This WASA course unit [CM-W-INT] describes the content and the organization of the lecture and practical course "Web Applications and Service-oriented Architecture" (WASA) provided by the research group Cooperation & Management (C&M, Prof. Abeck). Current concepts of software development and architectures (including Behavior-Driven Development, Domain-Driven Design, Microservices, RESTful Webservices, 12 Factor App, CI/CD Build Pipelines, DevOps, Container-virtualized Infrastructures) as well as related standards and technologies (including HTTP, Java, JavaScript/TypeScript, Angular, Spring, GitLab-CI, Docker, Kubernetes, Prometheus) are introduced which are needed to develop advanced (i.e. microservice-based, IoT aware, cloud-native, mobile) web applications. The web applications stem from the connected car domain which includes concepts from the domain of Internet of Things. Since the concepts presented in the lecture must be practically applied to really understand them the WASA lecture is only offered in a combination with the WASA practical and/or seminar course.

IMPORTANT: In winter semester 2020/21, the lecture and practical/seminar course are organized as an online event. This new format is called WASAOnline.

The WASAOnline kickoff lecture will take place on Wednesday, 4th November 2020 on 9:45 am.

Each student who wants to take part in WASAOnline and in the kickoff lecture should send an email to Sebastian Abeck (sebastian.abeck@kit.edu) to receive the web link to the web session which is made available for his/her personal use. Please use your depseudonymized KIT student email address (see https://my.scc.kit.edu/shib/pseudonymisierung.php for further information). Thank you!

The lecture material is made available in English. During the online lecture, the content is presented and discussed in German. The oral examination is conducted exclusively in German. Each student can choose if he/she wants to write his/her practical/seminar thesis in German or in English. Thesis templates are made available in LaTeX.

C&M Cooperation & Management
KIT Karlsruhe Institute of Technology
WASA Web Applications and Service-oriented Architectures

The research work carried out by C&M can be divided into two main areas:

(Connected Car, Internet of Things) In this area connected car application and IoT application based on the concept of domain modeling and microservice architectures are developed. Relevant concepts applied in the development process include Behavior-Driven Development (BDD) and Domain-Driven Design (DDD).

(Identity and Access Management, SecDevOps) SecDevOps concerns the continuous integration (CI) and continuous deployment (CD) of (hopefully not monolithic, but microservice-based) software systems deployment into a container-virtualized (Docker/Kubernetes-based) cloud infrastructure. The Identity and Access Management (IAM) is a highly relevant crosscutting concern appearing in almost every web application. One of the projects is concerned with the provisioning of IAM as a flexible cloud service.

BDD Behavior-Driven Development
CI/CD Continuous Integration / Continuous Deployment
DDD Domain-Driven Design
IAM Identity and Access Management
IOSB Fraunhofer Institute of Optronics, System Technologies, and Image Exploitation (Optronik, Systemtechnik und Bildauswertung)
IoT Internet of Things
SecDevOps Security Development Operations
The software development process applied by C&M combines the concepts of Behavior-Driven Development (BDD) and Domain-Driven Design (DDD). Both concepts provide complementary contributions to the layered microservice architecture as the figure illustrates.

(1) An implemented feature can be seen as a deployable increment of the software system. (Feature 1, Feature 2, ...) The ordering of the features in the figure implies that the first feature should cover the core functionality of the software system.

(2) The domain model makes sure that the static and dynamic domain knowledge is consistently used by each feature. This ensures that the features build a consistent whole although each feature is developed and deployed independently from other features.

BDD  Behavior-Driven Development
DDD  Domain-Driven Design
The activity diagram gives an overview of the whole engineering process which in fact is divided into two sub-processes: According to our understanding of DDD, a domain model is the foundation of all software applications belonging to this domain. We regard the domain model as the knowledge of a domain expert he or she makes available to the software developer by the model. This knowledge is the application-agnostic part of the application meaning that each application of the domain implements this knowledge in its domain logic layer. We understand this domain knowledge as the "physics" that each application must adhere to. Evans calls this part of functionality the heart of the software system [Ev03].

(Starting Points) (Strategic Modeling of the Initial Context Map) The starting point of the domain modeling process is set earlier as the starting point of the application development process. The reason is that it makes sense to have an initial context map of the domain before the development of the first application starts. In the following we describe the main characteristics of the application development process.

(Context Map) The main artifact of the domain modeling process is the context map in which all micro-service-based applications belonging to the domain must fit into.

(Strategic Modeling of the Bounded Contexts as Part of the Domain's Context Map) In the application development process, the integration of the application under development into the context map of the domain is done in the second step after the first step in which the the BDD-based requirements analysis is carried out.

(Tactical Modeling of a Bounded Context) A bounded context is part of a context map. It represents a candidate for a microservice which can be developed by an independent team [Ne15]. DDD considers the modeling of the content (i.e. the functionality) of a bounded context as tactical modeling.

(Relation View) The most relevant artifact of the tactical modeling is the relation view which is built in the third step. By the relation view central DDD concepts (such as entity and value objects) including their relationships and operations are represented.

(Implementation of each Bounded Context as a Microservice Accessible via its Web API) The functionality modeled by the relation view is implemented in the domain logic layer of the microservice architecture. A systematic approach to implement the bounded context as a microservice (fourth step) is a core characteristic of the proposed development process. One of the key demands of DDD is to keep the model and its implementation in sync in order to assure that the model and the implementation is the same.

The two remaining steps concern development activities around the core implementation of the microservice, namely the implementation and test and the deployment of the application.

The main dimension according which the development tools can be ordered are the development phases (from analysis to deployment). The tools supporting the analysis and design and the implementation and testing can be grouped according to the software architecture (which in our case is a microservice architecture).

(Project Management and Version Control) These tools support the overall organization of the software project and the communication between the project members. At C&M two different tools sets are used: the Atlassian toolset extended by Microsoft tools, esp. SharePoint on which the C&M Teamserver is based.

(Analysis and Design) Analysis requirements at C&M is done by taking the approach of behavior-driven development (BDD) based on the tool Cucumber. For the design the two most relevant tools are Enterprise Architect for the domain model and Swagger for the API specification. In addition to these tools the Microsoft Office tools (Word, PowerPoint) and Atlassian Confluence (in the case of iCC/xdi projects) are applied for documentation purposes.

(Implementation and Testing) In this phase the frontend and backend of the web applications are constructed. The integrated development environment (IDE) used for frontend development is JetBrains WebStorm and frontend frameworks are Angular and Bootstrap. As IDE for backend development Eclipse is used and Apache Spring (esp. Spring Boot for the microservice implementation) is used as backend framework.

(Build and Deployment) The build and deployment of the microservices is carried out via a build pipeline by which the concept of continuous integration and continuous deployment is provided. The result of the build pipeline is a Docker image (= application container image) since Docker is used as the container environment at C&M. Kubernetes is used to manage the Docker containers in order to reach a high scalability and robustness of the service landscape.

BDD Behavior-Driven Design
IDE Integrated Development Environment
(PredictiveCarMaintenance, Connected Car) The whole case study centers around the PredictiveCarMaintenance (PCM) which is a microservice-based application from the connected car domain.

(User, PredictiveCarMaintenance, PCM) The connected car application implements the predictive maintenance of a vehicle. The functionality of the PredictiveCarMaintenance (PCM) is specified by Gherkin features.

(IAM Service, Auth0) The Gherkin features include Identity and Access Management (IAM) requirements, such as authentication of the user of the PCM application. The IAM service Auth0 is used to implement these requirements.

(PCM Microservice, The Twelve-Factor App) The application itself is based on the microservice architecture. The development of the PCM microservice should follow the Twelve-Factor App requirements resulting in a cloud-native application.

(Message Broker, RabbitMQ) The communication between the Backend-For-Frontend (BFF) and microservices is based on the exchange of events. The event bus is provided by a message broker tool named RabbitMQ.

(PCM Microservice, Connected Car API, OpenAPI, IoT, IoT Service, SensorThings, Web of Things) The information needed for predictive maintenance is provided by a connected car's API specified in the OpenAPI format. A connected car today is one of the most relevant "Things of the Internet". Therefore, IoT standards, such as SensorThings API from OGC and Web of Things from W3C, are taken into account in the development of the PCM application.

(API Security, Connected Car API) The API offered by the connected car must be protected against unauthorized access. The IAM service provides the authentication and authorization information needed to take the access decision at the API.

(DevOps, Build Pipelines, Kubernetes Cluster, Prometheus Monitoring) IC2020 will also make relevant contributions to the highly relevant DevOps topic. The goal is to develop a build pipeline based on GitLab-CI by which the PCM application is continuously integrated (CI) and deployed (CD) on a Kubernetes cluster. The PCM application provides metrics which are monitored using the tool Prometheus.

PCM PredictiveCarMaintenance
The table describes the planned sequence of course units presented during the lecture.

(Content) All titles printed in upper case are course units that are the basis for the oral examination carried out at the end of the semester. After the course unit was held a range of page numbers are added. This indicates which pages must be prepared by the student for the oral examination.

COURSE UNIT (P. X – P. Y) All course units written in upper case letter are primary examination material. If page ranges are indicated only these pages are treated in the oral exam. No page range means that questions of the complete course unit can be asked in the oral exam.
The acronym WASA stands for "Web Applications and Service-oriented Architectures". Four different types of WASA courses are offered: (i) lecture courses WASA1 and WASA2 (ii) practical courses WASA1 and WASA2 associated to the lecture courses (iii) prosemnar course associated to WASA1 lecture course and seminar associated to WASA2 lecture course (iv) key qualification course (germ. Schlüsselqualifikation SQ).

(1) The lecture courses WASA1 and WASA2 each comprise 2 semester hours. A student who attends one of the lectures acquires 4 credit points (germ. Leistungspunkt).

(2) The practical course runs in parallel with the lecture course and counts 5 credit points meaning a workload of 150 hours. The capacity of students C&M can offer the practical course depends on the current projects carried out in the research group.
Hint: In the Wirtschaftsinformatik study programme the name of the module is "Microservice-basierte Web-Anwendungen".

(3) The examiners are Prof. Abeck and one of the C&M's PhD Researchers. Since the examination is in the lecture term the students should have a good personal resource management in order to have enough time for the preparation of the examination.

SQ Schlüsselqualifikation (Key Qualification)
On this page the specifics of the WASA practical/seminar courses which are offered in parallel to the WASA lecture are described.

1. A project consists of about 4 to 6 practical/seminar students.
   1.1 A SeniorStudent is a student who is writing his/her bachelor thesis or master thesis at C&M.
   1.2 The meetings take place at a defined time which is fixed at the beginning of the semester.

2. The topics dynamically evolve from the work done by the SeniorStudent in their bachelor/master thesis.
   2.1 The JuniorStudent should actively participate in the discussion and make own proposals how the topic should be treated.
   2.2 Reviews are an integral part of the work in the project team.
   2.3 This means that the practical/seminar work has a dynamic characteristic.
   2.4 There is a high flexibilty and liberty concerning the focal points of the practical/seminar work.
The project team agenda is a markdown document by which the work of all project team members is coordinated.

(1) The project team leader takes care that all team members contribute to the agenda which is available for each project team meeting.

(1.1) For each project team a subgroup exists in the C&M GitLab. In the README file of the repository "Projektteamtreffen" the agendas of the project team meetings are collaboratively worked out.

(1.2) The project team meets for one hour every week. The day and time are defined by the project team members.

(1.3) That is why the agenda is worked out in German.

(2) There are templates available for the different variants in which a JuniorStudent can write his/her practical/seminar thesis.

(3) C&M-TEAMARBEIT is a document [CM-CMT] which describes how the members of the research team C&M efficiently work together.

(Screen dump on the right hand side) This is an excerpt of the GitLab document "Projektteamtreffen" which provides an agenda draft of the first two project team meetings.

Each project team makes a specific contribution to the show case.

(PT NN) Implementation of PCM frontend providing the presentation logic and the backend microservices including the application logic.

(PT SchneiderMüller) Provide a streaming-oriented enhancement for the given Car API; Generation of realistic vehicle data through simulation of vehicle sensors

(PT PiatkowskiMidedji) (PT SidlerNN) Design of a BFF microservice that supports the overall PCM Architecture by merging request calls from the PCM Frontend to the backend microservices

(PT HippchenNN) Introduction of an event-driven architecture for the communication between the developed microservices

(PT ThronerNN) Provide the necessary infrastructure for the other project teams for the deployment to a Kubernetes cluster
(1) The goal of this project team is the implementation of PCM frontend providing the presentation logic and the backend microservices including the application logic. The main task for the ICD2020 is to deliver a detailed overview for at least one vehicle component including relevant values required by "Garage" for a diagnosis. This will enable a mechanic repairing a vehicle component to diagnose the condition and apply required actions for restoring a good condition.

(1.1) Based on the mockups and the feature descriptions, the GUI presentation is to be developed. Later on, the presentation logic including data fetching from the PCM BFF will be implemented. This implementation will be tested by end-to-end tests extracted from the Gherkin feature descriptions.

(1.2) First, the Diagnosis microservice will be implemented, which is responsible for generating deductions from current and historical sensor data. The Maintenance microservice, which takes data from Diagnosis microservice as an input to determine state definitions for the vehicle and its components, is expected to be taken care of after the ICD2020.

(get diagnostic data) Garage uses the PCM application to get diagnostic data relevant for repairing a Vehicle.

(deduct) The Diagnosis microservice generates deductions for observed sensor values. In this example the tire pressure reduction is deducted from tire pressure values.

(produce diagnoses) The Maintenance microservices receives events for produced diagnoses via a message broker.

(2) The work packages specified for the ICD2020.

(2.1) The component detail view contains static and dynamic vehicle component information. The dynamic information is shown as current and deducted values as numbers and historical values in the shape of graphs.

(2.2) The Diagnosis microservice must create deducted values, such as the tire pressure reduction per month. This is an important indicator for the condition of a component.

(2.3) This part of the PCM is concerned with the business process which organizes the maintenance appointment.
The goal of this project team is to introduce an event-driven architecture for the communication between the microservices developed during the ICD2020. The basis of the event-driven architecture is the use of a message broker. RabbitMQ is used within the ICD2020.

In the context of IDC2020 this project team must conclude different work packages. Members of PT Hippchen fulfill a cross-team purpose while designing and developing microservices. Each project team member is responsible for designing and implementing the web APIs of the PCM's microservices.

(2.1) In order to enable the communication of microservices to be triggered by events, an instance of RabbitMQ must first be provided. One means of providing this is to use a Docker image [Doc-Doc]. Based on the Docker image, the necessary configurations can be made.

(2.2) The project team members take over the implementation of the connection to the RabbitMQ in the microservices Diagnosis and Maintenance. So, a close collaboration with the project team PT Piatkowski is necessary.

(2.3) Similar to a RESTful API, the communication via events must be specified in detail. For this purpose, there is already the domain event pattern that help in the design.

(2.4) Finally, the specified web API in the form of the domain events is implemented in the microservices.

/1/ The Diagnosis microservice publishes domain events that provide information about activities within the microservice. An example of a domain event is "updatedComponentState" which is published when the state of a component of the vehicle is changed. A domain event therefore always describes when the state of a domain object changes. The features of the PCM also serve as the basis for identifying relevant domain events.

/2/ The Maintenance microservice subscribes to domain events that it needs for processing its own features. For example, it needs the information given by "updatedComponentState" to determine whether a vehicle requires maintenance.

/3/ The protocol for communication with the RabbitMQ is the AMQP (i.e. no HTTP).

AMQP Advanced Message Queuing Protocol
(1) The goal of this project team in the context of the ICD2020 is the design of a BFF microservice that supports the overall PCM architecture by merging request calls from the PCM frontend to the backend microservices. Therefore, the complexity of the frontend is simplified by isolating the data formatting in the BFF and by reducing the number of requests the frontend has to perform.

(1.1) A suitable BFF API has to be defined. It should closely relate to the functionality of the frontend.

(1.2) The BFF receives requests from the frontend and processes them by sending requests on the backend microservices.

(1.3) The BFF retrieves data from the backend microservices and delivers this data in a desired format to the frontend.

(2) In the following, work packages for the project team are defined.

(2.1) The first task is to understand all features and functionality of the frontend. Based on this analysis, necessary endpoints are identified.

(2.2) Based on the identified endpoints in (2.1), the API has to be specified including URLs, query parameters and data formats. The API must be a REST API and the specification should be designed by using the tool OpenAPI.

(2.3) The BFF is implemented by the students based on the previous tasks.

(2.4) A further task is the implementation of a concept to serve the different roles that exist within the PCM application. Depending on these roles, the frontend offers different views and therefore, different data from the backend has to be fetched and returned to the frontend.
(1) The goal of this project team is to allow the generation of the realistic vehicle data by simulating the car sensors. Providing realistic data is essential to test developed solutions against their applicability in the real world.

(1.1) This must be done based on the Load Generation for Testing (LGT) app which is developed at C&M and allows to generate load using dynamic and flexible scheduling strategies. In this context, only two components of LGT are relevant: the load generation and the scheduling microservices. Both services can be used together to simulate a vehicle sensor, e.g. oil pressure sensor.

(1.2) A Sensor Car Data Management (SCDM) app has also been developed at C&M. Its vehicle microservice can be used as a foundation to collect and process the sensor data generated from the simulated sensors.

(2) The work for this project team is split into three packages:

(2.1) First, the load generation microservice has to be adapted to simulate a sensor. This includes providing a streaming API and the communication with the scheduling microservice.

(2.2) This package includes the investigation of realistic sensor data trends and defining these trends for the implemented sensors. These trends can be defined in coordination with PT Schneider, who is responsible for the IoT communication aspect in ICD2020, to achieve satisfying results.

(2.3) The scheduling microservice controls the sensors. It decides when and what value the sensor has to trigger based on the defined scheduling strategies. In this project, a new type of scheduling strategy has to be implemented to allow the definition of the sensor data distribution.

(2.4) A vehicle microservice has to be implemented. It must collect the data from the different sensors using a streaming API and process them to deliver realistic data, e.g. a correlation between current vehicle speed and the decreasing tank level should be mapped. As mentioned above, the vehicle microservice of SCDM can be used as a foundation for the development of the new vehicle microservice because the current microservice can only generate random values and is not yet capable of collecting and processing data from multiple sensors.

LGT Load Generation for Testing
SCDM Sensor Car Data Manager
The goal within this project team and ICD2020 is to provide a streaming-oriented enhancement for the given car API [IC19 Car]. A connected car can be seen as a thing which is connected to the internet and provides a large amount of sensor information. A first approach to handle streamed data within a car is provided by SensorCarDataManager (SCDM). However, SCDM does not use a uniform data format.

(1.1) It is beneficial to use a standardized format, because otherwise, different data formats will be used (e.g. by car manufacturers). This leads to the typical interoperability problem within the Internet of Things (IoT).

(1.2) Since SCDM does not provide or utilize a uniform data format (in contrary to the SensorThings API), the services need to be adjusted to match the SensorThings API data format.

(2) The following points show the work packages of the IoT project team.

(2.1) Especially for the ICD2020, there should be a clarification regarding the IoT-based streaming API and the resource-oriented car API. At the current state, the resource-oriented API should be extended by the project team by implementing a streaming-based extension. During this work package, it must be clarified which functionality is offered by which API part.

(2.2) The adjustments should be utilized by following the SensorThings API standard.

(2.3) The SensorThings API standard leaves room for unprecise implementations. Therefore, guidelines for implementing the SensorThings API in order to assist developers should be created.

(2.4) Not every subject is allowed to access the provided data. Therefore, there is a need to protect the streaming and resource-oriented data. The considered solution presumes that policies are defined in the vehicle, which determine which data streams are available at the streaming API.

(2.5) Fraunhofer’s FROST server should be used to implement the SensorThing API for the connected car.

<table>
<thead>
<tr>
<th>FROST</th>
<th>Fraunhofer Open Source SensorThings API Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>LGT</td>
<td>Load Generation for Testing</td>
</tr>
<tr>
<td>OGC</td>
<td>Open Geospatial Consortium</td>
</tr>
<tr>
<td>SCDM</td>
<td>Sensor Car Data Manager</td>
</tr>
</tbody>
</table>
1. To ensure smooth operation of applications, it is important to define and implement a uniform process for delivery. A software artifact for this is the CI/CD pipeline. This enables certain framework conditions and processes to be enforced and adhered to. Without an appropriate delivery and uninstallation process, an undesired state on the target system can occur. The correct installation and cleanup is especially important due to the agile development process as changes and rollouts of new software artifacts occur in short time intervals. Kubernetes represents a modern and widespread platform for the provision of applications. The aim of the project is a sustainable development of a process for delivering applications to a Kubernetes cluster.

2. For the preparation of the ICD2020, a platform is required to ensure easy and reliable delivery of software artifacts. This is to be achieved by using a Kubernetes cluster and the use of defined and automated processes. The PT ThronerGogel project team will create the necessary conditions for this. This includes the following work packages.

2.1 The build artifacts include the finished Docker images, as well as the Helm charts. The images and charts contain the complete logic of the application and should therefore be protected against manipulation. To achieve this, each application should have separate read and write access to the build artifacts.

2.2 For the CI/CD pipeline, pipeline templates shall be developed which represent individual and reusable pipeline steps. The steps should be useable across all the projects. Each step should be documented in the pipeline repository.

2.3 Docker images should be created for the pipeline steps, which contain fixed components for the delivery process, such as the connection to the cluster and the processing of secrets. For the individual project the PT ThronerGogel also should create Dockerfiles for all the programming languages used in the ICD2020 show case. This is necessary to ensure the quality and security of the images.

2.4 To make it easier for the other project teams to deliver applications to the Kubernetes cluster, project templates will be provided for the projects. These should contain all relevant pipeline steps as well as the structures and necessary artifacts. The precise procedure for using the templates should be documented.

CI     Continuous Integration
CD     Continuous Delivery/Continuous Deployment
This page summarizes all relevant dates that are relevant for each participant of the practical / seminar course offered in combination with the WASA lecture.

(1) It is absolutely necessary that a student has a free slot in his/her personal time table in order to be able to take part in these weekly meetings. The meetings start in the next week (i.e. the second week of the lecture period).

(2) This preliminary version is reviewed by the co-supervising PhDResearcher/SeniorStudent.

(3) The precise date is communicated in the project teams. The date is also fixed in the team calendar of the C&M Teamserver.

(4) The content produced for the presentation should conform to the WASA course material.

(5) The delivery date is the last day of the lecture period.
It is absolutely important that a student who participates in the WASA lecture and practical/seminar course has the necessary resources to cope with the workload (lecture: 120 hours, practical course: 150 hours).

(2.1) Check on the page of the KIT Steinbuch Computing Centre if your email is already depseudonymized.

(2.3) The motivation and the experiences should be summarized in at least one or two paragraphs.

(2.4) This means that the first project team in the list is your favorite team.

(3) This email will be sent by the co-supervisor of the members of the project team which usually is a SeniorStudent.

(3.1) The current lecture material is stored on the C&M Teamserver in the following folder: https://team.kit.edu/sites/cm-tm/Mitglieder/2-0.Aktuelles_Semester
This function can be found in the Studierendeportal by clicking on "Meine Benutzerdaten" > "De-/Pseudonymisierung" and accepting "Ich stimme der Sichtbarkeit meiner namensbezogenen Daten zu". The name-related E-Mail-Adresse <prename><surname>@student.kit.edu" exists additionally to the "uxxx@student.kit.edu" email address.
The C&M Teamserver is described in detail in the document C&M-TEAMARBEIT (in German) which is available
(i) on the C&M web site: https://cm.tm.kit.edu/
(ii) on the C&M Teamserver: https://team.kit.edu/sites/cm-tm/Mitglieder/1-1.Teamarbeit