ADAPTIVE 2023

The Fifteenth International Conference on Adaptive and Self-Adaptive Systems and Applications

ISBN: 978-1-68558-047-6

June 26 - 30, 2023

Nice, France

ADAPTIVE 2023 Editors

Andreas Rausch, Technische Universität Clausthal, Clausthal-Zellerfeld, Germany

Christoph Knieke, Technische Universität Clausthal, Clausthal-Zellerfeld, Germany

ADAPTIVE 2023

Forward

The Fifteenth International Conference on Adaptive and Self-Adaptive Systems and Applications (ADAPTIVE 2023), held on June 26 - 30, 2023, continued a series of events targeting advanced system and application design paradigms driven by adaptiveness and self-adaptiveness. With the current tendencies in developing and deploying complex systems, and under the continuous changes of system and application requirements, adaptation is a key feature. Speed and scalability of changes require self-adaptation for special cases. How to build systems to be easily adaptive and self-adaptive, what constraints and what mechanisms must be used, and how to evaluate a stable state in such systems are challenging duties. Context-aware and user-aware are major situations where environment and user feedback is considered for further adaptation.

The conference had the following tracks:

- Self-adaptation
- Adaptive applications
- Adaptivity in robot systems
- Fundamentals and design of adaptive systems
- Computational Trust for Self-Adaptive Systems
- Assurances and metrics for adaptive and self-adaptive systems

Similar to the previous edition, this event attracted excellent contributions and active participation from all over the world. We were very pleased to receive top quality contributions.

We take here the opportunity to warmly thank all the members of the ADAPTIVE 2023 technical program committee, as well as the numerous reviewers. The creation of a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors that dedicated much of their time and effort to contribute to ADAPTIVE 2023. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations and sponsors. We also gratefully thank the members of the ADAPTIVE 2023 organizing committee for their help in handling the logistics and for their work that made this professional meeting a success.

We hope ADAPTIVE 2023 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in the area of adaptive and self-adaptive systems and applications. We also hope that Nice provided a pleasant environment during the conference and everyone saved some time to enjoy this beautiful city

ADAPTIVE 2023 General Chair

Jaime Lloret Mauri, Universitat Politecnica de Valencia, Spain

ADAPTIVE 2023 Steering Committee

Constantin Paleologu, University Politehnica of Bucharest, Romania
Claudia Raibulet, University of Milano-Bicocca, Italy
Sebastian Herold, Karlstad University, Department for Mathematics & Computer Science, Sweden
Andreas Rausch, TU Clausthal, Clausthal-Zellerfeld, Germany
Marc Kurz, University of Applied Sciences Upper Austria, Faculty for Informatics, Communications and Media, Austria
Valerie Camps, Paul Sabatier University - IRIT, Toulouse, France
Yukio Hayashi, Japan Advanced Institute of Science and Technology, Japan

ADAPTIVE 2023 Publicity Chair

José Miguel Jiménez, Universitat Politecnica de Valencia, Spain
Sandra Viciano Tudela, Universitat Politecnica de Valencia, Spain

ADAPTIVE 2023

Committee

ADAPTIVE 2023 General Chair

Jaime Lloret Mauri, Universitat Politecnica de Valencia, Spain

ADAPTIVE 2023 Steering Committee

Constantin Paleologu, University Politehnica of Bucharest, Romania
Claudia Raibulet, University of Milano-Bicocca, Italy
Sebastian Herold, Karlstad University, Department for Mathematics & Computer Science, Sweden
Andreas Rausch, TU Clausthal, Clausthal-Zellerfeld, Germany
Marc Kurz, University of Applied Sciences Upper Austria, Faculty for Informatics, Communications and Media, Austria
Valerie Camps, Paul Sabatier University - IRIT, Toulouse, France
Yukio Hayashi, Japan Advanced Institute of Science and Technology, Japan

ADAPTIVE 2023 Publicity Chair

José Miguel Jiménez, Universitat Politecnica de Valencia, Spain
Sandra Viciano Tudela, Universitat Politecnica de Valencia, Spain

ADAPTIVE 2023 Technical Program Committee

Habtamu Abie, Norwegian Computing Center/Norsk Regnesentral-Blindern, Norway
Harvey Alférez, Universidad de Montemorelos, Mexico
Raid Al-Nima, Northern Technical University, Iraq
Nik Bessis, Edge Hill University, UK
Glen Bright, University of KwaZulu Natal, Durban, South Africa
Antonio Brogi, University of Pisa, Italy
Barbara Buck, Boeing Global Services, USA
Valerie Camps, Paul Sabatier University - IRIT, Toulouse, France
Constantin F. Caruntu, “Gheorghe Asachi” Technical University of Iasi, Romania
Angel P. del Pobil, Jaume I University, Spain
Laura-Maria Dogariu, University Politehnica of Bucharest, Romania
Ibrahim Abdallah Abbas Atwa Elgendy, School of Computer Science and Technology | Harbin Institute of Technology, China
Holger Eichelberger, University of Hildesheim, Germany
Fairouz Fakhfakh, University of Sfax, Tunisia
Francisco José García-Peñalvo, University of Salamanca, Spain
Zhiwei Gao, Northumbria University, UK
Yukio Hayashi, Japan Advanced Institute of Science and Technology, Japan

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

Tailored Digital Twins for Lifecycle Assessment & Management

Stakeholder centered Digital Twin Framework Design for Product Lifecycle Managements and Assessment

Dominique Briechle, Marit Mathiszig, Nelly Nicaise Nyeck Mbialeu, Ali Piriyaie, Argianto Rahartomo
Institute for Software and Systems Engineering
University of Technology Clausthal
Clausthal-Zellerfeld, Germany
Email: dominique.fabio.briechle@tu-clausthal.de; marit.elke.anke.mathiszig@tu-clausthal.de;
nelly.nicaise.nyeck.mbialeu@tu-clausthal.de; ali.piriyaie@tu-clausthal.de; argianto.rahartomo@tu-clausthal.de

***Abstract*— The notion of sustainability is gaining more attention across the whole world, thereby serving as a predictor of future economic gain. Emerging technologies like Digital Twins (DT) could help meet Circular Economy objectives such as the UN Sustainable Development Goals. Although using a framework to develop DT is widespread, it is often not directly suited to answer the information requirement that the stakeholders desire fully. Shortcomings in design are therefore discovered to late when the concept of the Twin is already implemented. As such, this paper aims to provide a requirement-based framework for designing a sustainable Digital Twin that satisfies several parties involved in its conception and use. The proposed framework is therefore designed on the one hand to filter out the most valuable information requests for the stakeholders. On the other hand, the design requests are turned in the design phase to recommendations for the outlaying of the architectural implementation. This is shown as an example application for an e-bike with different stakeholders involved. Therefore, the use case features five different stakeholder groups with their own information request to show how these influence the final setup of such a twin. The proposed framework lays the path for improving the current lifecycle assessment while ensuring the optimization of Digital Twin design flexibility and therefore its application for a variety of different products.**

Keywords - Stakeholder Information Requirement; Digital Twin; Product Lifecycle; Framework Design, Circular Economy

I. INTRODUCTION

The current product usage leads globally more and more to a massive production of waste. Therefore, the question of product life prolonging techniques and measurements is on an all-time high. Especially in the context of Circular Economy, the condition assessment of products is a frequently mentioned topic.

Sustainable consumption and production are also important topics for society and ecology in general regarding the increasing world population [1]. Furthermore, the UN defined 17 sustainability goals in order to accelerate environmental and socially friendly usage of resources and products [2]. However, in order to apply the measurements detailed information about a product's current state, product life and usage are required. For example, the correct assessment of a product's condition is needed to decide, if a product can be used further or if the manufacturing of a new product would be more beneficial. This is where Digital Twins can come into usage; digital counterparts of physical products which can be used to monitor its state and optimize its usage. However, it is not simply sufficient to just monitor a product randomly. In order to gain the most out of a Digital Twin, its usage and functionality should be determined before implementing the potential architecture. The framework is therefore laid out to help stakeholders designing a Digital Twin for their expense with the most beneficial information generation capabilities. The target of this paper is to support the conception process of Digital Twins for a product and satisfy specifically the stakeholders information request due to a new framework design. The question of a suitable approach for Digital Twin Design in order to improve information value by simultaneously decreasing expense is tried to be answered while also addressing the limitations of the framework. This lie especially in the limited generation of new, unpredicted information that may be gained due to correlation of sensor data. This is addressed in a subchapter in order to discuss the limitations of the framework approach. It should be stated, however, that this is a concept which still needs validation and testing.

The paper consists of different subsections, describing the initial structure of the research work. In section II the current State of the Art of the Digital Twin-based research in terms of framework design is summarized. The section III describes the initial problem, followed by the used research methodology described in section IV. The main part consists of the framework design explained in section V and its use case application in section VI. The paper closes with a final conclusion in section VII.

 ISBN: 978-1-68558-047-6

II. STATE OF THE ART

This section aims to provide a brief overview of some of the currently proposed frameworks for digital twins, highlighting relevant studies and works related to the present paper. Over the years, the digital twin concept has evolved from just being the virtual mirror of a physical asset [3] to include digital twin data and services for static as well as dynamic information flow between both entities [4][5]. Over the time various digital twin frameworks have been proposed as the digital twin concept has been extending to more scientific and industrial fields.

Tao et al. [4] define a framework known as Digital Twin-driven Product Design (DTPD), which focuses on creating the digital twin of a physical asset, then applying the acquired generated knowledge in the redesigning process of the asset. In essence, it is a design framework that transforms big data into useful information that are used in the following process to guide decisions during the different design phases of the asset.

Zhang et al. [6] worked on a data- and knowledge-driven framework for the Digital Twin of a Manufacturing Cell (DMTC). The physical, digital, data, knowledge, and social spaces make up the five-dimensional space of this framework, which collectively promote the capacity of the digital twin for self-control functionalities as well as self-assessing capabilities. Thereby, rendering the manufacturing cell autonomous while maintaining flexibility and lowering cost.

D'Amico et al. [7] propose a conceptual framework for the evaluation of the Remaining Useful Life (RUL) of a product. The aim is to improve the availability of the product and minimize expenses throughout its life cycle by increasing the level of understanding of this asset. Thus, this framework focuses on an efficient exchange of information through its three main layers, which include the data architecture, modules, and connection of the modules to one another.

Onaji et al. [8] provide a framework that supports some essential functionalities in digital twin applications such as tools and functionalities for prognosis and diagnosis of behavior, simulations of the related system, monitoring and controlling of the system as well as its optimization [6][9][10]. It is made up of six components that work together to enable the manufacturing sector to benefit from features like integration, interconnectivity, flexibility, analytics, and supported decision-making.

A three-layered structure is conceived by Traoré [11] that combines the various perspectives on Digital Twins to clarify the concept that already has diverse understandings. This framework has a data layer, a capability layer, and a service layer, which provide a modular conceptual foundation that can aid in the adaptive or dynamic design of specific solutions. Practically, the framework was sampled to create the Digital Twin of an energy-efficient building and a smart manufacturing shop floor.

A proposal of a structure which incorporates the basic functionalities, initial requirements and guiding definitions for standardized components of Digital Twin is proposed by Nwogu [12]. It is a requirement-driven, technology-agnostic structure that can be applied to different situations based on their unique requirements. Hence, creating a close relationship between the Digital Twin requirements and the components in the suggested framework.

Zhao et al. [13] noticed issues with the usage of digital twins in the Construction industry, such as the misalignment of data integration and data standards, and a lack of information within each component. To tackle those obstacles, a conceptual framework is suggested to enable a broader application and implementation of digital twins for facility management throughout the Operation and Maintenance phase. Also, the framework takes into consideration stakeholders who are struggling with facility management decision-making processes. Moreover, the framework has six layers: preparation layer, data acquisition layer, data processing layer, data transmission and modeling layer, model logic layer with intelligence tools, and application presentation layer.

The aforementioned brief literature review shows that multiple specific frameworks for various uses exist that integrate existing Digital Twin viewpoints in one way or another. Our research is distinctive in that it suggests a framework that facilitates the process of evaluating product conditions to improve sustainability with special consideration of the stakeholder requirements.

III. PROBLEM STATEMENT

Stakeholders have in general a high interest in acquiring information regarding the product and its usage by the consumer. In the case of a Digital Twin this is usually the virtual counterpart of the physical product which, when combined to one another, creates a cyber-physical entity. The kind of information extracted from the Digital Twin and received by the stakeholders can therefore vary widely and is usually limited by the technological borders of the respective product like data acquisition methods, product features or transmission capabilities [31]. However, before the information can be transmitted and processed, the governing question regarding the kind of information the stakeholders want to receive out of their Digital Twin must be answered in order to achieve a model which not only satisfies the specific stakeholders' needs but also is sustainable in terms of the selection and shortage of an unnecessary information overflow (redundancies). This leads to the two assumptions that can therefore be stated:

 ISBN: 978-1-68558-047-6

1. The selection of stakeholder information requirements is a highly important factor when designing Digital Twins
2. The use of a "smart" digital twin framework design could prevent insufficient or over-dimensioned DT designs that lead to more resource wastage and consumption than necessary

The later proposed framework takes the physical entity properties and its environment into account and ties them directly to the framework design considerations. The information generation is therefore highly guided by the stakeholders needs and can be separated based on the two governing principles of the utilizability of a specific type of information for the targeted Stakeholder group and the technical capabilities of the information generation system [32]. To highlight the needs of the stakeholders, the type of stakeholder connected to a specific product must be determined first. Based on the number of information interests *in(1…m)(1…n)* for each of the *stakeholders (1….m)*, the result of the total number of relevant information factors can thereby be determined. This setup/configuration has a significant impact on the general sensing capabilities of the digital twin, as well as the overall structure of the framework [33]. The next section will describe the initial situation of a product with a diverse stakeholder composition.

A. Initial Scenario: E-Bike Sharing Service

In order to explain the procedure of designing a tailor-made Digital Twin, a e-bike sharing ecosystem is selected as use case. E-bikes are on an all-time rise since the increased transition of urban traffic ecosystems towards more sustainable mobility services [22].

In general, such ecosystems consist of a variety of different stakeholders [23][24]. Manufacturers, Clients, Service Providers and Product Maintainers as well as Recyclers and even Resellers can be viewed as examples of stakeholders that are part of the bike sharing ecosystem. In terms of the application of a digital twin of an e-bike, all of those stakeholders have specific information requests which they are interested in being fulfilled. On the one hand, these demands are not necessarily the same and are therefore of different value for each of the stakeholders. On the other hand, there are information demands, which are shared mutually by different shareholders and therefore have a higher value for all of them. Therefore, it is necessary to deal with the information requests of the stakeholders in order to find the highest information benefit at a given cost for the achievement of the former. Disregarding the stakeholder's information request can lead to unsatisfying decisions concerning the product's life because of a lack of information, that is not in the sense of a sustainable product usage and the Circular Economy in general.

IV. Research Methodology

In order to acquire a sufficient knowledge base on different types of information requests served by Digital Twin technology and framework designs, the authors used research papers and articles which were acquired by searching for different keyword combinations in order to create search strings to find suitable research papers. The search was conducted using google scholar and science direct as main tools in order to find suitable references. For the acquiring process the reference search for Digital Twin related papers was conducted by the setup sections of the digital twin and their relevance for Circular Economy related topics. The proposed frameworks in the state of the art section were assessed based on the following five key features [28][29]:

- Communication (between Digital Twin, physical entity and entity environment)
- Data acquisition
- Data processing
- Application Services
- Adjunct Adaption and reutilization

Based on the above-named features, the referenced papers were selected as research foundation for the designed Digital Twin framework. The main setup of the proposed frameworks was analyzed and the above-mentioned key features were further concretized in order to specify the layer structure of the later proposed information requirement-driven Digital Twin Framework.

Digital Twins have been used vastly within the last few years. However, most of the proposed Digital Twin Designs are centered around the product and the information that can be extracted by configuring data acquisition tools for the specific cause [30]. The goal of this paper is the design approach for a stakeholder requirement-driven framework which allows the configuration based on the final information that is desired by the stakeholders in order to use them in their field of application. The section therefore clarified the used research methodology and outlined the purpose of the work. The following section of the paper outlines the various requirements that stakeholders may have based on their branch and endeavors to establish a categorization of the required information.

V. Framework Components

The design of a framework for Digital Twins must necessarily address the needs of its stakeholders, that require the information which will be generated by it. Therefore, it is of uttermost importance to take the environment and its stakeholders into account when setting up the framework. The general environment of the Digital Twin is highly versatile and will change drastically depending on the product which is subject to the Twin. This is not only caused by the different products affecting the properties of a specific Twin but as well by the types of stakeholders and their specific information requests. Further, the

 ISBN: 978-1-68558-047-6

environmental influence is governed by the technical limitations of the sensing and acting devices, which results in a highly specific environmental composition for each product type. The idea of requirement-driven frameworks, however, is not totally new and has already been described by Nwogu et al. [34]. The basic difference between the approach presented by Nwogu et al. and the framework design stated here is the integration of the Stakeholder Layer necessary to form not only the requirements addressed to the required features but as well the requirements based on the basic information type. The Digital Twin Framework is as well not restricted merely to the data flow from the Framework to the adjunct sensing and acting systems but is receiving, as well a stream based on additional information and knowledge generated by preceding systems. However, the general foundation of the proposed framework is based on the one presented 2020 by Lu, C. et al. [35].

The proposed framework, shown in Figure 1 integrates a ***Product Stakeholder Layer (1)***, where the Stakeholders along with their information requests will directly contribute to the setup of the Digital Twin. The classification of the Stakeholders as well as their individual information request are thereby guiding the information generation boundaries of the resulting Digital Twin. Possible Stakeholders of physical products are for example the Designers, Manufacturers, Distributors, Maintenance Service Providers, Remanufacturers, Recyclers and of course the actual users of the product. The Stakeholders have different requests for information since they influence the product at different stages of its lifecycle [36]. The goal of the Layer is therefore the determination of the stakeholders' information demands as well as the selection of those demands based on the governing cost of the information generation. This will result however in a catalogue of information demands, which are then used in the adjunct Design & Conception Layer to create the architecture of the Digital Twin.

Due to the main driving requirement of information-based framework design, the Design phase of the actual structure is integrated as a central part of the concept. This results in an initial ***Design Block (2)*** integrated into the framework. In the Design & Conception Layer the Stakeholder request are evaluated and sorted in order to select the proper design of the resulting Digital Twin. Therefore, the exact setup of the respective Twin is modelled in terms of the establishment of communication between the different subsystems (synchronization frequency, type of transmission), the usage of different tools to produce the required information and the implementation of the generated results and the thereby resulting control over the actuation systems [37]. The planned design is therefore affecting the subsequent systems of the framework to a large extend. After setting the governing features of the respective Digital Twin for a product, the in-depth planning of the subsequent systems is conducted. The first section which is thereby accounted is the ***Data Block (3)*** with the Data acquisition, Data processing and Data validation Layer. The type of data acquisition is naturally limited by the technical limitations of the information request type and the necessary data collection system which is used in this context. Data acquisition therefore covers all types of data transmission whether it might be collected via sensing devices or by other data sources like historical usage data, production and design data or data concerning the product's usage after reaching the end of its lifecycle. The Data acquisition Layer is hereby the foundation of the Digital Twin and is therefore highly affecting the quality of the resulting information and knowledge. Data processing is equally important to the acquisition of the data since it adds the necessary properties to generate the desired information which allows the classification of the raw data. The processing is therefore a crucial task in order to use the data in the subsequent applications. The first step is hereby the data preparation, where the raw data is cleansed of flawed values and redundancies in order to use them in the process of information generation. The type of processing used is based as well on the initial information requirement and can as well differ drastically between its processing cycle and its amount of processed data in one iteration. The location of the processing step is also a governing factor for the Digital Twin, although the development of scalable far- and near-edge technology are currently bridging the selection of the processing location and therefore enabling system scalability [38]. The next step is the validation of the data and its transmission to the adjunct Evaluation block. Data validation is the third column of the data concerned block and is the "reality check" of the generated information to prevent the transmission of illogical and false information. The validation is therefore the second iteration of the transmission check. The basic difference between the processing and validation layer is the "item" which is evaluated: whereas the processing layer evaluates and filters the raw data, the validation layer accounts the plausibility of the initial databased and generated information. As well as the adjunct processing layer, the other data and information transferred to the acquisition layer by the different systems are also evaluated [39]. The ***Service Block (4)*** is the subsequent unit after the data-concerned section. In this section, the previously created data and information are transmitted to the different service branches in order to generate follow-up information to answer the initial information requests. For example, the question for lifecycle estimation, future product life assessment, and construction optimization can be answered. The block is separated into three distinguished layers: Preassessment, Prognosis, and Evaluation. The preassessment layer is used for the preparation of the gathered information in order to use it in the subsequent application. This includes the refrainment of the validated information and the classification in order to use them for simulations, prognosis, or recommendation functionalities. The selection of the right setup is here as well highly dependent on the type of required information and, subsequently the digital service which is selected

 ISBN: 978-1-68558-047-6

afterward [40]. These services are implemented in the Service Layer. This layer is in contrast to the beforementioned layer more versatile since it can consist of a high variety of different tools, tailor-made for the specific use case, and therefore complex in its internal structure. The layer is the core section of the Digital Twin in terms of adjunct functionalities, and combines the features around the utilization of the generated information for the specific application [41]. Therefore, the layer is built in a modular approach which can integrate different types of simulation systems that can be separated based on their calculation methodology and the overall model, on which they are built. Examples here for are modular dynamic simulations and Monte-Carlo Simulations, which are used in a variety of different simulation applications [42]. This concept is already in use and is carried out for different types of products and systems. As an example, the usage of Object-Oriented Modelling and Simulation (OOMS) can be seen as an approach to tackle highly complex simulation environments [43]. Further, the usage of prognosis tools is highly anticipated in order to assess the products behavior throughout its lifetime. As an example, this was already described in order to predict the behavior of bearing components used in the railcar industry. The conducted research clearly showed, how Artificial Intelligence (AI) is already in use in order to prognosticate the behavior and wear of components and can therefore be integrated and coupled to its adjunct layers [44]. The last layer of this block is the evaluation layer. This layer can as well be based on AI technology, but in contrast to the prediction layer, it is the third subsystem of the proposed framework and is used for validation of the two preceding layers of the block. The newly generated, and mostly artificially created information is therefore undergoing a last reality check in order to assess meaningfulness. This step is highly important regarding the overall acceptance of AI based systems and their capability of managing and deciding process steps.

The last block implemented in the internal framework is

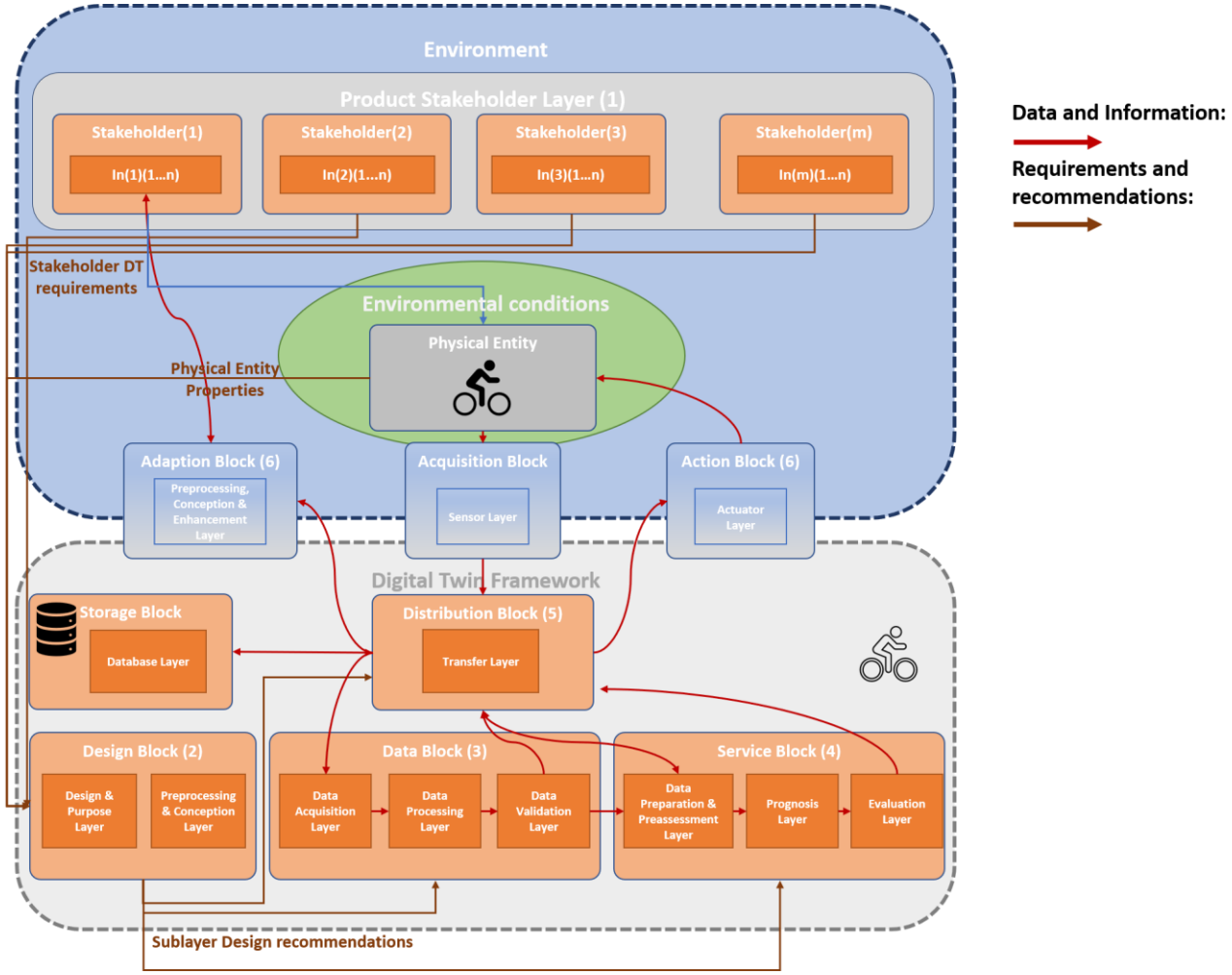

Figure 1: Stakeholder Driven Digital Twin Framework Design (inspired by [35])

the ***Distribution Block (5)***. This layer ensures the storage and communication with the adjunct entities outside the internal Digital Twin framework towards the environment. The Transfer layer is affected as well by the neighboring subsystems, especially in terms of the processing of the different types of data and information. Due to the different processing requirements of the product the overall complexity of the Transfer Layer can vary. This variety includes the usage of edge and cloud computing and the distribution of tasks to the different processing devices [38]. Further, the selection of a fitting transmission technic is as well part of the Layer.

The two sections that complete the overall framework proposal are the ***Action and Adaption Blocks (6)***. These blocks are responsible of adjusting the Twins behavior via the actuators and managing the generated information in order to transform it into wisdom which is reused by the Stakeholders in the different application like design, manufacturing or maintenance [45]. The actuator block consists therefore of the actuator systems of the physical product and carries out the control tasks in order to optimize the product usage in case of automated subsystems. Therefore, a life-prolonging operation state of the product can be enabled and changes in the product's environment can be tackled accordingly to prevent increased wear [46]. The Adaption concerned block consists of all methods that implement generated wisdom, lessons learned, and guidelines into the Stakeholders' processes. The transformation of the transferred information is thereby as well highly dependent on the stakeholders' need and is subject of the overall design process of the Digital Twin. However, the two beforementioned blocks are hybrids between the environment, the physical product, and the digital twin and are therefore located in the overall framework between the different subframes. In the following subchapter, the presented Framework and its Layers illustrated in Figure 1 are applied onto the initially mentioned use case.

VI. USE CASE APPLICATION

The application of the framework is utilized in the following section onto the initially described use case and is shown graphically in Figure 2 As described in the framework introduction, the first step is the determination of the information demands. Therefore, the participating stakeholders of the specific product environment have to be assessed at first.

In order to reduce the complexity of the presented use case, the Stakeholders are summed up in groups based on their common information requirements [33][47, p. 4]:

1) Designer & Manufacturer

While the product designer and the product manufacturer don´t have to be necessarily the same Stakeholder, this is the case for a variety of different products. The information requirements for this Stakeholder group cycles around the possibility to increase the monetary value and the improvement of their product. This leads us to the general assumption, that the majority of the information a producer is interested in will be information on current usage lifecycles of a product as well as the overall condition of products throughout the same. Further, the Stakeholder might be interested in information regarding the saving of material by optimized design based on the information of the Digital Twin.

2) Distributor and Maintenance

The products' distributor is the second Stakeholder group and is directly in contact with the user of the given product. While the distributor has a high interest in increasing its monetary outcome by selling the product, the overall information required regarding the actual product and its usage might not be as high when compared to the other stakeholder groups. This is mainly caused by the fact that the main interest lies in the actual distribution and sale of the product. This changes when the maintenance of the

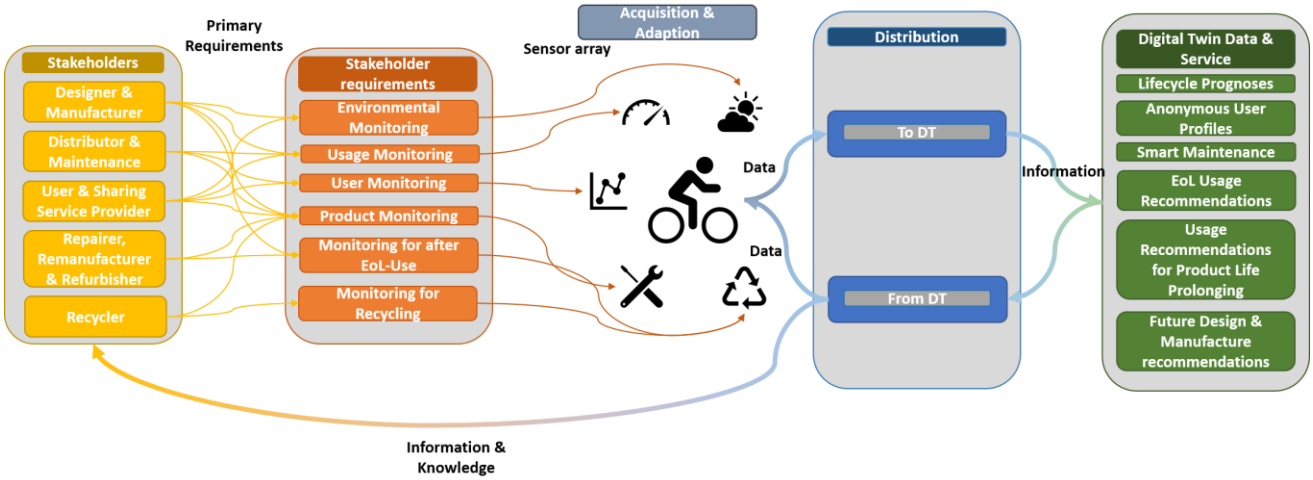

Figure 2: Framework applied to E-Bike

product is included as well in the assumption. The information requirement is therefore shifted to a lifecycle-centered one with focus on duration and maintenance optimization information of the regarding Product.

3) User & Sharing Service Provider

The third basic stakeholder group is the user & the service provider. The user's interaction with the product will greatly influences its overall condition and thus directly influences the product lifecycle. Nonetheless, the information interest of the digital twin is requested in terms of current usage behavior and information for lifecycle prolonging measures. Further the possible failure of different parts under the current usage profile is a highly valued information for the user of a given product. In the case of a ridesharing service, users are mainly focused on the reliability and availability of the product. These two factors will as well directly affect the sensing and transmission design of the hardware setup and therefore the Digital Twin itself.

4) Repairer, Remanufacturer & Refurbisher

The fourth basic stakeholder group consists of the stakeholders, that are involved in the handling of the e-bike after reaching its End-of-Life (EoL). However, the focus of this group is to enable the product to be used further in its original context. The focus is therefore reinstating the original functionality of the bike. The stakeholders have therefore a major information request regarding the current and future condition of the e-bikes and its parts as well as the potential time of disfunction in order to assess the requirements for reinstating the functional state. For the Digital Twin, this requires a high accuracy of the prognosis service as well as the EoL recommendation in order to satisfy the information requirements.

5) Recycler

The last stakeholder group deals with the inevitable fate of a product, when it is damaged beyond any form of repair. Recyclers are therefore mostly interested in information regarding the product type and its potential hazards in order to plan the dismantling and recycling process as well as the quantity of valuable resources which were used in the products manufacturing process. The recycler requires the information when the product reaches its EoL and the reinstating of its functionality is not desirable.

Generally speaking, the most desirable information for the different stakeholders can therefore be summarized as all information concerning the product life, usage and optimization. These information types are subsets of a lot of information labels and can therefore be seen as supergroups for the subsequent information [48].

For the product of the mentioned use case this concerns especially the monitoring of the motor, the accumulator and the frame of the bike, since these are the most expensive parts of an e-bike. Therefore, the sensor array should be designed to acquire condition information about the bike based on usage behavior and environmental influences [49]. The sensor array should on the one hand monitor the condition of the parts and on the other hand it must monitor the user behavior and environmental impact on the components. In order to deliver the data required to answer the questions above, the sensor array could consist of temperature, humidity, and photosensitive sensors for the evaluation of the environmental conditions and of strain gauge, triaxial sensors, and data from the Battery Management System (BMS) for the monitoring of the components [50].

In case of the E-Bike, the location of the Digital Twin and the transmission frequency will influence the overall setup greatly. As mentioned before, the Stakeholders may have different requirements based on the frequency of the transmissions and must find an agreement. For example, for the E-Bike, the highest frequency need would be the advisable way to plan the data transmission installations. For the previously described use case of Bike Sharing, the transmission could be established, for example either by the user's mobile device via mobile communication or while loading the E-Bikes at the different sharing points via wireless LAN.

The service concerned applications can differ as well widely based on the composition of the Stakeholders. Possible service applications for the E-bike could be for example DT data-based Finite Element Method (FEM) simulations to determine component wear over time or the prognose of accumulator loading capacity decreasement over time due to the usage behavior (Loading cycles, loading environments etc.) [51]. These applications can deliver information desirable for the different Stakeholders and can thereby enable subsequent services like predictive maintenance or future-life-planning of the components.

The transmission technology onto the Adaption and Action concerned block will usually in the case of E-Bikes be the same as the one that connects the Data and Service concerned block to one another (since the E-Bike will most properly not be outfitted with a processing device powerful enough to carry the simulations out on spot). Therefore, the information will be transmitted once again via the Transfer Layer to the above-mentioned Layers. The Actuator Layer is in case of the E-Bikes restricted to the optimization of the running conditions of the electrical motor and the battery system, since these two components are the only devices outfitted with controlling and managing subsystems. The Adaption concerned Layer integrates, as described in the corresponding section in the previous chapter, the information into knowledge or lessons learned. In case of the use case, the Layer could provide specific recommendations on how to optimize the components in order to increase their durability and overall lifetime as well as create patterns in order to enhance the stakeholder's services like the maintenance operations carried out by the provider or the business model adjustments regarding the distribution of the rented E-Bikes after reaching their technical obsolescence [51].

 ISBN: 978-1-68558-047-6

B. Tailor Made vs Research Driven DT-Design

The presented design is faced to provide a framework solution under two major premises: the first one is the DT-design for Digital Twins which are used in applications were a scarcity of resources is expected and therefore careful design approaches are mandatory. The second one is the extensive domain knowledge of the stakeholders regarding the overall Digital Twin Design. This includes as well the requirement of a well-researched domain, where there are nearly no unknown factors which could affect the physical entity in its usual environmental behavior. However, it should be noticed that the framework is therefore not suitable for use cases which are still not well researched and where domain knowledge is limited. The framework can be deployed in digital twin applications like the above-mentioned E-Bike, where the physical counterpart and its usual environmental influence have already been researched widely. In other applications however a different Digital Twin design approach, which is targeted more clearly to increase the overall data acquisition by measuring a lot of different parameters with a broad sensor array might be the better solution. The described tailor-made framework has on the other hand the benefit of decreasing resource input by pre-selecting the measuring capabilities.

VII. CONCLUSION

The main motivation of this research is to provide a framework usable to design tailor-made Digital Twin systems based on the initial requirement of the stakeholders. The proposed framework was set in context with a described use case regarding the lifecycle management of an E-Bike. The application of the initially described use case showed clearly, on one hand, the necessity of requirements- driven Digital Twin Frameworks in order to fulfill the needs of the Stakeholders as well as the functionality of an architecture setup for Digital Twins based on the proposed design pattern. The governing factor of the information requirement proved to be essential for designing fitting and energy optimized Digital Twins. The integration of the design of the Digital Twin as central part of the Framework design allows the initial collection of information requests and allows simultaneously the retrofitting of already in use architectures for the own purpose by reconsidering the previously made selections in the design phase. However, the selection of the required information by the Stakeholders in order to assess the final architecture of the physical entities' corresponding Digital Twin is still up for debate. The information selection phase therefore requires its own set of rules and algorithms in order to grant a fair and even distribution of information interest among the different Stakeholders. In general, the hereby proposed framework concept can be seen as a first step in order to conduct research in the area of stakeholder information requirement driven DT design. Using requirement driven frameworks could therefore not only pave the way for an enhancement of current lifecycle assessments but can as well optimize Digital Twins in general in terms of energy consumption, sensor implementation, and data and information redundancy based on the approach of tailor-made architecture designs for different products. However, it should be stated, that the concept still needs testing and validation in order to confirm its general expressiveness.

REFERENCES

[1] L. Elke, F. Swiaczny, A. Genoni, N. Sander, and R. Westermann, "Global population development. Facts and Trends" Bundesinstitut für Bevölkerungsforschung, Jul. 2021. doi: 10.12765/bro-2021-01.

[2] Redaktion: Lois Jensen, Vereinte Nationen, "Goals for a sustainable development Report 2022." 2022. [retrieved: Mai, 2023] Available: https://www.un.org/Depts/german/millennium/SDG-2022-DEU.pdf

[3] M. Grieves and J. Vickers, "Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems," in *Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches*, F.-J. Kahlen, S. Flumerfelt, and A. Alves, Eds. Cham: Springer International Publishing, 2017, pp. 85–113. doi: 10.1007/978-3-319-38756-7_4.

[4] F. Tao *et al.*, "Digital twin-driven product design framework," *Int. J. Prod. Res.*, vol. 57, no. 12, pp. 3935–3953, 2019.

[5] Q. Lu *et al.*, "Developing a Digital Twin at Building and City Levels: A Case Study of West Cambridge Campus," *J. Manag. Eng.-ASCE*, vol. 36, no. 3, 2020.

[6] C. Zhang, G. Zhou, J. He, Z. Li, and W. Cheng, "A data-and knowledge-driven framework for digital twin manufacturing cell," *Procedia CIRP*, vol. 83, pp. 345–350, 2019.

[7] D. D'Amico *et al.*, "Conceptual framework of a digital twin to evaluate the degradation status of complex engineering systems," *Procedia CIRP*, vol. 86, pp. 61–67, 2019.

[8] I. Onaji, D. Tiwari, P. Soulatiantork, B. Song, and A. Tiwari, "Digital twin in manufacturing: conceptual framework and case studies," *Int. J. Comput. Integr. Manuf.*, pp. 1–28, 2022.

[9] G. S. Martinez, S. Sierla, T. Karhela, and V. Vyatkin, "Automatic generation of a simulation-based digital twin of an industrial process plant," in *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*, 2018, pp. 3084–3089.

[10] C. Zhang, W. Xu, J. Liu, Z. Liu, Z. Zhou, and D. T. Pham, "A reconfigurable modeling approach for digital twin-based manufacturing system," *Procedia Cirp*, vol. 83, pp. 118–125, 2019.

[11] M. K. Traoré, "Unifying Digital Twin framework: simulation-based proof-of-concept," *IFAC-Pap.*, vol. 54, no. 1, pp. 886–893, 2021.

[12] C. Nwogu, G. Lugaresi, A. Anagnostou, A. Matta, and S. J. Taylor, "Towards a Requirement-driven Digital Twin Architecture," *Procedia CIRP*, vol. 107, pp. 758–763, 2022.

[13] J. Zhao, H. Feng, Q. Chen, and B. G. de Soto, "Developing a conceptual framework for the application of digital twin technologies to revamp building operation and maintenance processes," *J. Build. Eng.*, vol. 49, p. 104028, 2022.

[14] F. Tao, B. Xiao, Q. Qi, J. Cheng, and P. Ji, "Digital twin modeling," *J. Manuf. Syst.*, vol. 64, pp. 372–389, Jul. 2022, doi: 10.1016/j.jmsy.2022.06.015.

[15] D. Piromalis and A. Kantaros, "Digital Twins in the Automotive Industry: The Road toward Physical-Digital Convergence," *Appl. Syst. Innov.*, vol. 5, no. 4, p. 65, Jul. 2022, doi: 10.3390/asi5040065.

[16] A. Martínez-Gutiérrez, J. Díez-González, R. Ferrero-Guillén, P. Verde, R. Álvarez, and H. Perez, "Digital Twin for Automatic Transportation in Industry 4.0," *Sensors*, vol. 21, no. 10, p. 3344, May 2021, doi: 10.3390/s21103344.

 ISBN: 978-1-68558-047-6

[17] Y. K. Liu, S. K. Ong, and A. Y. C. Nee, "State-of-the-art survey on digital twin implementations," *Adv. Manuf.*, vol. 10, no. 1, pp. 1–23, Mar. 2022, doi: 10.1007/S40436-021-00375-W/TABLES/8.

[18] K. Schade, M. Hübscher, F. zur Lage, J. Schulze, and J. Ringel, "Integrating Retail into an Urban Data Platform from a Stakeholder Perspective: Network Approaches in Leipzig (Germany)," *Sustainability*, vol. 14, no. 10, p. 5900, May 2022, doi: 10.3390/su14105900.

[19] A. Luthfi, M. Janssen, and J. Crompvoets, "Stakeholder Tensions in Decision-Making for Opening Government Data," in *Business Modeling and Software Design*, vol. 391, B. Shishkov, Ed. Cham: Springer International Publishing, 2020, pp. 331–340. doi: 10.1007/978-3-030-52306-0_23.

[20] V. Gitelman, "Exploring safety-related behaviours of e-cyclists on urban streets; an observational study," *Eur. Transp. Eur.*, no. 85, pp. 1–15, Dec. 2021, doi: 10.48295/ET.2021.85.2.

[21] J. Jiao, H. K. Lee, and S. J. Choi, "Impacts of COVID-19 on bike-sharing usages in Seoul, South Korea," *Cities*, vol. 130, p. 103849, Nov. 2022, doi: 10.1016/j.cities.2022.103849.

[22] T. Koska, "The Path to Sustainable Mobility Systems – 8 Theses on a digital mobility transition," p. 33.

[23] L. Aarikka-Stenroos, D. Chiaroni, J. Kaipainen, and A. Urbinati, "Companies' circular business models enabled by supply chain collaborations: An empirical-based framework, synthesis, and research agenda," *Ind. Mark. Manag.*, vol. 105, pp. 322–339, Aug. 2022, doi: 10.1016/j.indmarman.2022.06.015.

[24] L. Saari, J. Heilala, T. Heikkilä, J. Kääriäinen, A. Pulkkinen, and T. Rantala, *Digital product passport promotes sustainable manufacturing*. VTT Technical Research Centre of Finland, 2022.

[25] X. Chen and H. Jiang, "Detecting the Demand Changes of Bike Sharing: A Bayesian Hierarchical Approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 5, pp. 3969–3984, May 2022, doi: 10.1109/TITS.2020.3037791.

[26] C. H. Lee, J. W. Lee, and Y. J. Jung, "Practical method to improve usage efficiency of bike-sharing systems," in *ETRI Journal*, Apr. 2022, vol. 44, no. 2, pp. 244–259. doi: 10.4218/etrij.2021-0408.

[27] M. Bertoni and A. Bertoni, "Designing solutions with the product-service systems digital twin: What is now and what is next?," *Comput. Ind.*, vol. 138, Jun. 2022, doi: 10.1016/j.compind.2022.103629.

[28] W. Hu, T. Zhang, X. Deng, Z. Liu, and J. Tan, "Digital twin: a state-of-the-art review of its enabling technologies, applications and challenges," *J. Intell. Manuf. Spec. Equip.*, vol. 2, no. 1, pp. 1–34, Aug. 2021, doi: 10.1108/JIMSE-12-2020-010.

[29] M. Bertoni and A. Bertoni, "Designing solutions with the product-service systems digital twin: What is now and what is next?," *Comput. Ind.*, vol. 138, p. 103629, Jun. 2022, doi: 10.1016/j.compind.2022.103629.

[30] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, "Characterising the Digital Twin: A systematic literature review," *CIRP J. Manuf. Sci. Technol.*, vol. 29, pp. 36–52, May 2020, doi: 10.1016/j.cirpj.2020.02.002.

[31] D. Adamenko, S. Kunnen, R. Pluhnau, A. Loibl, and A. Nagarajah, "Review and comparison of the methods of designing the Digital Twin," *Procedia CIRP*, vol. 91, pp. 27–32, 2020, doi: 10.1016/j.procir.2020.02.146.

[32] A. Kantaros, D. Piromalis, G. Tsaramirsis, P. Papageorgas, and H. Tamimi, "3D Printing and Implementation of Digital Twins: Current Trends and Limitations," *Appl. Syst. Innov.*, vol. 5, no. 1, p. 7, Dec. 2021, doi: 10.3390/asi5010007.

[33] S. Lawrenz, M. Nippraschk, P. Wallat, A. Rausch, D. Goldmann, and A. Lohrengel, "Is it all about Information? The Role of the Information Gap between Stakeholders in the Context of the Circular Economy," *Procedia CIRP*, vol. 98, pp. 364–369, 2021, doi: 10.1016/j.procir.2021.01.118.

[34] C. Nwogu, G. Lugaresi, A. Anagnostou, A. Matta, and S. J. E. Taylor, "Towards a Requirement-driven Digital Twin Architecture," *Procedia CIRP*, vol. 107, pp. 758–763, 2022, doi: 10.1016/j.procir.2022.05.058.

[35] Q. Lu *et al.*, "Developing a Digital Twin at Building and City Levels: Case Study of West Cambridge Campus," *J. Manag. Eng.*, vol. 36, no. 3, p. 05020004, May 2020, doi: 10.1061/(ASCE)ME.1943-5479.0000763.

[36] R. Rasor, D. Göllner, R. Bernijazov, L. Kaiser, and R. Dumitrescu, "Towards collaborative life cycle specification of digital twins in manufacturing value chains," *Procedia CIRP*, vol. 98, pp. 229–234, 2021, doi: 10.1016/j.procir.2021.01.035.

[37] J. Friederich, D. P. Francis, S. Lazarova-Molnar, and N. Mohamed, "A framework for data-driven digital twins of smart manufacturing systems," *Comput. Ind.*, vol. 136, p. 103586, Apr. 2022, doi: 10.1016/j.compind.2021.103586.

[38] J. Protner, M. Pipan, H. Zupan, M. Resman, M. Simic, and N. Herakovic, "Edge Computing and Digital Twin Based Smart Manufacturing," *IFAC-Pap.*, vol. 54, no. 1, pp. 831–836, 2021, doi: 10.1016/j.ifacol.2021.08.098.

[39] M. Pregnolato *et al.*, "Towards Civil Engineering 4.0: Concept, workflow and application of Digital Twins for existing infrastructure," *Autom. Constr.*, vol. 141, p. 104421, Sep. 2022, doi: 10.1016/j.autcon.2022.104421.

[40] Y. Yu, D. M. Yazan, V. Junjan, and M.-E. Iacob, "Circular economy in the construction industry: A review of decision support tools based on Information & Communication Technologies," *J. Clean. Prod.*, vol. 349, p. 131335, May 2022, doi: 10.1016/j.jclepro.2022.131335.

[41] Benjamin Schleicha*, Marc-André Dittrichb, Till Clausmeyerc, Roy Damgraved, John Ahmet Erkoyuncue, Benjamin Haefnerf, Jos de Langed, and Denys Plakhotnikg, Wieben Scheidelh, Thorsten Wuest, "Shifting value stream patterns along the product lifecycle with digital twins." [retrieved: Mai, 2023] https://reader.elsevier.com/reader/sd/pii/S2212827120300639?token=4058F61DD11A9AB2896848BF7756F22597A54B0D6AD3B05F9AABF1DF8AA149180EF2E6521509C796DBDD1B53895F6BE9&originRegion=eu-west-1&originCreation=20220609142727

[42] K. Binder, "Ein drittes Standbein der Forschung neben Experiment und (analytischer) Theorie," *Phys. J.*, p. 6, 2004.

[43] C. Cimino, A. Leva, E. Negri, and M. Macchi, "An integrated simulation paradigm for lifecycle-covering maintenance in the Industry 4.0 context," *IFAC-Pap.*, vol. 53, no. 3, pp. 307–312, 2020, doi: 10.1016/j.ifacol.2020.11.049.

[44] I. Daniyan, R. Muvunzi, and K. Mpofu, "Artificial intelligence system for enhancing product's performance during its life cycle in a railcar industry," *Procedia CIRP*, vol. 98, pp. 482–487, 2021, doi: 10.1016/j.procir.2021.01.138.

[45] Y.-X. Zhang *et al.*, "Digital twin accelerating development of metallized film capacitor: Key issues, framework design and prospects," *Energy Rep.*, vol. 7, pp. 7704–7715, Nov. 2021, doi: 10.1016/j.egyr.2021.10.116.

[46] D. Romero, T. Wuest, R. Harik, and K.-D. Thoben, "Towards a Cyber-Physical PLM Environment: The Role of Digital Product Models, Intelligent Products, Digital Twins, Product Avatars and Digital Shadows," *IFAC-Pap.*, vol. 53, no. 2, pp. 10911–10916, 2020, doi: 10.1016/j.ifacol.2020.12.2829.

[47] Ostfalia University of Applied Sciences, 38302 Wolfenbüttel, Germany, L. Kintscher, S. Lawrenz, H. Poschmann, and P. Sharma, "Recycling 4.0 - Digitalization as a Key for the Advanced Circular Economy," *J. Commun.*, pp. 652–660, 2020, doi: 10.12720/jcm.15.9.652-660.

[48] S. Blömeke *et al.*, "Recycling 4.0: An Integrated Approach Towards an Advanced Circular Economy," in *Proceedings of the 7th International*

Conference on ICT for Sustainability, Bristol United Kingdom, Jun. 2020, pp. 66–76. doi: 10.1145/3401335.3401666.

[49] A. P. Barquet *et al.*, "Sustainable Product Service Systems – From Concept Creation to the Detailing of a Business Model for a Bicycle Sharing System in Berlin," *Procedia CIRP*, vol. 40, pp. 524–529, 2016, doi: 10.1016/j.procir.2016.01.127.

[50] C. Kiefer and F. Behrendt, "Smart e-bike monitoring system: real-time open source and open hardware GPS assistance and sensor data for electrically-assisted bicycles," *IET Intell. Transp. Syst.*, vol. 10, no. 2, pp. 79–88, Mar. 2016, doi: 10.1049/iet-its.2014.0251.

[51] T. Buchert *et al.*, "Design and Manufacturing of a Sustainable Pedelec," *Procedia CIRP*, vol. 29, pp. 579–584, 2015, doi: 10.1016/j.procir.2015.02.168.

 ISBN: 978-1-68558-047-6

Emergent Software Service Platform and its Application in a Smart Mobility Setting

Christoph Knieke, Eric Nyakam,
Andreas Rausch, Christian Schindler
Technische Universität Clausthal
Institute for Software and Systems Engineering
Clausthal-Zellerfeld, Germany
Email: {christoph.knieke, eric.douglas.nyakam.chiadjeu,
andreas.rausch, christian.schindler}@tu-clausthal.de

Christian Bartelt, Nils Wilken
Institute for Enterprise Systems
University of Mannheim
Mannheim, Germany
Email: {bartelt, wilken}
@es.uni-mannheim.de

Nikolaus Ziebura
Anaqor AG
Berlin, Germany
Email:
nikolaus.ziebura@anaqor.io

***Abstract*—The development dynamics of digital innovations for industry, business, and society are producing complex system conglomerates that can no longer be designed centrally and hierarchically in classic development processes. Instead, systems are evolving in DevOps processes in which heterogeneous actors act together on an open platform. Influencing and controlling such dynamically and autonomously changing system landscapes is currently a major challenge and a fundamental interest of service users and providers, as well as operators of the platform infrastructures. In this paper, we propose an architecture for such an emergent software service platform. A software platform that implements this architecture with the underlying engineering methodology is demonstrated by a smart parking lot scenario.**

***Index Terms*—Software Services; Service Composition; Self-adaptive Platform; Emergent Systems.**

I. Introduction

The runtime environment of current software systems, such as modern embedded systems or information systems, consists of a large number of complex software components that run on different hardware components in a distributed architecture. In a distributed system architecture, each of these components can use and provide different applications or services over a network connection. This environment is also often referred to as the *Internet Of Things (IoT)* [1]. One important feature of an IoT runtime environment is that the connected components can be from very different domains (e.g., social events, transportation, home automation, etc.).

From a technical point of view, in such an environment, a software platform, which enables software service providers to offer their services and enables customers to make use of these services, is required. We refer to such a software platform as a *Platform Ecosystem*. In this work, we focus on *software services* as these software components. A major challenge in the context of such Platform Ecosystems is the composition of the available software services so that they function together as a more complex, higher-value software component.

Due to the dynamic nature of an IoT environment, another major challenge for such Platform Ecosystems is to maintain their functionality also if runtime conditions suddenly change, which is often the case in IoT environments [2]. Recent self-adaptive software systems, which are able to adapt their architecture and behaviour automatically to changes in the environment to a certain extent, try to solve this challenge. Such self-adaptive systems can be designed from existing software services, as proposed in the DAiSI component model [3]. However, such self-adaptive systems still have to be designed manually to a large extent. Hence, we propose the concept of an *Emergent Software Service Platform*, which is able to design software services from the set of available software services completely automatically at runtime.

This vision includes, that such an Emergent Software Service Platform has to have at least the following capabilities:

1) It has to be able to elicit the current user requirements automatically at runtime.
2) It has to be able to automatically compose a software service, which meets the elicited user requirements, from the set of available software services at runtime.
3) It has to be able to execute an automatically composed software service at runtime and provide the result to the user of the Emergent Software Service Platform.

The paper is structured as follows: Section II gives a short overview on related work. In Section III, we introduce our platform architecture for emergent software service composition. We apply our software platform on a mobility use case, described in Section IV. Finally, Section V concludes.

II. Related Work

Influencing and controlling dynamically and autonomously changing system landscapes is currently a major challenge and a fundamental interest of service users and providers, as well as of platform infrastructure operators. For this reason, numerous platform and middleware technologies supporting controlled self-adaptation of dynamically adaptive systems have emerged in research and development over the last decade [4]–[8]. PORSCE II [9] is one of the first semantic composition systems for web services with the particularity that it takes advantage of semantic information to improve the planning, as well as the composition of software components. iServe [10], on the other hand, is not directly concerned with the composition of Web Services but describes a new and

 ISBN: 978-1-68558-047-6

open platform for publishing web services to better support their discovery and use.

However, although these system approaches already support both dynamic networking at runtime, self-adaptation, and openness, their reliable operation depends on the presence of a central logical entity (e.g., a platform operator and/or a standardization body). Such an actor is currently needed to standardize data, services, and processes in such a way that component providers can specify semantically compatible interfaces on their basis. Furthermore, strategies for functional and non-functional assurance of self-adaptive IT systems also require a central body to guarantee reliable cooperation among components. Current technology platforms centrally specify the configuration rules according to which system components can network with each other. However, the emergence of far-reaching emergent systems is severely restricted by such centrally anchored coordination mechanisms.

III. Emergent Software Service Platform

In this section, we describe a revised version of the architecture of a software service platform, which was introduced in [11], that has all the capabilities that were described above.

A. Definitions

First, we begin with defining the core concepts of an emergent software service platform in the context of an IoT runtime environment.

Definition 1 (Software Service Description): A software service description defines the required and provided interfaces of a software service instance.

Definition 2 (Software Service Instance): A software service instance is a software entity that implements the interfaces that are defined by the corresponding software service description. In addition, this software service entity is already deployed and ready for use.

Definition 3 (Process): A process is a composition of software service descriptions to describe the composed system behavior, which can be executed by an execution engine.

Definition 4 (Software Service Platform): A software service platform is a software platform that provides a library of software services (not software service instances) to potential users. Providers that host software service instances can register their service instances on the platform with their corresponding software service descriptions.

Definition 5 (Emergence): A software service platform is called emergent if it automatically and dynamically composes available software services to an executable software service in response to a trigger event (user requirement). The executable software service is not predefined at design time and cannot be anticipated by the individual components.

B. Architecture

The core idea of a dynamic adaptive IoT ecosystem is the reuse of software components and the ability to fulfill the expected system behavior by using emergent sequences of available software components to provide higher-value services. Compared to the previous work [11], as shown in Figure 1, the architecture additionally includes the *Domain* as an internal component of the platform. The emergent platform as a whole interacts with users to determine formal user requirements through interactions and monitoring (A). The formal user requirement is then passed on to the composition mechanism (B) to compose a sequence of Software Service Descriptions to fulfill the expected system behavior demanded by the formal user requirement. The Service Descriptions that are allowed to be used in the composition are registered in the Service Registry (D). The composed sequence of components is forwarded to the Execution Engine (C). The responsibility of this component is to call Service Instances that "use/implement" the given software components of the composed sequence, to incorporate necessary user feedback into the execution, and to return process results to the user. The Domain is a central part of the architecture, as it is the foundational vocabulary to express user requirements and the foundation for a semantic description of the software components.

Requirements Handler: The requirements handler component is responsible for the automatic elicitation and formalization of the current user requirements. This can be done following two different modes of interaction with the user. First, the user can explicitly state his/her user requirements in a request to the platform. However, the majority of users are not capable of expressing their requirements in a formalized format.The task of the requirements handler component is to convert the unformalized user requirements into a formalized format. Second, there is the option of implicit user requirements recognition. This means that the user does not proactively interact with the platform but the user requirements are extracted from a sequence of sensor measurements that the emergent software platform is able to record from the environment of the user.In this case, the task of the user requirements handler component is to extract the current user requirements from the sequence of sensor measurements.

Self-adaptive Composition Mechanism: The composition component handles the fulfillment of a user request as a planning problem. The user requirement is the goal of the planning problem. A sequence of software components is to be determined (the plan) to fulfill the expected behavior. As the goal and the available software components are semantically described in terms of concepts defined in the Domain a matching of (parts of the) user requirement to software components is possible without the need to explicitly describe compatibility of software components.

Execution Engine: The execution engine provides a browser-based flow editor to integrate external services into the platform via the concept of an interpretable flow. In addition, it provides a runtime environment with an interpreter to execute those flows. Each flow consists of a set of nodes, wired together to define the order of execution and the way data is transferred from node to node. With the concept of service-nodes, flows can be combined into a new, more complex, (meta-) flow. This is the base concept for the automatic

 ISBN: 978-1-68558-047-6

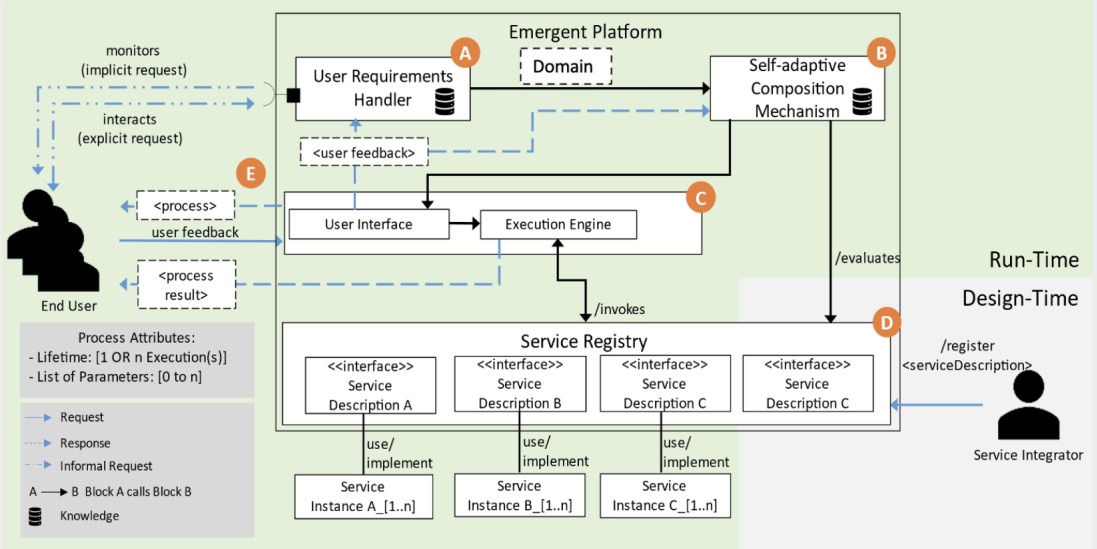

Fig. 1. Platform Architecture.

creation of a flow based on a composition result. For each service description in the service repository, there has to exist at least one flow tagged with the related action-reference. Now, the composition result can be mapped to an executable meta-flow by finding a matching flow for each action and wiring them together in the order given by the composition result.

Service Registry: The Service Registry contains the Service Descriptions to be offered by the platform. This is primarily a set of unique service descriptions that can be associated with different Service Instances. The Service Registry manages both the location and the style of interaction with the Service Instances. For example, available Service Descriptions can be queried by the Self-adaptive Composition Mechanism during composition. In addition, new Service Descriptions can be added to the Service Registry or the associated instances can be invoked by the execution engine.

IV. Application to a Smart Mobility Setting

We will demonstrate the introduced architecture in the application domain of a smart mobility IoT ecosystem, other domains are also feasible, as the introduced architecture is not limited to the demonstration domain. It consists of different services offered in the context of a parking lot that are able to be used on their own and give additional value when used in combination. We have defined web services that can have an impact in a real-world parking lot and trigger physical actions. In addition, we have defined services to place a reservation in a parking lot, charging of an electric vehicle, book a car wash, get tire pressure measurement, and get navigation directions to a specific spot in the parking lot. The services are described as OpenAPI-specified REST-endpoints and prototypical implementations are available. Based on the OpenAPI-specifications we have integrated the services as Service Descriptions in the Service Registry of the prototype.

In addition to implementing all the components shown in the platform architecture, we have also implemented software components, a formal domain, the semantic integration of the software components in the service registry, a GUI illustrating a parking lot in Figure 2, and an additional GUI part to accept explicit user requests, see Figure 3.

For each parking spot in Figure 2, icons indicate the available services (green) and the booked services (red). Figure 3 shows the Request Configurator, which an end user can interact with to express requests the platform should fulfill. Each icon on the UI represents a certain desire and each row refers to a certain request in the same parking lot (e.g., the selected checkboxes in the first row mean, that tire pressure measurement and charging station are desired, the parking spot should be booked and the navigation directions to the same parking spot should be given). The request will be formulated in a formal format, given in Figure 4. The request contains environment information, such as specified initial values for objects referenced in the request, an initial state, and the goal state. Based on this request, the Self-adaptive Composition Mechanism is able to compute a Service Description sequence, given in Figure 5. The message of the composition forwards

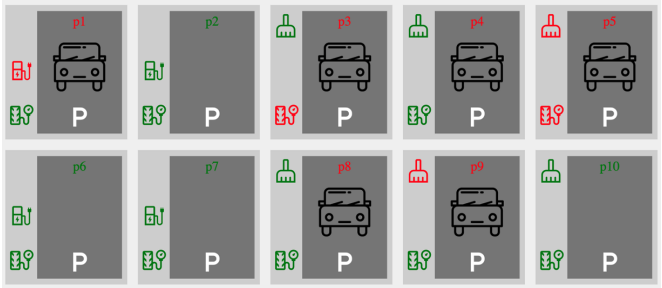

Fig. 2. Parking Lot UI.

 ISBN: 978-1-68558-047-6

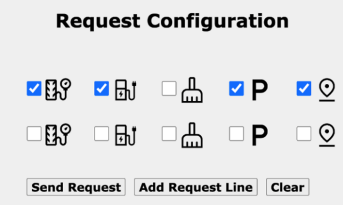

Fig. 3. User Request UI

the objects and their initial values and the service description sequence toward the executing part of the platform, which selects suitable Service Instances for the descriptions and calls them according to the composition plan.

V. Conclusion and Future Work

We presented a solution architecture for an Emergent Software-Service Platform and a smart mobility use case for which a prototypical implementation of the platform exists. First experiments with the existing prototype showed that it is able to automatically elicit the user requirements from an explicit request and is able to automatically compose and execute a software service that meets the recognized user requirements. Hence, these results show that it is viable to interpret the problem of autonomous self-adaptive software-service composition as a classical planning problem.

Nevertheless, the presented evaluation use case and the prototypical implementation of course still have some limitations. One limitation is that the considered use case is rather small, and only one possible form of interaction between a user and the software platform (i.e., via the Request Configurator) was evaluated. In future research, it would be interesting to extend this to other, possibly even more complex, forms of interaction like recognizing user requirements directly from natural language. Another limitation is that it is not possible to roll back the execution of the composed software service when the platform detects at execution time that the calculated software composition does not work. This might happen as it cannot be checked during the composition process whether the software services that are composed into a new service will be available at execution time. Also interesting to consider for future work is to implement and evaluate a more sophisticated feedback loop between the software service platform and the user(s). This would potentially not only increase the acceptance of the users to use such a platform but also enables the possibility for the platform to learn from more detailed user feedback to improve future software service compositions.

```
{"environment": [
 {"value":"", "type":"parkingid", "name":"p1"},
 {"value":"", "type":"operatorid", "name":"b1"},
 {"value":"", "type":"reservationnr", "name":"r1"},
 {"value":"", "type":"maxparkingtime", "name":"m1"},
 {"value":"", "type":"bookedservice", "name":"g1"}],
"init": [],
"goal": "(and (tirepressurecheck r1)
(bookeparking p1 r1 m1)
(navigation p1))"}
```

Fig. 4. Formalized User Request.

```
{"composition": [
 {"name":"get_parking-e-available",
    "params":["p1","b1"]},
 {"name":"post_book-parking-e", "params":
    ["p1", "r1", "b1", "m1"]},
 {"name":"book-tirepressurecheck",
    "params":["p1", "m1", "r1"]},
 {"name":"get_parking-navigation-parkingid",
    "params":["p1"]}],
"environment": [
 {"value":"", "type":"parkingid", "name":"p1"},
 {"value":"", "type":"operatorid", "name":"b1"},
 {"value":"", "type":"reservationnr", "name":"r1"},
 {"value":"", "type":"maxparkingtime", "name":"m1"},
 {"value":"", "type":"bookedservice", "name":"g1"}]}
```

Fig. 5. Formalized Composition Result.

Acknowledgment

This work was funded by the German Federal Ministry of Education and Research (Research Grant: 01IS18079, Project: BIoTope).

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE communications surveys & tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[2] A. Bröring, S. K. Datta, and C. Bonnet, "A categorization of discovery technologies for the internet of things," in *Proceedings of the 6th International Conference on the Internet of Things*, 2016, pp. 131–139.

[3] K. Rehfeldt, M. Schindler, B. Fischer, and A. Rausch, "A Component Model for Limited Resource Handling in Adaptive Systems," in *Proceedings of the 9th International Conference on Adaptive and Self-Adaptive Systems and Applications*. IARIA, 2017, pp. 37–42.

[4] H. Klus, A. Rausch, and D. Herrling, "DAiSI – Dynamic Adaptive System Infrastructure: Component Model and Decentralized Configuration Mechanism," *International Journal On Advances in Intelligent Systems*, vol. 7, no. 3, pp. 595–608, 2014.

[5] V. Cardellini *et al.*, "MOSES: A Platform for Experimenting with QoS-Driven Self-Adaptation Policies for Service Oriented Systems," in *Software Engineering for Self-Adaptive Systems III. Assurances*, R. de Lemos, D. Garlan, C. Ghezzi, and H. Giese, Eds. Cham: Springer International Publishing, 2017, pp. 409–433.

[6] S. Hallsteinsen *et al.*, "A development framework and methodology for self-adapting applications in ubiquitous computing environments," *Journal of Systems and Software*, vol. 85, no. 12, pp. 2840–2859, 2012.

[7] R. M. Andrade *et al.*, "Multifaceted infrastructure for self-adaptive IoT systems," *Information and Software Technology*, vol. 132, 2021.

[8] A. Burger, C. Cichiwskyj, S. Schmeißer, and G. Schiele, "The Elastic Internet of Things - A platform for self-integrating and self-adaptive IoT-systems with support for embedded adaptive hardware," *Future Generation Computer Systems*, vol. 113, pp. 607–619, 2020.

[9] O. Hatzi, D. Vrakas, N. Bassiliades, D. Anagnostopoulos, and I. Vlahavas, "The PORSCE II framework: Using AI planning for automated semantic web service composition," *The Knowledge Engineering Review*, vol. 28, no. 2, pp. 137–156, 2013.

[10] Á. Villalba, J. L. Pérez, D. Carrera, C. Pedrinaci, and L. Panziera, "servIoTicy and iServe: A scalable platform for mining the IoT," *Procedia Computer Science*, vol. 52, pp. 1022–1027, 2015.

[11] N. Wilken *et al.*, "Dynamic adaptive system composition driven by emergence in an iot based environment: Architecture and challenges," in *Proceedings of the 12th International Conference on Adaptive and Self-Adaptive Systems and Applications*, 2020, pp. 25–29.

 ISBN: 978-1-68558-047-6

Towards Transforming OpenAPI Specified Web Services into Planning Domain Definition Language Actions for Automatic Web Service Composition

Christian Schindler, Christoph Knieke, Andreas Rausch, Eric Douglas Nyakam Chiadjeu
Technische Universität Clausthal
Institute for Software and Systems Engineering
Clausthal-Zellerfeld, Germany
Email: {christian.schindler, christoph.knieke, andreas.rausch, eric.douglas.nyakam.chiadjeu}@tu-clausthal.de

***Abstract*—Today, more and more highly complex Internet Of Things (IoT) ecosystems are emerging that can no longer be centrally designed and controlled, but must self-adapt to new environments and user requirements. An approach to achieve this self-adaptation are so called emergent software service platforms that must be able to react continuously at runtime to changes in the runtime environment, such as changes in user requirements or the addition/removal of software services. Therefore, all software service compositions are created only at runtime of the platform on the basis of the automatically detected user requirements. In a previous work, we introduced our vision of a software architecture for such an emergent software service platform. A core part of such a platform is the service composition mechanism. In this paper, we present a set of rules that can be used to transform web services specified in OpenAPI into Planning Domain Definition Language (PDDL) actions. This allows web service compositions to be performed automatically with common PDDL solvers.**

***Index Terms*—Web Service Composition, AI Planning, PDDL, OpenAPI, Emergent Systems.**

I. Introduction

IoT ecosystems are complex system conglomerates of autonomous and interacting individual systems that are adaptive as a whole because they exhibit a special capacity for adaptation [1]. Analogous to their model in nature, such ecosystems can only be vital in the long term if a balance of needs and interests is achieved with regard to data, services and processes. The different life cycles of the individual systems, whose behavior and interactions change over time, are also decisive here. These changes (e.g., in data models / domain ontologies) cannot usually be centrally pre-planned, but result from independent processes and decisions within and outside the IoT ecosystem [2]. In addition, it is becoming increasingly apparent that industry-wide standardization efforts are too slow and inflexible when it comes to the speed of innovation in IoT components. This poses major challenges to traditional approaches and technologies for self-adaptive systems, as these systems rely on automated (semantic) interpretation of data and functions in system composition.

Many of the currently realized mechanisms for self-adaptation are based on maximizing the interface couplings of the participating components of an IoT ecosystem or similar demand-independent criteria (see, e.g., [3], [4]). This results in self-functioning systems at runtime, but in many cases these systems are not able to address the changing functional and non-functional needs of the users. In particular, the same system guarantees are not useful or necessary for all applications on a platform; rather, they are highly application- or user-specific. For this reason, the rules governing self-adaptation within IoT ecosystems in the future must be increasingly introduced into the platform by the actual service demanders. Intervention by the platform operator, on the other hand, should be limited to moderating the communication and composition of services and components.

An emergent software platform must be able to react continuously at runtime to changes in the runtime environment, such as the addition/removal of software services or changes in user requirements [5]. Therefore, all software service compositions emerge only at runtime of the platform on the basis of automatically detected user requirements. In this paper, this property is defined as "emergence" in the context of a software platform. More formally, we define emergence as follows: A software platform is called emergent if the platform is able to automatically and dynamically compose the available software services into a higher-value software service in response to a triggering event. The emergent behavior of the platform is not predefined at design time and cannot be predicted by individual software services [5].

In a previous paper [5], we already proposed a vision of an architecture as a so-called Emergent Software Service Platform. Using the architecture, a software engineer is able to develop a dynamic adaptive system which fulfills the emergent properties of an IoT-based environment. The main building blocks of the architecture are separated into run-time and design-time. At run-time, the building blocks are capable to determine an application for user requirements, which emerge from the IoT environment based on available services.

A challenging task in the Emergent Software Service Platform is the web service composition mechanism. A common way to describe web services is using OpenAPI [6]. The OpenAPI specification defines an open and vendor-neutral description format for Application Programming Interface

 ISBN: 978-1-68558-047-6

(API) services. In particular, OpenAPI can be used to describe, develop, test, and document REST-compliant APIs. In the paper at hand we present an approach to transform web services specified in OpenAPI into PDDL [7] actions. The advantage of this approach is that web service compositions can be performed with common PDDL solvers (e.g., ENHSP [8]).

The paper is organized as follows: Section II gives an overview on the related work. In Section III, we introduce our architecture vision for emergent service composition, as well as the PDDL language. Our approach is proposed in Section IV. Finally, Section V concludes and gives an outlook on future work.

II. Related Work

Extensive research has been done on the topic of semantic integration of service interfaces in the last two decades. In particular, work in the area of semantic integration of web services [9]–[11], as well as in the area of dynamically adaptive systems in the IoT and Industry 4.0 environment should be mentioned here [12]–[14]. Unfortunately, until today, the supply of components and platforms that allow semantics-based networking of industrial distributed service systems falls far short of the initially high goals and expectations or only covers the data layer [15] [16].

As emergence offers the opportunity to take advantage of the composability of individual software components, we will comment on previous works in the field of service composition dealing with the translation of web service composition problem into planning problem, or the integration of web services.

PORSCE II [17] is one of the first semantic composition systems for web services with the particularity that it takes advantage of semantic information to improve the planning as well as the composition of software components. The realization of this feature is based on an external domain ontology coupled with OWL-S [17]. In addition, PORSCE II also provides a mechanism to replace services that fail during composition.

iServe [18], on the other hand, is not directly concerned with the composition of Web Services, but describes a new and open platform for publishing web services to better support their discovery and use. It provides a common vocabulary that can be used to describe different services in such a way that they can be found automatically by machines and their functionality is independent of the form used for the original web service description. An immediate advantage is that through iServe a large set of different Web Services can be discovered and integrated for composition.

To sum up, iServe offers an open platform for publishing Web services, but does not provide composition of Web services. PORSCE II, on the other hand, addresses planning and composition but requires an additional description of Web services in OWL-S. In contrast, our approach leverages the widely used and standardized description of Web services in OpenAPI which can thus be used immediately with less effort.

III. Fundamentals

Dynamic adaptive software systems in an IoT-based environment can be designed from reusable software components [19], e.g., as proposed in the DAiSI component model [20], which describes the structure and the behavior of the system. Therefore, software components and interfaces are used to describe the building blocks of the architecture. The behavior is described on the basis of a contract based approach. The contracts are used by the system to check required and provided interfaces of software components for semantic compatibility at run-time.

A. Architecture

Figure 1 illustrates the architecture of our emergent platform as introduced in a previous work [5]. The architecture consists of five major parts (see letters A-E in Figure 1) which are either associated to the run-time or design-time part of the platform and will be briefly explained in the following:

Run-Time: The emergent platform as a whole interacts with users to determine formal user requirements through interactions and monitoring (A). The Domain is a central part of the architecture, as it is the foundational vocabulary to express user requirements and the foundation for a semantic description of the software components. The formal user requirement is passed on to the composition mechanism (B) to compose a sequence of Software Service Descriptions to fulfill the expected system behavior demanded by the formal user requirement. The composed sequence of components is forwarded to the Execution Engine (C). The responsibility of this component is to call Service Instances that "use/implement" the given software components of the composed sequence, to incorporate necessary user feedback into the execution, and to return process results to the user (E).

Design-Time: The Service Descriptions are developed and maintained by a service integrator at design-time. All those Service Descriptions that are allowed to be used in the composition are registered in the Service Registry (D).

B. Self-adaptive Composition Mechanism

The Self-adaptive Composition Mechanism has to compose services provided in the IoT ecosystem to achieve a given higher-level goal. This includes determining which services are needed and how these services can be invoked sequentially to achieve the goal. Computing an appropriate composition is formulated as a planning problem where the user's requirement is the goal and the service descriptions of the IoT ecosystem are possible actions that can be taken.

In the overall architecture, the Self-Adaptive Composition Mechanism has the following interfaces to other components: it has an interface from the User Requirements Handler, from which the target of an end user is provided. From the Service Registry, the component can query all currently available Service Descriptions and compute the composition based on them. Another interface is responsible for passing the calculated composition to the Execution Engine.

 ISBN: 978-1-68558-047-6

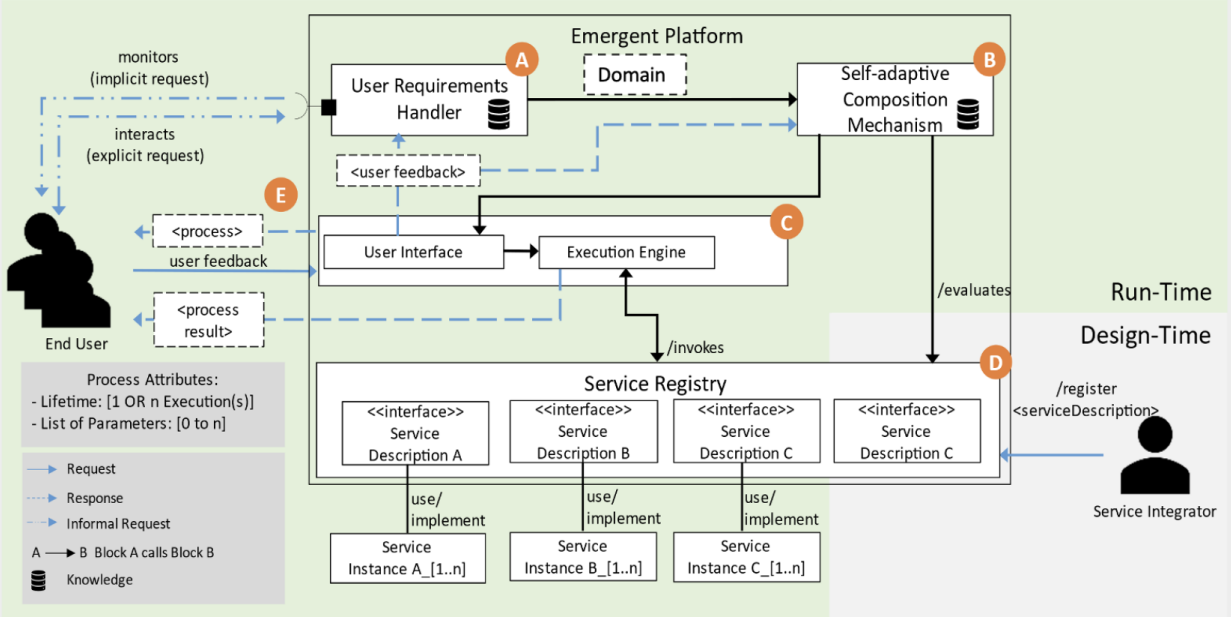

Fig. 1. Architecture of the emergent software service platform

C. Planning Domain Definition Language

PDDL is a standard encoding language for "classical" planning.

The components of PDDL files are:

- *Requirements:* Defining levels of abstraction in the language, temporal, probabilistic effects, etc.
- *Types:* Sets of the things of interest in the world
- *Objects:* Instances of types
- *Predicates:* Properties representing a state or condition in the world that can be true or false
- *Predicates:* Facts about objects that can be true or false
- *Initial state of the world*: Before starting the planning process
- *Goal:* Properties of the world true in goal states and achieved after the planning process
- *Actions/Operators:* Ways of changing states of the world and going from the initial state to goal states

Predicates are used to define the initial state of a planning problem, as well as to represent the goal state that the planner is trying to achieve. In addition, predicates can be used to describe properties of objects, relationships between objects, and other conditions that are relevant to the planning problem.

A planning task in PDDL is specified in two text files: A *domain file* for requirements, types, predicates and actions. A *problem file* for objects, initial state and goal specification.

PDDL plans from a state S_0 (initial state) to a S_z (target state). S_0 is our initial state which can be empty. S_z is the higher order requirement that can be expressed in terms of domain concepts. A is a set of actions available to the platform. The planner calculates a valid path (sequence of $a_i \in A$) from S_0 to S_z. The sequence of A_i can be called one after the other to reach the state S_z.

An important point in the computed sequence of A_i is that at the appropriate point in the execution sequence, all the information required at that time, which is needed as parameters, is available. These can either be given in the S_0 or obtained by executing an $a_j \in A$ with $(j < i)$.

IV. Rule-based Web Service Composition Approach

A. Motivating example

With a motivating example, we want to show the expressive power that the PDDL brings to the problem of web service composition, having the respective description of such planning tools needed to perform their planning. We want to give an example covering the Domain, available PDDL actions representing Service Descriptions, a user requirement needing to be composed of multiple available Service Descriptions, and finally the composed sequence of Service Descriptions a planing tool (such as the one we use) can compute.

The example reflects some core interactions of components of the architecture shown in Figure 1. Starting with the User Requirement Handler (A) giving the formal request (Figure 4) to the Self-Adaptive Composition Mechanism (B). The computed sequence of Service Description is shown in Figure 3. The example also shows the available Service Descriptions contained in the Service Registry (D) in Figure 3 and the Platforms Domain. The example does not cover the user interaction with the platform, neither the executing part (C). These aspects are addressed in a further paper [21] and shown together with the implementation of the platform and an application scenario. The example is in the application domain of a parking lot, offering to book a parking spot, and offering to get a car wash. The PDDL domain in this example (Figure 2) defines the types parkingid, reservationnr and bookedservice. The second part of the domain are the

 ISBN: 978-1-68558-047-6

```
(:types
    parkingid
    reservationnr
    bookedservice
)
(:predicates
    (parkplatz
        ?p - parkingid)
    (bookedparking
        ?p - parkingid
        ?r - reservationnr)
    (bookedcarwash
        ?p - parkingid
        ?g - bookedservice)
)
```

Fig. 2. PDDL Domain

predicates, which are defined similar to first order logic with an identifying name followed by the typed arguments.

Figure 2 contains the available actions given in the scope of the example, referring to types and predicates from the domain. Parameters are objects that will be passed to actions in the plan. The predicates given in an actions precondition need to evaluate to true for the passed objects to be able to be scheduled by the planning tool. The effects on the other hand are then enforced by the planner. The assignment of truth values to the predicates in the effects essentially allow the planning tool to finally reach the goal state of a planing problem. Figure 4 and 5 give a requirement and the respective calculated composition. The syntax of those is no plain PDDL, as this is the payload format of the platform itself. The requirement nevertheless describes available objects and their types and defines a goal to be reached. This goal's value is a PDDL conform String. An additional information that could be provided is an initial state (of the world) so the planner does not start from en empty state for the composition of actions.

The composition in Figure 5 also contains the objects originally received and in addition the composition. The composition is a sequence of actions, with valid assignments of available objects to the actions parameters, so that by following the sequence the original goal state is reached.

Coming back to the idea of treating web services as actions that can be executed in order, planning tools (e.g., ENHSP [8])) offer a good foundation on finding suitable action sequences to reach a given goal. In addition to the calculation of a sequence, the plan also contains objects that need to be passed to the actions in a given step of the plan to reach the goal state. To use such an approach for the web service composition part of this platform we want to emphasize the benefit of having the ability to derive a PDDL domain and actions reflecting the web services, so a composition can be calculated. As the manual translation can be cumbersome and error prune, we will give a set of rules on how a domain and actions similar to the ones in the example can be derived from given web service descriptions available in an OpenAPI specification.

```
(:action get_available-parkingspot
    :parameters (?p - parkingid)
    :precondition ()
    :effect (parkplatz ?p)
)
(:action post_book-parking
    :parameters (
        ?p - parkingid
        ?r - reservationnr)
    :precondition (parkplatz ?p)
    :effect (bookedparking ?p ?r)
)
(:action post_book-carwash
    :parameters (
        ?p - parkingid
        ?r - reservationnr
        ?l - bookedservice)
    :precondition (bookedparking ?p ?r)
    :effect (bookedcarwash ?p ?l)
)
```

Fig. 3. Available Service Descriptions

```
{ "environment": [
  {"type": "parkingid", "name": "p1"},
  {"type": "reservationnr", "name": "r1"},
  {"type": "bookedservice", "name": "g1"}
  ], "init": [],
  "goal": "(and (bookedparking p1 r1) (bookedcarwash
       p1 g1))"
}
```

Fig. 4. Requirenment

B. PDDL Actions

An action has three important sections that are required for planning. These are Parameters, Preconditions and Effects. Parameters are the objects that are passed to the action. These objects are used to check preconditions and to apply effects. The preconditions allow the planner to determine if an action can be scheduled and the effect allows the planner to determine if the action helps in getting towards the target state.

A web service, on the other hand, is not described in the same way. Here, the inputs and outputs are essentially described, along with interface details, such as Hypertext Transfer Protocol methods, content types, and response codes, which play a minor role for our approach.

```
{ "composition": [
  {"name": "get_available-parkingspot",
   "params": ["p1"]},
  {"name": "post_book-parking",
   "params": ["p1","r1"]},
  {"name": "post_book-carwash",
   "params": ["p1", "r1", "g1"]}
  ], "environment": [
    {"name": "p1", "type": "parkingid"},
    {"name": "r1", "type": "reservationnr"},
    {"name": "g1", "type": "bookedservice"}
]}
```

Fig. 5. Composed Sequence of Service Descriptions

The task now is to find suitable assignments of the inputs and outputs of the web services to the described sections of the PDDL actions, so that planning can be based on them.

C. Approach

The focus of this paper is on how to allow the semantical integration of available Web Services Descriptions into the platform's Service Registry to enable the platform's internal processing (calculation of compositions and incorporation into the platform domain to express user requirements).

We have defined multiple rules for the transformation of OpenAPI specifications into PDDL actions and the corresponding PDDL domain, to enable planning tools to use the actions and domain to plan according to provided problems. Figure 7 shows the algorithm on how to transform given paths of an OpenAPI specification into PDDL actions. We distinguish between primitives and more complex types (*object*) in the schema defined in the OpenAPI specification. Primitives in the sense of this approach are key-value pairs with a simple data type (e.g., string, number or boolean). Elements of type *object* are treated differently in our rules. To identify a given data type of an object in the algorithm, we introduce *isPrimitive()* and *isObjectType()*.

Rule Creation of an action: We create an action for each method of each path in the given specification. For each method of each path, we create an action (lines 4, 5, and 6 in the figure). The name of the action is a concatenation of the path and the currently handled method (e.g., *get_status*) for the path *status* and the method *GET*. Line 25 collects all the actions. To omit the risk of duplicates, it would be feasible to also incorporate a unique identifier, which is not part of this pseudo code, but would be attached in line 6.

Rule Precondition collection: To schedule an action into a plan, it is necessary to gather the preconditions required for the action to be executed successfully. To collect the preconditions, we process the requestBody of the OpenAPI specification. The idea behind this is that the requestBody also needs to be provided to the web service to be used, meaning the information needs to be present prior to the actual call. In the algorithm (Figure 7) this is shown in lines 8 to 15. The *objects()* in the algorithm collects and returns (recursive) all objects contained in the context of the calling element of the OpenAPI specification. Starting in line 8 we iterate over all objects and check if they are of type *object* (line 9). If this is the case a new precondition is found and added (line 10). In the next step all the child objects get processed, to find the precondition's parameters. To find them we again process all the contained objects recursive (line 11) and check if they are primitive (line 12) and add the parameter to the precondition in line 14. For simplification, we do not distinguish different contentTypes that can be offered by a requestBody (we only handle one).

Rule Effect collection: Similar to the collection of preconditions, we collect the effects, but by looking at the response specified in the OpenAPI specification. We focus on the good case responses (e.g., 200 or 201) which is not mentioned specifically in the algorithm. The underlying idea is that the response reflects what the web service has done. This is reflected for example by retrieved objects from the backend. The function block in the algorithm is line 17 to 23 and following the same logic than the collection of preconditions, with the mentioned difference of interacting over the response instead of the request body. A limitation of this transformation approach is if something happens while executing the web service which is not reflected by the response, this is not automatically covered by the actions. It is possible to extend the actions if more knowledge about the functionality of the web service is available (e.g., by experts) having a deeper understanding of the service and the correlation of its input and output.

Rule Parameter collection: The third building block of PDDL actions are the parameters. In comparison to the preconditions and effects, which are predicates, the parameters are typed variables, with types from the PDDL domain. There is a relationship between the parameters (of the action) and the predicates used in the preconditions and effects. In the algorithm the preconditions are simultaneously collected with the primitive child objects of preconditions (line 13) and the primitive child objects of effects (line 22).

Rule Creation of the PDDL Domain: The second algorithm given in Figure 8 describes how to derive the PDDL domain from OpenAPI specification. We iterate over all objects of the requestBody and the response, but this time we do not distinguish where the objects originate, as this does not matter for the domain. Primitive objects are transformed into domain types and objects of type *object* with their parameters are transformed into predicates in the domain. Iterating over the objects is done starting in line 7. Primitive types are identified in line 14 and added to the domain types in line 15. *Objects* are identified in line 8. Predicates get created in line 9. Lines 10, 11, and 12 collect the primitive parameters belonging to the domain predicate. This is the same logic, which is applied to collecting the preconditions and effects (compare Figure 5, lines 11, 12 and, 14). Lines 13 and 15 add the predicates and types to the domain.

D. Example transformation of an OpenAPI specified service

The following section gives a concrete example of how an OpenAPI-specified web service is transformed into a PDDL action and the corresponding domain. Figure 2 lists the OpenAPI specification (left-hand side) with the resulting PDDL action and domain (right-hand side). The highlighted parts visualize the corresponding parts in both formats, with the applied rules assigned. The grey box indicates a legend on the highlighting to the specific rules applied. We have one path with one method (*post*) transformed into a single action (orange box highlighting). The preconditions of the action are highlighted in green. On the right side, we see the resulting precondition *parkingspot* with its parameter *p*. On the left-hand side, we see that this is derived from the schema of the requestBody (and the according schema elements that are referred to by the requestBody). The same applies to the effect

 ISBN: 978-1-68558-047-6

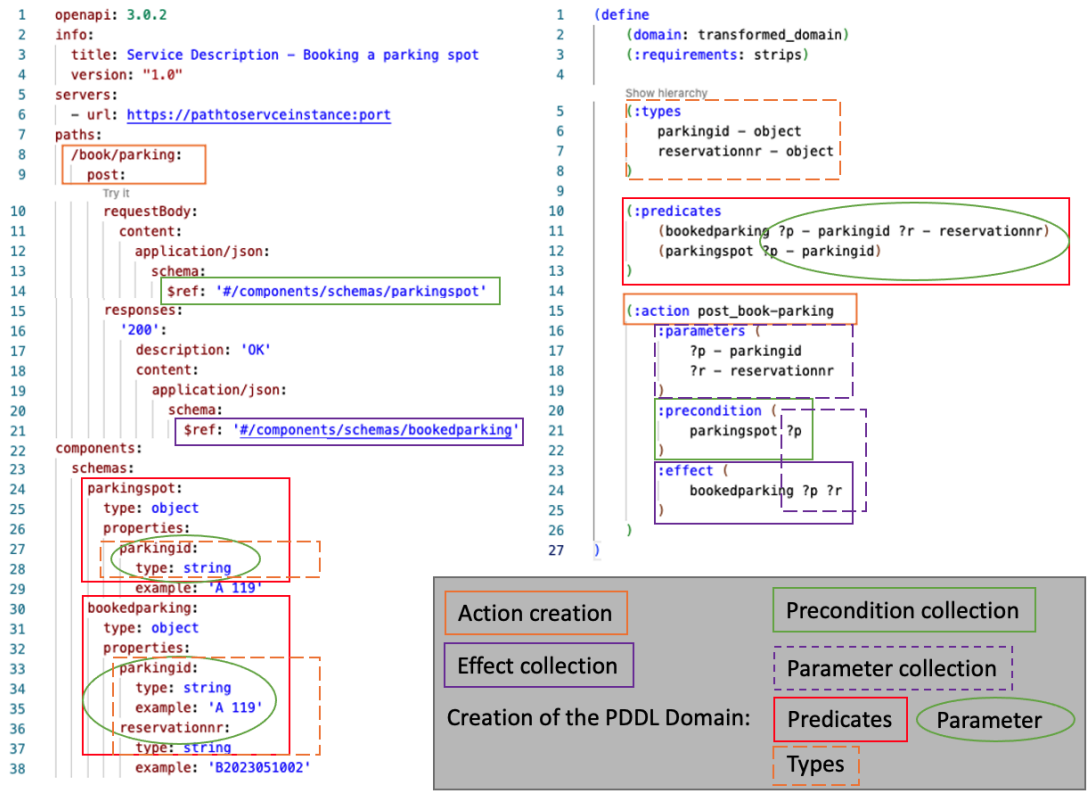

Fig. 6. Example REST API to PDDL

```
oas: OpenAPI
ps: PDDL

for path in oas.paths:
 for m in path.methods:
  action <- path.m + "_" + path.name.toLowecase()
  // Preconditions
  for object in m.requestBody.content.schema.objects():
   if object.isTypeObject():
    precondition.name <- object.name
    for childObject in object.objects():
     if childObject.isPrimitive():
      action.parameters <- childObject
      precondition.parameters <- childObject
    action.preconditions <- precondition
  // Effects
  for object in m.response.content.schema.objects():
   if object.isTypeObject():
    effect.name <- object.name
    for childObject in object.objects():
     if childObject.isPrimitive():
      action.parameters <- childObject
      effect.parameters <- childObject
    action.effects <- effect
  ps.actions <- action
```

Fig. 7. Rule Action Preconditions

highlighted in purple. In the upper part of the right-hand side, we see the definition of the domain types (orange dashed highlighting), the predicates (red highlighting), and their parameters (green ellipse) with their respective counterparts in the schema to which requestBody and the response refer.

```
oas: OpenAPI
ps: PDDL

for path in oas.paths:
 for m in path.methods:
  for context in { m.requestBody, m.response}:
   for object in context.content.schema.objects():
    if object.isTypeObject():
     predicate <- object.name.toLowercase()
     for childObject in object.objects():
      if childObject.isPrimitive():
       predicate.parameters <- childObject
     ps.domain.predicates <- predicate
    else if object.isPrimitiveType():
     ps.domain.types <- object.name.toLowercase()
```

Fig. 8. Rule PDDL Domain

V. CONCLUSION AND FUTURE WORK

In this paper, we have shown how PDDL can be used to compose web services to fulfill a more complex user request than any of the given services are capable of satisfying in its own way. We have given an example on how such a request for a given domain and given actions can look like. We have also described the differences in an OpenAPI specified web service and an action specified in PDDL. We have defined a set of transformation rules and described the pseudo code.

Nevertheless, the presented approach still has some limitations. As the OpenAPI specification does not explicitly describe coherence between the objects in the requestBody and

 ISBN: 978-1-68558-047-6

the response, the usage of given parameters in an action, and their occurrence in preconditions and effects massively affect the semantic meaning of an action. A possible quality gate is that an expert can ensure the quality and the correct semantics, editing the parameters, used parameters in preconditions and effects, if this collected information is not correct. A second limitation is the focus on specific response codes such as 200 or 201, as this is often the expected good case of such a web service. Other status codes in the range of 400 or 500 are not taken into account. This could easily be incorporated by also iterating over all given status codes per response, creating actions for each.

As future work, we want to investigate how the service integrator can be supported by the rules and the transformation (compare Figure 1, bottom right). Taking into account the stated limitations of the current approach, it is worth noting that a service integrator can offer supplementary knowledge beyond what is included in the OpenAPI specification.

ACKNOWLEDGMENT

This work was funded by the German Federal Ministry of Education and Research (Research Grant: 01IS18079, Project: BIoTope).

REFERENCES

[1] P. Pradeep, S. Krishnamoorthy, and A. V. Vasilakos, "A holistic approach to a context-aware IoT ecosystem with Adaptive Ubiquitous Middleware," *Pervasive and Mobile Computing*, vol. 72, 2021.

[2] M. Zdravković *et al.*, "Domain framework for implementation of open IoT ecosystems," *International Journal of Production Research*, vol. 56, no. 7, pp. 2552–2569, 2018.

[3] R. Rouvoy *et al.*, "Music: Middleware support for self-adaptation in ubiquitous and service-oriented environments," *Software engineering for self-adaptive systems*, pp. 164–182, 2009.

[4] I. Corredor, J. F. Martínez, M. S. Familiar, and L. López, "Knowledge-aware and service-oriented middleware for deploying pervasive services," *Journal of Network and Computer Applications*, vol. 35, no. 2, pp. 562–576, 2012.

[5] N. Wilken *et al.*, "Dynamic adaptive system composition driven by emergence in an iot based environment: Architecture and challenges," in *Proceedings of the 12th International Conference on Adaptive and Self-Adaptive Systems and Applications*. IARIA Press, 2020, pp. 25–29.

[6] D. Sferruzza, J. Rocheteau, C. Attiogbé, and A. Lanoix, "Extending openapi 3.0 to build web services from their specification," in *International Conference on Web Information Systems and Technologies*, 2018.

[7] D. M. McDermott, "The 1998 AI planning systems competition," *AI magazine*, vol. 21, pp. 35–35, 2000.

[8] E. Scala, P. Haslum, S. Thiébaux, and M. Ramirez, "Interval-based relaxation for general numeric planning," in *ECAI 2016*. IOS Press, 2016, pp. 655–663.

[9] N. F. Noy, "Semantic integration: a survey of ontology-based approaches," *ACM Sigmod Record*, vol. 33, no. 4, pp. 65–70, 2004.

[10] H. Nacer and D. Aissani, "Semantic web services: Standards, applications, challenges and solutions," *Journal of Network and Computer Applications*, vol. 44, pp. 134–151, 2014.

[11] M. Klusch, P. Kapahnke, S. Schulte, F. Lecue, and A. Bernstein, "Semantic web service search: a brief survey," *KI-Künstliche Intelligenz*, vol. 30, pp. 139–147, 2016.

[12] M. D'Angelo, M. Caporuscio, and A. Napolitano, "Model-driven engineering of decentralized control in cyber-physical systems," in *2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS* W)*. IEEE, 2017, pp. 7–12.

[13] M. U. Iftikhar, G. S. Ramachandran, P. Bollansée, D. Weyns, and D. Hughes, "Deltaiot: A self-adaptive internet of things exemplar," in *2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, 2017, pp. 76–82.

[14] I. Crnkovic, I. Malavolta, H. Muccini, and M. Sharaf, "On the use of component-based principles and practices for architecting cyber-physical systems," in *2016 19th International ACM SIGSOFT Symposium on Component-Based Software Engineering (CBSE)*. IEEE, 2016, pp. 23–32.

[15] V. Bauer and A. Vetro', "Comparing reuse practices in two large software-producing companies," *Journal of Systems and Software*, vol. 117, pp. 545–582, 2016.

[16] P. Bonte *et al.*, "The massif platform: a modular and semantic platform for the development of flexible iot services," *Knowledge and Information Systems*, vol. 51, pp. 89–126, 2017.

[17] O. Hatzi, D. Vrakas, N. Bassiliades, D. Anagnostopoulos, and I. Vlahavas, "The PORSCE II framework: Using AI planning for automated semantic web service composition," *The Knowledge Engineering Review*, vol. 28, no. 2, pp. 137–156, 2013.

[18] C. Pedrinaci *et al.*, "iserve: a linked services publishing platform," in *CEUR workshop proceedings*, vol. 596, 2010, pp. 2–13.

[19] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE communications surveys & tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[20] K. Rehfeldt, M. Schindler, B. Fischer, and A. Rausch, "A component model for limited resource handling in adaptive systems," in *ADAPTIVE 2017: The Ninth International Conference on Adaptive and Self-Adaptive Systems and Applications*. IARIA Press, 2017, pp. 37–42.

[21] C. Knieke *et al.*, "Emergent Software Service Platform and its Application in a Smart Mobility Setting," in *15th International Conference on Adaptive and Self-Adaptive Systems and Applications*, 2023, *to appear*.

 ISBN: 978-1-68558-047-6

Towards Self-Adaptive User Interfaces by Holding Posture Recognition for Smartphones

Rene Hörschinger, Marc Kurz, Erik Sonnleitner
Department for Smart and Interconnected Living (SAIL)
University of Applied Sciences Upper Austria
4232 Hagenberg, Austria
email:{firstname.lastname}@fh-hagenberg.at

***Abstract*—People interact with their smartphones in many different ways depending on the phone size, the application itself, or simply whether the user is left or right-handed. From the developer side of view, it can be beneficial to know how the user operates the phone in order to adjust and adapt the user interface accordingly. Therefore, this paper proposes a model that can predict the holding posture of a smartphone at runtime. The model gets the smartphone IMU data on each interaction with the display as well as the location where the user pressed onto the display as input and outputs the most likely holding posture. By using each interaction individually, the model is able to predict which hand is holding the phone at runtime with an accuracy of 89.3%.**

***Index Terms*—posture recognition, self-adaptiveness, smartphone sensing**

I. Introduction

People interact with their mobile devices in many different ways. This makes designing user interfaces challenging as the interface will only suit one type of group while others may struggle because of the way they hold their smartphone. If the smartphone could be aware of how the user currently interacts with the device, user interfaces and the general handling of the mobile device could autonomously adapt to this specific situation enabling self-adaptiveness of the smartphone on a real-time basis.

Steven Hoober conducted a survey in 2013 by observing 1,333 people in public places (streets, airports, bus stops, cafes, trains, busses) and analyzing how they interact with their smartphones [1]. People nowadays use their mobile devices in many different scenarios for example while walking, riding a bus, or standing still, and therefore they adapt their grasp on the smartphone accordingly. After his observation Hoober found out that 49% of the people hold their phone in one hand, 36% hold it cradled (one hand holding the phone, one hand interacting with the touchscreen either with the thumb or the index finger), and 15% use it two-handed. These are the numbers of people actively interacting with the mobile device, while we see in Figure 1 the distribution of all interactions including 22% accounting for voice calls and 18.9% listening to music or watching videos.

Looking at the one-handed interactions, interestingly only 67% of the people use their right hand while 33% use their left hand which does not correlate with the number of left-handed people in the general population of about 10% [2].

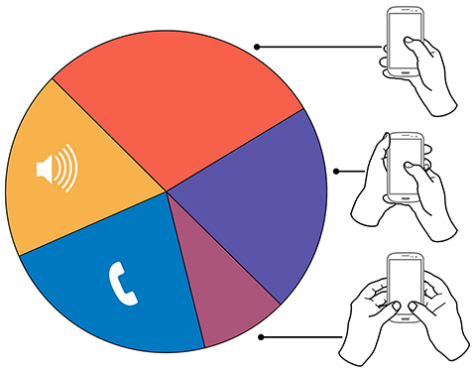

Fig. 1. General distribution of active and passive smartphone interactions [1].

This observation shows that right-handed people are likely to operate their mobile devices from time to time to free up their dominant hand for other tasks. Taking this into consideration, UI designs should not only be focused on right-handed people as nearly 1/3 of the one-handed interactions are done by the left hand.

After studying the one-handed interactions, it can also be seen that there are two ways of holding a phone with one hand:

- Thumb on the display, all 4 other fingers on the side
- Thumb on the display, pinkie on the bottom, the other 3 fingers on the side

Those differences result in different screen coverage, as with the pinkie on the bottom it is harder to reach the top left corner, while the other holding posture makes it tough to reach the bottom right corner which we can see in Figure 2.

Looking at the two-handed interactions with both thumbs it can be seen that people only use this method for gaming, video consumption, and keyboard typing.

Smartphones nowadays tend to get bigger and bigger:

- iPhone 5s, 2013, 4 inch display [3]
- iPhone X, 2017, 5.8 inch display [4]
- iPhone 13 Pro Max, 2022, 6.7 inch display [5]

That makes one-handed interactions with the mobile device harder. Apple for example offers a one-handed mode with a swipe down from the bottom of the display to shift the whole UI downwards. Otherwise, the classic return button on the top

 ISBN: 978-1-68558-047-6

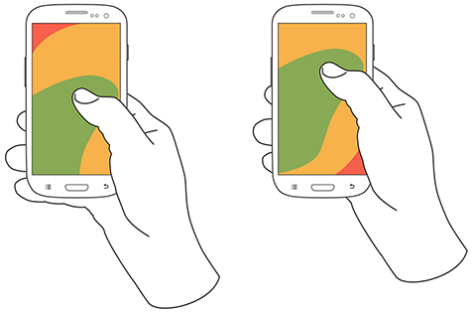

Fig. 2. Thumb coverage of one-handed interaction with the right thumb but different holding postures [1].

left of the screen would be nearly impossible to reach for a user with an average size hand on an iPhone 13 Pro Max. But this practice needs an active interaction of the user which is sub-optimal.

In this paper, we propose a way to detect how the user interacts with the smartphone at runtime in the background so the UI designer can take advantage of that information when designing the interface. A practical example would be the zoom operation in a maps application: To zoom in or out the standard way is to swipe two fingers apart or together which tends to be quite complicated with a big smartphone operated with only one hand. The zoom in that case could be adjusted to work with a long press on the touchscreen and then swiping up- and downwards while still pressing on the screen to zoom in and out. That interaction can easily be done with one hand. Additionally, knowing how the user currently interacts with his/her smartphone, would open possibilities in terms of self-adapting the UI mode of the device in an automated way.

To achieve this recognition, a machine learning (ML) model is trained with the sensor data from the inertial measurement unit (IMU) of a smartphone. The dataset used for the training was recorded with different participants of different sex and age while interacting with the smartphone.

The rest of the paper is structured as follows: Section II gives an overview of related work. Section III describes the methodology applied in our work. Section IV provides the evaluation containing (i) feature engineering, and (ii) model selection. Section V summarizes the achieved results. The final Section VI concludes the paper with a discussion and an outlook on future work.

II. Related work

Detecting the holding position of a mobile device is an active field of study with many different approaches. Wimmer et al. [6] utilize capacitive sensors in their prototype ”HandSense” to determine which hand is holding the device with an accuracy of 80% and 6 different holding positions. Another approach from Hinckley et al. [7] takes advantage of a self-capacitance touchscreen to detect the grip on the smartphone (one-handed and two-handed) as well as to recognize multiple fingers hovering over the touchscreen.

Goel et al. [8] present the prototype ”GripSense” for detecting holding patterns by combining built-in inertial sensors as well as the vibration motor of the mobile device. After a sequence of touchscreen interactions (tapping, swiping) the model reaches 84.4% accuracy at detecting left-, right- and two-handed interactions. Goel et al. utilize ”GripSense” for their next model ”ContextType” [9] and add additional posture-specific touch pattern information to detect the hand posture after each tap with an accuracy of 89.7%.

Park et al. [10] not only use gyroscope and accelerometer data for their Support Vector Machine (SVM) but also touchscreen data like the coordinates of interaction and the size. After training and testing on 6 participants, they reach an accuracy of 87.7% and 92.4% for 5 and 4 hand postures respectively. Löchtefeld et al. [11] have a similar approach to Park et al. but they also include the device orientation and focus on the detection of the holding posture during device unlocking. With a k-nearest neighbor model and Dynamic Time Warping (DTW), they achieve 98.51% accuracy. A comparable model to Löchtefeld et al. comes from Avery et al. [12]. They detect the holding position of the smartphone prior to the first touchscreen interaction with the use of the built-in orientation sensors and DTW with an accuracy of 83.6%.

Summing up these approaches, some of them need external sensors like Wimmer et al., some of them only focus on grabbing the phone and the unlocking process like Löchtefeld et al. and Park et al., and others detect the holding posture during actual user interaction like ”GripSense” from Park et al. but require a certain amount of interactions in order to work properly.

In contrast to the approaches mentioned above, the model presented in this paper does not need any external sensors, does not focus only on the unlocking process of a smartphone, and furthermore detects holding posture changes during the usage of an application at runtime.

III. Approach

Nowadays a lot of mobile applications do not support landscape mode, especially social media apps, for example, Instagram, Twitter, Snapchat, and Facebook. With that in mind and with the information retrieved from the survey of Steven Hoober in Section I we decided to differentiate between 4 different holding postures, seen in Figure 3: Right single-handed, left single-handed, cradle with the left index finger, cradle with the right index finger.

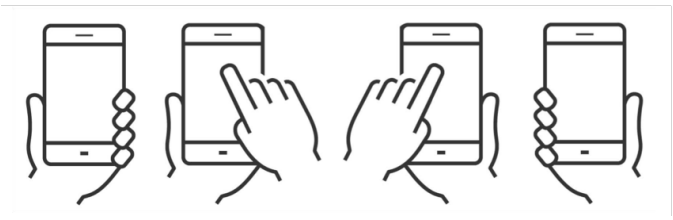

Fig. 3. Different holding postures of a smartphone [12].

 ISBN: 978-1-68558-047-6

The idea is to gather IMU data before and after user interaction with the touchscreen and identify the holding posture only by that event. IMU data is available on every new smartphone which means the model is platform-independent and can be used on all iOS and Android devices.

A. Data recording

The setup for the data recording was as follows: 9 participants (7 male, 2 female) between 23 and 58 were asked to perform multiple tap actions with different holding postures within an iOS app on predefined locations which were: left, middle and right each at the top, middle, and bottom of the touchscreen, as seen in Figure 4.

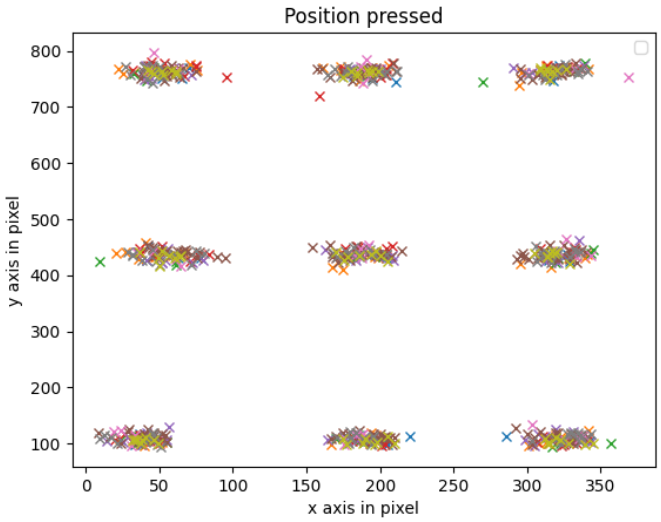

Fig. 4. Position in x and y pixels where the participants tapped onto the touchscreen during data recording.

The iOS app fetched IMU data at 100 Hz in the background and saved a time series of 0.5 seconds before and after each tap action. Furthermore, the orientation (yaw, pitch, roll) of the device got also tracked at 100 Hz, as well as the location of the tap interaction. In total, the following data got recording after each interaction:

- 3 axis accelerometer @ 100 Hz: 300 data points
- 3 axis gyroscope @ 100 Hz: 300 data points
- 3 axis magnetometer @ 100 Hz: 300 data points
- roll, pitch, yaw @ 100 Hz: 300 data points
- x-axis and y-axis (in pixels): 2 data points

The participants pressed multiple times on each defined location which resulted in a dataset of 723 samples with each having 1,202 features. The number of samples per holding posture was nearly equally distributed as seen in Figure 5.

B. Data preprocessing

After the dataset was complete, the approach was followed as shown in the flowchart in Figure 6. On the training set, multiple feature extraction methods were performed:

- **Statistical features**: The time series of each axis were used to extract the following features: mean, median, maximum, minimum, standard deviation, skewness, kurtosis, interquartile range, mad, integrate along the given axis using the composite trapezoidal rule, range, root mean square, variance and the number of peaks. Furthermore, each sensor was used to extract the mean standard deviation and the mean of variance.
- **Frequency features with Fast Fourier Transform (FFT)**: FFT calculation from all axis was done to calculate the following features: mean of FFT magnitude, the standard deviation of FFT magnitude, mean of FFT angle, the standard deviation of FFT angle.
- **Magnitude features**: For each sample the magnitude for each axis was calculated to derive some more statistical features: mean of magnitude, maximum magnitude, minimum magnitude, variance of magnitude, standard

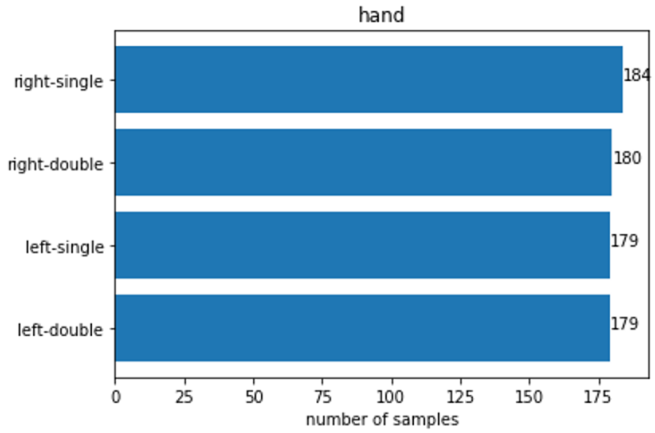

Fig. 5. Samples per posture in the dataset.

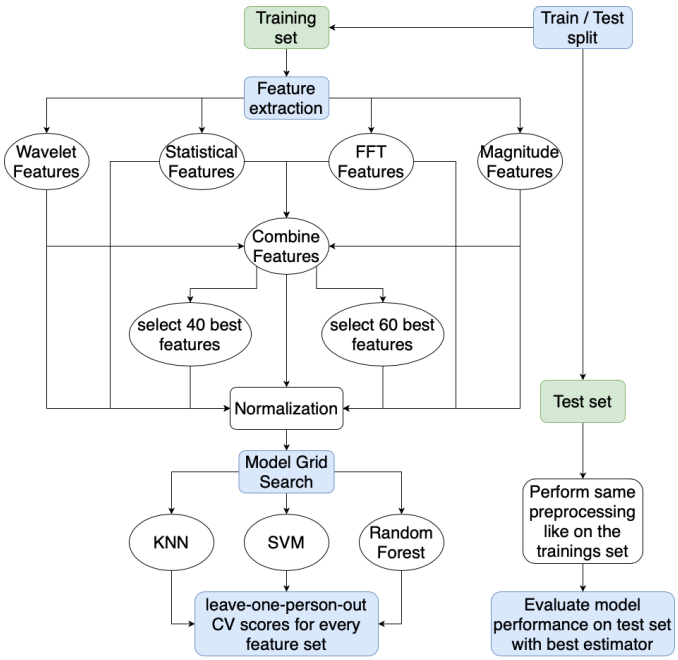

Fig. 6. Flowchart from data to ML model.

deviation of magnitude, and number of peaks.
- **Wavelet features**: To combine time and frequency domain the db3 wavelet was used to get the number of wavelet levels and then the approximated coefficients instead of the detailed ones were used.

IV. Evaluation

In order to find the best model suited for this task, several feature extraction methods were used in combination with various machine learning models.

A. Feature Engineering

First of all, the raw time series data was used as a feature set to find the baseline of the model and see how different models would perform without any preprocessing and feature engineering except for scaling and filtering. The extracted features mentioned in Section III-B were used individually for training and also combined together to a big feature set that accounted for 370 different features. Furthermore, not only each sensor but also each sensor axis was used individually for a feature set, as some axis might perform better and have more relevance in solving the problem than others. For example, the wavelet feature set actually consisted of several different feature sets: only wavelets of accelerometer/gyroscope/magnetometer/motion for only x/y/z axis, all wavelets of each sensor individually, all wavelets of the IMU combined, all wavelets of gyroscope and accelerometer combined, all wavelets of all sensors combined and 3 of these feature sets where the correlation between features above 0.8 and under -0.8 has been removed. That way the wavelet feature set actually consisted of 22 different sets. With a scoring function (ANOVA F-value), the best 40, 80, 120, and 160 features were selected to find out if these new feature sets can increase the accuracy further. In total, this setup accounted for 92 different feature sets:

- raw time series (scaled)
- raw time series (scaled + filtered)
- only statistical features
- only FFT features
- only magnitude features
- only wavelet features
- all features combined
- best 40/80/120/16 features with the scoring function
- all features combined but correlation above 0.8 and under -0.8 removed (114 features)

B. Model selection with gridsearch

The models k-nearest neighbor (KNN), random forest, and support vector machine (SVM) were used for a hyperparameter grid search in Python using the models of scikit-learn. KNN was trained with 96 different hyperparameter variations, random forest with 576, and SVM with 56. Each model was trained with each feature set and with 5-fold cross-validation as well as leave-one-group-out cross-validation. Because the number of samples per participant was not equally distributed, the leave-one-group-out cross might introduce some bias, and since the accuracy numbers were only slightly different at the end, all following numbers from the training refer to 5-fold cross-validation.

The hyperparameters of each model were chosen as follows.

1) KNN:
- **neighbors**: in the range of 3 to 30 excluding multiples of 4 (because of 4 different holding postures)
- **weights**: uniform and distance
- **metric**: euclidean and manhattan

2) Random Forest:
- **bootstrap**: true
- **max depth**: 10, 20, 30, 40
- **max features**: 2, 3, 4, 5
- **min samples leaf**: 2, 4, 5, 8
- **min samples split**: 4, 8, 12
- **n estimators**: 100, 200, 300

3) SVM:
- **kernel**: rbf and linear
- **gamma**: only for rbf kernel: 3^{-8}, 3^{-7}, 3^{-6}, 3^{-5}, 3^{-4}, 3^{-3}, 3^{-2}
- **C**: 3^{0}, 3^{1}, 3^{2}, 3^{3}, 3^{4}, 3^{5}, 3^{6}, 3^{7}, 3^{8}

V. Results

Figure 7 shows different evaluation metrics for different feature sets and models. The first two columns are the best models with the raw time series feature set (scaled, scaled, and filtered) to find the baseline of the model. The third column is the best overall model where all features were tested, and the last column is the best model which only used feature sets with removed correlation. Overall, wavelet features performed extremely well on their own while the magnitude features achieved a low accuracy.

Model Feature set Sensor	Random Forest scaled timeseries IMU	Random Forest filtered timeseries IMU	SVM combined features all	Random Forest wavelets corr < 0.9 all
left-single precision, recall, f1	0.908, 0.937, 0.922	0.865, 0.938, 0.9	0.906, 0.96, 0.932	0.885, 0.885, 0.885
right-single precision, recall, f1	0.865, 0.882, 0.874	0.889, 0.842, 0.865	0.915, 0.796, 0.851	0.85, 0.911, 0.879
left-double precision, recall, f1	0.904, 0.922, 0.913	0.831, 0.86, 0.845	0.839, 0.839, 0.839	0.889, 0.873, 0.881
right-double precision, recall, f1	1.0, 0.923, 0.96	0.962, 0.909, 0.935	0.764, 0.824, 0.792	0.863, 0.815, 0.838
weighted average precision, recall, f1	0.919, 0.917, 0.918	0.887, 0.885, 0.885	0.855, 0.855, 0.855	0.871, 0.871, 0.871
cv score	0.846	0.842	0.883	0.893
accuracy	0.917	0.885	0.852	0.871

Fig. 7. Different evaluation metrics for different ML models on the test set with 5-fold cross-validation.

While the cross-validation score of the raw time series dataset was already at 84.6%, utilizing all features SVM achieved 88.3% and a random forest got an even better result with 89.3% when correlated features were removed.

The last model without the correlated features is ultimately preferable, not only because of the highest cross-validation score but also because of the lowest complexity as it requires fewer features than the other models. Also, looking at the

 ISBN: 978-1-68558-047-6

confusion matrix in Figure 8, the ROC curve and AUC value in Figure 9 there are no abnormalities between the different holding postures, as all perform very similarly.

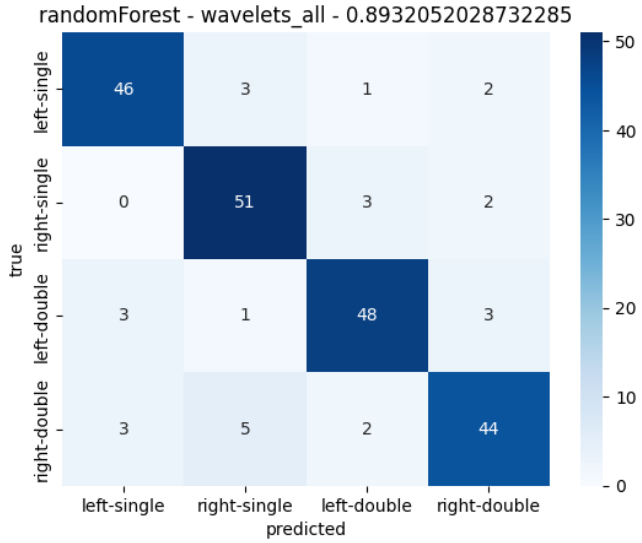

Fig. 8. Confusion matrix of the final model (Random Forest).

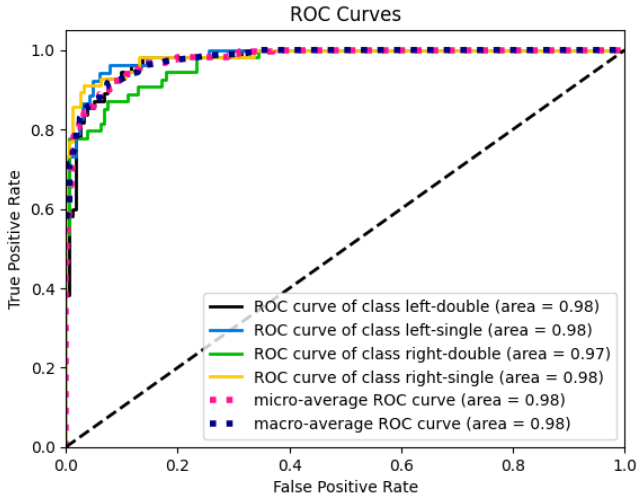

Fig. 9. ROC Curves and AUC value for the final model (Random Forest).

VI. Conclusion and Outlook

By analyzing the sensor data of the built-in IMU of smartphones we were able to detect the holding posture during the usage of an app with an accuracy of 89.3% without the need of external sensors. Therefore, the model is hardware and platform-independent and can be shipped on Android and iOS applications. Interestingly, the raw time series data performed way better than initially expected, feature engineering did increase the overall accuracy and reduced the complexity of the model though.

With the knowledge of the holding posture of the user at runtime, the UI designer can take this information into consideration while designing an intuitive interface. Also, self-adaptive capabilities in terms of the UI of a smartphone could be imaginable with real-time recognition of holding postures.

While this model now only works for tap gestures, we are currently working on adding swipe gestures with a Dynamic Time Warping approach to find out if swipe gestures can further improve the accuracy of the model. Besides, the dataset will be enlarged with new data recordings to compare the machine learning models with neural networks.

References

[1] S. Hoober, "How do users really hold mobile devices?" 2013, accessed: 2023-05-08. [Online]. Available: https://www.uxmatters.com/mt/archives/2013/02/how-do-users-really-hold-mobile-devices.php

[2] C. Hardyck and L. F. Petrinovich, "Left-handedness." *Psychological bulletin*, vol. 84, no. 3, p. 385, 1977, accessed: 2023-05-08. [Online]. Available: https://psycnet.apa.org/record/1978-00208-001

[3] "iPhone 5s - Technical Specifications," https://support.apple.com/kb/sp685?locale=en_GB, accessed: 2023-05-08.

[4] "iPhone X - Technical Specifications," https://support.apple.com/kb/sp770?locale=en_GB, accessed: 2023-05-08.

[5] "iPhone 13 Pro Max - Technical Specifications," https://support.apple.com/kb/SP848?locale=en_GB, accessed: 2023-05-08.

[6] R. Wimmer and S. Boring, "Handsense: discriminating different ways of grasping and holding a tangible user interface," in *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*. New York, NY, USA: Association for Computing Machinery, 2009, pp. 359–362. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/1517664.1517736?

[7] K. Hinckley, S. Heo, M. Pahud, C. Holz, H. Benko, A. Sellen, R. Banks, K. O'Hara, G. Smyth, and W. Buxton, "Pre-touch sensing for mobile interaction," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2016, pp. 2869–2881. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/2858036.2858095

[8] M. Goel, J. Wobbrock, and S. Patel, "Gripsense: using built-in sensors to detect hand posture and pressure on commodity mobile phones," in *Proceedings of the 25th annual ACM symposium on User interface software and technology*, ser. UIST '12. New York, NY, USA: Association for Computing Machinery, 10 2012, p. 545–554. [Online]. Available: https://doi.org/10.1145/2380116.2380184

[9] M. Goel, A. Jansen, T. Mandel, S. N. Patel, and J. O. Wobbrock, "Contexttype: using hand posture information to improve mobile touch screen text entry," in *Proceedings of the SIGCHI conference on human factors in computing systems*. New York, NY, USA: Association for Computing Machinery, 2013, pp. 2795–2798. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/2470654.2481386

[10] C. Park and T. Ogawa, "A study on grasp recognition independent of users' situations using built-in sensors of smartphones," in *Adjunct Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. New York, NY, USA: Association for Computing Machinery, 2015, pp. 69–70. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/2815585.2815722

[11] M. Löchtefeld, P. Schardt, A. Krüger, and S. Boring, "Detecting users handedness for ergonomic adaptation of mobile user interfaces," in *Proceedings of the 14th international conference on mobile and ubiquitous multimedia*. New York, NY, USA: Association for Computing Machinery, 2015, pp. 245–249. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/2836041.2836066

[12] J. Avery, D. Vogel, E. Lank, D. Masson, and H. Rateau, "Holding patterns: detecting handedness with a moving smartphone at pickup," in *Proceedings of the 31st Conference on l'Interaction Homme-Machine*, ser. IHM '19. New York, NY, USA: Association for Computing Machinery, 12 2019, p. 1–7. [Online]. Available: https://doi.org/10.1145/3366550.3372253

 ISBN: 978-1-68558-047-6