Citizens as Developers and Consumers of Smart City Services: A Drone Tour Guide Case

Christian Muck
Faculty of Computer Science, University of Vienna
Vienna, Austria
christian.muck@univie.ac.at

Alexander Voelz
Faculty of Computer Science, University of Vienna
Vienna, Austria
alexander.voelz@univie.ac.at

Danial M. Amlashi
Faculty of Computer Science, University of Vienna
Vienna, Austria
danial.mohammadi.amlashi@univie.ac.at

Dimitris Karagiannis
Faculty of Computer Science, University of Vienna
Vienna, Austria
dk@dke.univie.ac.at

ABSTRACT

The trend of urbanization has started over two centuries ago and is no longer limited to high-income countries. As a result, city population growth has led to the emergence of applications that manage complex processes within cities by utilizing recent technological advances, thereby transforming them into *smart cities*. Besides automating complex processes within a city, technology also enables a simplified integration of citizens into identifying problems and creating corresponding solutions. This paper discusses an approach that enables citizens to design and later execute their own services within a smart city environment by employing conceptual modeling and microservices. The overall aim is to establish the role of a *citizen developer*. The proposed approach is then discussed within our proof of concept environment based on a drone tour guide case.

CCS CONCEPTS

• **Human-centered computing** → **Interaction design process and methods**; Interaction paradigms; • **Networks** → *Cyber-physical networks*.

KEYWORDS

Smart city, citizen developer, conceptual modeling, microservices, drone scenario

ACM Reference Format:
Christian Muck, Alexander Voelz, Danial M. Amlashi, and Dimitris Karagiannis. 2022. Citizens as Developers and Consumers of Smart City Services: A Drone Tour Guide Case. In *Companion Proceedings of the Web Conference 2022 (WWW '22 Companion), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3487553.3524848

1 INTRODUCTION

According to the *United Nations*, almost 55 % of the World's population already lives in cities, and predictions state that the urbanization rate will rise to more than 65 % within the next 30 years [29]. Consequently, a growing amount of research is investigating the challenges and opportunities resulting from the steady population growth within cities worldwide. Such challenges emerge due to the increased complexity of handling processes within growing cities [1, 19, 26, 28]. Transportation, housing, and sustainable resource management are only some of the many challenges megacities face.

One approach to tackle these increasingly complex processes is utilizing recent advances regarding digital technologies. The objective of these approaches is to exploit the potential benefits that result from integrating technologies into a city's infrastructure. Such benefits can be realized by either optimizing existing processes and services or generating completely new ones. The domain of complementary approaches is often grouped under the term *smart city*. However, no universal definition has been established yet, and many different terminologies are used in the literature to distinguish between different smart city domains [5, 7].

Although the incorporation of technologies usually forms the basis for developing a solution, they alone cannot solve the complex problems of modern cities. In addition, a comprehensive understanding of the core problem is needed to develop a solution. As part of understanding the problem, it is crucial to identify the involved stakeholders whose perspectives need to be considered. Following the low-code principle (cf. [32]), we propose an approach for creating new services within a city without comprehensive software engineering knowledge. The overall aim is to enable citizens to develop problem-based solutions on their own. To support this aim, models are used within our approach to capture the citizen knowledge and execute it in the smart city environment.

For this purpose, we first describe the relevant elements of the smart city setting before introducing a smart city-based *drone tour guide case* scenario in section 3, to highlight the challenges we aim to address. The scenario was created based on experience with smart city teaching cases (cf. [3, 4]), also applied during the NEMO Summer School*. Based on this scenario, we present our approach and associated requirements in section 4. Afterwards, our proposed solution is discussed within a proof-of-concept environment (see section 5). Finally, we conclude our study with a SWOT analysis.

2 RELATED WORK

The related work of the present study encompasses a general elaboration of smart cities before the theoretical foundation of microservices, and conceptual modeling is established.

*https://nemo.omilab.org/, last visited: 16-03-2021

WWW '22 Companion, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9130-6/22/04.
https://doi.org/10.1145/3487553.3524848

2.1 Smart City Setting

For the general elaboration of the smart city setting, we first substantiate our understanding of the term *smart city* since no universally accepted definition exists in the scientific literature. Afterwards, we use established application dimensions to further classify our case within the smart city setting. In general, the term *smart city* refers to a set of approaches that use technologies to optimize existing or shape new processes in cities [1]. Such services are often provided by the city administration and consumed by other entities within the city. The overall goal is that either administrative processes, the life of citizens, or both, improve through the incorporation of *Information and Communication Technologies* (ICTs) [39].

Nevertheless, a widely accepted definition of the term *smart city* has yet to be established, which is why various terms exist to describe different concepts of smart cities [5, 7]. Namely, *Wired, Intelligent, Digital*, or *Knowledge City* are all terms used in the scientific literature to distinguish between different areas and concepts of smart cities. Camero and Alba [5] conducted an extensive review on the smart city literature and concluded that no universally used definition of the term *smart city* exists among their analyzed publications. To contribute to a more common understanding of the smart city domain, we adapted the following definition by the European Parliament [24] instead of establishing our own:

> "A smart city is a city seeking to address public issues via ICT-based solutions on the basis of a multi-stakeholder, municipally based partnership." [24, p.9]

We deemed this definition most suitable for this study's drone tour guide case while still leaving room for context-specific adaptions. Nevertheless, context-specific adaptions must be clarified during the design phase of applications within this broad definition. Moreover, we argue that a holistic view of the smart city setting is needed, in which citizens, technologies, and the environment are not treated as independent but interacting entities [27].

To further classify our case study, we use six application dimensions proposed within the report on European smart cities that the European Parliament commissioned (e.g., [24]). Since then, these dimensions have been adopted in the smart city literature [1, 5, 26]. A common approach to distinguish between the different application dimensions is to assess who provides the respective application, which infrastructure is needed, and whether it aims at generating benefits for administrative purposes, for the citizens, or both [39]. Considering the six application dimensions, we classify the drone tour guide case as an interface between *Smart Living*, *Smart Economy*, and partially *Smart Mobility*.

Smart Living describes applications in a smart city that enable new lifestyles and consumption through technology [5, 24]. In our case, the consumption of touristic attractions is enabled by combining drones with microservice-enriched city infrastructures. Additionally, we consider our tour guide to offer an innovative service in the sense of a *Smart Economy* application, which, among others, encompasses all kinds of technology-enabled innovations and new services [5, 24]. Finally, we classify the drone tour guide at least partially as a *Smart Mobility* application, which generally describes the enhancement of transport or logistics systems through technologies [5, 24]. Even though the drone tour guide doesn't enhance transportation systems, it still depends on an interconnected system to safely navigate the city.

To sum up, we define a smart city in the context of our drone tour guide case as a city, in which ICTs are incorporated into a city's infrastructure and combined with other digital technologies to generate benefits for the citizens.

2.2 Microservices in Smart Cities

The following section covers the fundamentals of microservice architecture. The aim is not to provide an analysis of microservices in smart cities but rather to illustrate how the related concepts are used in the context of this paper.

As discussed in section 2.1, we understand a smart city as a city that tries to overcome its challenges by applying ICT-based solutions. ICT in smart cities often includes applications associated with the Internet of Things (IoT) [39]. These applications lead to increased complexity for software systems within smart cities. One way to tackle this complexity is the application of web services in the form of a microservice architecture [21].

The functionality of a software system in a microservice architecture is separated into decoupled services, each focusing on an independent functionality [6, 23]. Each microservice can be written in its own programming language and possesses its own technology stack. This architecture reduces the dependency within the software system and supports individual deployment, maintenance, and replacement. Each service runs independently, and communication between services is done through light-weighted communication protocols (e.g., REST-like interfaces). As discussed later (see *citizen developer* in section 3.1), communes, businesses, or even citizens can offer such services. The workflows of the software system are then executed through the connection of different microservices. Two ways of accomplishing this execution are orchestration, and choreography [6]. Orchestration means that a centralized service exists, which handles the execution of the workflow. In choreography, on the other hand, such a centralized service does not exist, and the workflow is embedded into the microservices.

We assume for our use case that different services in a smart city are provided by multiple stakeholders, like other citizens, businesses, or the municipality. The stakeholders should then be able to define their own services by orchestrating existing ones, as it is often done in microservice architectures. The business logic of the created services is then described through conceptual models, which are executed in a microservice environment (cf. [37, 38]). In our paper, we analyze how such an approach can look like in the smart city domain and what the resulting requirements consist of. We focus our analysis of such services in a smart city on two viewpoints: consumers and developers of a service.

2.3 Conceptual Modeling

We use conceptual models in this work to capture the knowledge exchanged between humans and the smart city environment. This approach provides an easy way for humans to describe their desired objectives in the smart city environment. In return, the environment needs to enable the execution of such descriptions, which implies that both sides can process the knowledge captured in the models.

Models are defined as conceptual models if the concepts from the domain under study and their relations are abstracted and represented in the model itself [25]. Thereby, conceptual models reduce the complexity of a given system under study through abstraction [13, 25]. This is achieved by omitting unnecessary characteristics and only including the essential ones in the model. Which characteristics are deemed important depends heavily on the creation purpose of the respective model [10].

In computer science, conceptual models are used to represent existing or planed information systems [25]. Therefore, knowledge is described using dedicated modeling languages, which define the notation, syntax, and semantics of the models [12, 16, 35]. The benefit of using such a language is that everyone familiar with its' definition can understand the models created with it. Karagiannis et al. [16] further argue that the value of models is increased if they can be automatically processed by tools to put the captured knowledge to work. This vital part distinguishes the modeling language (notation, syntax, and semantic) from a modeling method (including automated processing and guidelines to create models). In our paper, we use this feature of models for the interaction between humans and the smart city. Humans describe their knowledge using dedicated modeling languages, which can then be processed and executed in the smart city environment.

One way of processing models was introduced in [37, 38] where *Cyber-Physical Systems* (CPS) are controlled using conceptual models. The idea of this approach is to capture both the domain knowledge and the tasks that should be executed in the models. The CPS offer their capabilities without having a deeper understanding of the domain and its processes. In [38] these models are called *Smart Models* and used to decouple the CPS from the domain knowledge. In the context of this paper, we aim to build upon this idea while focusing on how the models can be used as means of communication between humans and a smart city environment. Nevertheless, it needs to be emphasized that we do not aim to design a new modeling method for these interactions. Instead, the objective is to analyze which requirements the models and the smart city environment must offer to allow such interactions. Therefore, we introduce the *drone tour guide case* in the next section as a smart city environment, in which models are used as means for communication.

3 THE DRONE TOUR GUIDE CASE

This section covers the relevant requirements and challenges of the drone tour guide case within the previously discussed smart city setting before the smart city environment, and the drone tour guide scenario itself is described.

3.1 Requirements and Challenges in the Smart City

First, we focus on requirements and associated challenges during the design phase of the drone tour guide. Before doing so, the different entities present in this scenario have to be determined.

As mentioned in the previous section, a smart city application usually consists of an interaction between the three entities (i) human, (ii) technology, and (iii) city infrastructure. Before assessing the individual requirements and key challenges for each of these entities, we have to differentiate between two possible expressions of the entity human. At this point, the ongoing trend towards *citizen development* needs to be introduced for better understanding.

A *citizen developer* is considered a developer with little to no experience regarding software engineering or coding. Yet, they use so-called *low-code* or *no-code platforms* designed to enable fast, easy, and cost-efficient application development [32]. The idea behind citizen development originates from the problem of shadow IT systems within organizations where employees use applications other than those provided by the organization itself to compensate for specific shortcomings [11]. Consequently, employees have been encouraged to develop such workarounds through in-house low- or no-code platforms in order to distributed resulting applications within the whole organization. Since then, the concept of citizen developers has also been applied outside of organizational borders, for example, with mobile app development [30] or process mining applications [36]. A similar approach is present in the smart city literature, namely in the form of participation platforms [34]. Based on this elaboration, we distinguish between humans that use the drone tour guide service and those that develop the tours. In the following, we will refer to the first as *citizen tourist* while referring to the latter as *citizen developer*.

A general requirement during the process of designing smart city applications is to answer the question of how and to which extent the human will be incorporated into the respective application architecture [27]. In our case, both the citizen tourist and the citizen developer are part of the application architecture (see Figure 3). The main requirement of citizen tourists is awareness about the available smart city services and applications. Accordingly, the lack of awareness is a key challenge for smart city development [19]. The subsequent challenges resulting from a lack of awareness affect both citizen tourists and citizen developers. The citizen tourists need to be aware of the offered drone tour guide service in order to use it, while the citizen developers need to have awareness in the first place to create tours. Therefore, the awareness and participation of citizen developers is a key requirement in the context of this study. Participation platforms have been proposed to raise citizen engagement to meet this general challenge in smart cities [34].

The technology requirements of our drone tour guide are primarily related to the capabilities a drone needs to possess for performing its tasks. Voice recognition combined with natural language processing, image recognition, temperature sensors, and access to open data are only some of the requirements that need to be fulfilled to ensure the functionality of the drone tour guide. Another technological and, at the same time, a general challenge of smart cities are the continuous and sometimes drastic advances of the technologies used in this context, which in return cause difficulties regarding the interoperability and long-term planning perspective within these cities [20]. Consequently, an ICT-enhanced and connected infrastructure is a key requirement to be fulfilled. Otherwise, deficits in the infrastructure can become a significant barrier during the development of smart city services [26]. Finally, the lack of an operational framework has to be mentioned as a general challenge of smart city development [19] that we aim to address by providing a framework for describing our smart city service on a high abstraction level. After assessing the requirements and challenges of the drone tour guide in the smart city setting, the specifications of the smart city environment have to be determined.

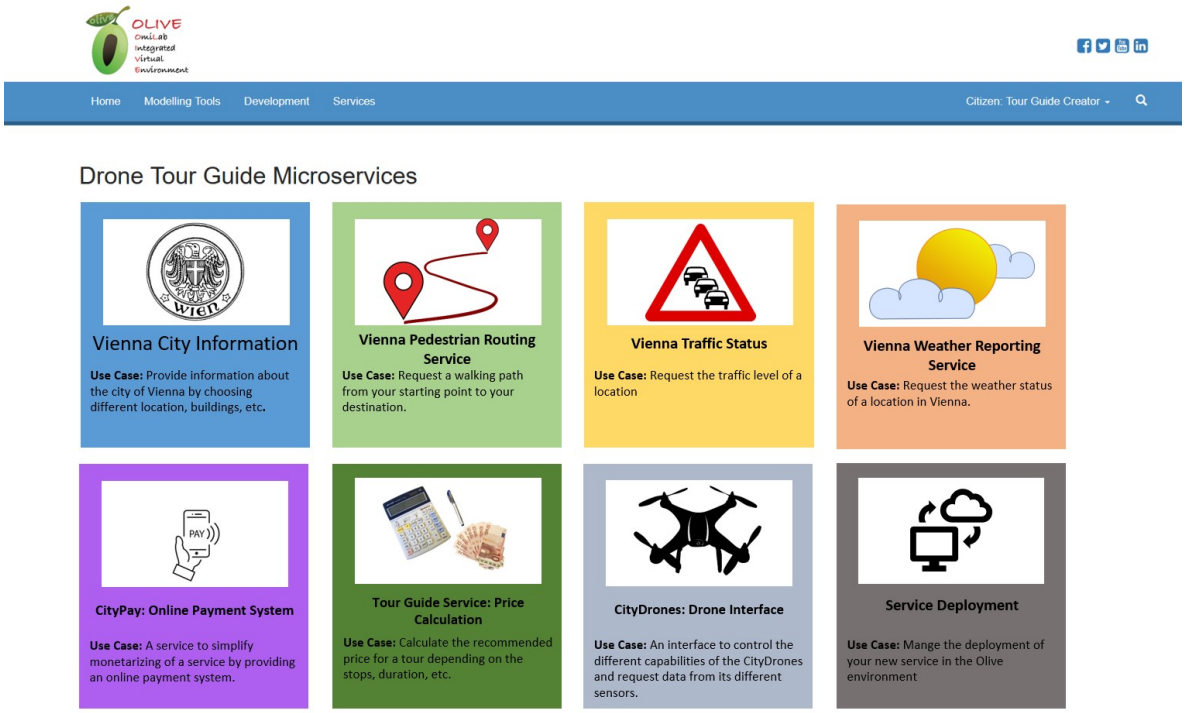

Figure 1: Developer Interface of Microservice Configuration Environment

3.2 The Smart City Environment

For a comprehensive understanding of the drone tour guide case and the possibilities of the microservice architecture, the environment of our scenario needs to be determined.

The drone tour guide scenario takes place in a smart city environment equipped with CPS, which offers a microservice-oriented architecture for the city infrastructure. Due to this open infrastructure, the CPS can communicate with each other and gather necessary information while being structurally independent. This lack of a dependency on basic microservices allows developers to create complex services without interfering with existing ones. In our scenario, the smart city uses dynamic and static CPS to gather and distribute information in the ecosystem. Static CPS, such as smart trash cans and surveillance cameras, are strategically placed in defined locations. In contrast, dynamic CPS, such as flying drones, gather and use the information on traffic and weather conditions while moving through the city.

3.3 The Drone Tour Guide Scenarios

The next step in this section on the drone tour guide case is to present two scenarios. We describe the respective procedure relevant to realizing a drone service in the smart city while also highlighting the previously discussed requirements. The first scenario illustrates the *tour creation* process by a citizen developer, while the second one depicts the *tour consumption* by a citizen or tourist.

Tour creation. For exemplification, let us assume a citizen of Vienna decides to use its expertise in Viennese architecture to create a tour specifically designed to include the city's most famous architectural buildings. To do so, the citizen first creates a profile on the citizen developer platform for drone tours. At this point, the previously described requirements of awareness and participation come into play since the service cannot be offered without the contributions of citizens. The platform provides an interface for these citizen developers, containing a selection of most commonly used and best-fitting microservices, as displayed in Figure 1. Further elaborations on this exemplary microservice interface of our drone tour guide scenario are provided in section 5. After registration, the citizen developer can create a drone tour according to his ideas using the conceptual modeling languages offered by the citizen developer platform. This is achieved by combining individual microservices among the available ones, which range from *CityDrones: Drone Interface* and *Vienna Pedestrian Routing Service* up to *Vienna Weather Reporting Service* in our scenario.

Tour consumption. Following the tour creation process, let us now assume that a tourist is visiting the city of Vienna and decides to get to know the most famous attractions of the city. Again, the previously described awareness requirement is highlighted since the tourist must be aware of the drone tour guide service to use it. To select a tour, the tourist first needs to create a user account and browse the library of tours made available by citizen developers through the previously described process of tour creation. After browsing through the library of available tours, the tourist selects a tour fitting their preferences, created by the architectural domain expert. Upon selection, the tourist is asked to check-in at the nearest drone station while also receiving a tour summary, information about the individual stops, and general information about

the weather. The current weather information for the different locations in the city is gathered by other drones currently moving around the city. As soon as the platform confirms the arrival at the pickup point, the tourist is assigned a personal tour guide drone, which introduces itself, explains the tour process, and kindly asks the tourist to follow along for the duration of the tour. In the last step, the drone flies to the nearest drone station after completing the tour to charge and be available for the next customer.

4 MODEL-BASED DOMAIN-SPECIFIC SERVICES THROUGH MICROSERVICE ORCHESTRATION

In the context of the introduced *drone tour guide case*, we now discuss our proposed solution on how conceptual modeling combined with a microservice architecture can facilitate the creation of new services by citizens of a smart city. In the following, *city services* describes services that citizens created to offer specific functionalities to consumers. Such city services are created with the help of conceptual models. Thereby multiple microservices are used to offer this functionality. A microservice in this context is a reusable service provided within the smart city to create new services.

4.1 Relevant Smart City Entities

To begin with, we introduce the relevant entities of our use case. Figure 2 displays the different interactions between these entities in an interdependence model. *Human* represents human actors who can engage with a smart city through two distinct interactions. The first of these interactions is labeled *Citizen as consumer* and means that citizens consume city services or microservices offered by the city. Depending on the specific service, the consumers interact with the interface (e.g., web interface) to accomplish a particular task. The second interaction is labeled *Citizen as developer* and represents the possibility of citizens to create their own city services within a smart city through the use of conceptual models.

The developer relation is the core focus of this paper, in which we analyze how a citizen can define their city services. The aim is that humans without well-founded programming skills are provided with an opportunity to create new city services through conceptual modeling. To accomplish this aim, we first discuss the requirements a smart city must fulfill to allow for service creations as described in our case. Therefore, we follow a low-code approach (see section 3.1) to reduce the needed programming skills for developers.

The entity *Smart City* represents the services offered and the physical environments they interact with. The physical part is represented through *Environment* and contains all aspects which are not directly connected to technology (e.g., trees, streets, buildings, etc.). The applications and hardware directly related to computing power are represented through *Technology*. These components offer the services' APIs and implement the application logic. Examples for hardware in *Technology* would be CPS like the drone or sensors.

Possible interactions between *Technology* and *Environment* are represented by *Sensing* and *Actuation Interaction*, which both represent relations between the *Environment* and fitting hardware in *Technology*. Sensing means that some aspects of the smart city are captured and transformed into digital signals, like a temperature sensor or a video stream. On the other hand, actuation describes the

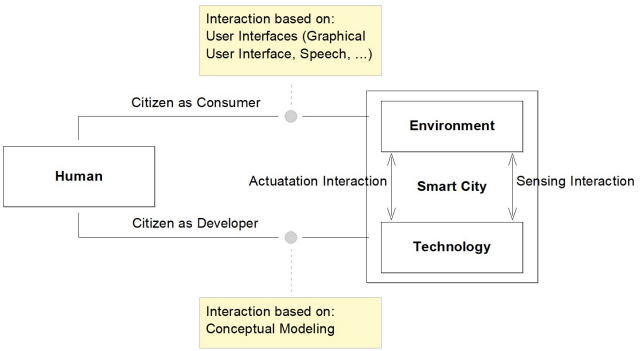

Figure 2: Smart City entities: An Interdependence Model

change of an element in the smart city, like traffic lights changing their color. Although these relations mark an important aspect of smart cities, they are not the focus of this work, which is why these capabilities are considered to be provided.

In total, two roles can be identified regarding the relation between humans and the smart city: consumers and developers. In this context, consumers use predefined services to satisfy a need for which they have to interact with the smart city. More precisely, a tourist must interact with the drone tour guide service to reserve a drone or adapt the route according to their needs. Corresponding interactions need to be considered while creating new city services.

The developer is defined through the *Citizen as Developer* relation and has the goal to create a new city service to be used by the inhabitants of a smart city. As already mentioned, developers in the context of this paper do not need extensive knowledge in software engineering or coding. By following a low-code approach, developers are enabled to create new city services through conceptual modeling. Since models operate on a higher abstraction level than code, they should be easier to comprehend. However, developers still have to be familiar with the city infrastructure and the conceptual modeling methods to some extent.

4.2 Model-based City Services

After discussing the important entities and their roles within our smart city, we now elaborate on how conceptual modeling can support developers in creating city services.

City services generally need basic functionalities to operate, for example, processing payments, finding routes, ascertaining weather forecasts, and similar tasks. Consequently, such functionalities should be encapsulated and offered to developers by other parties, like the municipality, private businesses, or other citizens. Reusing functionalities eases the work of developers and gives providers of functionalities the opportunity to specialize in their tasks. This can be accomplished by using a microservice architecture, where each functionality is encapsulated in one microservice [6, 23].

The microservices can either be created through a low-code approach or software engineering. As defined in this paper, city services can also be created through the reuse of already existing microservices, which in our case is done by a developer through conceptual models. A microservice can provide and require information, and both ways must be supported by the modeling tool.

Between the interaction with two different microservices, it can be necessary to prepare the information from one microservice to be used as input for the next one. To acknowledge the consumer's interaction with the city service, it is necessary that a microservice provides the corresponding capabilities. This can be accomplished either by dedicated communication services or if the interaction is part of the microservice, offering the desired functionality in the first place. In our drone tour guide example, this can mean that communication with the tourist is either handled over a dedicated communication service or part of the routing service itself.

The models can be used with microservices for a more specific or general purpose. An example of one specific microservice would be a routing service for the old town, while a payment microservice would be more general. Therefore, different models with different degrees of domain specificity are needed [9, 15]. This also means that the extent to which concepts in a modeling language are oriented on the domain of the microservice (e.g., routing) varies just as the number of different microservices that can be used.

4.3 Smart City Infrastructure Requirements

Based on the drone tour guide case (see section 3) and the previous subsections, we now extract requirements towards the proposed infrastructure, with which we mean the modeling tool for creating and executing the conceptual models and the environment for managing the microservices.

First, a modeling tool for creating and processing models must be available. Such a tool should provide not only generic but also domain-specific model processing capabilities that are tailored to the needs of a specific domain [16]. In our case, domain-specific processing is represented by the execution of a model in the smart city environment. Therefore, model processing is an essential requirement for the proposed infrastructure. The modeling tool must process, send, and receive information to enable the interaction with the microservices and thereby create city services. Nevertheless, the modeling tool does not have to be a single application, as different modeling methods with different modeling tools can be used to describe various aspects of a city service. Such tools must be able to link the modeled information to create a coherent view throughout the models, assuming that the information is separated in multiple modeling methods or even modeling tools. Further, the modeling methods should evolve alongside the smart city to always offer the right modeling method for the right problem [14].

Parts of already modeled knowledge should be reusable within the model. This can support the developers even further, as it enables structuring of the models and reusing parts from other developers. For example, the model on how to interact with a routing service can be shared with and used by other developers to easily retrieve a route for their own city service. The respective microservices must not only be able to execute their own application logic but also interact with the consumer to tailor specific executions. An example from the drone tour guide would be that the drone does not know where to go next and therefore asks the consumer to decide. These interaction capabilities must be included in the model, as they hold the application logic of a city service.

Lastly, the consumer must be able to trigger the defined city service to decide when the respective service starts. Therefore, the modeled knowledge must be provided to the environment to be callable and executable. The microservices also need a platform where they are managed and configured for integration into city services. Besides storing microservices, it is vital that their status can be controlled to interfere if problems occur and a service is not responsive. Further, the microservices should be scaled if there is exceptionally high traffic for a specific service.

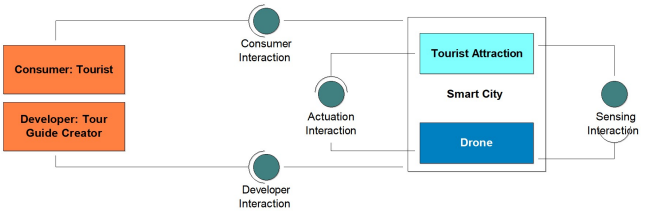

Figure 3: The Interdependence Model: A Drone Tour Guide Instance

5 THE DRONE TOUR GUIDE: PROOF OF CONCEPT

In this section, we apply the insights from the previous section in the context of the introduced drone tour guide case. The objective is to outline the used proof of concept system to evaluate our solution for handling comparable cases in smart cities by employing conceptual modeling and microservices.

For this purpose, we adapted the interdependence model from Figure 2 to fit the drone tour guide case. The result is displayed in Figure 3, where the two roles, the consumer as tourist and the developer as tour guide creator, are represented, each with their corresponding interaction types to the smart city. In this section, we concentrate on the developer interaction since the citizen developer's ability to describe new tours through conceptual modeling is essential to reduce the required programming skills. We assume that the interaction between humans and the drone and between the drone and the smart city is already established through microservices. Hence we focus on how the existing microservices can be utilized collectively to create new city services for consumers. In our case, the newly created city service is represented by the drone tour guide that guides tourists through the city while providing insightful information about selected points of interest (POIs).

The two relations between citizens and the smart city are visualized in the *Usage Layer* of Figure 4. If citizens interact with the *Smart City Environment* and consume services, they take on the role of consumers. If citizens interact with the execution environment to create new services, they take on the role of a developer, which can configure existing services through the *Configuration Environment*. By doing so, new city services are created through the collaborative interaction of citizen developers. From the *Design Layer* the modeling tools can be used to create the models, which are later used in the *Execution Layer* to execute the services.

Within the introduced drone tour guide scenario, the developer of a new tour has to consider all necessary capabilities needed for the tour before finding fitting microservices. Only after can identified microservices be configured and orchestrated. The orchestration of existing microservices reduces the skill requirements

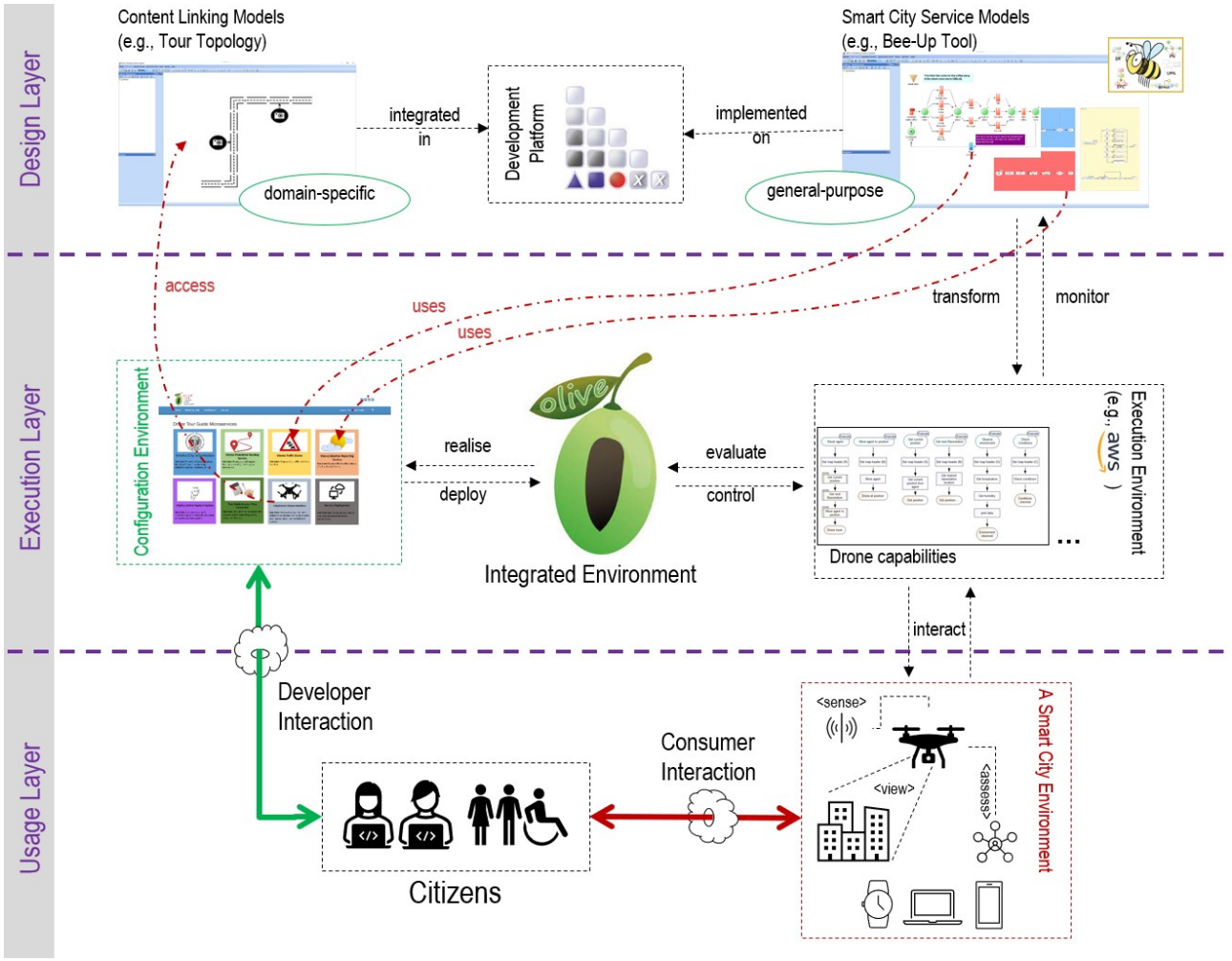

Figure 4: The Drone Tour Guide: A three-layered Component Architecture

for defining new city services since they don't need to be created from scratch. For this to be possible, different conceptual models need to be created that capture the essence of the new city service.

In the proof of concept system, the *Olive* microservice framework (see [2, 8]) was used to provide the microservices of the smart city. Olive stands for *OmiLab Integrated Virtual Environment* and is a microservice framework for creating model-aware web applications. It offers a common interface structure for its services, allowing the integration of external web services through a wrapper and the implementation of dedicated Olive microservices, which can then be accessed over a REST API. All microservices of Olive have a similar API structure to standardize the interaction. The wrapper enables services not implemented in Olive to be integrated and offered over the harmonized API structure. A standardized API structure eases the offering of interfaces to external web services within a modeling environment.

In Figure 1 we showed a web interface of configurable microservices for creating drone tours in an exemplary Olive instance. The overview can be automatically generated if the needed information is provided during microservice implementation. Each tile represents a microservice with a corresponding description provided by the creator. All the information for integrating the microservices in a new city service can be gathered from this interface. Additionally, it can be used to configure and interact with the microservices to better incorporate them in new city services.

For our proof of concept, we not only discuss the benefits of microservice architecture in the given case but also how conceptual modeling can be used in a low-code fashion to reduce the entry requirements for potential developers of city services and through that, increase the available services within a smart city in the long run. Therefore, the models, or more specifically the modeling tools, must be able to (i) represent the knowledge in a processable way and (ii) interact with the microservices, which are the REST APIs of Olive in our case. As we do not want to limit our proof of concept to only one modeling method, we use the ADOxx metamodeling platform [16] to create domain-specific modeling methods or employ ready-to-use modeling tools created with ADOxx. Many domain-specific modeling methods have already been implemented with

the ADOxx platform [17, 18] and some modeling methods were used to execute use-cases with CPS [38], or in the context of the Olive microservice environment [8].

A domain-specific modeling method prototype was created and tailored for linking parts of a route to their POIs and the POI's information. A diagrammatic model can be created within the modeling tool and passed on to the tour guide service to extract the needed information for integrated POIs. An example can be seen in the top left corner of Figure 4.

Other modeling methods can be used to orchestrate microservices (e.g., payment, routing, weather, etc.). We use general-purpose languages to define this part in the proof of concept. Notably, we use their implementation in the Bee-Up tool. In Figure 4 the general-purpose modeling is shown in the upper-right corner under *Smart City Service Models*, which visualizes that these models use the available microservices specific to our drone tour guide scenario by interacting with their API in the *Execution Layer*. For the drone tour guide scenario, additional information like weather or traffic is included in the tour execution through the orchestration of different services. Further, it is shown that services can access information from models (e.g., content linking models) to use this knowledge to execute a specific service.

In the *Execution Layer*, the implementation aspects in our proof of concept environment are visualized. An execution environment like *Amazon Web Serveries* (AWS) must be available to execute and manage the different services and their instances. The services are then integrated into a microservice environment, like *Olive* in our tour guide scenario. The microservice environment can control the running services in the execution environment and get evaluation information back, which can be forwarded to the user interface or handled automatically. *Olive* then offers an overview of the available services (cf. *Configuration Environment*). This overview realizes the user interface of the Olive services and can support the deployment of new services. The modeling environment (e.g., Bee-Up) can interact with the *Execution Environment* (e.g., AWS) over its interface. Therefore, the knowledge of the models must be transformed into a sequence of processing steps that can be executed. Further, the information returned from the execution environment can be used to monitor the execution within the modeling environment and make decisions based on real-time data.

With our proof of concept, we evaluated our idea and analyzed how the interaction of models and microservices can be achieved to support the creation of novel city services without the need for extensive coding knowledge.

6 CONCLUSION

In this paper, we introduced the basic principles of an approach that enables citizens of a smart city to develop and offer their own services without the need for comprehensive software engineering knowledge by reusing existing microservices. For this purpose, we propose using conceptual models as the essential foundation for describing new services. These models are created by citizen developers and later processed by the smart city infrastructure to execute the new services, which are then offered to be consumed by other citizens. The introduced *drone tour guide case* was used to motivate and discuss the essential aspects of our approach.

To conclude, a SWOT analysis is presented to evaluate the findings of our study in a structured manner:

Strengths: The proposed microservice architecture enables reusability on different levels. On one hand, microservices can be reused as discussed in section 4.3. On the other hand, related scenarios could also be modeled and implemented using our approach, such as defibrillator delivering drones [33]. These advantages are based on the loose coupling of the models to the execution environment. The application knowledge is captured in the models and can be interpreted by different microservices. This implies that the defined case is executable if the microservice interface can be connected to the models, even in case the initial microservices are modified. For example, the hardware of drones might change, but as long as the interface stays similar, new drones can still be used within our proposed architecture.

Weaknesses: The introduced approach relies on an environment in which existing microservices are already available and can be reused in a meaningful way. Still, the infrastructure enabling the connection of microservices and models is not standardized. Therefore, different ways of creating and executing models or microservices exist, which are not easily integrated. Additionally, the practical application of the proposed architecture has yet to be tested in the real-world scenario of a smart city. Until today, the approach was only used in teaching and prototype environments. Further research is needed so that the approach can be analyzed in and adapted to more realistic environments.

Opportunities: The major opportunity of our approach is the offered support for citizens who want to develop smart solutions for problems within the city they live in, even if they lack the technical knowledge to do so. This advancement towards citizen developers is enabled through abstract models, which reduce the involved complexity and lower the required skills for creating smart city services in regards to software engineering and coding. As a result, city administrations can be supported with identifying and solving the respective city's challenges. Metamodeling and the modeling methods created with it offer great opportunities for this purpose since they have been confirmed as sufficient tools for supporting the digital transformation process [22]. Accordingly, future research should aim at incorporating our approach of an open and model-based microservice platform into the innovation strategy of smart cities to advance the digital transformation and, at the same time, foster citizen engagement through the openness of the platform. Offering usage-based monetary compensations to the citizen developers for their created city services provides an opportunity to support citizen engagement even further.

Threats: The threats regarding the drone tour guide correspond to the challenges discussed in section 3.1. Namely, ICT infrastructure deficits and lacking awareness among citizens (developers and consumers) pose existential challenges for most smart city services. Additionally, the increased risk of cyber attacks must be considered in this context [31], as our approach uses open data and is build on an open platform. Since the introduced approach relies on an existing system of reusable microservices, the whole approach will not reach its full potential if this system cannot be established. Further, integrating different systems and services from multiple providers increases the probability that errors occur, which can be hard to solve in such complex environments.

REFERENCES

[1] Michael Batty, Kay W Axhausen, Fosca Giannotti, Alexei Pozdnoukhov, Armando Bazzani, Monica Wachowicz, Georgios Ouzounis, and Yuval Portugali. 2012. Smart cities of the future. *The European Physical Journal Special Topics* 214 (2012), 481–518. https://doi.org/10.1140/epjst/e2012-01703-3

[2] BOC Asset Management GmbH. n.d.. Olive: Microservice Framework. https://www.adoxx.org/live/olive. last-visited: 03-02-2022.

[3] Dominik Bork, Robert Buchmann, Igor Hawryszkiewycz, Dimitris Karagiannis, Nikolaos Tantouris, and Michael Walch. 2016. Using conceptual modeling to support innovation challenges in smart cities. In *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, New York, NY, 1317–1324.

[4] Dominik Bork, Hans-Georg Fill, Dimitris Karagiannis, Elena-Teodora Miron, Nikolaos Tantouris, and Michael Walch. 2015. Conceptual Modelling for Smart Cities: A Teaching Case. *Interaction Design and Architecture(s)* (2015), 10–28. http://eprints.cs.univie.ac.at/4621/

[5] Andrés Camero and Enrique Alba. 2019. Smart City and information technology: A review. *Cities* 93 (2019), 84–94. https://doi.org/10.1016/j.cities.2019.04.014

[6] Tomas Cerny, Michael J. Donahoo, and Michal Trnka. 2018. Contextual Understanding of Microservice Architecture: Current and Future Directions. *SIGAPP Appl. Comput. Rev.* 17, 4 (2018), 29–45. https://doi.org/10.1145/3183628.3183631

[7] Annalisa Cocchia. 2014. Smart and Digital City: A Systematic Literature Review. In *Smart City: How to Create Public and Economic Value with High Technology in Urban Space*, Renata Paola Dameri and Camille Rosenthal-Sabroux (Eds.). Springer, Cham, 13–43. https://doi.org/10.1007/978-3-319-06160-3_2

[8] Damiano Falcioni and Robert Woitsch. 2021. OLIVE, a Model-Aware Microservice Framework. In *The Practice of Enterprise Modeling*, Estefanía Serral, Janis Stirna, Jolita Ralyté, and Jānis Grabis (Eds.). Springer, Cham, 90–99.

[9] Ulrich Frank. 2013. Domain-Specific Modeling Languages: Requirements Analysis and Design Guidelines. In *Domain Engineering: Product Lines, Languages, and Conceptual Models*, Iris Reinhartz-Berger, Arnon Sturm, Tony Clark, Sholom Cohen, and Jorn Bettin (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 133–157. https://doi.org/10.1007/978-3-642-36654-3_6

[10] Nicola Guarino, Giancarlo Guizzardi, and John Mylopoulos. 2020. On the Philosophical Foundations of Conceptual Models. In *Information Modelling and Knowledge Bases XXXI*, Ajantha Dahanayake et. al. (Ed.). IOS Press, Amsterdam, 1–15. https://doi.org/10.3233/FAIA200002

[11] Mark J. Handel and Steven Poltrock. 2011. Working around Official Applications: Experiences from a Large Engineering Project. In *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work (CSCW '11)*. Association for Computing Machinery, New York, NY, USA, 309–312. https://doi.org/10.1145/1958824.1958870

[12] David Harel and Bernhard Rumpe. 2004. Meaningful modeling: what's the semantics of" semantics"? *Computer* 37, 10 (2004), 64–72. https://doi.org/10.1109/MC.2004.17

[13] Brian Henderson-Sellers, Jolita Ralyté, Pär J Ågerfalk, and Matti Rossi. 2014. *Situational method engineering*. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-41467-1

[14] Dimitris Karagiannis. 2015. Agile Modeling Method Engineering. In *Proceedings of the 19th Panhellenic Conference on Informatics* (Athens, Greece) *(PCI '15)*. Association for Computing Machinery, New York, NY, USA, 5–10. https://doi.org/10.1145/2801948.2802040

[15] Dimitris Karagiannis, Robert Andrei Buchmann, Patrik Burzynski, Ulrich Reimer, and Michael Walch. 2016. Fundamental Conceptual Modeling Languages in OMiLAB. In *Domain-Specific Conceptual Modeling: Concepts, Methods and Tools*, Dimitris Karagiannis, Heinrich C. Mayr, and John Mylopoulos (Eds.). Springer, Cham, 3–30. https://doi.org/10.1007/978-3-319-39417-6_1

[16] Dimitris Karagiannis and Harald Kühn. 2002. Metamodelling Platforms. In *E-Commerce and Web Technologies*, Kurt Bauknecht, A. Min Tjoa, and Gerald Quirchmayr (Eds.). Springer, Berlin, Heidelberg, 182. https://doi.org/10.1007/3-540-45705-4_19

[17] Dimitris Karagiannis, Moonkun Lee, Knut Hinkelmann, and Wilfrid Utz. 2022. *Domain-Specific Conceptual Modeling*. Springer, Heidelberg.

[18] Dimitris Karagiannis, Heinrich C Mayr, and John Mylopoulos. 2016. *Domain-Specific Conceptual Modeling*. Springer, Heidelberg.

[19] Huma H. Khan, Muhammad N. Malik, Raheel Zafar, Feybi A. Goni, Abdoulmohammad G. Chofreh, Jiří J. Klemeš, and Youseef Alotaibi. 2020. Challenges for sustainable smart city development: A conceptual framework. *Sustainable Development* 28, 5 (2020), 1507–1518. https://doi.org/10.1002/sd.2090

[20] N. Komninos, C. Kakderi, A. Panori, and P. Tsarchopoulos. 2019. Smart City Planning from an Evolutionary Perspective. *Journal of Urban Technology* 26, 2 (2019), 3–20. https://doi.org/10.1080/10630732.2018.1485368

[21] Alexandr Krylovskiy, Marco Jahn, and Edoardo Patti. 2015. Designing a Smart City Internet of Things Platform with Microservice Architecture. In *2015 3rd International Conference on Future Internet of Things and Cloud*. IEEE, New York, NY, 25–30. https://doi.org/10.1109/FiCloud.2015.55

[22] Susanne Leist, Dimitiris Karagiannis, Florian Johannsen, and Hans-Gert Penzel. 2021. From Business Engineering to Digital Engineering: The Role of Metamodeling in Digital Transformation. In *Engineering the Transformation of the Enterprise*, Stephan Aier, Peter Rohner, and Joachim Schelp (Eds.). Springer, Cham, 51–71. https://doi.org/10.1007/978-3-030-84655-8_4

[23] James Lewis and Martin Fowler. 2014. Microservicesa definition of this new architectural term. https://martinfowler.com/articles/microservices.html. last-visited: 03-02-2022.

[24] Catriona Manville, Gavin Cochrane, Jonathan Cave, Jeremy Millard, Jimmy Kevin Pederson, Rasmus Kåre Thaarup, Andrea Liebe, Matthias Wissner, RA Massink, and Bas Kotterink. 2014. Mapping Smart Cities in the EU. (2014), 1–196.

[25] Heinrich C Mayr and Bernhard Thalheim. 2021. The triptych of conceptual modeling. *Software and Systems Modeling* 20 (2021), 7–24. https://doi.org/10.1007/s10270-020-00836-z

[26] Andres Monzon. 2015. Smart cities concept and challenges: Bases for the assessment of smart city projects. In *2015 International Conference on Smart Cities and Green ICT Systems (SMARTGREENS)*. IEEE, New York, NY, 1–11.

[27] Catherine E. A. Mulligan and Magnus Olsson. 2013. Architectural implications of smart city business models: an evolutionary perspective. *IEEE Communications Magazine* 51, 6 (2013), 80–85. https://doi.org/10.1109/MCOM.2013.6525599

[28] Paolo Neirotti, Alberto De Marco, Anna Corinna Cagliano, Giulio Mangano, and Francesco Scorrano. 2014. Current trends in Smart City initiatives: Some stylised facts. *Cities* 38 (2014), 25–36. https://doi.org/10.1016/j.cities.2013.12.010

[29] United Nations | Department of Economic and Social Affairs. 2018. 68% of the world population projected to live in urban areas by 2050, says UN. https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html. last-visited: 03-02-2022.

[30] Marten Oltrogge, Erik Derr, Christian Stransky, Yasemin Acar, Sascha Fahl, Christian Rossow, Giancarlo Pellegrino, Sven Bugiel, and Michael Backes. 2018. The Rise of the Citizen Developer: Assessing the Security Impact of Online App Generators. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, New York, NY, 634–647. https://doi.org/10.1109/SP.2018.00005

[31] Javier Pastor-Galindo, Félix Gómez Mármol, and Gregorio Martínez Pérez. 2021. Nothing to Hide? On the Security and Privacy Threats Beyond Open Data. *IEEE Internet Computing* 25, 4 (2021), 58–66. https://doi.org/10.1109/MIC.2021.3088335

[32] Apurvanand Sahay, Arsene Indamutsa, Davide Di Ruscio, and Alfonso Pierantonio. 2020. Supporting the understanding and comparison of low-code development platforms. In *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, New York, NY, 171–178. https://doi.org/10.1109/SEAA51224.2020.00036

[33] Sofia Schierbeck, Jacob Hollenberg, Anette Nord, Leif Svensson, Per Nordberg, Mattias Ringh, Sune Forsberg, Peter Lundgren, Christer Axelsson, and Andreas Claesson. 2021. Automated external defibrillators delivered by drones to patients with suspected out-of-hospital cardiac arrest. *European Heart Journal* ehab498 (2021), 1–10. https://doi.org/10.1093/eurheartj/ehab498

[34] Anthony Simonofski, Emile Hertoghe, Michiel Steegmans, Monique Snoeck, and Yves Wautelet. 2021. Engaging citizens in the smart city through participation platforms: A framework for public servants and developers. *Computers in Human Behavior* 124 (2021), 106901. https://doi.org/10.1016/j.chb.2021.106901

[35] Bernhard Thalheim and Ivor Nissen. 2015. *Wissenschaft und Kunst der Modellierung: Kieler Zugang zur Definition, Nutzung und Zukunft*. Vol. 64. Walter de Gruyter GmbH & Co KG, Berlin/Boston.

[36] Carolin Ullrich, Teodora Lata, and Jerome Geyer-Klingeberg. 2021. Celonis Studio–A Low-Code Development Platform for Citizen Developers. *CEUR Workshop Proceedings* 2973 (2021), 102–105.

[37] Michael Walch. 2017. Knowledge-driven enrichment of cyber-physical systems for industrial applications using the KbR modelling approach. In *2017 IEEE International Conference on Agents (ICA)*. IEEE, New York, NY, 84–89. https://doi.org/10.1109/AGENTS.2017.8015307

[38] Michael Walch. 2018. Operating cyber-physical systems with microservices: the s* IoT conceptual modelling approach. In *2018 7th International Congress on Advanced Applied Informatics (IIAI-AAI)*. IEEE, New York, NY, 787–792. https://doi.org/10.1109/IIAI-AAI.2018.00162

[39] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. 2014. Internet of Things for Smart Cities. *IEEE Internet of Things Journal* 1, 1 (2014), 22–32. https://doi.org/10.1109/JIOT.2014.2306328