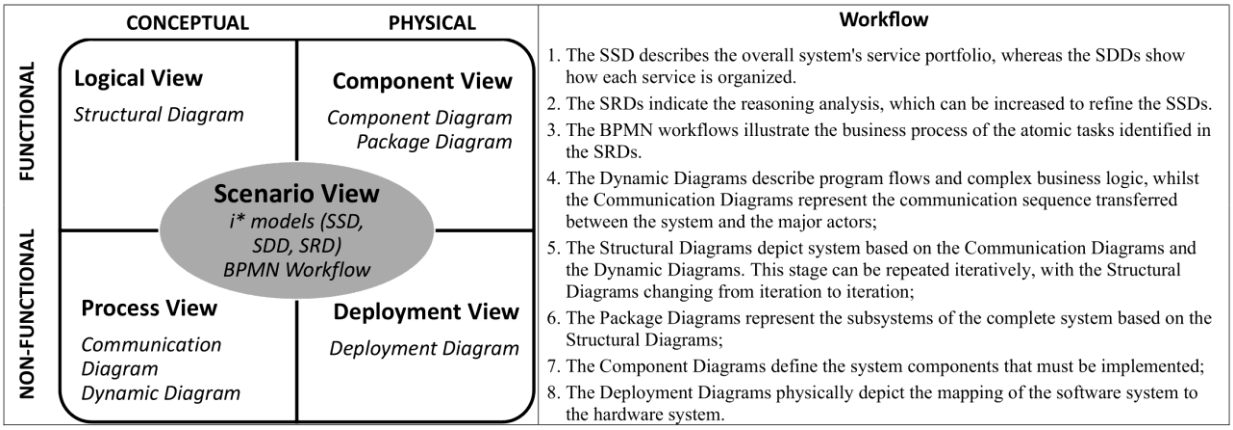

Fig. 1. The 4+1 Architecture Views of the ArcMS model.

Fig. 1 depicts the views of the ArcMS model. The Logical View and the Process View are used to conceptually characterize the system at the analysis and design stages, respectively. The Component View and the Deployment View are used to explain the implementation and deployment stages at the physical level, respectively. The Logical View and the Component View are functionally aligned. They represent the modeling and implementation of system functionalities. Non-functional aspects are realized in the Process View and the Deployment View through behavioral and physical modeling. The Scenario View leads to the Logical View analyzing structural elements and implementing them in the Development View. The Scenario View's services are realized in the Process View and deployed in the Deployment View. The model's workflow can be seen as a guideline for model implementation. The workflow starts with the Scenario View and works its way through the Process View, Logical View, Component View, and Deployment View. Each view of this CASE tool is sequentially illustrated in the Use Case section.

5. Use Case

In this section, the ArcMS model is used with MFin [15], a FinTech application. The management of agents, properties, insurance, and finance departments by real estate enterprises is made possible by this cutting-edge FinTech application, which is built on a microservices architecture. A domain layer and a microservice layer are the two levels that make up MFin's architecture. The former models the IT (back-end), while the latter illustrates the business (front-end). The following microservices make up the microservice layer:

- *User Authentication*: Controls user profile creation, login, and logout;
- *Account*: Controls the establishment, administration, and retrieval of a user's banking accounts;
- *Transaction*: Takes care of user-created transaction production and retrieval;
- *Card*: Controls the creation, maintenance, and retrieval of a user's cards;
- *Wallet*: Controls the creation, management, and retrieval of a user's wallet;

- *Property*: Creates, manages, and retrieves a user's properties.

For each of the main branches, MFin divides the business logic into services. For the property branch, it might, for instance, create a "domain-property" service. This service would utilize a number of microservices, like *User Authentication*, *Account*, *Property*, and *Transaction*.

5.1. The Scenario View

The Scenario View is the initial view developed during the system development lifecycle. This view represents the scenarios that tie the other four views together and explains why the other views exist. It represents important architectural requirements in the form of services. It also helps to ensure that all essential services are provided.

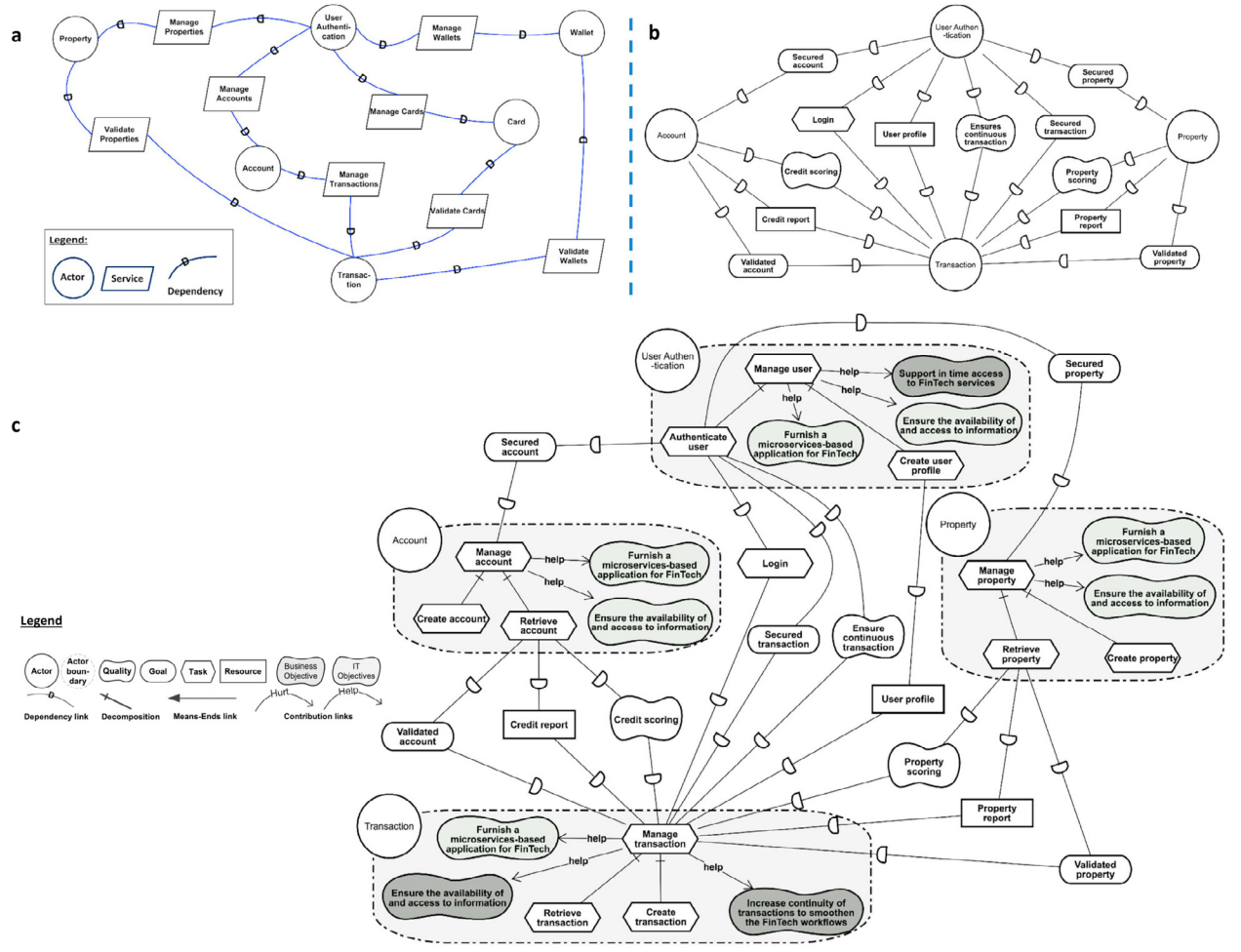

Fig. 2. (a) Strategic Service Diagram; (b) Strategic Dependency Diagram; (c) Strategic Rationale Diagram [15].

The i* framework and its extensions, in this view, provide the Strategic Service Diagram (SSD), the Strategic Dependency Diagram (SDD), and the Strategic Rationale Diagram (SRD) to represent the system at the strategic, tactical, and operational levels, respectively. The Strategic Service Diagram allows us to depict the system's core services. Using organizational concepts, the Strategic Dependency Diagram allows you to depict each service in a static manner. The Strategic Rationale Diagram allows for service implementation via atomic tasks. Each atomic job is described by a business process that the BPMN workflow executes.

The SSD in Fig. 2(a) is used to represent all "domain-property" microservices as strategic services. It briefly identifies and characterizes the many roles that individuals or groups of individuals perform in this domain, as well as the interactions between these various positions.

Fig. 2(b) shows an SDD used at the tactical level in "domain-property" service while Fig. 2(c) depicts the SRD used at the operational level in the "domain-property" service for the property branch. It is a refinement of the SDD.

To handle users, the **User Authentication** task in the SR diagram conducts two particular actions (*Create user profile* and *Authenticate user*). To manage accounts, the **Account** does two unique activities (*Create account* and *Retrieve account*). To handle properties, the Property does two specified activities (*Create property* and *Retrieve property*). To manage transactions, the **Transaction** conducts two specified operations (*Create transaction* and *Retrieve transaction*).

Fig. 3 represents the BPMN workflow that specified atomic tasks of the *Create transaction* task.

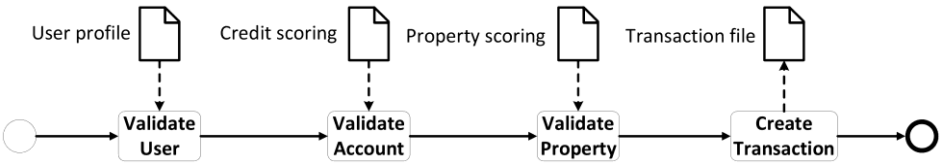

Fig. 3. Create transaction Workflow.

5.2. The Process View

The Process View can be expressed at several levels of abstraction depending on the requirement, such as interactions between systems, subsystems, and objects, and so on. The Communication Diagrams and Dynamic Diagrams express this viewpoint.

- Communication Diagrams: depict the sequence of messages exchanged between objects on a vertical timeline;
- Dynamic Diagrams: depict program flows and complicated business logic with actions, decision points, branching, merging, and parallel processing.

Fig. 4 (a) describes the sequence of messages passed between the main actors in the *Create transaction* while Fig. 4 (b) describes the program flows of the *Create transaction*.

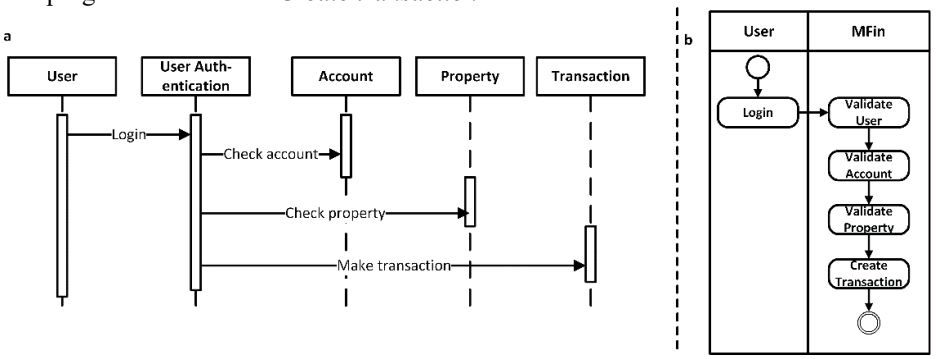

Fig. 4. (a) Communication Diagram; (b) Dynamic Diagram.

5.3. The Logical View

The Logical View depicts structural aspects like classes and objects, as well as their relationships. The Structural Diagrams express this point of view. Structural diagrams depict system domains using basic building blocks that focus on each particular class, the major actions and relationships to other classes' associations, usage, composition, inheritance, etc.

The *Transaction Microservice* is a subsystem of the whole system within the context of the **MFin**. The entire system consists of various domains. Due to the limitation of space, we only illustrate a structural diagram of the Transaction domain in Fig. 5.

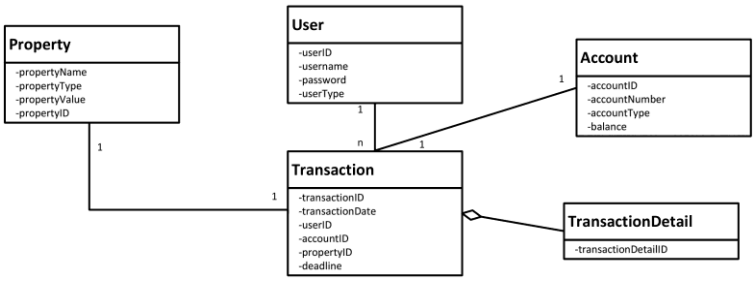

Fig. 5. Transaction Microservice Structural Diagram.

5.4. The Component View

In the development context, the Component View focuses on configuration management and the actual system modules. The system is actually bundled into components that the development team may develop and test. It describes the tangible objects created by the development team. The Package Diagrams and Component Diagrams depict this viewpoint.

Fig. 6 (a) describes the packages diagram while Fig. 6 (b) illustrates the components of the *Transaction Microservice* package.

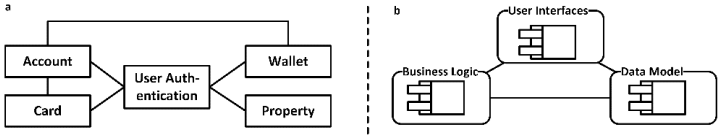

Fig. 6. (a) Packages Diagram; (b) Component Diagram.

5.5. The Deployment View

The Deployment View displays all potential hardware configurations and maps the Component View components to these configurations. The Deployment Diagrams express this point of view. These diagrams depict the physical placement of the items in their surroundings.

The Transaction Microservice package's deployment diagram is shown in Figure 7.

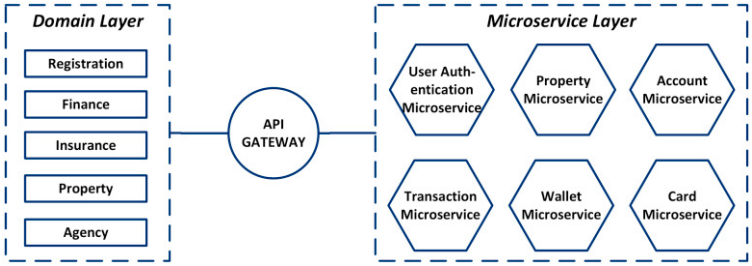

Fig. 7. Transaction Microservice Deployment Diagram [15]

6. Evaluation

A single depiction of an architectural view cannot be seen by all the related professional stakeholders (end-users, developers, system architects, and project managers). The suggested architecture view model provides a complete foundation for understanding and developing microservice-based systems. It allows architects to think about a wide

range of system features, from high-level functions to low-level deployment and runtime behavior. This model can be used by systems architects to efficiently communicate the architecture of the system to varied stakeholders, solve design issues, and make informed decisions throughout the development lifecycle. This model is a framework for organizing information that comprises layouts (logical, process, component, and deployment) as well as end-user perspective information of an application (scenario). The scenario allows the architecture to be linked together. Furthermore, a variety of diagrams can be used to show how an activity flows through a system and which logical, process, component, and deployment viewpoints are collaborating to achieve the desired result.

The model is referred to be a generic process that may be modified for specific projects to ensure validity. The technique must be improved further as a result of experience with several case studies. After each case study, we discuss the benefits and drawbacks of creating the general process description.

7. Conclusion

A model with a specification that concentrates on the development and use of microservices-based systems is one of the contributions of this study. We proposed ArcMS, an architectural view model built on the 4+1 Views model by Philippe Kruchten and utilizing organizational needs for strategic modeling. The development of a useful microservices architecture model view that satisfies stakeholder expectations for IT/Business alignment has been the main objective of this model. The architecture (design) of software-intensive systems can be expressed using the 4+1 viewpoints paradigm, which contains several concurrent perspectives. Viewpoints are used to explain the system from the views of the system's end users, developers, system engineers, and project managers. The ArcMS model gives software development concepts that can be applied in a microservices architecture, in contrast to the 4+1 Views model, which does not describe how to establish development standards for microservices-based applications.

Future research suggests that in order to present a comprehensive template that considers, for example, managing daily activities and responding to changing requirements and input, other practices, such as project management and agile methods [1, 2, 3, 11], must be included in this model. Additionally, a CASE tool has to be created to help with the creation and application of all the models described in this study.

References

[1] Ambler, S., Nalbone, J., Vizdos, M. (2005*) Enterprise Unified Process: Extending the Rational Unified Process*. Prentice Hall.

[2] Ambler, S. (2007) “Agile software development at scale. In: IFIP Central and East European Conference on Software Engineering Techniques”, 1-12, Springer.

[3] Ambler, S., Lines, M. (2012) *Disciplined Agile Delivery: A Practitioner’s Guide to Agile Software Delivery in the Enterprise*, IBM Press.

[4] Dalpiaz, F., Franch, X., Horkoff, J. (2016) *iStar 2.0 language guide*, CoRR, abs/1605.07767.

[5] Dragoni, N. et al. (2017) “Microservices: Yesterday, Today, and Tomorrow. In: Mazzara M., Meyer B. (eds.) Present and Ulterior Software Engineering”, 195-216. Springer.

[6] Dingsøyr, T. et al. (2012) *A decade of agile methodologies: Towards explaining agile software development*, Elsevier.

[7] Hevner, A. et al. (2004) “Design Sciene in Information Systems Research.” *Management Information Systems Quarterly* **28**(**1**): 75-105.

[8] Kilu, E. et al. (2019) “Agile software process improvement by learning from financial and fintech companies: LHV bank case study. In: International Conference on Software Quality”, 57-69, Springer.

[9] Kruchten, P. (1995) “The 4+1 View Model of Architecture” IEEE Software, **12**(**6**): 42-50.

[10] Kruchten, P. (2004) *The Rational Unified Process: An Introduction*. Addison-Wesley.

[11] Kruchten, P. (2013) “Contextualizing agile software development.” *Journal of Software: Evolution and Process* **25**(**4**): 351-361.

[12] Nadareishvili, I. et al. (2016) *Microservice architecture: aligning principles, practices, and culture*. O'Reilly Media, Inc.

[13] Peffers, K. et al. (2008) “A Design Science Research Methodology for Information Systems Research.” *Journal of Management Information Systems* **24**(**3**): 45–77.

[14] Taibi, D. et al. (2017) “Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation.” *IEEE Cloud Computing*, **4**(**5**), 22--32

[15] Vu, H. A. Nguyen (2022) “An Agile Approach for Managing Microservices-Based Software Development: Case Study in FinTech. In: Information Systems (EMCIS 2021)”, *Lecture Notes in Business Information Processing* **437**: 723-736, Springer.

[16] Wautelet, Y., Kolp, M. (2016) “Business and model-driven development of BDI multi-agent systems” *Neurocomputing*, **182**: 304-321.

[17] Yu, E., Giorgini, P., Maiden, N., Mylopoulos, J., (2011) Social Modeling for Requirements Engineering. MIT Press.