Antunes, P., Nguyen, T., & Tate, M. (2026). BizDevOps Blueprint: A Light Model Supporting BizDevOps Teams Planning Digital Services. First Monday, 31(3). https://doi.org/10.5210/fm.v31i3.13818.

BizDevOps Blueprint: A Light Model Supporting BizDevOps Teams Planning Digital Services

Pedro Antunes[1]*, Nguyen Hoang Thuan[2], and Mary Tate[3]
[1]Faculty of Sciences, University of Lisbon
Campo Grande, 1749-016 Lisbon, Portugal
padantunes@fc.ul.pt
ORCID: 0000-0002-5411-8798

[2]The Business School, RMIT University Vietnam
702 Nguyen Van Linh District 7, Ho Chi Minh City, Vietnam
thuan.nguyenhoang@rmit.edu.vn
ORCID: 0000-0002-6241-8205

[3]School of Information Management, Victoria University of Wellington
PO Box 600, Wellington 6140, New Zealand
mary.tate@vuw.ac.nz
ORCID: 0000-0002-4284-7467

Abstract

Many organizations nowadays depend on digital services to deliver business value to customers. BizDevOps is a new approach that helps organizations quickly develop and adapt digital services to changing business needs, evolving user demands, and new technologies. However, BizDevOps brings new challenges and opportunities. In particular, BizDevOps teams face the challenge of self-managing the technical/business development while keeping it in sync with a continuous flow of change requirements, user feedback, diversity of interests (business and technical), and constant collaborative decisions. This paper proposes the BizDevOps blueprint, a light model that offers a simple, flexible, and strategic map for BizDevOps teams to plan digital services. Expert interviews were used to evaluate the blueprint. The results indicate that the blueprint can help build and maintain an overview of digital service offerings.

Keywords: BizDevOps, BizDevOps Planning, BizDevOps Blueprint, Light Model, Microservices.

1 Introduction

Digital services have played essential roles in modern organizations; most services are now digital (Tuunanen, Lumivalo, Vartiainen, Zhang, & Myers, 2023). Modern organizations depend on digital services to deliver customer value, integrate and optimize work processes, and explore new business models (Linde, Frishammar, & Parida, 2021; Wirtz, 2019). This dependence challenges the organizations' capacity to adapt digital services to endlessly changing settings, including fluctuating business requirements, unique user demands, and evolving technologies.

BizDevOps is a new approach that helps organizations quickly develop and adapt digital services by cutting back the lag between vision and implementation (Antunes & Tate, 2024). The BizDevOps approach brings Biz (business development), Dev (software development), and Ops (operational deployment) expertise into unified, self-organized, cross-functional, and value-oriented teams (Lohrasbinasab, Acharya, & Colomo-Palacios, 2020). BizDevOps builds on top of the benefits recognized in the Agile and DevOps software development approaches, including, for instance, quick response to changes, daily collaboration between business experts and software developers, continuous software delivery, and extensive automation and tooling (Hemon, Lyonnet, Rowe, & Fitzgerald, 2020; Humanitec, 2023; Misra, Kumar, Kumar, Fantazy, & Akhter, 2012). BizDevOps builds upon and further intensifies the trend toward orchestrating small independent services (designated microservices) rather than operating large monolithic systems (Bucchiarone et al., 2020). Finally, BizDevOps also enables quicker decision-making feedback (Gruhn & Schäfer, 2015; Hernández, Moros, & Nicolás, 2023; Lohrasbinasab et al., 2020).

However, the adoption of BizDevOps also brings with it some challenges. In particular, BizDevOps teams face the challenge of self-managing the technical/business development of digital services, as it must be kept in sync with a constant flow of change requirements, user feedback, business and technical interests, and continuous and collaborative decision-making.

A recent industry survey indicates that most DevOps teams can deploy software in a few hours (77% of teams do it within four hours) and can do it multiple times per week (28% of teams) (Gearset, 2022). BizDevOps teams are expected to perform similarly, if not better, with the additional challenge of dealing with business requirements. Considering that modern digital services are supported by a complex mesh of software technology (Stack Overflow, 2022), managing business and technical changes while keeping a coherent perspective over what is committed by the teams can be challenging. Sanjurjo et al. (2020, p. 200) note that "there are no standard methods and processes nor standard practices to guide that [BizDevOps] implementation." We reviewed the BizDevOps literature and could not find relevant models supporting BizDevOps teams planning service offerings (Chasioti, 2019; Ebert, Gallardo, Hernantes, & Serrano, 2016). Existing approaches have weak links to the all-hands-on-deck, integrated, and continuous practices adopted by BizDevOps teams (Alt, Auth, & Kögler, 2021). Existing approaches also make it challenging to embrace microservices, a key enabler of continuous software delivery (Dudjak & Martinović, 2020).

This study concerns how BizDevOps teams can plan service offerings. It is important to note that we do not claim that existing planning approaches, namely the ones used by DevOps, cannot support this endeavor. However, they do not offer the light approach required for effective communication and rapid, iterative, and collaborative decision-making.

We propose a *light approach, designated BizDevOps blueprint, that supports BizDevOps teams in planning digital services*. We emphasize the *light* feature, as it aims to support a quick, flexible, and "fast fail" mindset. The term *blueprint* refers to a high-level script intended to provide a strategic understanding of digital services (Camilli, Bellettini, Capra, & Monga, 2017).

This study contributes to the emerging BizDevOps field with insights into the key elements that should be considered by BizDevOps teams when planning digital services in a continuous deployment environment. Adopting the interview method, the study also offers practitioners feedback about the blueprint. The feedback indicates that the blueprint can help build and maintain an overview of digital services. Finally, taking a more applied view, this study contributes a set of guidelines on how BizDevOps teams can plan digital services.

The rest of this paper is organized as follows. Section 2 briefly discusses the movement from DevOps to BizDevOps. Section 3 gives background on BizDevOps for planning digital services. Section 4 defines the challenges and requirements. Section 5 justifies our approach toward a solution. Section 6 presents the BizDevOps blueprint. Section 7 summarizes the practitioners' feedback. Finally, the paper offers a discussion and conclusions.

2 From DevOps to BizDevOps

The BizDevOps approach emerged from the success of DevOps in improving software delivery processes (Hemon et al., 2020; Lohrasbinasab et al., 2020; Zhang, Zhao, & Jaskolka, 2025). DevOps offers a variety of advantages, including extensive automation and tool support; alignment with the current trend towards microservices (to reduce complexity); continuity between software development and operation; short incremental development cycles; quick releases and recalls (in case of failure); and fast feedback cycles supported by analytics (Antunes & Tate, 2024; Dörnenburg, 2018; Ebert et al., 2016; Leite, Rocha, Kon, Milojicic, & Meirelles, 2019). The adoption of DevOps has changed the role of IT in organizations from a supporting, mainly responsive role to a proactive business enabler as IT becomes the business (Dörnenburg, 2018).

Considering this backdrop, the extension to BizDevOps seems highly compelling for organizations, as it brings another level of integration, where Biz concerns (for example) with business requirements, value generation, time-to-market, user experiences, and product and service designs are integrated with Dev and Ops. This allows teams to address problems in a cohesive, collaborative, hands-on, and highly tooled-up approach (Alt et al., 2021).

Research on BizDevOps highlights that it reinforces a culture of collaboration, increases quality and speed, and stimulates innovation and competitiveness (Antunes & Tate, 2024; Lohrasbinasab et al., 2020). Industry surveys also indicate that BizDevOps can successfully tackle organizational silos and persistent lack of alignment between the technical and business domains, improving decision-making and controlling the complexity of IT (Harvard Business Review Analytic Services, 2020; Lohrasbinasab et al., 2020).

3 BizDevOps for planning digital services

3.1 BizDevOps and digital services

Many organizations are adopting a service-oriented logic, exploring strategic capabilities that enable the organization to co-create value with customers (Karpen, Bove, & Lukas, 2012). While DevOps enables the continuous development and innovation of digital services from a technical stance, BizDevOps enables this capability from an integrated business-technical stance: “[t]he emergent tendency is to build BizDevOps teams for the ideation, creation, development, and operation of new digital services” (Petana & Rosa, 2020, p. 180). BizDevOps provides a mechanism for thinking and acting towards the realization of services (Karpen et al., 2012). It also brings a multidisciplinary, agile, small-scale, highly focused mentality into service design (Petana & Rosa, 2020).

3.2 BizDevOps and microservices

The BizDevOps approach is also carried by the recent trend towards microservices. Microservices emerged as key technical developments in service-oriented computing, breaking barriers among proprietary software, reducing complexity, simplifying communication, and facilitating agile and highly flexible software development (Bucchiarone et al., 2020; Larrucea, Santamaria, Colomo-Palacios, & Ebert, 2018).

Microservices architectures decompose monolithic software applications into small, independent applications. A microservice implements a single responsibility (functionality and data) exclusively supplied through an application interface (API) (Waseem, Liang, Shahin, Di Salle, & Márquez, 2021). The API tells everything about the microservice: it strictly operates on a set of requests-responses, and the execution is black-boxed (Cinque, Corte, & Pecchia, 2022). This way, BizDevOps teams can focus on small, cohesive sets of business and technical requirements and independently design microservices.

Microservices can request functionality from other microservices; this interaction is again done through APIs. API gateways provide anchor points and manage service discovery, availability, and access, adding flexibility and autonomy to service interactions (Cinque et al., 2022). This combination of features provides a developer-friendly, highly reusable, and manageable service catalog for BizDevOps teams to work with.

3.3 BizDevOps and planning

The emergence of Agile approaches has significantly changed how software development is planned. Core Agile values include the capacity to respond to changes rather than follow plans, working with short timescales, focusing on simplicity, self-organized teams, and constant collaboration (Misra et al., 2012). Considering the spectrum of software development processes, they can range between highly-planned and unbounded models (Abrahamsson, Salo, Ronkainen, & Warsta, 2002). Highly-planned models define with significant detail what has to be done and how it will be done. Unbounded models let developers define the best approach and change it whenever needed. The Agile worldview leans towards the latter type of model. For instance, Extreme Programming and Scrum adopt a kind of model, designated backlog, which contains a prioritized list of requirements of a system to be developed or updated (Abrahamsson et al., 2002). Agile teams use the backlog to decide on the required activities for each development cycle, turning the backlog items into features to be developed. Changes to the backlog drive the system updates.

BizDevOps also follows this model, where BizDevOps teams use the planning stage to decide what must be developed or updated. Backlogs usually support this; the innovation is that BizDevOps teams bring a stronger focus on business requirements and deeper automation into backlog management. A stronger focus on business requirements is consequential of the broader scope and responsibilities taken by the team, for instance regarding time-to-market, value delivery, and customer feedback (Alt et al., 2021). Deeper automation is a consequential reaction to increasingly complex socio-technical responsibilities, such as monitoring service operations, demand, bottlenecks, and failures and adapting whenever necessary (Hemon et al., 2020; Katal, Bajoria, & Dahiya, 2019).

3.4 Research on BizDevOps and planning

Research on BizDevOps and planning is scarce. Searching in Scopus for papers mentioning BizDevOps and planning in the title, abstract, and keywords returned three papers. A search in the ACM Digital Library considering the full document returned 15 papers. A search in IEEE Xplore, viewing all metadata, returned three papers. After screening the search results, 31 papers were selected for further analysis, and eight were selected to characterize the current state of the research (Chasioti, 2019; Fitzgerald & Stol, 2017; Forbrig, 2016; Fuentes-Quijada, Ruiz-Gonzalez, & Caro, 2023; Hemon et al., 2020; Johanssen, Kleebaum, Paech, & Bruegge, 2018; Ravichandran, Taylor, & Waterhouse, 2016; Wiedemann, Wiesche, Gewald, & Krcmar, 2019).

Studies on BizDevOps indicate that continuous planning is a required capability of BizDevOps teams, as they must continuously check the dependencies between Biz, Dev, and Ops (Chasioti, 2019; Fuentes-Quijada et al., 2023; Johanssen et al., 2018). Planning involves multiple stakeholders but is accomplished as a team autonomously (Chasioti, 2019; Wiedemann et al., 2019).

Plans are open-ended artifacts that continuously evolve in response to changes (Fitzgerald & Stol, 2017). They help identify, discuss, prioritize, and track project items (e.g., requirements, features, and APIs) (Hemon et al., 2020; Wiedemann et al., 2019). As with DevOps (Chasioti, 2019), BizDevOps project items are pushed into a backlog. Plans have a limited scope and are tightly integrated into the project execution.

In mature BizDevOps environments, planning is integrated with the continuous practice of collecting automated feedback from technical and human sources, e.g., regarding service failure and user feedback (Chasioti, 2019; Fitzgerald & Stol, 2017; Forbrig, 2016). This allows teams to optimize continuous delivery (Ravichandran et al., 2016).

4 Challenges, requirements, and adopted strategy

The literature review also allowed us to identify challenges in how BizDevOps teams deal with planning. As noted above, planning has been essentially based on backlogs. They offer a simple mechanism for managing change by updating priorities. However, backlogs cannot roadmap the whole range of Biz, Dev, and Ops decisions to be made by BizDevOps teams. As such, business and technical decisions may end up being done in silos instead of being fully shared by the whole team.

BizDevOps teams work end-to-end on the analysis and design of digital services, including, in particular, transforming features into microservices, specifying APIs, and deciding on what other microservices to use (Delgado, García, & Ruiz, 2023). However, current planning approaches do not address the articulation of these design elements. They focus on limited and traditional aspects, such as requirements definitions (Forbrig, 2018; Forbrig & Dittmar, 2019; Lohrasbinasab et al., 2020).

Planning is shared by various stakeholders, including business and technical experts (Forbrig, 2017, 2018; Gruhn & Schäfer, 2015). Collaborative decisions depend on the team's capacity to develop common language, concepts, and artifacts (Chasioti, 2019). However, current planning approaches promote the lowest common denominator, e.g., in the form of requirements lists, feature lists, and metrics (Lohrasbinasab et al., 2020). Planning should provide a more inclusive baseline for business and technical experts to share knowledge.

Considering these challenges, we identify three requirements for planning digital services in BizDevOps settings (Table 1). The first requirement concerns strategic road mapping for all decisions BizDevOps teams make. The second requirement involves end-to-end analysis and design or redesign of digital services, which must be collaboratively performed by BizDevOps teams. Finally, the last requirement addresses the development of an inclusive common ground (language, concepts, and artifacts) for knowledge sharing.

Table 1. Requirements for planning digital services in BizDevOps settings

Strategic road mapping: Planning must support a dynamic, shared, and consistent snapshot of decisions made by the team.
Design orientation: Planning must support collaborative, end-to-end analysis and design or redesign of digital services.
Inclusive common ground: Planning artifacts must provide an inclusive knowledge-sharing baseline.

Addressing these requirements, we adopt a light modeling approach. The use of light models in software development has been increasing (Siau et al., 2022). For instance, personas, agile user stories, storyboards, scenarios, empathy maps, stakeholder maps, affinity maps, journey maps, and service blueprints are often adopted to analyze and design digital services (Wood et al., 2021). They help developers by focusing on key concepts like opportunities, pain/gain points, and user values.

Prior research on the properties of canvas models provides a framework for understanding light models (Antunes & Tate, 2022b). Firstly, a light model *operates strategically*, where users work with a small set of key concepts. Secondly, a light model emphasizes *visual thinking*, combining everyday visual elements with tacit knowledge to support decision-making. Finally, a light model takes a *general view* of the represented system. These properties align with the requirements for planning digital services in BizDevOps settings, as shown in Table 2.

Table 2. Adopted strategy and its alignment with requirements

Strategic road mapping: By operating strategically and visually, a light model helps to quickly put together the decisions made by the team
Design orientation: By providing a general view of a targeted system, a light model helps to quickly put together, and change whenever necessary, the various components of a digital service
Inclusive common ground: The visual thinking supported by a light model, along with the use of everyday visual elements, helps in sharing diverse knowledge about a digital service

Following this direction, we will elaborate on the BizDevOps blueprint.

5 BizDevOps blueprint

The proposal and explanation of the BizDevOps blueprint are divided into two parts. First, several components are defined to provide a foundation for digital services design. Second, the components are connected to provide a strategic, high-level view of a digital service.

A running illustrative example supports the discussion. This choice is appropriate for evaluating design propositions like the blueprint (Peffers, Rothenberger, Tuunanen, & Vaezi, 2012). It is also a common method in software research and practice (Senapathi, Buchan, & Osman, 2018). The illustrative case considers a stock trading application named eTrader. The application allows users to buy/sell stock items on stock markets and perform related actions, such as following market trends and managing a stock portfolio.

5.1 Blueprint components

Digital services are technically complex. This complexity has many origins, including architectural, code, feature, and method complexity (Auer, Lenarduzzi, Felderer, & Taibi, 2021; Molina-Ríos & Pedreira-Souto, 2020; Wakil & Jawawi, 2018). Such technical complexity cannot be entirely hidden or fully exposed, or the blueprint will fail the different needs of the BizDevOps team. A way to balance complexity is to work with abstract components, which can hide many secondary details.

A departing point towards abstraction is to model a digital service as two components: frontend and backend (Figure 1), where the frontend deals with user interactions and the backend deals with non-user-facing functionality (Dudjak & Martinović, 2020). This separation is common in software research and practice (Bonura, Culmone, & Merelli, 2002; Northwood, 2018; Waseem et al., 2021). It does not imply that all digital services must have both components. For instance, a digital service may only operate on the backend.

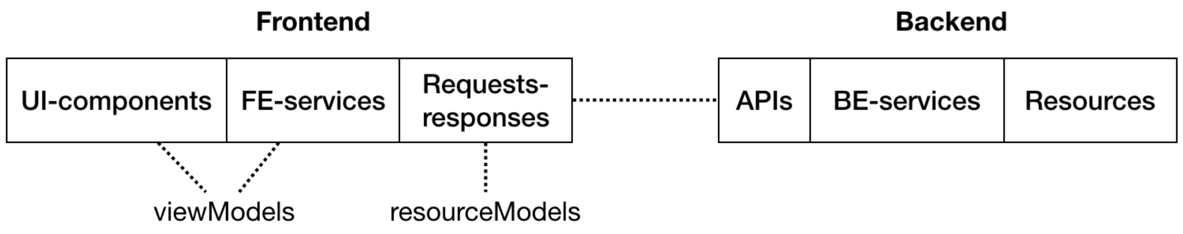


Figure 1: Blueprint components

5.2 Representing the backend

We adopted the backend-as-a-service abstraction to represent the backend (Figure 1, right), where BE-services provide operations on specific resources through APIs (De, 2017; Dudjak & Martinović, 2020). BE-services and resources are only named in the blueprint. For example, eTrader provides a BE-service named *marketWatch* that manages *exchanges* and *stocks* lists. Regarding the APIs, the blueprint considers three parts: operations, endpoints, and data in/out[1] (Dudjak & Martinović, 2020). For instance, in eTrade, the *marketWatch* API has operations for getting a list of exchanges and a list of stocks for a particular exchange. Data in/out is represented in the blueprint using name-attribute pairs[2].

5.3 Representing the frontend

The frontend defines the user-facing functionality of digital services (Bonura et al., 2002). We adopted the following framework (Figure 1, left)[3]: UI-components, FE-services, requests-responses, viewModels, and resourceModels. UI-components are templates for displaying the data supplied by viewModels[4]. ViewModels reduce coupling between UI-components and FE-services (Garofalo, 2011). FE-services handle events, manage the presentation logic, communicate with the backend, and pass data to UI-components through viewModels. Requests-responses characterize the interactions with the backend. ResourceModels decrease the coupling between the frontend and backend by regulating the exchanged data structures. This framework offers a flexible way to characterize modern frontends while black-boxing details related to specific technologies (Bragge, 2013; Garofalo, 2011). It allows developers to translate abstract concepts into concrete technologies without alienating the other stakeholders[5].

---

[1] This representation uses pragmatic REST, which is the de facto standard for microservices's APIs (Costa, Pires, Delicato, & Merson, 2016). It defines a small number of resource-oriented operations (GET, PUT, POST, DELETE), where resources are uniquely identified using a scheme that resembles a hierarchical file structure ("/…/…") (Pautasso, 2014).

[2] This folows JSON, the de facto standard for microservices communication (Baškarada, Nguyen, & Koronios, 2020).

[3] Based on two well-known approaches: MVC (Model-View-Controller) and MVVM (Model-View-ViewModel) (Garofalo, 2011). In some frontend frameworks, FE-services correspond to controllers, event handlers, or dispatchers.

[4] This follows the declarative approach adopted by recent frontend technologies, like React, where views are rendered on the screen based on data parameters rather than instructions (Madsen, Lhotak, & Tip, 2020).

[5] For instance, UI-components can be directly mapped into React and Angular templates (He, 2022).

The representation of UI-components, FE-services, and requests-responses in the blueprint is highly simplified. The purpose is to focus on resourceModels and viewModels, as they provide a strategic view of the frontend that helps BizDevOps teams focus on high-level decisions rather than implementation details. For instance, eTrader's *marketWatch* is centered on displaying a list of stocks (viewModel), which in turn is built from another list of stocks gathered from the backend (resourceModel).

5.4 Representing the frontend behavior

Understanding the frontend behavior is important for BizDevOps teams, as it supports discussions about the service logic. A set of principles defined by Jacobson et al. (2016) is adopted to represent the frontend behavior: keep it simple by telling use cases (simple stories), explain the big picture, and focus on value. Use cases capture essential aspects of how specific goals are achieved using a digital service. Representing multiple use cases is simpler and clearer than creating a unified but convoluted representation of all behavioral possibilities afforded by a digital service. For example, eTrader can be described using two use cases: *trading* (check the market and buy/sell stocks) and *accountManagement* (check accounts and transfer money).

The blueprint is not intended to represent fine-grained details but to build a strategic view of a digital service. Therefore, a high-level navigational map is adopted (Schewe & Thalheim, 2019). A use case identifies (Figure 2): what happens (events), when it happens (states), what can be done (actions), and where to go next (routing).

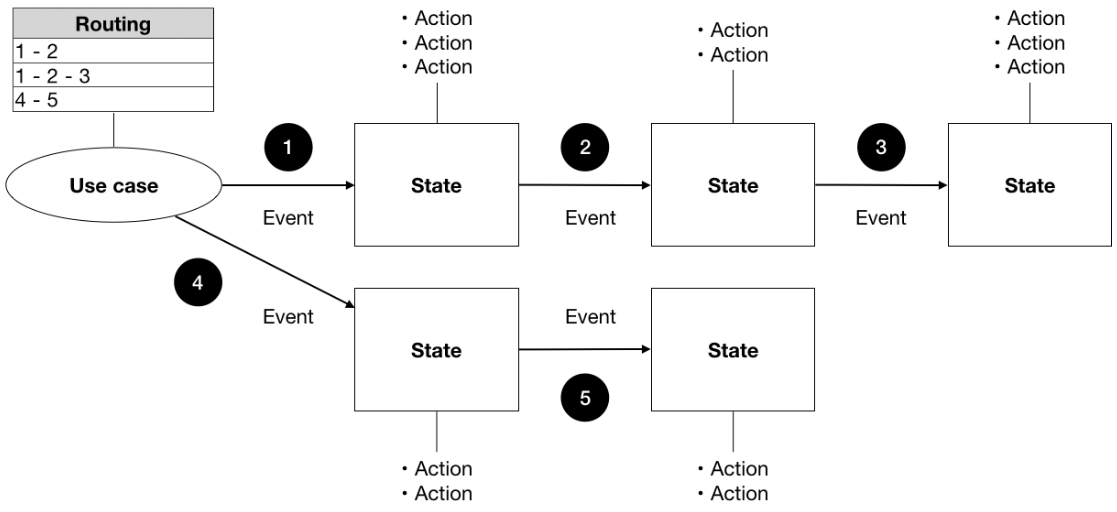


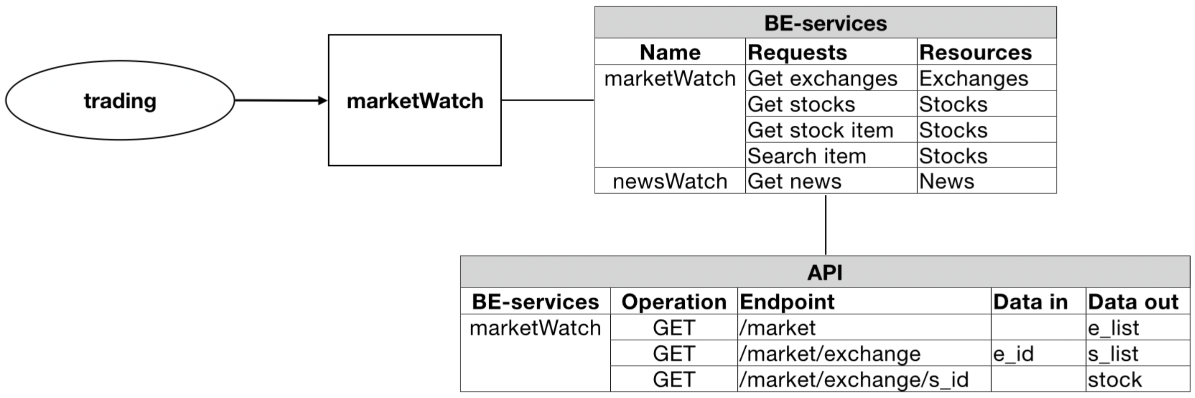
Figure 2: Representation of frontend behavior

A state defines a point in time where a set of actions and events may occur. For example, in eTrader, the *trading* use case involves three states: *marketWatch*, when users check what is going on with the stock markets; *trade*, when they buy/sell stocks; and *portfolio*, when they manage their collections of stocks. Routing defines how the use case goes from one state to another after a state-changing event. Considering eTrader, two routes can be considered: start in *marketWatch* to select a stock and then move to *trade* to put a buy/sell order; or instead, start in the *portfolio*, mark a stock, and then go to *trade* to buy/sell the stock.

Given the light nature of the blueprint, many behavioral details of the use case are not represented. For instance, it is expected that users can go back, jump to the beginning, and arrive somewhere when a use case finishes. These details can be assumed without compromising the usefulness of the blueprint. They are intentionally absent to keep the blueprint as simple as possible.

5.5 Representing the frontend-backend interactions

Having defined some essential aspects of the frontend and backend, it is time to connect them in the blueprint. This is accomplished through the use cases. For each state in a use case, a table is defined with the BE-services, considering names, requests, and managed resources. Figure 3 shows the BE-services table for eTrader's *marketWatch* state defined by the *trading* use case.



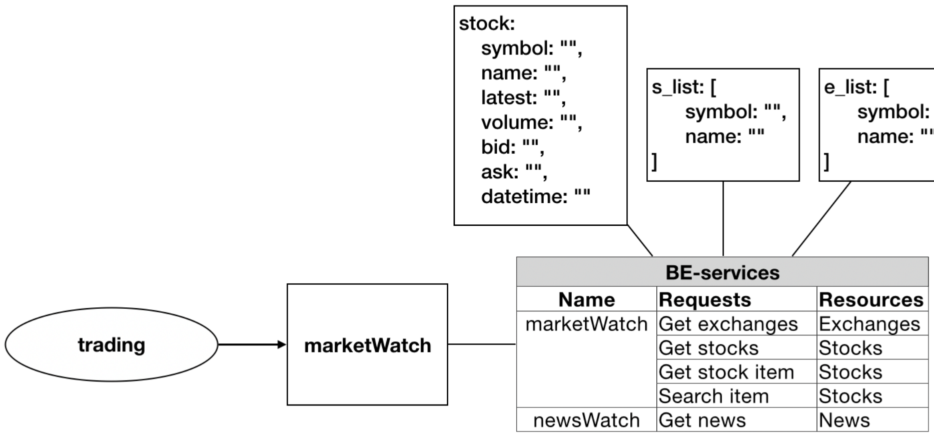
BE-services		
Name	**Requests**	**Resources**
marketWatch	Get exchanges	Exchanges
	Get stocks	Stocks
	Get stock item	Stocks
	Search item	Stocks
newsWatch	Get news	News

API				
BE-services	**Operation**	**Endpoint**	**Data in**	**Data out**
marketWatch	GET	/market		e_list
	GET	/market/exchange	e_id	s_list
	GET	/market/exchange/s_id		stock

Figure 3: BE-services table for eTrader's *marketWatch*

BE-services are accessed through APIs. The APIs are defined using another table specifying the operations, endpoints, and data in/out. API tables can be substituted by schemes generated by API design tools like Swagger. Figure 3 (bottom-right) shows the API table for eTrader's *marketWatch*. It has endpoints for getting a list of market exchanges (*e_list*), getting a list of stock items (*s_list*), and getting details about a specific stock (*stock*).

The communication between the frontend and the backend requires resourceModels. They are specified in the blueprint using attribute-value pairs[6] linked to the BE-services table. Figure 4 shows three resourceModels used by *marketWatch*: *e_list* has a list of exchanges, *s_list* has a list of stocks, and *stock* has details about a stock.



```
stock:
   symbol: "",
   name: "",
   latest: "",
   volume: "",
   bid: "",
   ask: "",
   datetime: ""
```

```
s_list: [
     symbol: "",
     name: ""
]
```

```
e_list: [
     symbol: "",
     name: ""
]
```

BE-services		
Name	**Requests**	**Resources**
marketWatch	Get exchanges	Exchanges
	Get stocks	Stocks
	Get stock item	Stocks
	Search item	Stocks
newsWatch	Get news	News

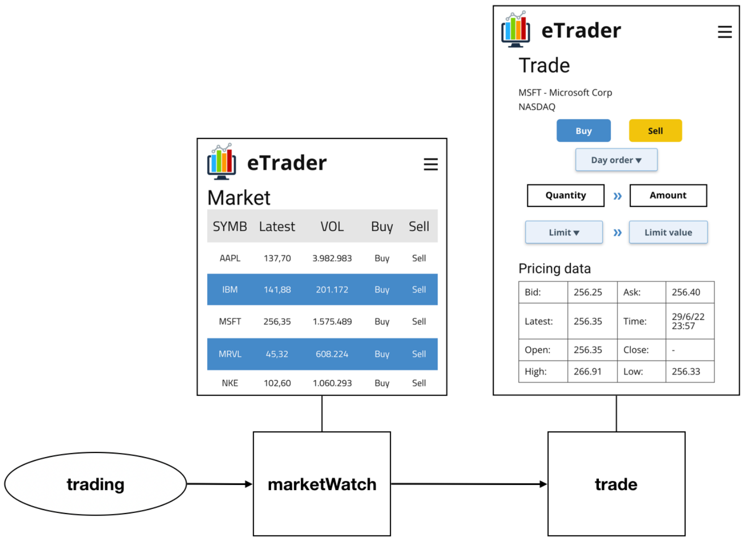
Figure 4: resourceModels used by eTrader's *marketWatch*

5.6 Completing the frontend

Having deviated from the frontend to the backend, we now complete the frontend with some missing features. First, UI-components are connected to states defined by use cases. In fact, UI-

[6] Following JSON, lists are represented using square brackets.

components visually materialize such states, providing presentations for users to interact with. Instead of representing UI-components abstractly, we assume that prototyping tools can be used to depict them realistically. Figure 5 shows eTrader's *marketWatch* and *trade* states and associated UI-components. They were prototyped using Nicepage and copied and pasted into the blueprint.



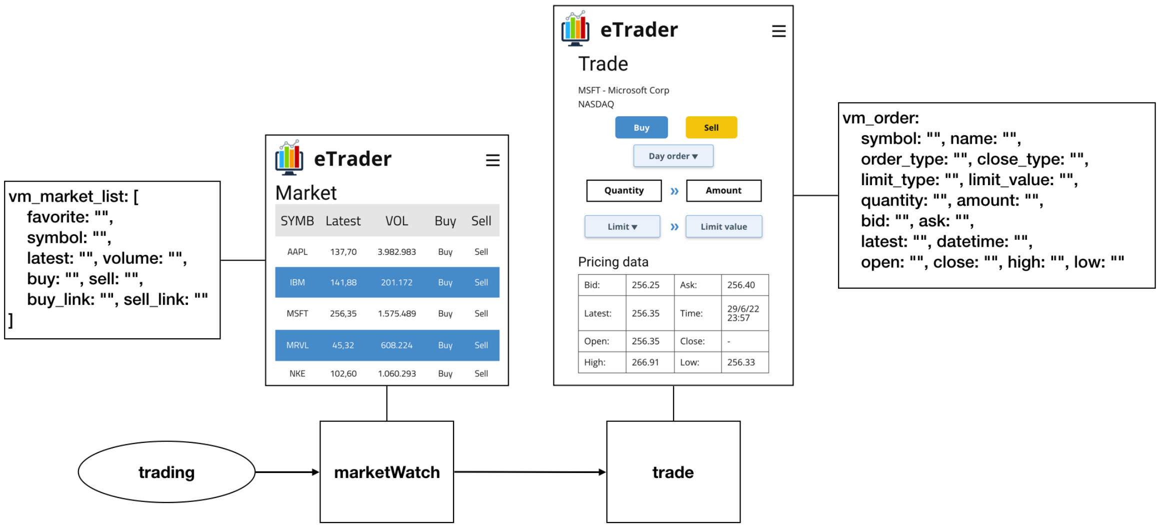
SYMB	Latest	VOL	Buy	Sell
AAPL	137,70	3.982.983	Buy	Sell
IBM	141,88	201.172	Buy	Sell
MSFT	256,35	1.575.489	Buy	Sell
MRVL	45,32	608.224	Buy	Sell
NKE	102,60	1.060.293	Buy	Sell

Bid:	256.25	Ask:	256.40
Latest:	256.35	Time:	29/6/22 23:57
Open:	256.35	Close:	-
High:	266.91	Low:	256.33

Figure 5: UI-components of eTrader's *marketWatch* and *trade* states

Only one UI-component is defined per state. Once again, this reflects the light nature of the blueprint. Modern digital services can use many secondary, usually transient user-interface elements, such as pop-ups, confirmations, and drop-down menus. Trying to integrate all these elements into the blueprint seems impractical. The adopted tradeoff emphasizes relevance over detail, focusing on the primary user-interface elements.

Second, as noted earlier, UI-components present data from viewModels. Therefore, we attach viewModel representations to UI-component using name-value pairs. Figure 6 shows the viewModels used by the *marketWatch* and *trade* UI-components.



```
vm_market_list: [
    favorite: "",
    symbol: "",
    latest: "", volume: "",
    buy: "", sell: "",
    buy_link: "", sell_link: ""
]
```

```
vm_order:
    symbol: "", name: "",
    order_type: "", close_type: "",
    limit_type: "", limit_value: "",
    quantity: "", amount: "",
    bid: "", ask: "",
    latest: "", datetime: "",
    open: "", close: "", high: "", low: ""
```

SYMB	Latest	VOL	Buy	Sell
AAPL	137,70	3.982.983	Buy	Sell
IBM	141,88	201.172	Buy	Sell
MSFT	256,35	1.575.489	Buy	Sell
MRVL	45,32	608.224	Buy	Sell
NKE	102,60	1.060.293	Buy	Sell

Bid:	256.25	Ask:	256.40
Latest:	256.35	Time:	29/6/22 23:57
Open:	256.35	Close:	-
High:	266.91	Low:	256.33

Figure 6: eTrader's UI-components and respective viewModels

Third, we have not yet added the FE-services to the blueprint. These are defined in tables and linked to states. Figure 7 shows the FE-services tables for the *marketWatch* and *trade* states in eTrader.

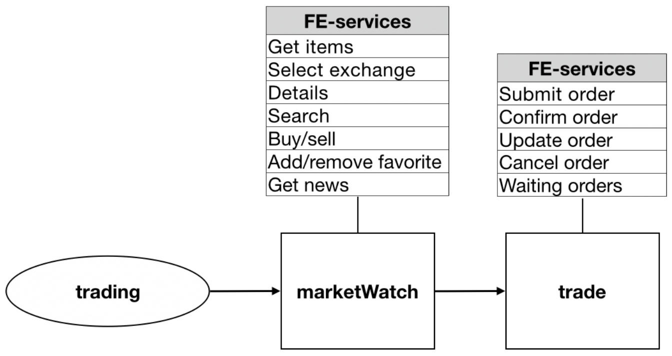


Figure 7: FE-services for eTrader's *marketWatch* and *trade* states

In summary, the blueprint maps the main components of digital services. Use cases are central in the blueprint, providing a strategic view of the digital service's behavior and connecting all other components. This provides a mechanism for designing, decision-making, checking consistency, and tracking changes by BizDevOps teams.

Even though the blueprint has a variety of components, it is scalable because it can be simplified by omitting components. Complex digital services can also be split into multiple use cases. Tradeoffs between business and technical views can also be achieved by emphasizing or omitting specific components, e.g., APIs can be removed to focus more on the business view.

Figure 8 shows the eTrader's blueprint for the *trading* use case. We note that the blueprint does not have to be constructed in a definite way, following specific steps. Its primary purpose is to bring the BizDevOps team on the same page, share a strategic view of the project, provide a general view, and support visual thinking.

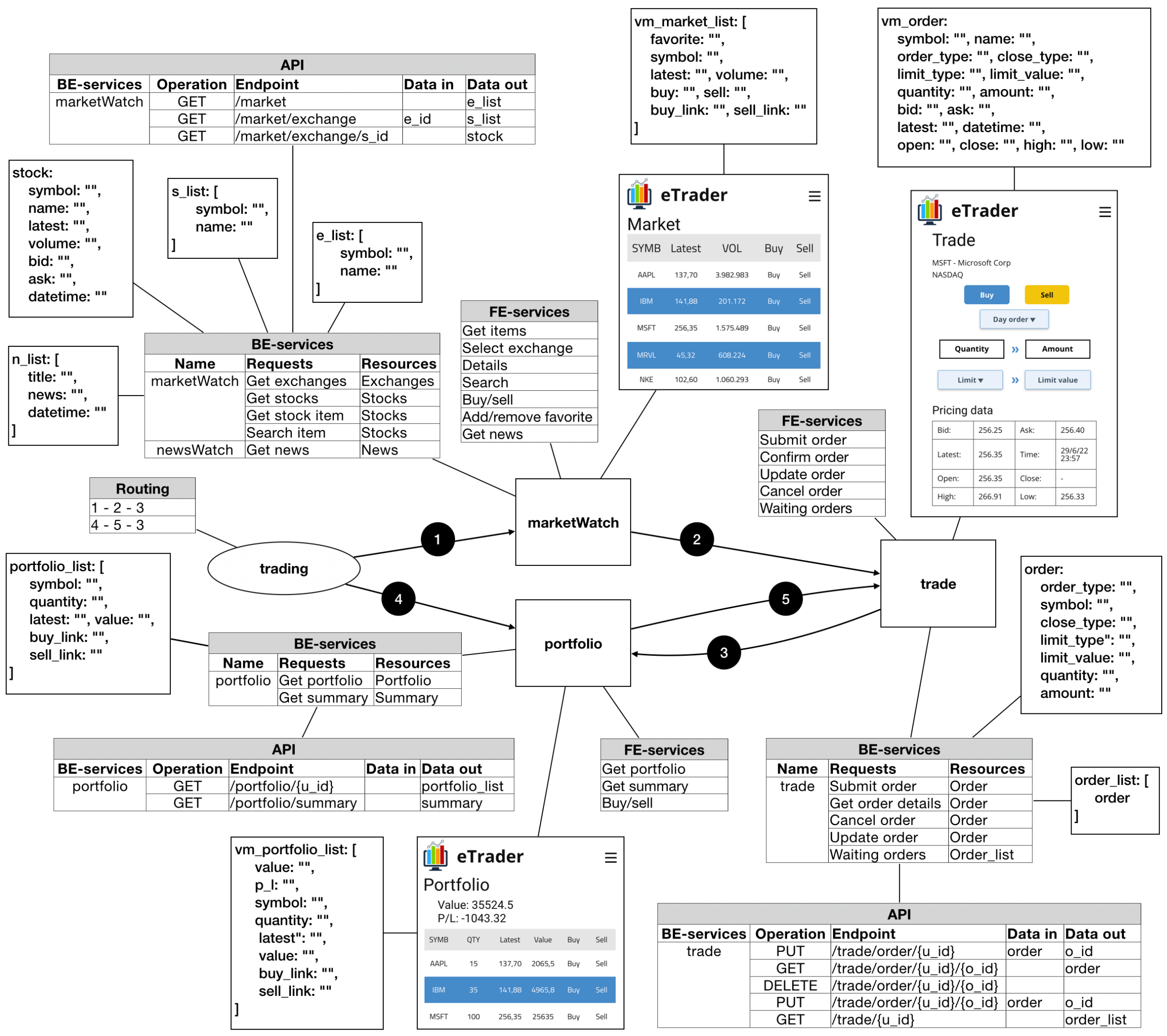

API
BE-services | Operation | Endpoint | Data in | Data out
marketWatch | GET | /market | | e_list
GET | /market/exchange | e_id | s_list
GET | /market/exchange/s_id | | stock
vm_market_list: [favorite: "", symbol: "", latest: "", volume: "", buy: "", sell: "", buy_link: "", sell_link: ""]
vm_order: symbol: "", name: "", order_type: "", close_type: "", limit_type: "", limit_value: "", quantity: "", amount: "", bid: "", ask: "", latest: "", datetime: "", open: "", close: "", high: "", low: ""
stock: symbol: "", name: "", latest: "", volume: "", bid: "", ask: "", datetime: ""
s_list: [symbol: "", name: ""]
e_list: [symbol: "", name: ""]
eTrader
Market
SYMB Latest VOL Buy Sell
AAPL 137,70 3.982.983 Buy Sell
IBM 141,88 201.172 Buy Sell
MSFT 256,35 1.575.489 Buy Sell
MRVL 45,32 608.224 Buy Sell
NKE 102,60 1.060.293 Buy Sell
eTrader
Trade
MSFT - Microsoft Corp
NASDAQ
Buy
Sell
Day order
Quantity
Amount
Limit
Limit value
Pricing data
Bid: 256.25 Ask: 256.40
Latest: 256.35 Time: 29/6/22 23:57
Open: 256.35 Close: -
High: 266.91 Low: 256.33
FE-services
Get items
Select exchange
Details
Search
Buy/sell
Add/remove favorite
Get news
n_list: [title: "", news: "", datetime: ""]
BE-services
Name | Requests | Resources
marketWatch | Get exchanges | Exchanges
Get stocks | Stocks
Get stock item | Stocks
Search item | Stocks
newsWatch | Get news | News
FE-services
Submit order
Confirm order
Update order
Cancel order
Waiting orders
Routing
1 - 2 - 3
4 - 5 - 3
trading
1
2
3
4
5
marketWatch
trade
portfolio
portfolio_list: [symbol: "", quantity: "", latest: "", value: "", buy_link: "", sell_link: ""]
BE-services
Name | Requests | Resources
portfolio | Get portfolio | Portfolio
Get summary | Summary
order: order_type: "", symbol: "", close_type: "", limit_type": "", limit_value: "", quantity: "", amount: ""
API
BE-services | Operation | Endpoint | Data in | Data out
portfolio | GET | /portfolio/{u_id} | | portfolio_list
GET | /portfolio/summary | | summary
FE-services
Get portfolio
Get summary
Buy/sell
BE-services
Name | Requests | Resources
trade | Submit order | Order
Get order details | Order
Cancel order | Order
Update order | Order
Waiting orders | Order_list
order_list: [order]
vm_portfolio_list: [value: "", p_l: "", symbol: "", quantity: "", latest": "", value: "", buy_link: "", sell_link: ""]
eTrader
Portfolio
Value: 35524.5
P/L: -1043.32
SYMB QTY Latest Value Buy Sell
AAPL 15 137,70 2065,5 Buy Sell
IBM 35 141,88 4965,8 Buy Sell
MSFT 100 256,35 25635 Buy Sell
API
BE-services | Operation | Endpoint | Data in | Data out
trade | PUT | /trade/order/{u_id} | order | o_id
GET | /trade/order/{u_id}/{o_id} | | order
DELETE | /trade/order/{u_id}/{o_id}
PUT | /trade/order/{u_id}/{o_id} | order | o_id
GET | /trade/{u_id} | | order_list

Figure 8: Blueprint of eTrader's *trading* use case

5.7 Meta-model

Figure 9 presents the meta-model of the BizDevOps blueprint, where the dominant elements are use case, connection, and state. The other elements are optional to confer flexibility to the blueprint, which is reflected in the connections' cardinality. The optional elements can be used or not by BizDevOps teams, depending on the stage of development and the nature of discussions.

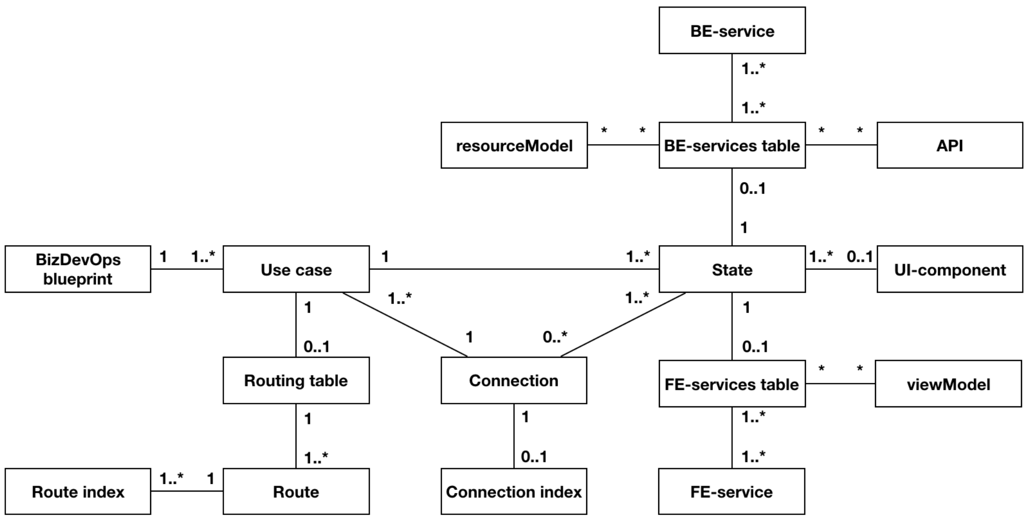


Figure 9: BizDevOps blueprint's meta-model

6 Interviews with experts on the BizDevOps blueprint

We conducted a set of interviews with experts working in BizDevOps environments to gather evaluative feedback about the blueprint. The interview method is appropriate as it enables experts to discuss their thinking and provides in-depth discussions about the blueprint. Further, interviews are suitable for evaluating new models and software practices (Bi et al., 2022; Zampetti et al., 2023). Next, we discuss the interview procedure covering participant recruitment, data collection, analysis, validity checks, and results.

The participants were recruited through professional networks, software development forums, and emails. Key experts in different BizDevOps roles were recruited. Roles included backend, frontend, full-stack development, and project management. Two criteria were used to invite participants: 1) working in a BizDevOps environment and 2) a minimum of one year of experience. All participants regard themselves as BizDevOps experts and have worked on multiple BizDevOps and DevOps projects. Five experts voluntarily participated in the interviews. This sample size is appropriate for three reasons. First, the emerging nature of BizDevOps gives us a limited number of experts in the field. Second, our sample size aligns with Creswell's (2009) recommendation that the qualitative sample size be between 5 and 25 participants. Finally, while this sample size may limit the findings' generalization, it enables us to evaluate the blueprint with detailed expert opinions.

Semi-structured interviews were used for data collection. The process was organized in the following way. Before asking any specific questions, the blueprint was introduced to the participant. Then, a set of questions was raised: Do you think the blueprint would be useful in your projects? How? Would the blueprint accommodate the project's specific architecture and technology? What do you think about the link between the frontend and the backend in the blueprint? Do you think the blueprint could give a complete overview of your project? Is there anything that the blueprint does not address? Would you adopt the blueprint? Why? Given the relatively straightforward nature of these questions, we reached saturation after five interviews, as the same feedback started to be repeated by the participants (Saunders et al., 2018).

All interviews were conducted online through MS Teams, using audio recording. Each took 30-45 minutes. The audio recordings were transcribed and/or translated by a research assistant and were cross-checked by the researchers. Table 3 summarizes the participants' profiles.

Table 3: Participant's profiles

ID	Roles	Experience	Gender
A	Backend developer, tester	>3 years	Male
B	CIO, project manager, full-stack developer	>12 years	Male
C	Operational officer, developer	>2 years	Male
D	Technical lead (both frontend and backend)	>2 years	Male
E	Frontend developer	>4 years	Male

A cross-interview procedure was adopted for the data analysis (Patton, 2014). We examined and labeled the codes extracted from the individual interviews and compared and categorized codes among the different interviews into themes. The procedure enabled us to find themes and patterns across interviews and discover key findings. The analysis focused on three key evaluation aspects: evaluative elements (e.g., usefulness); feedback about specific components (e.g., viewModels); and issues/problems (e.g., complexity). We considered not only supporting evidence but also non-supporting evidence and suggestions to improve the blueprint.

We checked reliability through triangulation. First, we performed investigator triangulation, where one researcher and one research assistant independently coded the interview transcripts. The outcomes were compared and discussed for any differences. Second, as mentioned above, we triangulated the outcomes across the interviews through the cross-interview procedure. Third, the interview procedure and semi-structured questions were conducted in a similar way in all interviews, which also contributed to data triangulation and reliability.

The interview results are presented in Table 4, according to two areas. In the first area, we consider four metrics: usefulness, understandability, strategic view, and future adoption. Usefulness and understandability are widely used to evaluate models such as our blueprint (Dikici, Turetken, & Demirors, 2018; Lübke, Ahrens, & Schneider, 2021). The strategic view metric is aligned with the first requirement (see Table 1). Regarding future adoption, we want to know whether the blueprint will be used by the study participants in the future. In the second area, we consider the participants' feedback according to the blueprint components and also consider any issues and/or problems for future improvements.

Table 4: Interview results

Interview elements	Selected feedback
Usefulness	+ "The good thing about this model is that when there are changes in one of the frontend services, these changes are not moved through all the application. It just has to focus on that particular service, which is really good" (A) + "The model maps with my working experience. It is comprehensive" (B) + "I think the model is suitable regarding the discussion [communication] between different teams" (C) + "It is similar [to what we are doing]" (E)
Strategic view	+ "all of these are independent, […] but there's always a part in which all of them have to be linked" (A) + "The model provides an overview picture, yet may not represent the details" (B) + "The model stays in an abstract level" (C) + "The model is enough" (D)
Understandability	+ "It is ok. I think it is enough" (A) + "The model is simple and understandable" (B) = "If we use the model, does it take more time for small projects?" (B) + "I think it is enough to design an application" (C)

	= “As I am already familiar with MVC, I can understand the model instantly” (E)
Future adoption	+ “Yes, one of the most advantageous things that this model has is that it can be recycled, especially in the backend” (A) + “The model can also be applied for mobile application development” (B) + “I will use this model as reference to discuss with other teams” (C) – “It is hard to say, as the model needs more components such as the communication between frontend and backend” (D) + “I want to try it” (E)
blueprint components	**Selected feedback**
UI-components	+ “[When we start,] we design the layout, user interface, and mock-up prototypes” (B) – “If we apply this model for mobile applications, the views need to be adapted according to mobile types” (B) + “It is certain that we have dynamic elements, yet we also need to identify static elements” (C) = “Frontend [business analytics and designers] will develop the mock-up” (D) + “Sometimes, we use Figma for the interface presentation” (E)
viewModels	– “I think it should show the databases” (C) + “In my project, frontend developers communicate with backend to configure data bindings” (D)
Use cases	– “May be hard to understand” (B) – “There may also be [changes] between different versions of deployments” (C) = “Frontend designers define paths” (D)
FE-services	+ “Developers can write different [frontend] services” (B) = “We mainly do [backend] microservices” (D)
Requests-responses	+ “As soon as we make the frontend that communicates with the APIs, it turns out to be open” (A)
APIs	+ “We define the APIs” (B) + “We link the backend and frontend through contracts and APIs (C) + “Backend links with frontend through APIs” (E)
BE-services	+ We can take it as individual services (A) = “We need to identify in the backend, whether it goes with monolithic or microservices” (C)
Issues/problems	**Selected feedback**
Technical complexity/ simplicity	– “It is too much detail” (A) + “The model is simple and understandable” (B) – “[The blueprint] may be more about development than operations, as there is no operational component here” (C) – “The model is too simple. More details are needed” (D) + “The model is detailed enough” (E)
Frontend framework	– “Change the model into MVC” (B) = “Some applications use MVC or MVVM” (D)
Missing components	– “There should be another service in the middle for authentication” (A) – “There are more than one backend and there are more than one frontend” (A) – “It lacks parts for deployment” (D) – “Frontend calls backend through a gateway, which may also take into account load balancing” (D) – “There should be UI/UX elements” (C) – “Continuous integration and continuous delivery is important in DevOps projects” (D)

Note: '+' indicates supporting evidence; '–' negative evidence; and '=' suggests some concerns.

As a result, the participants provided a set of comments and concrete remarks that indicate the blueprint is useful and provides a strategic view (labeled with '+' in Table 4). They also highlighted the understandability of the blueprint. Further, most participants would also consider adopting the blueprint in the future.

Regarding the blueprint components, the participants emphasized a clear representation of the blueprint components. Indeed, much-supporting evidence was noted for UI-components, viewModels, FE-services, requests-responses, APIs, and BE-services. Yet, use cases seem to be a polarizing aspect. This could be explained by the participants' prevailing technical views.

Regarding problems/issues, the participants identified a few concerns with the blueprint, of which two stand out most. First, the participants have opposing views about the blueprint's technical complexity: several suggested it is too detailed, while others suggested it is simple enough. This issue may come from the different projects discussed in the interviews, which have varied levels of complexity. Second, some participants suggested changing the frontend framework to more familiar abstractions, such as MVC and MVVM. While noting the suggestion, we kept the framework, as it is flexible and service-oriented.

Overall, the experts assert that the blueprint can support BizDevOps in planning digital services. Further, the blueprint and its components are useful, understandable, and ready for adoption in future projects.

7 Discussion

In this paper, we identified three requirements for supporting BizDevOps teams in planning the continuous development of digital services. We now discuss how the blueprint addresses these requirements.

Strategic road mapping. The blueprint supports strategic road mapping by connecting the components required to implement digital services. A small set of concepts establishes these connections: use cases, states, and FE- and BE-services. This conceptual structure caters to the diverse needs of BizDevOps teams. From the business development side, strategic road mapping defines use cases that deliver value to users. From the software development side, strategic road mapping involves connecting components using viewModels and resourceModels. From the operational deployment side, strategic road mapping involves tracking all functional dependencies during design and execution times. For instance, a failure in a BE-service may be connected to an error in a UI-component during the execution of a particular use case.

Aiming specifically at planning capabilities, the blueprint offers a more sophisticated structure than the commonly used backlogs. The blueprint can be seen as a collection of backlogs tailored to specific concerns, e.g., FE- and BE-services, data requirements, presentation, etc. However, the blueprint provides a means for tracking dependencies between elements in these different backlogs and harmonizes their dependencies.

The blueprint's meta-model provides a mechanism for BizDevOps teams to negotiate goals and processes and tailor the blueprint to specific practices. Another advantage is that it avoids conflicting with various tactical tools already used by BizDevOps teams to improve performance in areas such as infrastructure automation, testing, monitoring, and recovery (Senapathi et al., 2018).

The evaluation indicates that the blueprint aligns with the current practices of BizDevOps teams. It is perceived as understandable and useful (see Table 4). Even though the evaluation participants identified some missing elements, the blueprint's meta-model can integrate those elements into the existing structure, for instance, elaborating the communication between the frontend and backend.

Design orientation. The blueprint stimulates visual thinking using a set of components and a unifying logic based on simple concepts. This allows BizDevOps teams to design service offerings using a light modeling approach. As mentioned by one participant in the evaluation, one interesting aspect of the blueprint is that some parts (e.g., the backend) can be reused or recycled across projects. This aligns with the agile, continuous practices of BizDevOps.

The participants in the evaluation indicated they have different design practices. For instance, one participant noted they usually receive prototypes from others, while another noted they develop the prototypes themselves. The blueprint affords this variety of practices. For instance, mockups produced outside can be easily integrated into the blueprint.

Interestingly, the participants in the evaluation were divided between considering the blueprint as either too complex or too simple. This could be an indicator that a reasonable compromise has been achieved.

Inclusive common ground. The blueprint fosters inclusiveness by using a simple, common representation that cuts across the business and technical views. technical-related concepts, such as APIs and requests-responses, are connected to concepts understandable by non-technical experts like viewModels, resourceModels, UI-components, and use cases. Furthermore, the light modeling approach, which adopts a simple, intuitive, and pragmatic notation, helps combine the blueprint with the users' tacit knowledge to support decision-making.

In summary, when contrasted with the current state of the art, the blueprint stands out as one of very few approaches explicitly addressing the needs of BizDevOps teams. Furthermore, it does it uniquely, leveraging the light properties of models to support communicating about critical aspects of digital services and collaboration and consensus building based on a simple, strategic, and flexible project map.

8 Conclusion

This study addresses the emergence of BizDevOps and fulfills the research gap of how to support BizDevOps teams in planning the continuous development of digital services. It proposes and evaluates a blueprint that offers a simple, flexible, and strategic project map. The blueprint defines a set of abstract components and combines them using a set of abstract concepts that characterize essential aspects of digital services at a strategic level. Second, it defines a set of flexible relationships that fuse the business and technical views.

We consider the blueprint not a replacement but a complement to the current BizDevOps practices (Delgado et al., 2023). In particular, the blueprint does not impose precedence rules, development stages, or specific development processes. BizDevOps teams may adopt their own Biz, Dev, and Ops processes. They may also adopt specific technologies (Lohrasbinasab et al., 2020), as the blueprint is abstract enough to accommodate a variety of technical choices and technologies. Nevertheless, there is a fine line between the business and technical views. If a model gets too technical, the business stakeholders can be alienated. If it gets too non-technical, there is also the risk of disaffecting the technical stakeholders. The blueprint offers a baseline to define compromises around this fine line. Teams can add, modify, or remove details without compromising the baseline. Furthermore, the notation adopted for the blueprint avoids intricate formalisms and is highly pragmatic.

The expert interviews indicate that the blueprint provides a helpful project overview (Table 4). At the same time, the results also highlight a diversity of viewpoints and suggestions. Nevertheless, instead of seeing this as an obstacle, we see it as an opportunity for improvement. That is, rather than promoting standardization, we emphasize openness and flexible uses of the blueprint.

The blueprint adopts a light modeling approach, considering three properties: operating strategically, supporting visual thinking, and providing an inclusive common ground. These properties are essential for BizDevOps practice and align with critical requirements for planning digital services in BizDevOps settings. First, modeling should focus on the strategic level, considering abstract rather than concrete features around which BizDevOps teams can collaborate and build consensus. Second, modeling should allow digital services to be visually thought out using a set of components that, on the one hand, expose their complex nature and, on the other hand, can be adopted by stakeholders with diverse needs. Finally, modeling must support a close integration between Biz, Dev, and Ops.

Regarding practical implications, the blueprint has two features contributing to improving BizDevOps practices in planning digital services. First, the blueprint covers the whole digital service design cycle. It shows technical details in a way that is consistent with the strategic road map, where a digital service is envisaged as a collection of use cases enacted by a set of technical elements. Second, addressing the complexity of digital services, the blueprint stimulates visual thinking using a small set of key components. The blueprint clearly connects these components, allowing BizDevOps teams to prioritize and decide on specific designs and design changes while checking interdependencies across the whole component set.

Nevertheless, some limitations of the current research should be recognized. The blueprint is anchored in use cases as the primary mediator between the business and technical views. However, the ever-increasing complexity of digital services can make it challenging to work with cases. Future research is necessary to understand if anchoring the blueprint in even more flexible concepts, such as events (Antunes & Tate, 2022a), could be advantageous. Future research is also necessary to study concrete blueprints developed by BizDevOps teams.

In summary, the current research contributes a strategic map that offers a simple, flexible baseline for planning digital services in BizDevOps settings. Considering modeling epistemology, the blueprint is unique because it embraces a light whole-of-a-system approach, which trades the faithfulness of the system representation in favor of the purpose of representation.

Statements and Declarations

Conflicts of interest: The authors declare they have no competing interests.

Ethical approval: Ethical approval was obtained for the interviews.

Funding: The author(s) received no financial support for this research.

References

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). *Agile software development methods: Review and analysis* (No. VTT publications 478). Espoo, Finland.

Alt, R., Auth, G., & Kögler, C. (2021). DevOps for Continuous Innovation. In *Continuous Innovation with DevOps* (pp. 17–36). Springer.

Antunes, P., & Tate, M. (2022a). Business Process Conceptualizations and the Flexibility-Support Tradeoff. *Business Process Management Journal*, *28*(3), 856–875. https://doi.org/10.1108/BPMJ-10-2021-0677

Antunes, P., & Tate, M. (2022b). Examining the Canvas as a Domain-Independent Artifact. *Information Systems and E-Business Management*, *20*, 495–514. https://doi.org/10.1007/s10257-022-00556-5

Antunes, P., & Tate, M. (2024). "What's Going On" with BizDevOps: A Qualitative Review of BizDevOps Practice. *Computers in Industry*, *157–158*(104081), 1–14. https://doi.org/10.1016/j.compind.2024.104081

Auer, F., Lenarduzzi, V., Felderer, M., & Taibi, D. (2021). From monolithic systems to Microservices: An assessment framework. *Information and Software Technology*, *137*, 106600.

Baškarada, S., Nguyen, V., & Koronios, A. (2020). Architecting Microservices: Practical Opportunities and Challenges. *Journal of Computer Information Systems*, *60*(5), 428–436. https://doi.org/10.1080/08874417.2018.1520056

Bi, T., Xia, X., Lo, D., Grundy, J., Zimmermann, T., & Ford, D. (2022). Accessibility in Software Practice: A Practitioner's Perspective. *ACM Transactions on Software Engineering and Methodology*, *31*(4), 66:1-66:26. https://doi.org/10.1145/3503508

Bonura, D., Culmone, R., & Merelli, E. (2002). Patterns for web applications. *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering*, 739–746. ACM.

Bragge, M. (2013). *Model-View-Controller architectural pattern and its evolution in graphical user interface frameworks*. LUT University, Finland.

Bucchiarone, A., Dragoni, N., Dustdar, S., Lago, P., Mazzara, M., Rivera, V., & Sadovykh, A. (2020). *Microservices Science and Engineering*. Springer Nature. https://doi.org/10.1007/978- 3- 030- 31646- 4

Camilli, M., Bellettini, C., Capra, L., & Monga, M. (2017). A formal framework for specifying and verifying microservices based process flows. *International Conference on Software Engineering and Formal Methods*, 187–202. Springer.

Chasioti, K. (2019). *BizDevOps: A process model for the Alignment of DevOps with Business Goals* (Master Thesis). Utrecht University.

Cinque, M., Corte, R., & Pecchia, A. (2022). Microservices Monitoring with Event Logs and Black Box Execution Tracing. *IEEE Transactions on Services Computing*, *15*(1), 294–307. https://doi.org/10.1109/TSC.2019.2940009

Costa, B., Pires, P. F., Delicato, F., & Merson, P. (2016). Evaluating REST architectures—Approach, tooling and guidelines. *Journal of Systems and Software*, *112*, 156–180. https://doi.org/10.1016/j.jss.2015.09.039

Creswell, J. (2009). *Research design: Qualitative, quantitative, and mixed methods approaches* (3rd ed.). SAGE Publications.

De, B. (2017). API governance. In *API Management* (pp. 179–188). Springer.

Delgado, A., García, F., & Ruiz, F. (2023). BizDevOps Support for Business Process Microservices-Based Applications. In J. Troya, R. Mirandola, E. Navarro, A. Delgado, S. Segura, G. Ortiz, … A. Ruiz-Cortés (Eds.), *Service-Oriented Computing – ICSOC 2022 Workshops* (pp. 274–286). Cham: Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-26507-5_22

Dikici, A., Turetken, O., & Demirors, O. (2018). Factors influencing the understandability of process models: A systematic literature review. *Information and Software Technology*, *93*, 112–129.

Dörnenburg, E. (2018). The path to devops. *IEEE Software*, *35*(5), 71–75.

Dudjak, M., & Martinović, G. (2020). An API-first methodology for designing a microservice-based Backend as a Service platform. *Information Technology and Control*, *49*(2), 206–223.

Ebert, C., Gallardo, G., Hernantes, J., & Serrano, N. (2016). DevOps. *IEEE Software*, *33*(3), 94–100.

Fitzgerald, B., & Stol, K. (2017). Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software*, *123*, 176–189.

Forbrig, P. (2016). Continuous Software Engineering with Special Emphasis on Continuous Business-Process Modeling and Human-Centered Design. *Proceedings of the 8th International Conference on Subject-Oriented Business Process Management*, 1–4. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/2882879.2882895

Forbrig, P. (2017). Does Continuous Requirements Engineering Need Continuous Software Engineering? *REFSQ Workshops*.

Forbrig, P. (2018). Use Cases, User Stories and BizDevOps. *REFSQ Workshops*.

Forbrig, P., & Dittmar, A. (2019). Integrating HCD into BizDevOps by Using the Subject-Oriented Approach. In C. Bogdan, K. Kuusinen, M. Lárusdóttir, P. Palanque, & M. Winckler (Eds.), *Human-Centered Software Engineering* (pp. 327–334). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-05909-5_21

Fuentes-Quijada, G., Ruiz-Gonzalez, F., & Caro, A. (2023). Towards Agile IT/Business Alignment at BizDevOps. *Proceedings of the 25th International Conference on Enterprise Information Systems*, 608–614.

Garofalo, R. (2011). *Building enterprise applications with Windows Presentation Foundation and the model view ViewModel Pattern*. Microsoft Press.

Gearset. (2022). *The State of Salesforce DevOps 2022*.

Gruhn, V., & Schäfer, C. (2015). BizDevOps: Because DevOps is not the end of the story. *International Conference on Intelligent Software Methodologies, Tools, and Techniques*, 388–398. Springer.

Harvard Business Review Analytic Services. (2020). BizOps: Connecting IT to Business Outcomes. *Harvard Business Review*. Retrieved from https://hbr.org/sponsored/2020/06/bizops-connecting-it-to-business-outcomes

He, M. (2022). *Creating Apps with React Native*. Apress Media, LLC.

Hemon, A., Lyonnet, B., Rowe, F., & Fitzgerald, B. (2020). From agile to DevOps: Smart skills and collaborations. *Information Systems Frontiers*, *22*, 927–945.

Hernández, R., Moros, B., & Nicolás, J. (2023). Requirements management in DevOps environments: A multivocal mapping study. *Requirements Engineering*, *28*(3), 317–346. https://doi.org/10.1007/s00766-023-00396-w

Humanitec. (2023). *DevOps Benchmarking Study 2023*. Germany: Humanitec GmbH.

Jacobson, I., Spence, I., & Kerr, B. (2016). Use-case 2.0. *Communications of the ACM*, *59*(5), 61–69.

Johanssen, J., Kleebaum, A., Paech, B., & Bruegge, B. (2018). Practitioners' eye on continuous software engineering: An interview study. *Proceedings of the 2018 International Conference on Software and System Process*, 41–50.

Karpen, I., Bove, L., & Lukas, B. (2012). Linking service-dominant logic and strategic business practice: A conceptual model of a service-dominant orientation. *Journal of Service Research*, *15*(1), 21–38.

Katal, A., Bajoria, V., & Dahiya, S. (2019). DevOps: Bridging the gap between Development and Operations. *2019 3rd International Conference on Computing Methodologies and Communication*, 1–7. https://doi.org/10.1109/ICCMC.2019.8819631

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

Leite, L., Rocha, C., Kon, F., Milojicic, D., & Meirelles, P. (2019). A survey of DevOps concepts and challenges. *ACM Computing Surveys*, *52*(6), 1–35.

Linde, L., Frishammar, J., & Parida, V. (2021). Revenue models for digital servitization: A value capture framework for designing, developing, and scaling digital services. *IEEE Transactions on Engineering Management*, *70*(1), 82–97.

Lohrasbinasab, I., Acharya, P., & Colomo-Palacios, R. (2020). BizDevOps: A Multivocal Literature Review. *International Conference on Computational Science and Its Applications*, 698–713. Springer.

Lübke, D., Ahrens, M., & Schneider, K. (2021). Influence of diagram layout and scrolling on understandability of BPMN processes: An eye tracking experiment with BPMN diagrams. *Information Technology and Management*, *22*(2), 99–131.

Madsen, M., Lhotak, O., & Tip, F. (2020). A Semantics for the Essence of React. *European Conference on Object-Oriented Programming*.

Misra, S., Kumar, V., Kumar, U., Fantazy, K., & Akhter, M. (2012). Agile software development practices: Evolution, principles, and criticisms. *International Journal of Quality & Reliability Management*, *29*(9), 972–980. https://doi.org/10.1108/02656711211272863

Molina-Ríos, J., & Pedreira-Souto, N. (2020). Comparison of development methodologies in web applications. *Information and Software Technology*, *119*, 106238.

Northwood, C. (2018). *The Full Stack Developer: Your Essential Guide to the Everyday Skills Expected of a Modern Full Stack Web Developer*. Springer.

Patton, M. (2014). *Qualitative research & evaluation methods: Integrating theory and practice*. Sage publications.

Pautasso, C. (2014). RESTful web services: Principles, patterns, emerging technologies. In *Web Services Foundations* (pp. 31–51). Springer.

Peffers, K., Rothenberger, M., Tuunanen, T., & Vaezi, R. (2012). Design science research evaluation. In *LNCS*: *Vol. 7286*. *Design Science Research in Information Systems. Advances in Theory and Practice* (pp. 398–410). Berlin Heidelberg: Springer.

Petana, G., & Rosa, C. (2020). Digital Transformation and the Impact in Knowledge Management. *Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, 180–187.

Ravichandran, A., Taylor, K., & Waterhouse, P. (2016). *Devops for digital leaders: Reignite business with a modern devops-enabled software factory*. Springer Nature.

Sanjurjo, E., Pedreira, O., García, F., & Piattini, M. (2020). Measuring the Maturity of BizDevOps. In M. Shepperd, F. Abreu, A. Silva, & R. Pérez-Castillo (Eds.), *Quality of Information and Communications Technology* (pp. 199–210). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-58793-2_16

Saunders, B., Sim, J., Kingstone, T., Baker, S., Waterfield, J., Bartlam, B., … Jinks, C. (2018). Saturation in qualitative research: Exploring its conceptualization and operationalization. *Quality & Quantity*, *52*, 1893–1907.

Schewe, K., & Thalheim, B. (2019). *Design and development of web information systems*. Springer.

Senapathi, M., Buchan, J., & Osman, H. (2018). DevOps capabilities, practices, and challenges: Insights from a case study. *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018*, 57–67.

Siau, K., Woo, C., Story, V., Chiang, R., Chua, C., & Beard, J. (2022). Information Systems Analysis and Design: Past Revolutions, Present Challenges, and Future Research Directions. *Communications of the Association for Information Systems*, *50*(1), 33.

Stack Overflow. (2022). *Stack Overflow Annual Developer Survey*.

Tuunanen, T., Lumivalo, J., Vartiainen, T., Zhang, Y., & Myers, M. (2023). Micro-Level Mechanisms to Support Value Co-Creation for Design of Digital Services. *Journal of Service Research*.

Wakil, K., & Jawawi, D. (2018). A new adaptive model for web engineering methods to develop modern web applications. *Proceedings of the 2018 International Conference on Software Engineering and Information Management*, 32–39. Casablanca, Morocco: ACM.

Waseem, M., Liang, P., Shahin, M., Di Salle, A., & Márquez, G. (2021). Design, monitoring, and testing of microservices systems: The practitioners' perspective. *Journal of Systems and Software*, *182*, 111061. https://doi.org/10.1016/j.jss.2021.111061

Wiedemann, A., Wiesche, M., Gewald, H., & Krcmar, H. (2019). Implementing the Planning Process within DevOps Teams to Achieve Continuous Innovation. *Hawaii International Conference on System Sciences*. Retrieved from https://aisel.aisnet.org/hicss-52/st/agile_development/9

Wirtz, B. W. (2019). *Digital business models: Concepts, models, and the alphabet case study*. Springer.

Wood, K., Lauff, C., Hui, W., Teo, K., Png, S., Swee, A., … Vargas, B. (2021). *Design Innovation Methodology Handbook–Embedding Design in Organizations*. Design Innovation Programme/Team, Singapore University of Technology and Design-Massachusetts Institute of Technology International Design Centre.

Zampetti, F., Tamburri, D., Panichella, S., Panichella, A., Canfora, G., & Di Penta, M. (2023). Continuous Integration and Delivery Practices for Cyber-Physical Systems: An Interview-Based Study. *ACM Transactions on Software Engineering and Methodology*, *32*(3), 73:1-73:44. https://doi.org/10.1145/3571854

Zhang, X., Zhao, P., & Jaskolka, J. (2025). Navigating the DevOps landscape. *Journal of Systems and Software*, *223*, 112331.