DAI-DSS ARCHITECTURE AND INITIAL DOCUMENTATION AND TEST REPORT

D4.1

Editor Name	Rishyank Chevuri (Jotne EPM Technology)
Submission Date	February 28, 2023
Version	1.0
State	Final
Confidentially Level	PU

Co-funded by the Horizon Europe
Framework Programme of the European Union

EXECUTIVE SUMMARY

This deliverable "D4.1 – FAIRWork Architecture and initial Documentation and Test Report" for is the first description of the architecture, which will be updated in M20 and M30 in form of updated deliverables.

First the high-level architecture is mapped to the use case scenarios that are for FLEX "Automated Test Building", "Worker Allocation" and "Machine Maintenance After Breakdown" as well as for CRF "Workload Balance", "Delay of Material" and "Quality Issues". Based on the process analysis described in more detail in "D2.1 - Specification of FAIRWork and Initial DAI-DSS Architecture" we identified generic challenges to the aforementioned use case scenarios. This ensures that the architecture and the implemented solutions will not exclusively fit to those use cases but also serve similar use cases that are out of the project. The challenges are: "finding similar projects", "find relevant experts", "simulate production process", "allocate worker", "map workers with profiles", "find similar problems", "reschedule production line", "allocate order to production line", "assess the impact". Such challenges are targeted by Microservices that may use Artificial Intelligence (AI) when appropriate. Hence, we propose a list of services that either individually or in a cooperative manner target the aforementioned list of challenges.

The architecture distinguishes therefore between (a) the core components that allow the selection, configuration, deployment, and operation of selected (AI) services and (b) a list of (AI) services the user can select and compose a solution that targets the requirements of the particular use case.

The (a) cores components are (i) the user interface that is a framework enabling user interface widgets to be deployed in environment like web-pages of MS TEAMS, (ii) the orchestrator that is a framework enabling the controlling of individual services or the orchestration of services, (iii) the knowledge base that integrates data from legacy applications, sensor data streams that require no protection and sensor data stream that require protection in form of an Intelligent Sensor Box (e.g. for human-related information); (iv) the externa data asset marketplace complements the data coming from inside the use case with available data form outside, (v) the configurator that enables the selection and configuration of services to a use-case specific solution and finally (vi) the so-called AI-enrichment which is a service catalogue providing different AI-based services that fulfill the end users' needs.

The (b) list of (AI) services is a collection of available services, commercial products and research prototypes that are partly used from outside the consortium where feasible and partly created during the project. Hence, we currently propose an initial list of services that target the aforementioned challenges and propose different AI realizations to demonstrate the flexibility of our core components and to enable a selection of appropriate AI solutions. This list is therefore seen as indicative, and it is expected that in the duration of the project it will evolve. The final set of services will also be provided as projects results in the so-called innovation shop.

An initial plan for testing and reporting procedures is presented, which is currently seen as a plan and will be detailed once the deliverable will be updated in M20. The updated version of the deliverable includes the details of implemented components and services and updated security implementations. as well as implemented architecture testing and reporting methodologies.

PROJECT CONTEXT

Workpackage	WP4: Development of DAI-DSS
Task	T4.1 Architecture, Documentation and Testing
Dependencies	This task produces a generic architecture that will be instantiated in WP4 as a reference implementation

Contributors and Reviewers

Contributors	**Reviewers**
Remi Lanza, Rishyank (Jotne) Herwig Zeiner, Lucas Paletta, Michael Schneeberger (JR) Robert Woitsch, Magdalena Dienstl (BOC) Jose Barbosa, Gutavo Vieira, Higor Rosse (MORE) Sylwia Olbrych (RWTH), Alexander Nasuta (RWTH) Christian Muck (OMiLAB)	Wilfrid Utz (OMiLAB) Damiano Falcioni (BOC) Daniel Lütticke (RWTH)

Approved by: Robert Woitsch [BOC], as FAIRWork coordinator

Version History

Version	**Date**	**Authors**	**Sections Affected**
1.0	February 28, 2022	Remi Lanza/Rishyank (Jotne) and Contributors	All

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

Abbreviation	Meaning
AI	Artificial intelligence
ANN	Artificial neural networks
API	Application program interface
BPMN	Business Process Model and Notation
CAD	Computer aided design
CE	cognitive-emotional
DAI-DSS	Democratized AI decision support system
DHS	Digital Human Sensor
DMN	Decision Model Notation
EDM	Express data manager
EIP	Enterprise Integration Pattern
FIPA	Foundation for Intelligent Physical Agents
GUI	Graphical user interface
HLD	High level design
HTTP	Hypertext Transfer Protocol
IT	Information technology
IIOT	Industrial internet of things
IOT	Internet of things
ISB	Intelligent sensor box
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
KPI	Key performance indicator
LLD	Low level design
MAS	Multi Agent System
ML	Machine Learning
NLP	Natural Language Processing
OGSi	Open Service Gateway Initiative
OPC-UA	Open Platform Communications - Unified Architecture
PLCS	Product Life Cycle Support
PLM	Product Lifecyle management
PSI	physiological strain index

QA	Quality assurance
RDF	Resource Description Framework
RDL	Reference data library
REST	Representational State Transfer
RL	Reinforcement Learning
SDAI	Standard Data Access Interface
SDLC	Software development life cycle
STEP	Standard for the Exchange of Product model data
TCP	Transmission Control Protocol
UI	User interface
URI	Uniform Resource Identifier
WP	Work package
XML	Extensible Markup Language

1 INTRODUCTION AND PROBLEM STATEMENT

1.1 Project Introduction

Manufacturing companies are to face challenges than ever to international competition in the modern world due to advancements and incorporation of Digital technologies like AI and Robotics in production lines. Different objectives can be used for production system optimization, it might cover all technological components, including the use of robots, machines, AI, or it can apply to the improvement of processes and systems using important criteria like the number of workers, the number of stations, the line's balance between stations, and others. All these criteria may be employed in the optimization process to govern the process depending on some decisions.

All optimization tasks need evaluation of performances of process, machines, workers. The most common way used in most of the manufacturing companies to evaluate the production systems are Key Performance Indicators. These indicators show the performance of the process in all needed aspects. These indicators in business have the preferable value which is a target of process performance. Decisions are made based on these Measurable performance indicators to control the process due to cost, expected efficiency and energy consumption. The main challenge is often these decisions overlap or are in conflict to each other.

Most of the production process are complex, thus there is a high likelihood that the decisions made lead to unwarranted result. Additionally, decision-making in the modern production environment results in a great deal of uncertainty due to lack of knowledge about something or lack of precision data. It is crucial because, in the case of complicated production processes, the choice about the process' desired performance is not clearly stated. This makes it difficult to determine with precision if a method's assistance can provide managers with the chance to grasp the situation's entirety and base decisions on objective information. Based on the situation presented above, making a fast and good decision is important to make the profitability and effective business in the world of customization and short lead times.

Current automated and hierarchical structured production processes can only insufficiently deal with the upcoming flexibilization. To tackle this complex production setting we are introducing a decentralized AI system by “democratization” of decision making in production processes, where all relevant stake holders are involved. The proposed Democratic AI-based Decision Support System (DAI-DSS) finds the appropriate decision for a concrete situation during production. Each human or technical actor is represented by an agent who negotiates based on the status provided by the digital shadow and twins. The future situation is predicted by AI algorithms for each individual actor considering the modelled knowledge base that defines each negotiation strategy. A multiple optimization algorithm finds the most appropriate solution considering the needs of all involved human and technical stakeholders.

FLEX have been selected, both providing a complex combination of human, AI, robots, and data acting as a role-model for flexibilization and optimization of production processes, and aiming to consider energy efficiency and workforce safety, production efficiency, workload management in their decision making. The two use cases with complex cyber-physical smart and data driven processes are:

(1) The car manufacturer CRF, need to optimize the throughput between two succeeding production lines where automatic lines, static and mobile robots, and human workers with and without machines are cooperatively working together.

(2) The Design and manufacturing company FLEX wants to speed up and improve the programming of manufacturing robots for low volume production. The cooperative production by humans and robots is influenced by factors like energy consumption for moves, machine hours, safeness of cooperating workers

within the co-working area, selection, and tooling of grippers in combination with production plan and the required flexibility in low volume production.

To address the issues in the use cases the FAIRWork will use proposed Democratic AI-based Decision Support System (DAI-DSS) to optimize the production process. The overall objective of the DAI-DSS is shown below.

Democratic AI-based Decision Support System (DAI-DSS) - Objectives:

- **Speed-Up the Configuration of complex Decision-Making:** Models speed up the configuration and maintenance.
- **Enable Compliance in Decision-Making by approved Models:** Using methods and tools to approve decision models
- **Improve the Co-creation Capabilities:** Using physical laboratories enabling participatory and user-centred design.

FAIRWork brings "Human, AI, Data and Robots" together, by introducing: (a) Design Decision Models and assess if those models are "appropriate / fair", (b) decentralize the decision-making, by representing involved – human or technical – actors within a Multi Agent System and configure each agent with previously approved decision models, (c) broaden the view to optimize the production process, by also introducing social and energy related parameters to the existing technical and business related aspects. These will be identified and valuated in a participatory and user-centered design process. This enables a more balanced decision-making in complex situations, as the flexibility opens-up "space for maneuverer".

The goal is to change the present production process's decision-making to cooperative decision-making so that human workers may trust the decisions made regardless of whether they were made by humans, artificial intelligence (AI), or in a hybrid fashion. The DAI-DSS focuses providing support for making decisions in the production process under uncertainty, strong dependencies on unknown future events, affecting the balance of work between humans and machines, and affecting the overall success of the production process while using the available data with "best effort" in frequently a very complex and conflicting situation for human decision makers.

1.2 Motivation For DAI-DSS

The DAI-DSS stands for democratized AI decision support system. As the DAI-DSS aims at improving decision-making processes using AI, different aspects must be considered within this decision-making process regarding which part (human or AI) makes which decisions and decisions made by AI need to be trusted by the humans. This can be seen on a spectrum which ranges from decisions which are made completely by humans (where the human decides every aspect directly by himself) to decisions which are made complete automated (where a system or agent or AI decides without the input from a human). In this context another factor that must be considered is the granularity of the decisions, as within a process decisions on different levels. Decisions on a higher level can include multiple fine-granular decisions.

To better discuss these aspects, let's take the simple scenario of making coffee and a system, that supports humans in making coffee. On the one side of the introduced spectrum, the human must make all the decisions which are necessary for making the coffee. This includes deciding how many shots of espresso to use, as well as the amount of milk, sugar, and hot water needed. In addition, the order of preference must be determined. On the other spectrum, the (AI or agent) system makes all decisions on its own. In this case the human gets a coffee decided by the system.

Both extremes are unlikely to occur in a real scenario, but they are used as the boundaries of the introduced spectrums. But feasible solutions will occur somewhere between these two extremes by using AI based services.

Therefore, we will discuss three cases which show different manifestations of this spectrum. The cases are placed on the first side of the spectrum, which are those where humans still have a decisive say in decisions. Therefore, we omit cases which are too far on the fully automated side of our spectrum. Figure 1 shows flow charts of the three cases which are introduced below. These are used to as a visual representation and to allow an easier comparison of the cases. The cases themselves are used to discuss different decision on the introduced spectrum and do not introduce cases which can be directly executed in the real world.

In the first case the user must decide on each of the ingredients, like the coffee type, or if milk or sugar will be added. Here the basic functionality of making the coffee is provided by the system, but the decisions are all done by humans.

In the second case, the system offers a selection of possible coffees, and the user must choose one of them. For example, the system can offer a choice between espresso, americano, latte or cappuccino. Once the decision made, the machine prepares the coffee on its own. Here a user must make fewer decisions, as they do not have to choose every ingredient of the coffee. The system makes the lower-level decisions on how to prepare the coffee, but the higher-level decision on which coffee to make is decided by the human.

In the third case, the system makes a preselection for users, based on available knowledge about them. The premise is the same as for the second case, namely that a set of possible coffees is available within the system, but the system itself makes a preselection on which coffee options are offered to a specific user. For example, the system knows that a user only drinks black coffee, therefore it only offers espresso and americano. Latte and cappuccino are not offered, as the contain milk.

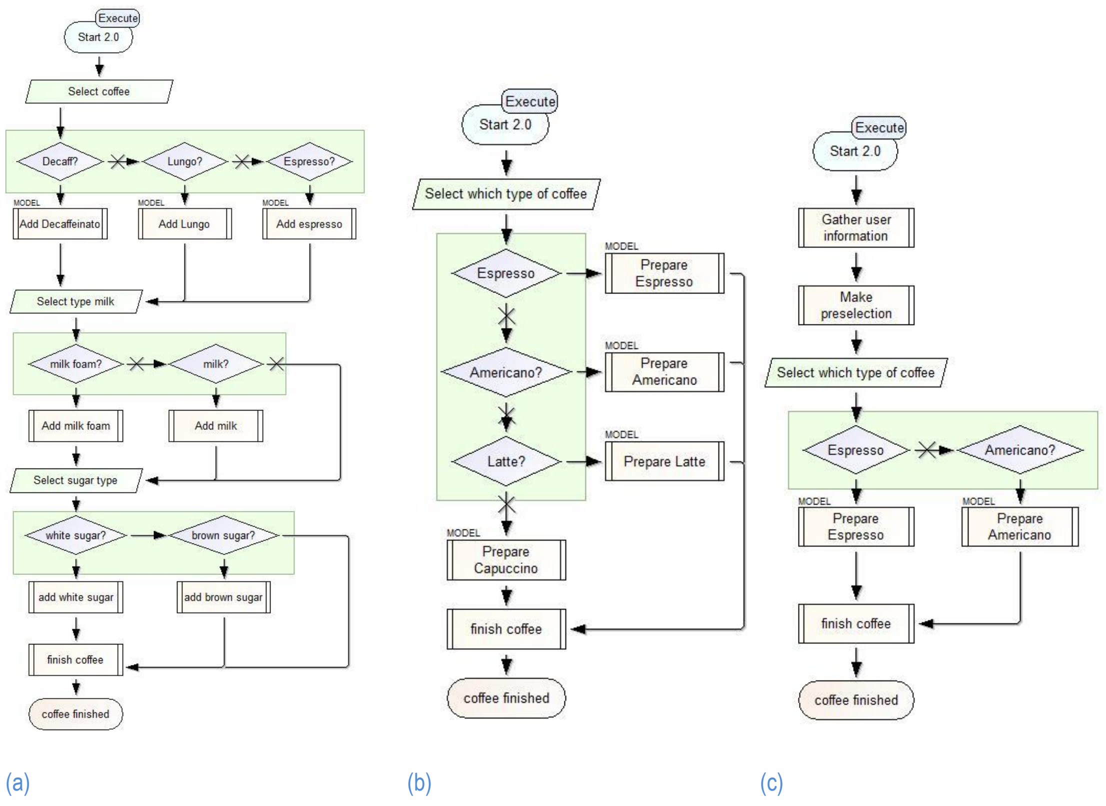


Figure 1 Flowchart Representation of different decision making cases

These three cases should only be seen as examples and not a complete set of possible cases, as there are many different other cases possible along the introduced spectrum. Within the FAIRWork project, different experiments will be executed to find a fitting cases on where decisions should be divided between humans and supportive systems, so that a sufficient decision support for humans can be established. Deliverable D2.1 covers the identified business process scenarios for the use cases of CRF and FLEX where DAI-DSS can aid humans in decision making process.

Another aspect introduced above can also be discussed in the context of the introduced cases. This is the how the granularity of the decision can influence the system and how the decisions are divided between humans and systems. Till now we did not talk about important decisions on how the creation process of the different coffees is managed. We abstracted the coffee making process into aspects we deemed important for humans. But other aspects, like how does the system refill its resources to create the coffees were neglected. In these decisions humans could be involved, by letting them check if the resources are (nearly) empty and if new ones should be obtained. Or such decisions can be automated to let humans focus on the important decisions and the other decisions are done by the system to support them.

The decision on which decisions are made automatically and in which the humans are integrated cannot be made generically but must be decided based on the domain and orientation of the system. To support humans, they should be able to make important and more high-level decisions, like which type of coffee should be produced and the system should make the decisions which are needed to allow the human to make his choices. For example, the system should automatically decide when to restock, so that humans can always make their decisions for the coffees. Of course, which decisions are important depends on the specific system.

1.3 Document Structure

This deliverable is structured as follows:

- **Chapter 1:** this section introduces the project objectives with focus on the main goals and problem statement, together with main proposed solutions and an example of decision-making scenario which involve human and system.
- **Chapter 2:** contains a high-level architecture overview and description. It shows how the different components within the architecture, without going into the details of those and mapping the high-level architecture with use cases.
- **Chapter 3:** details the individual core components of the architecture. The description is broken down into template sections.
- **Chapter 4:** gives the list of potential AI services that can be used for providing decision support in DAI-DSS.
- **Chapter 5:** lists the functional capabilities for DAI-DSS architecture.
- **Chapter 6:** provides the overview of security considerations within DAI-DSS.
- **Chapter 7:** introduces the initial plan for DAI-DSS Architecture testing and reporting methods.
- **Chapter 8:** summarizes the content of this document and provides the conclusion.

2 HIGH LEVEL ARCHITECTURE

The following chapter gives an overview of the high-level architecture of the proposed Democratic AI-based Decision Support System (DAI-DSS), while details about each component will be described in chapter 3.

2.1 High Level Architecture Overview

In general, the high-level architecture is based on the fowling factors:

- a complex legacy system,
- Microservices, Micro-Frontend and industrial IoT approaches,
- and decentralized decisions.

This means that the priority is to expand upon current systems. To comply with the second assumption, we use a loosely coupled approach of independent services that offer required functionalities. In addition, we use a Micro-Frontend Framework to create interfaces that may be integrated on various platforms. The entire decision-making process is supported using negotiations carried out by decentralized agents who look out for the needs and protect the interests of a specific digital twin and by the provision of various AI services. The following Figure 2 gives an overview of the high-level architecture, which follows the axioms mentioned above and a short description of the components and their interactions should give a broad overview.

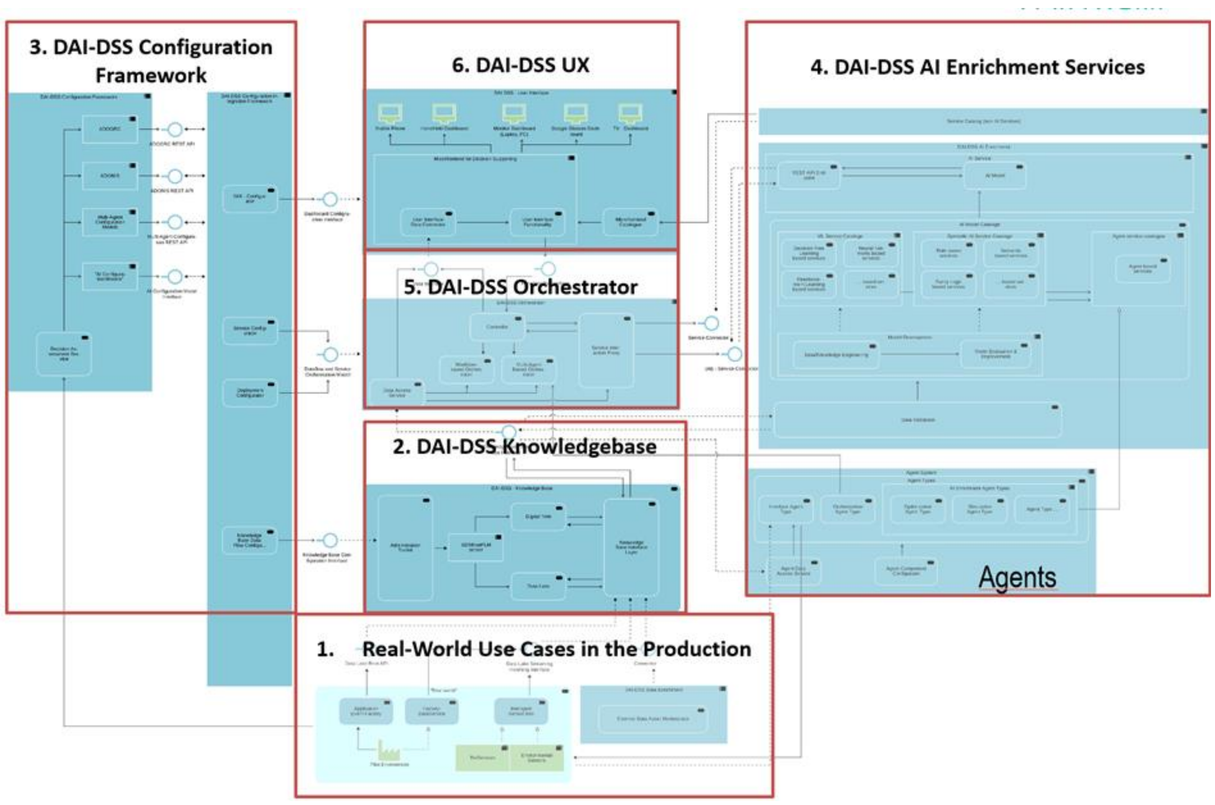

3. DAI-DSS Configuration Framework
6. DAI-DSS UX
4. DAI-DSS AI Enrichment Services
5. DAI-DSS Orchestrator
2. DAI-DSS Knowledgebase
Agents
1. Real-World Use Cases in the Production

Figure 2 Overview of high-level architecture with key components (HQ figure in Annex B.1)

Decision models are created using various model environments and after their assessment they serve as input for the DAI-DSS Configuration Framework. The models not only describe processes and workflows but may include information for the Multi Agent System and needed AI services.

Depending on what should be modelled the model environment could be specialized to depict business processes, dashboards, or meta models. Using the DAI-DSS Configuration Framework model information can be extracted and summarized in the form of component's specific configuration files. These configuration files are used to create instances of different Microservices, which can be chosen from a service catalogue. Components dependent of the DAI-DSS Configurator are mainly the DAI-DSS Orchestrator containing the Multi Agent System and the DAI-DSS User Interface.

The DAI-DSS Orchestrator is a controlling component and oversees handling the interaction with the services and accessing the data from the knowledge base when required. If an individual service is called by the user a simple retrieval of data via the Data Access Service may be sufficient, while a combination of services must be orchestrated using a workflow engine or the multi agent system.

An important part of the services is enrichment through AI and are accessible in the form of an AI model catalogue. How this AI algorithms and models for assisting decision-making are developed is represented in the DAI-DSS AI Enrichment Services component. This component should enable the provision of a broad collection of AI services to support various decision-making challenges.

For the visualization the DAI-DSS User Interface uses a Microservice Framework to display results and information relevant for the decision maker in a web application. The Microservice framework consists of modular frontend Microservices, called widgets, which enable a versatile composition for the user interfaces. Widgets can be responsible for representing data and results from services, but they can have also a more interactive nature and allow the decision maker to interact with the different (AI)- services.

The DAI-DSS Knowledge Base has an important role when it comes to decision making, because huge amounts of data from various sources may influence the decision makers choices. The storage and provision of data relevant for the decision process is provided by the DAI-DSS Knowledge Base. Not only decision models are retrievable, but also data needed to train AI models are stored. The knowledge base includes digital shadows and a data lake, where data, e.g., retrieved from sensors, coming from the "real world" is stored. This sensor data can come not only from physical objects (e.g., machines or robots), but also from humans wearing specific sensors, which are able to capture human factors. The laboratory environment is used to evaluate experiments in advance, therefore abstraction challenges of the "real world" or factory environment, allowing for the execution of experiments during a decision-making phase on one hand, and the import of real-world data for experiments on the other. In the laboratory different decision-making approaches can be tested and compared with each other to find a fitting configuration method. Access to the experiments is possible both remotely and directly from the lab

2.2 Use Case Mapping with the DAI-DSS High Level Architecture.

Note: This section will be further detailed in the next version to be delivered in M20.

The DAI-DSS high-level architecture is mapped with the Use Cases of FLEX and CRF which are identified and developed via the Design Thinking and Business Process Models in section 2 of deliverable D2.1. The Use Cases are outlined in below Table 1. For each Use Case a detailed business process scenario is developed where the following is identified:

- data source requirements
- KPIs
- where DAI-DSS is required in the decision-making process

The use cases are abstracted to following challenges:

- “Finding similar projects”,
- “Find relevant experts”,
- “Simulate production process”,
- “Allocate worker”,
- “Map workers with profiles”,
- “Find similar problems”,
- “Reschedule production line”,
- “Allocate order to production line”,
- “Assess the impact”.

This is done to create a decision support solution that is beneficial for the stated use cases but also tailored to the demands of other use cases, i.e., in the manufacturing sector.

Table 1 Use Case Scenarios

	FLEX	CRF
USE CASE SCENAROIS	Automated Test Building	Worker Allocation
	Worker Allocation	Delay of Material
	Machine Maintenance After Breakdown	Quality Issues

In chapter 4, a list of potential AI services is proposed which offers the required decision support for the use cases. These AI services can use different methods such as machine learning, fuzzy logics, knowledge graphs, reinforcement learning, etc. for providing decision support.

Note: The Application service catalogue will be extended and further detailed in the next revision of this deliverable (M20).

We identify and map process requirements with the appropriate AI services as shown in Figure 3 for providing decision support.

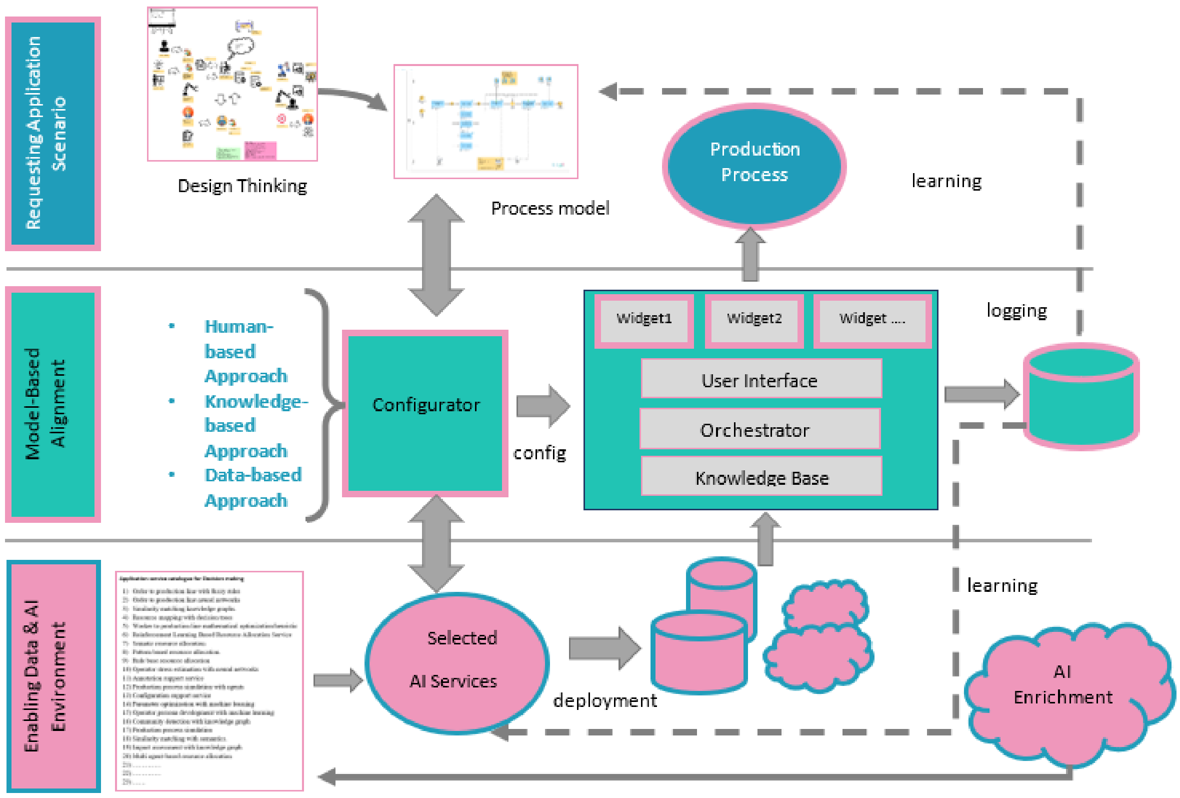


Figure 3 describes the model-based alignment and configuration of AI Services to address the use case specific need.

The process model describes the use case scenarios and the DAI-DSS AI enrichment represents a catalogue of services for decision making. To provide decision support for the use case scenarios, the AI services from the catalogue are selected. This configuration or data sources and AI services are then used by the orchestrator and the user interface to provide a solution to the end user. The data required for the services is provided by the knowledge base. For each use case scenario an overview is provided, how may DAI-DSS support decision making.

2.2.1 Flex Automated Test Building Process

The business process for creating automated test building is depicted in Figure 4 and starts with a new product order or the requirement for a manufacturing process update. The automation engineer oversees organizing and designing the inclusion of the robots in the manufacturing process for a given request. The program manager provides this information together with order specifics (such as the needed cycle time, quantity, and client details). Cooperative mode is the recommended method of operation for robots, but collaborative mode is also a possibility. Because there are no people involved or present in cooperative mode, the robot may move more quickly, resulting in a reduced cycle time. In collaborative mode humans need to work close to the robots and the speed and motions of robots need to be reduced such that it will be safe for the humans in the range of motion. Here the automation engineer needs to consider several aspects like safety of workers, regulations, cycle time for designing a solution.

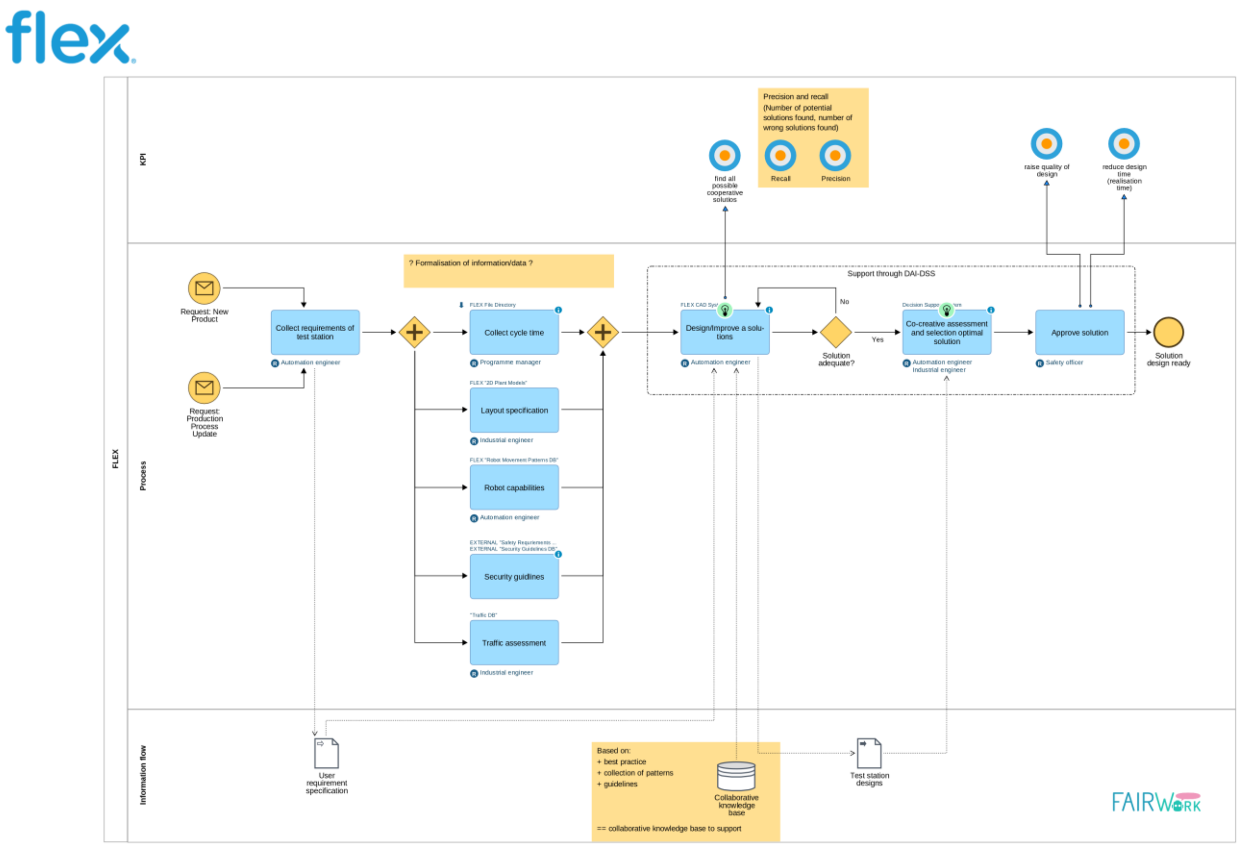


Figure 4 FLEX Automated Test Building Process

Here DAI-DSS may provide a decision support for the automation engineer for obtaining the best possible cooperative solution by considering the user requirements such as workers' safety, best usage of resources and robot capabilities, using the AI services described in chapter 4. The data required in this scenario is collected and stored in the DAI-DSS Knowledge Base. These AI services can have a functionality to search and match similar projects available in the Knowledge Base based on the input request received. If no good match is found, then requirements like cycle time, plant layout specifications and model configuration may be changed to retrieve a new result set from the Knowledge Base. AI services that predict the traffic in the shop floor may be added to support the selection of the best alternative. Finally, a cooperative decision from relevant experts with appropriate competences support the automation engineer. AI services to find the relevant experts with appropriate competences may be selected form the list of available services.

A list of potential AI Services from the application service catalogue for decision support are:

- **Find similar projects:** services like: “similarity matching with neural networks” or “Similarity matching with fuzzy rules”,
- **Find relevant expert:** service like: “Community detection with knowledge graph”,
- **Simulate of production process:** services like**:** “production process simulation” or “production process simulation with agents”.

2.2.2 Flex Worker Allocation Process

The second scenario concerns with worker allocation is shown in Figure 5. Allocating workers is required either before the shift begins or during the production when workers can’t continue to work due to different reasons. The challenge in both cases is determining the appropriate action of finding a suitable replacement for the unavailable worker. The first step is finding a suitable replacement, who is currently available and who has the required training to work on the machine. Each line has a "screenplay" that outlines the required abilities for operators, which must undergo a training to be capable in performing this “screenplay”.

The process starts by receiving a message that a work shift is cancelled due to a sick leave or another unexpected incident. Checking current workers availabilities and asking for the required "screenplay" to find those who are available. The workers' competencies are compared with the "screenplay". At the same time production plan must be considered. Additionally, it's crucial to assess the fairness factor and take worker preferences into account when reallocating resources. The best worker allocation is then determined after compiling a list of all possibilities. A risk assessment for not meeting production targets and a check on whether product quality requirements can be met should be included in this process.

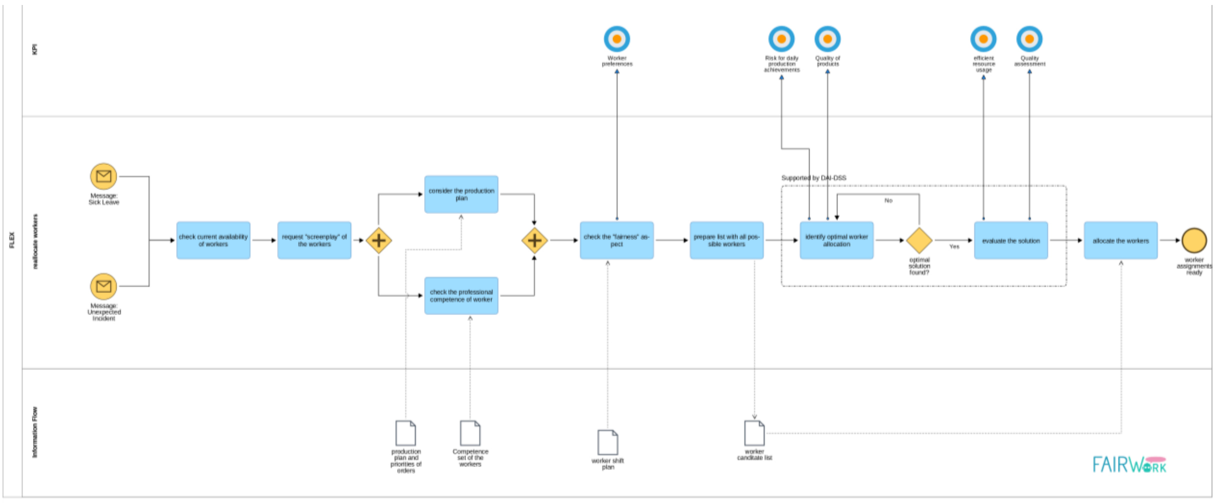

FLEX
KPI
reallocate workers
Information Flow
Message: Sick Leave
Message: Unexpected Incident
check current availability of workers
request "screenplay" of the workers
consider the production plan
check the professional competence of worker
check the "fairness" aspect
prepare list with all possible workers
Supported by DAI-DSS
identify optimal worker allocation
No
optimal solution found?
Yes
evaluate the solution
allocate the workers
worker assignments ready
Worker preferences
Risk for daily production achievements
Quality of products
efficient resource usage
Quality assessment
production plan and priorities of orders
Competence set of the workers
worker shift plan
worker candidate list
FAIRWORK

Figure 5 FLEX Worker Allocation Process

In this scenario we propose AI services for worker allocation process. These services must consider the aspects: worker preferences, risk for daily production achievements, quality of products, efficient resource usage and quality assessment, such that the solution obtained are not biased to an individual or to the process. The data from the legacy systems is collected and stored in the DAI-DSS Knowledge Base. The user can interact with the DAI-DSS framework using the DAI-DSS User Interface dashboard for viewing the results from AI services. The DAI-DSS Configurator will configure the dashboards for displaying the key results which helps the users for decision making.

The implemented AI services consider the fairness aspect in the optimization algorithms such that solutions obtained are fair and provide best working conditions for involved workers. This can be combined by using the AI services which can measure and maps the operators stress levels while working on a particular machine or in an environment. Based on this, worker profiles can be stored with preferred working conditions and machine information. This data can be used by the other allocation or optimization AI services to provide “fair” resource allocations. For worker and machine allocations the AI services can have a functionality to search historical databases for comparable situations such as the same missing worker, or the same machine incident and provide the required solutions. If there are no relevant matching cases found in the knowledge base, then the AI services can be used to predict the results for changes made to the allocation of workers or machines and based on these results, the required allocation can be made.

A list of potential AI Services from the application service catalogue for decision support are:

- **Allocate workers:** services like: “Workers to machines with neural networks” or “Worker to production line with mathematical optimization/heuristic”,
- **Map workers with specific profiles:** services like: “Operator stress estimation with neural networks” or “Multi agent-based resource allocation” or “Impact assessment risk, knowledge graph”.

2.2.3 FLEX Machine Maintenance After Breakdown Process

The machine maintenance scenario is shown in Figure 6. In this scenario a ticket is raised in the ticket system when a machine malfunctions. To ascertain if it was a straightforward human error or a technical problem, a technician will come to the machine. If it’s an operator failure, the issue can be quickly fixed; if not, the failure database must be reviewed to determine if the issue is already known. Issues that are known to exist in the database are resolved by initiating the relevant solution procedure.

If the issue is a novel kind, it will likely take longer to fix, and all available technicians must be contacted to find a fix for the issue as soon as possible. While the technicians seek to find a solution, countermeasures must be determined and put into place by rescheduling to other machines and reallocating the workforce while taking the production schedule into account. If the rescheduling operations are doable, they need to be carried out, and the production schedule must be modified. The new production plan must allow for the effective use of resources, permit the creation of items of the same quality, and fulfill orders on time. The new production plan must provide efficient resource usage, enable to production of same quality products, and fulfil the orders within the deadlines.

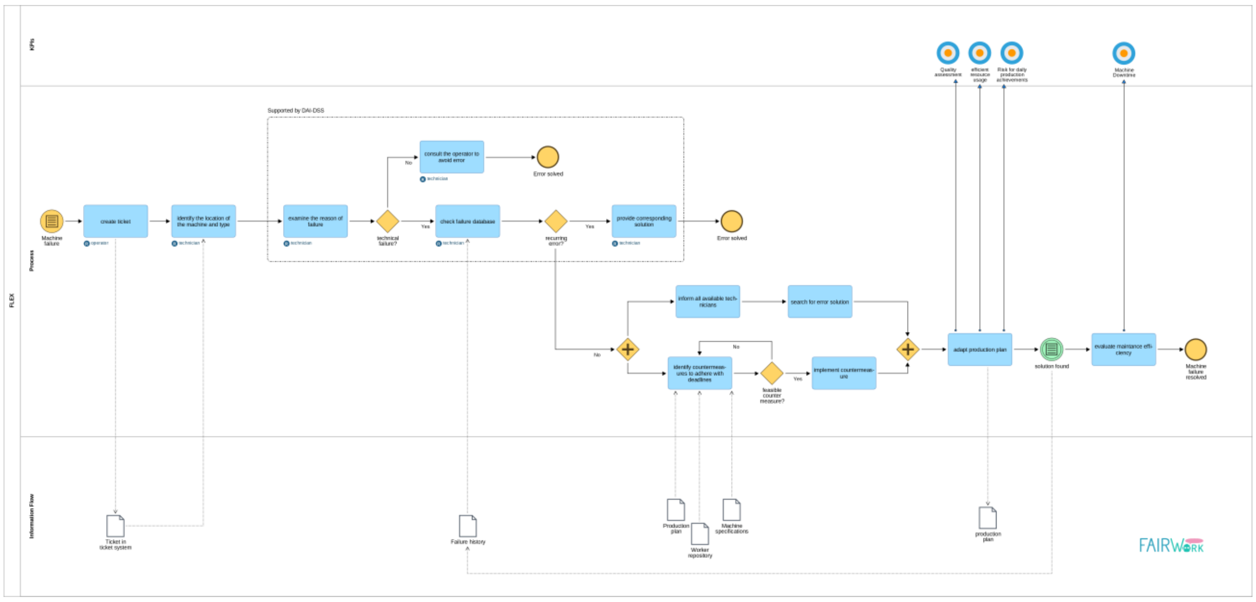


Figure 6 FLEX Machine Maintenance After Breakdown Process

In this scenario the DAI-DSS Architecture uses AI services from the Application Service Catalogue. The DAI-DSS Knowledge Base needs to collect the information from the ticket system, and it also need to store the failure history, production plan, worker information and machine specifications which are specified in the use case scenario. The AI services to support for this use case can have functionality for searching similar types of problems in the DAI-DSS Knowledge Base and provide recommended solutions. If the suggested solution involves replacing a machine part that must be ordered and is likely to take a few days to get and install on machine, in this situation the DAI-DSS Orchestrator can invoke AI services which have functionality for production rescheduling such that production capacity is not affected due to the failed machine in a line. These AI services will try to find best production rescheduling plan by considering quality assessment, efficient resource usage, risk for daily production achievements.

A list of potential AI Services from the application service catalogue for decision support are:

- **Find similar problems:** services like: “similarity matching with neural networks” or “Similarity matching with semantics”,
- **Rescheduling production line after machine break down:** services like**:** “production process simulation” or “production process simulation with agents” or “Production scheduling with reinforcement learning”.

2.2.4 CRF Workload Balance Process

The workload balance scenario is shown in Figure 7 and the describes a scenario for finding a good workload balance. The process is triggered by an event which makes an allocation or reallocation of workers necessary (e.g., planning the weekly production, a change in the production lines, etc.). The start event is followed by two parallel tasks, where one concerns availability of the operators and the other the production plan. The production plan specifies the production requirements (e.g., requested number of parts) and influences how many people are needed to fulfil all orders in time. Before assigning people to certain lines or positions, their availability and their capabilities need to be checked by accessing a database. Capabilities of operators include passed trainings, skills, and medical conditions. After all input data is collected, the allocation of workers is done in two stages, where both stages should be supported by the decision support system.

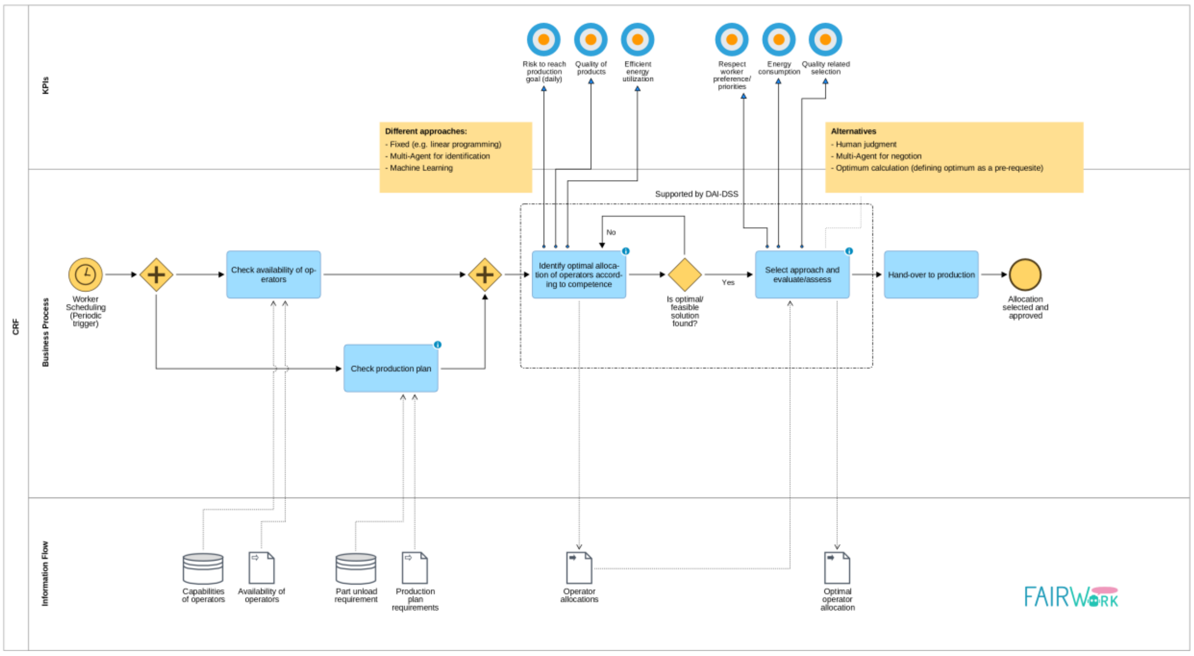

CRF
KPIs
Business Process
Information Flow
Risk to reach production goal (daily)
Quality of products
Efficient energy utilization
Respect worker preference/ priorities
Energy consumption
Quality related selection
Different approaches:
- Fixed (e.g. linear programming)
- Multi-Agent for identification
- Machine Learning
Alternatives
- Human judgment
- Multi-Agent for negotion
- Optimum calculation (defining optimum as a pre-requesite)
Supported by DAI-DSS
Worker Scheduling (Periodic trigger)
Check availability of operators
Check production plan
Identify optimal allocation of operators according to competence
No
Is optimal/ feasible solution found?
Yes
Select approach and evaluate/assess
Hand-over to production
Allocation selected and approved
Capabilities of operators
Availability of operators
Part unload requirement
Production plan requirements
Operator allocations
Optimal operator allocation
FAIRWORK

Figure 7 Workload Balance Process

This scenario we propose AI services for workload balance. It is possible to have different allocations like order allocation, worker allocation, machine and robot allocations using AI Services form the catalogue listed in chapter 4. Worker allocation is assigning workers to machines by matching with the required capabilities to operate the machines. The data source requirements for these services will be provided by the DAI-DSS Knowledge Base and the user will interact with the DAI-DSS User Interface for decision support. Furthermore, the required allocations can be organized as a workflow, where the required sequence of AI services – e.g., a service to automatically annotate a production plan and then a service that find best matches using the annotation as an input - be invoked by DAI-DSS Orchestrator. The solutions provided by the AI services aim to meet the daily production goal, respect product quality and are realizable in terms of machine and resource capacities. The feasible solutions are then further evaluated according to KPIs (e.g., workers preferences, energy consumption, quality etc.) and compared with each other.

A list of potential AI Services from application service catalogue for decision support are:

- **Allocate orders to production line:** services like: “Order to production line with fuzzy rules” or “Order to production line neural networks” or “Order to production line decision tree”,
- **Allocate workers:** services like: “Workers to machines with neural networks” or “Worker to production line with mathematical optimization/heuristic”,
- **Map workers with a specific personal profile:** services like**:** “Operator stress estimation with neural networks” or “Multi agent-based resource allocation” or “Impact assessment with knowledge graphs”.

2.2.5 CRF Delay of Material Process

The scenario delay of material delivery is illustrated in Figure 8.This scenario starts with incoming orders with corresponding order specifications. The first step is to identify the amount and type of needed raw materials by using information about the level of inventory in the warehouse as well as a procurement list in case material is not yet stored. In this step material turnover rates might be calculated and used for information about material usage within the company. When there is not enough material stored at the warehouse, the procurement strategies must be adapted to these circumstances, as optimal warehouse management could prevent material shortages. If there is enough supply, identification of the line capacities and a ranking of the orders according to their priorities takes place. Also, the expected duration of one order is an important indicator for the allocation to the machines.

The optimal assignment to specific lines is done by UTE head (Elementary Technological Unit) and this person request the material from the logistic manager for delivery of material to the required line. If there is delay in receiving material due to shortage or other reasons the UTE head needs to reconsider the production plan to avoid down times. This could be preparing the line for different production by changing the machines stamps and reallocation of workers to production lines.

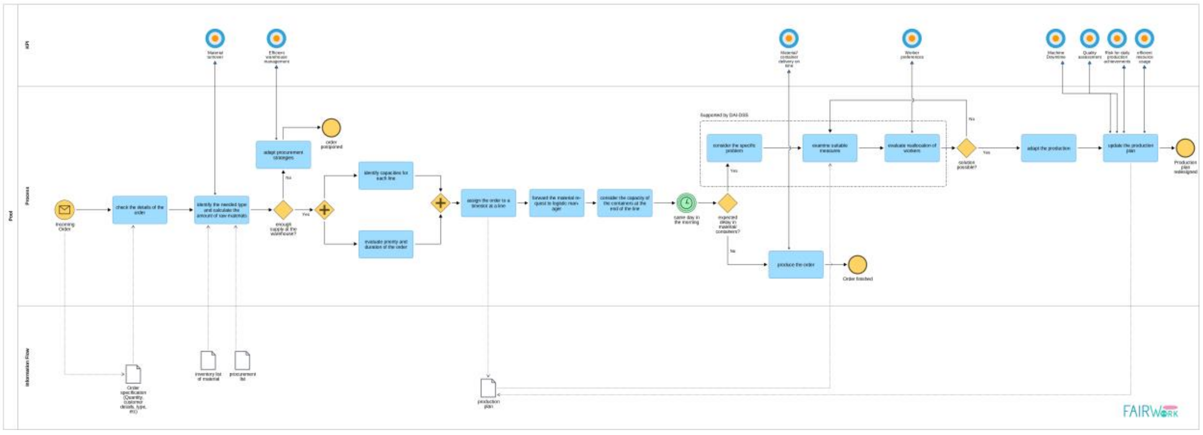

KPI
Process
Pool
Information Flow
Material turnover
Efficient warehouse management
Incoming Order
check the details of the order
identify the needed type and calculate the amount of raw materials
enough supply at the warehouse?
No
Yes
adapt procurement strategies
order postponed
identify capacities for each line
evaluate priority and duration of the order
assign the order to a timeslot at a line
forward the material request to logistic manager
consider the capacity of the containers at the end of the line
same day in the morning
expected delay in material/ containers?
Supported by DAI-DSS
consider the specific problem
examine suitable measures
evaluate reallocation of workers
solution possible?
adapt the production
update the production plan
Production plan redesigned
produce the order
Order finished
Material/ container delivery on time
Worker preferences
Machine Downtime
Quality assessment
Risk for daily production achievements
efficient resource usage
Order specification (Quantity, customer details, type, etc)
inventory list of material
procurement list
production plan
FAIRWORK

Figure 8 CRF Delay of Material Process

In this scenario we propose AI services which provides functionality for allocations such as order and worker allocation. For these allocations the AI services may use fuzzy rules or neural networks, and the feasibility of these solutions needs to be checked. The feasible solutions should be able to reach the daily production goal, respect product quality and should be realizable in terms of machine and resource capacities.

A list of potential AI Services from application service catalogue for decision support are:

- **Rescheduling a production line after material delays:** service like “production process simulation” or production process simulation with agents” or “Production scheduling with reinforcement learning” or “Order to production line with fuzzy rules” or “Worker to production line with mathematical optimization/heuristic”,

- **Reallocate workers:** service like “Workers to machines with neural networks” or “Worker to production line mathematical optimization/ Heuristic”,

- **Assess the production impact:** services like “Impact assessment risk knowledge graph” or “Production process simulation.”

2.2.6 CRF Quality Issue Process

The scenario "quality issues" is a recurring process of quality checks and is illustrated in Figure 9. It is either necessary for testing the quality of products that are produced after a line was set up for a new geometry and before production on full capacity or takes place during the ongoing production on a random sample basis. If a new geometry is produced the line must be set up and all necessary components for the line must be checked and prepared. The first molded products are then produced and checked for any quality issues. If the quality standards are fulfilled the product goes into full production. When the quality requirements are not met the deficiencies must be improved and the line must be adapted. The quality check is made by visually analyzing any surface defects on the part and based on the defects a categorization of the severity and type of defect is determined and if they defect can be rectified by the rework different ways of reworking the products are evaluated and then the products are reworked manually. If the rework not possible or feasible the products must be discarded.

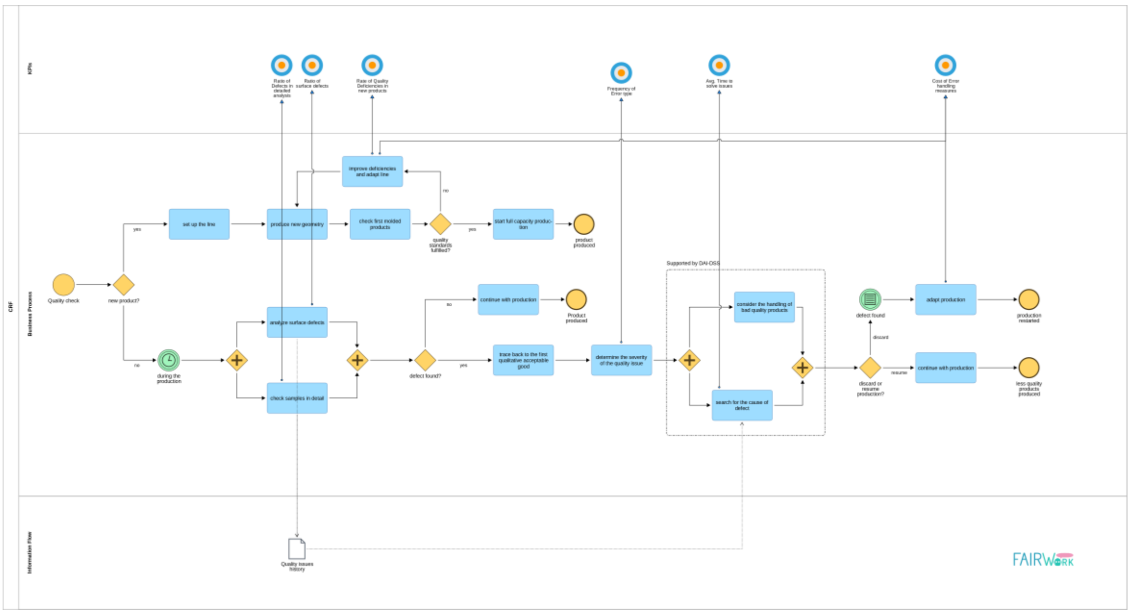

CRF
KPIs
Business Process
Information Flow
Ratio of Defects in detailed analysis
Ratio of surface defects
Rate of Quality Deficiencies in new products
Frequency of Error type
Avg. Time to solve issues
Cost of Error handling resources
Quality check
new product?
yes
no
set up the line
produce new geometry
check first molded products
quality standards fulfilled?
improve deficiencies and adapt line
start full capacity production
product produced
during the production
analyze surface defects
check samples in detail
defect found?
continue with production
Product produced
trace back to the first qualitative acceptable good
determine the severity of the quality issue
Supported by DAI-DSS
consider the handling of bad quality products
search for the cause of defect
defect found
discard
discard or resume production?
resume
adapt production
production restarted
continue with production
less quality products produced
Quality issues history
FAIRWORK

Figure 9 CRF Quality Issues Process

In this scenario we propose AI service which look for similar quality problems in the DAI-DSS Knowledge Base by matching the surface defect descriptions or image or categorization of the severity and type of defect. The AI services can also provide the cost of reworking a defective part. The user can interact with DAI-DSS User Interface dashboard and based on the input description from the user, DAI-DSS Orchestrator invokes required AI service to provide solutions. Although in general discarding a product is easier and more cost effective than reworking the product, the solutions provided by the services will also consider the sustainability factor.

A list of potential AI Services from application service catalogue for decision support are:

- **Find similar quality problems:** services like "Similarity matching with neural networks" or "Similarity matching with semantics",
- **Assess the impact and the cost of the quality issues:** services like "Impact assessment risk knowledge graph or "Production process simulation",
- **Rescheduling the production line after quality issue:** services like "production process simulation" or production process simulation with agents" or "Production scheduling with reinforcement learning" or "Order to production line with fuzzy rules" or "Worker to production line with mathematical optimization/heuristic".

3 ARCHITECTURE COMPONENTS

Each of the next sub-sections of this chapter describes the different DAI-DSS components introduced in chapter 1.3. The descriptions are structured in the following template sub-sections:

1. **Purpose**
   - The purpose of the component as part of the DAI-DSS framework
2. **Internal Architecture**
   - Component diagram describing the architecture of the component, also showing how it connects to the other components.
3. **Functions**
   - An overview of functional capabilities the component provides to either the user or to the other components within the framework.
4. **Dependencies**
   - How the component relates to, depend on, or is a dependency to other components.
5. **Interfaces and data**
   - Description of all external and internal interfaces, including mechanism for communication, and required import and export functionalities.
     - Input: Type of data passed through the interfaces, such as data structures, file formats, etc.
     - Output: Type of data passed through the interfaces, such as data structures, file formats, etc.
6. **Development focus area (optional)**
   - If this is based on an existing tool, application, library, or other type of module, which area within the component will require to be further developed to confirm to the project requirements.
7. **Other details (optional)**
   - If applicable, other specifics about the component that does not fit the template sections.

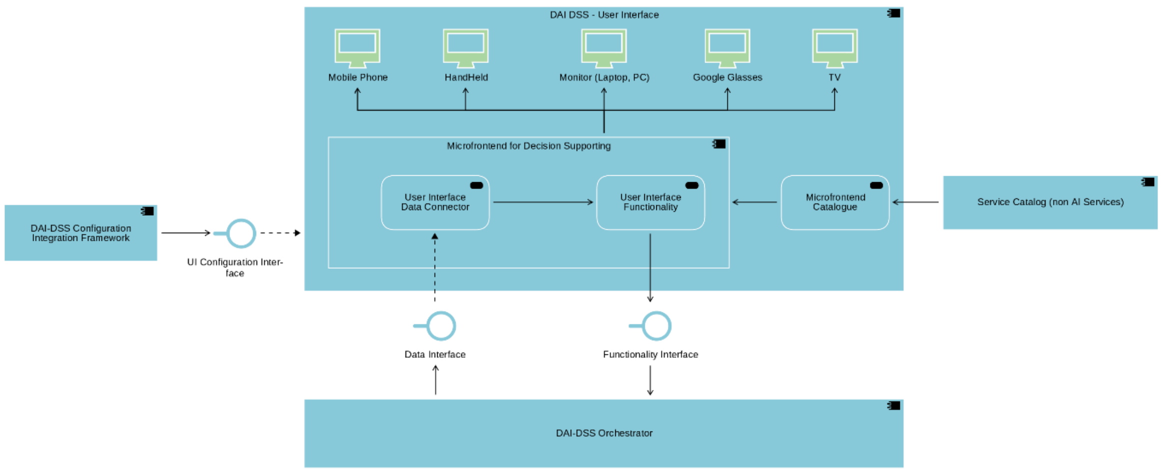
3.1 DAI-DSS User Interfaces

3.1.1 Purpose

The DAI-DSS User Interfaces provide an overview of the possibilities and options available to the user and recommend appropriate solutions for the decision maker. Depending on the situation the DAI-DSS User Interfaces can be displayed on various devices to fit the users need and working environment e.g., Mobile Phone dashboards, Monitor dashboards or Google Glasses dashboards. Besides from displaying valuable information to the decision maker, the user interfaces also enable the interaction with the available AI services (e.g., starting a simulation, trigger an allocation algorithm etc.). According to a high-level view, a DAI-DSS User Interfaces is an instantiation of model-based web widgets offered by a Micro-frontend using the OLIVE Framework.

3.1.2 Internal Architecture

The Micro-Frontend for decision supporting communicates with the orchestrator using Microservices and consists of two subcomponents. User Interface Data Connector consists of Microservices known as connectors, which are instantiations of pre-existing components offered by the Microservice framework. The Microservices are responsible to retrieve the data and information necessary for the UI provided by the DAI-DSS Orchestrator using the Data Interface. Not only the provision of data is relevant for the user, but also functionality is important. This part is provided by the second subcomponent, the User Interface Functionality, which communicates via the Functionality Interface with the DAI-DSS Orchestrator and enables the interaction between services and the users. The services, which are displayed in a customized user interface are chosen from a Micro frontend catalogue and need to be configurated before there deployment. The configuration files are created using the DAI-DSS Configuration Integration Framework and are provided through the UI Configuration Interface. The following Figure 10 gives an overview of the internal high-level architecture, and each component is described in more detail below.



DAI DSS - User Interface
Mobile Phone
HandHeld
Monitor (Laptop, PC)
Google Glasses
TV
Microfrontend for Decision Supporting
User Interface Data Connector
User Interface Functionality
Microfrontend Catalogue
Service Catalog (non AI Services)
DAI-DSS Configuration Integration Framework
UI Configuration Inter-face
Data Interface
Functionality Interface
DAI-DSS Orchestrator

Figure 10 High level internal architecture of DAI-DSS User interfaces

3.1.2.1 Back End for Decision Supporting Dashboards

The Micro-Frontend Framework extends the Microservice controller's backend methodology to the UI level. Utilizing the Micro-Frontend Framework, the various UI components known as Widgets are configured to create the application's appearance. The result of the configuration is a workbench containing the different widgets, where each widgets gives access to a different AI service or displays information coming from them. Figure 11 shows a mock-up of a possible user interface to give the reader a better impression how a workbench could look like. The Web Browser, on both desktop and mobile devices, as well as the MS Teams pages, are the presentation channels that the framework supports the most. For alternative channels like IoT devices and mobile apps, there is currently only a limited amount of support available. Widgets facilitate connections with Microservices defined in the framework to collect the necessary data or provide the desired user experience. They can be more generic, like visualizing layouts, tables, and charts, or more particular, like visualizing a DAI-DSS Dashboard. The Widgets can be chosen from the Micro frontend catalogue and then be configurated with certain parameters. Their configuration results in a new specific instance, which gets a new entry in the widget instances repository. Each instantiated widget or a combination can be finally exposed publicly by the framework to a specific endpoint, responsible to deliver the widgets and its configuration to the supported clients. The Micro-Frontend Framework allows the definition of different endpoints and associating them to the widget and its configuration to use. The web application is then automatically created through the Widget component in form of JavaScript file (for widgets targeting Web Browsers and MS Teams pages) and the Widget configuration in form of JSON file by rendering it in the right view. Each endpoint is associated with exactly one widget instance, consisting of the widget and its configuration, which results in the final layout refer Figure 11. The Micro frontend consists of two subcomponents, which are described in more detail below.

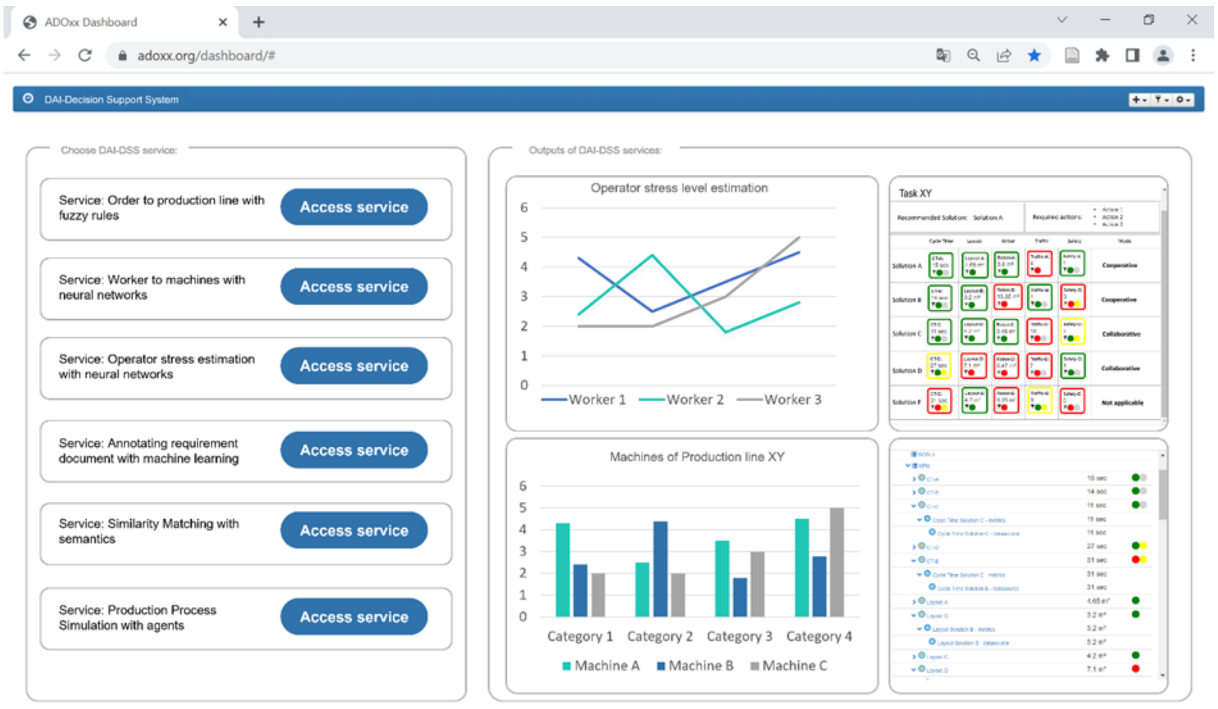

ADOxx Dashboard
adoxx.org/dashboard/#
DAI-Decision Support System
Choose DAI-DSS service:
Service: Order to production line with fuzzy rules
Access service
Service: Worker to machines with neural networks
Access service
Service: Operator stress estimation with neural networks
Access service
Service: Annotating requirement document with machine learning
Access service
Service: Similarity Matching with semantics
Access service
Service: Production Process Simulation with agents
Access service
Outputs of DAI-DSS services:
Operator stress level estimation
Worker 1
Worker 2
Worker 3
Task XY
Machines of Production line XY
Category 1
Category 2
Category 3
Category 4
Machine A
Machine B
Machine C

Figure 11 Mock-up of customizable User Interface

3.1.2.2 User Interface Data Connector

The accessible components for the backend are referred to as Connectors, and their configuration results in REST Microservices that are ready for usage. They are part of the OLIVE platform but, if necessary, they can be extended by using plug-ins. Connectors enable the connection to external systems and data sources. The OLIVE platform includes connection to more than twenty services and data storage systems right out of the box. After the configuration of a Connector a new Microservice instance is created, executed, and exposed through a REST interface, having a standardized input and output format. The User Interface Data Connector enables the data exchange between the external data source and user interface using configurated instances of Microservices. After their configuration these Microservices can access data from external sources e.g., the DAI-DSS Knowledge Base, the Multi Agent System or AI services by communicating with the DAI-DSS Orchestrator.

3.1.2.3 User Interface Functionality

Not only data is needed by the user interface, but also functionality should be offered to the user. This can be in form of providing basic functionalities like sorting or filtering entries by certain criteria but is also granting access to interact with more complex AI services and Multi Agent Systems. The User Interface Functionality subcomponent uses the Functionality Interface to interact with the DAI-DSS Orchestrator.

3.1.3 Functions

The DAI-DSS User Interface enables the interaction between one or more end users or decision makers and other important components of the DAI-DSS architecture, e.g., the DAI-DSS Knowledge Base or the Multi-Agent System. Clear representations of information and results on a dashboard facilitates decision making and should give the decision maker the possibility to react in an appropriate way.

Possible technical use cases are listed and described in the following:

- **Viewing different alternatives**
  Depending on the use case scenario the user interface can look quite differently, as it is formed out of a combination of various widgets. Some widgets could be interactive while others give the user an overview of the status or available data. The configuration of the web application takes place once during a set up phase, where needed Microservices, data and calculations are defined. After that widgets are chosen, and their appearance is configurated. The combination and set up of the widgets can be saved and forms the web application, which is regularly updated to show latest data and information refer Figure 12.

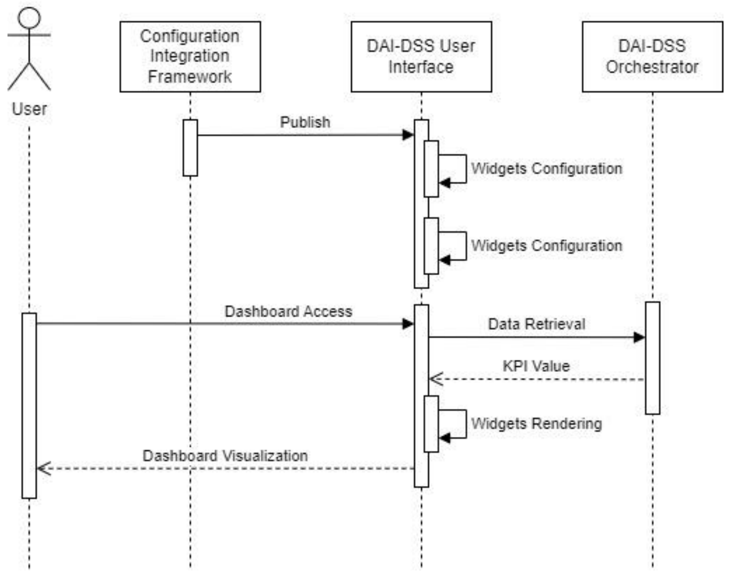


Figure 12 Use Case Dashboard View

- **Choosing an alternative:**
  As already mentioned above the widgets can also be interactive in the sense that the user can decide for different actions or requesting more detailed information by clicking on the corresponding element. This request is forwarded to the DAI-DSS Orchestrator and could triggering the access to the DAI-DSS Knowledge Base or a service, a workflow, or an interaction with the Multi Agent System. After the orchestrator has processed the request, the result is returned to the user interface component to display it in the corresponding widget refer Figure 13.

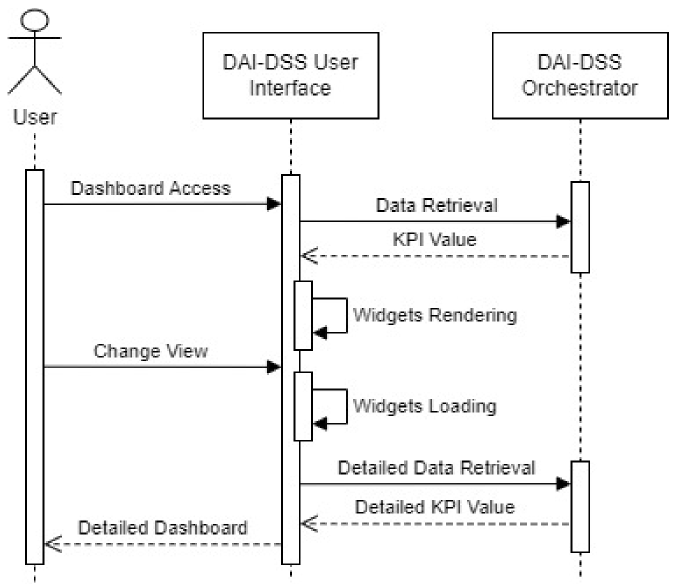


Figure 13 Use Case Alternative View

3.1.4 Dependencies

The DAI-DSS User Interfaces depend highly on two other components in the high-level architecture: the DAI DSS Configurator and the DAI-DSS Orchestrator. The DAI-DASS Configurator defines the dependencies between the User Interface, the business logic which is represented as a Microservice and the data.

The DAI-DSS Orchestrator provides data coming from different sources. Which data is needed it specified by the Microservices of the back end and/or the interaction with the user. Data can be values stored in the Knowledge base, results coming from different services or information resulting from the Multi-Agent System.

3.1.5 Interfaces and Data

As mentioned in the section above there are two interfaces to connect the DAI-DSS User Interface to its surrounding components. The Data Interface enables the communication between the User Interface and the DAI-DSS Orchestrator. The internal architecture consists of two main modules, which have the need to exchange data as well. The back end consists of three subcomponents which interact with each other and for the DAI-DSS User Interface to work the back end needs to communicate with its front-end. In the following Table 2 gives an overview of the inputs and outputs of the components and which kind of interfaces are used to interact.

Table 2 DAI-DSS User Interfaces API Overview

Interface	Input components	Output component	Input Data Type	Output Data Type
Configuration Interface	GUI Configurator	User Interface Model Connector	JSON containing configuration of widgets	nothing
Data Interface	Data Access Service, Orchestrator	User Interface Data Connector	JSON containing info of KPIs to retrieve	JSON containing the needed KPIs values

3.1.6 Development Focus Area

Development focus is to configure user interface based on pre-defined widgets or creating user interfaces from scratch using a low-code no-code environment and link the created user interface to Microservices and the corresponding dataflow. The goal is to enable the deployment of such user interface as flexible as possible hence enable the deployment on a Webpage or on legacy systems like Confluence or MS Teams.

3.2 DAI-DSS Configurator

3.2.1 Purpose

The DAI-DSS Configurator consists of two parts, the Configuration Framework, and the Configuration Integration Framework. The former enables the creation of decision models using several modelling environments (e.g., business process modelling tool, dashboard modelling tool, meta modelling tool etc.). Decision models can take various forms and manifestations. For example, by using meta modelling platforms like ADOxx[1], a process described in BPMN can be extended with different types of models, such as DMN, OWL, or Petri-Net and form an environment that describe decisions in an adequate way. The Configuration Framework enables the creations of these combined hybrid-models, and the model data serves as a basis for the second part of the component. The Configuration Integration Framework includes several subcomponents and enables the generation of configuration files derived from the before created models. In the configuration phase the appearance and functionality of a customized user interface, which can be envisioned as a workbench consisting of decision supporting services, is defined by configurating different aspects. This results in the creation of configuration files for Microservices, which are then used by the orchestrator and the user interface component. In addition, the configuration files should be stored in the knowledge base, so that they can be retrieved when needed. To have the possibility to evaluate and improve the configuration of the components, an assessment service retrieves feedback from stakeholders involved in the decision process and outcome. This feedback is then processed by updating the models and configuration files.

3.2.2 Internal Architecture

The DAI-DSS Configurator is based on the OLIVE[2] framework and its strength are its model awareness in that such configurations are abstract enough to be represented as models. Additionally, the out-of-the-box integration with the ADOxx modelling environment allows to design the appearance and functionality of your decision support system by creating models. The configurator receives various input in form of different models. These models can be resulting from different model environments and platforms, e.g., Dashboard Modelling Environment, Process Modelling Environment and Meta-modelling Platforms. These input models, which should represent the decision process, are then used by the configurator's internal subcomponents to create configuration files for the different Microservices. The internal subcomponents are the GUI Configurator, the Service Configurator, the Deployment Configurator, and the Knowledge Base Data Flow Configurator. The following Figure 14 gives an overview of the internal high-level architecture, and each subcomponent is described in more detail below.

[1] https://www.adoxx.org/

[2] https://www.adoxx.org/live/olive

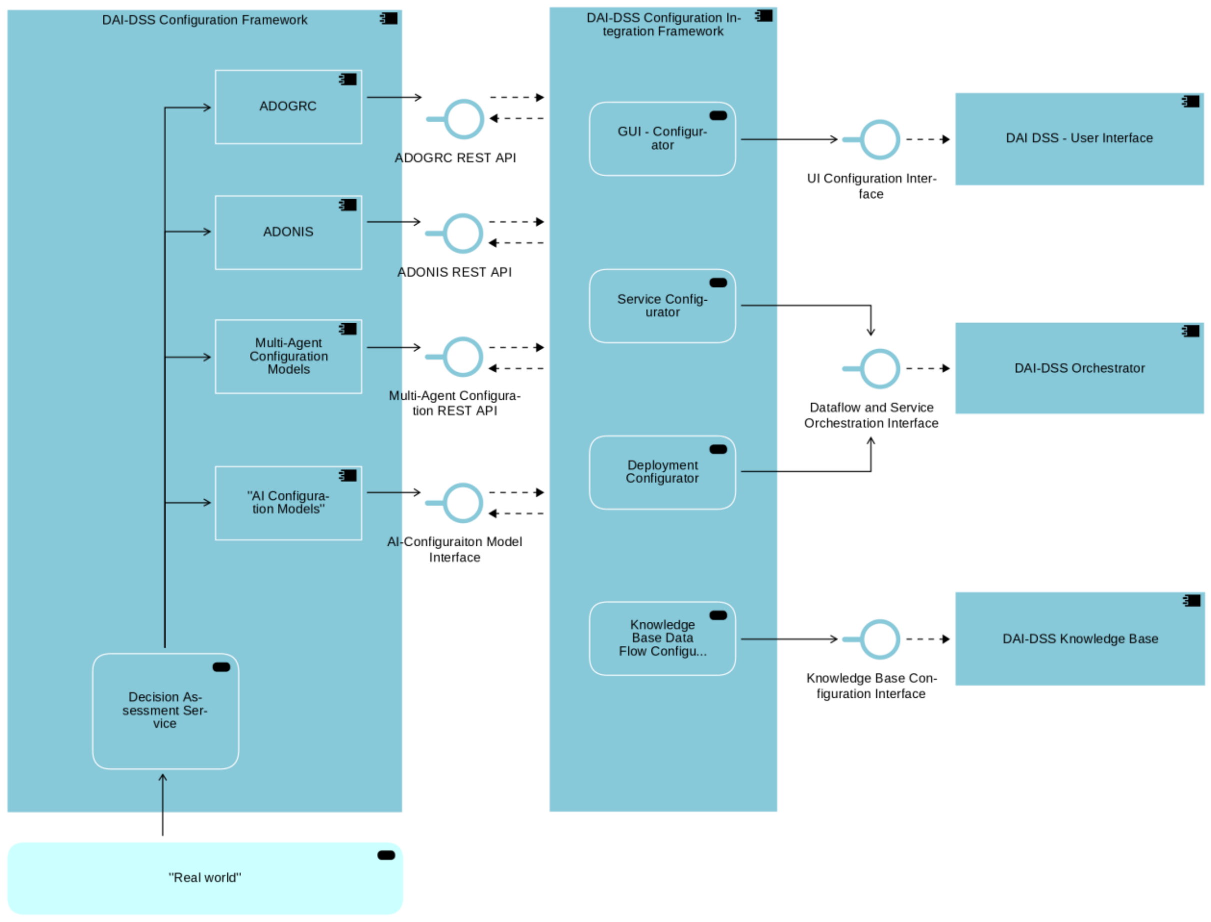

DAI-DSS Configuration Framework
ADOGRC
ADONIS
Multi-Agent Configuration Models
"AI Configuration Models"
Decision Assessment Service
"Real world"
ADOGRC REST API
ADONIS REST API
Multi-Agent Configuration REST API
AI-Configuraiton Model Interface
DAI-DSS Configuration Integration Framework
GUI - Configurator
Service Configurator
Deployment Configurator
Knowledge Base Data Flow Configu...
UI Configuration Interface
Dataflow and Service Orchestration Interface
Knowledge Base Configuration Interface
DAI DSS - User Interface
DAI-DSS Orchestrator
DAI-DSS Knowledge Base

Figure 14 High level internal Architecture of DAI-DSS Configurator

3.2.2.1 GUI Configurator

In the GUI Configurator environment, the appearance and layout of the individual services and the whole application can be defined. Models that include information about the visualization of services are relevant for this subcomponent. Such information could include how alerts should be displayed or what colors should be used if a KPI value exceed a defined threshold, but also how data should be represented on the monitoring dashboard. This component can be seen as a type of UI builder and allows the generation of customized user interfaces by configuring existing Micro-Frontend widgets.

3.2.2.2 Service Configurator

Decision models may include which data and services need to be triggered to find suitable solutions for the decision maker. This kind of information is relevant for the Service Configurator. The services which are needed to support a certain use case scenario are chosen from a service catalogue. The chosen services need to be configurated to fit the use case requirements using this subcomponent. Services can be autonomous or dependent from each other. In case services are dependent we can see them as a workflow, and they're inputs and outputs need to fit together. The configuration files for the services and workflows are forwarded to the DAI-DSS Orchestrator.

3.2.2.3 Deployment Configurator

After all services are configurated and the visual layout of the user interface is defined, the next step is to decide on which platform the web application should be deployed (e.g., Confluence, MS Teams etc.). This component so retrieves information not only specific for deploying the Microservices but also for their combination to achieve the needed business functionality.

3.2.2.4 Knowledge Base Data Flow Configurator

The Knowledge Base Data Flow Configurator accesses the Knowledge Base via Administration Toolkit to store configuration files and decision models for further use.

3.2.2.5 Decision Assessment Service

The different types of decision challenges represented in the decision model influence which services are needed. The decision models serve as a basis to generate the configuration files for the different components of the decision support system. The configuration then again can influence the outcome of the services and the decision process. Therefore, the decision models need to fulfil certain criteria and should be assessed before further processing. Also, feedback from the stakeholders, who are directly or indirectly involved with the decision support system, should be incorporated, to achieve continuous improvement. The assessment is done by the Decision Assessment Service by using for example questionnaires or polls. After the assessment the model is signed, to prevent unauthorized changes. The signed decision models are stored in the knowledge based, where they can be accessed if needed.

3.2.3 Functions

The DAI-DSS-Configurator enables creation of personalized decision models and the arrangement of the other architectural components by extracting the relevant model data and forward it to the specific component in form of a configuration file. Possible technical use cases are listed and described below.

- **Configuration of a DAI-DSS User Interface**:
  KPIs and services needed by the decision maker can be defined by using several modelling platforms. The output should be a model representing this information, which can then be further processed by the GUI Configurator. Model data relevant for the GUI Configurator concerns the frontend and the backend of the user interface component. As the user interface consists of front end Microservices, called widgets, each widgets needs to be configurated. If modifications are required, the user interface can be easily adapted by adding or removing widgets. This is accomplished by modifying the model to meet the new requirements. A new configuration file can be generated from the updated model. The user interface layout is defined by arranging widgets in the desired order. The configuration of the widgets and the layout is stored in the DAI-DSS Knowledge Base. If the configuration is finalized the user interface can be published on different devices. Figure 15 provide an overview of this configuration to express, how user interface, respecting the corresponding business logic that is encapsulated as a Microservice and the necessary data access is composed.
  The main idea of this DAI-DSS architecture is to enable the flexible configuration of user-specific interfaces and combine them with available AI service and the knowledge base. This type of flexibility enables the evolution of the use-case specific decision support system as both the user interface, the service and the knowledge base evolve.

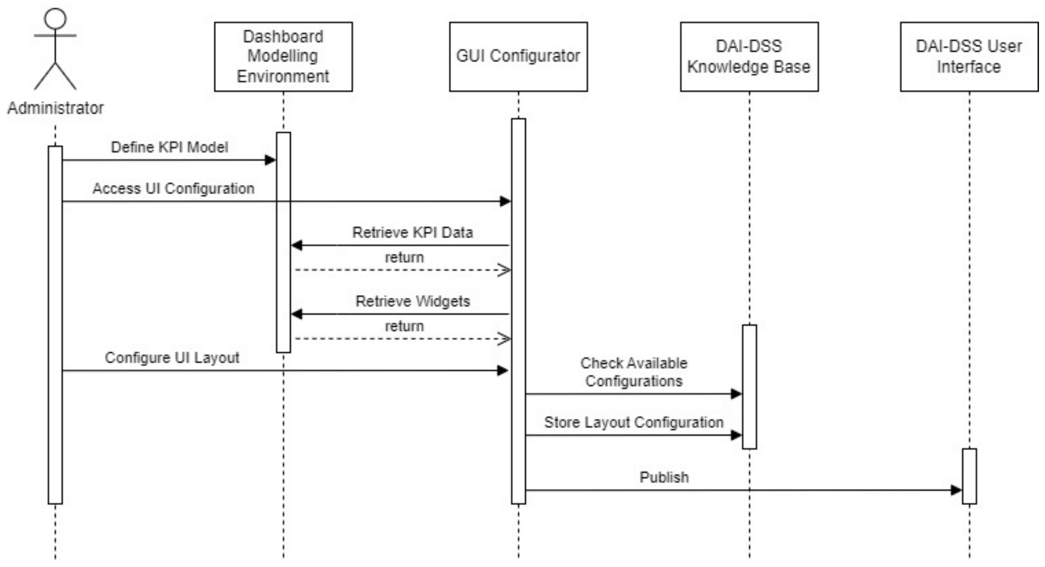


Figure 15 Sequence Diagram UI Configuration

- **Configuration of the DAI-DSS Orchestrator**:
  The modeller or administrator can use a process modelling tool to define services and their dependencies. From this model the needed services are chosen from a service catalogue and each service needs to be configurated using the service configurator. If services are dependent from each other the whole workflow needs to be configurated, so that service inputs and outputs fit together. The configuration files are stored in the DAI-DSS Knowledge Base. If all services are configurated, the configuration files are forwarded to the orchestrator component, which oversees the execution Figure 16. This counterpart to the UI configuration is necessary to not only design nice user interfaces but also select the corresponding business logic that is encapsulated as Microservices. The orchestrator hence not only combines the User Interface, the AI Enrichment services and the DAI-DSS Knowledge Base and hence is the central element of the DAI-DSS architecture, but also enables to build complex solutions by combining several AI services.

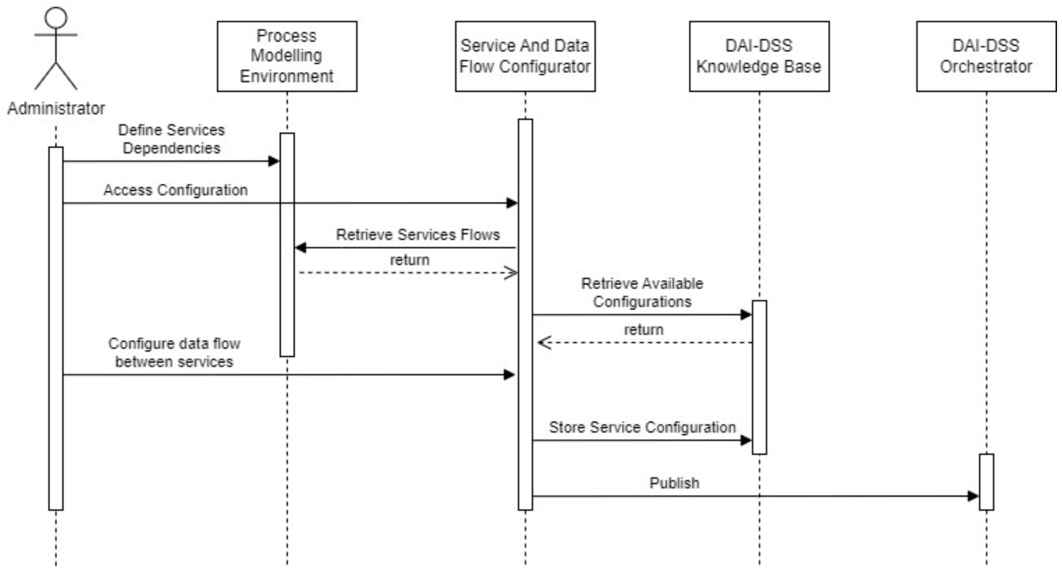

Administrator
Process Modelling Environment
Service And Data Flow Configurator
DAI-DSS Knowledge Base
DAI-DSS Orchestrator
Define Services Dependencies
Access Configuration
Retrieve Services Flows
return
Retrieve Available Configurations
return
Configure data flow between services
Store Service Configuration
Publish

Figure 16 Sequence Diagram Orchestrator Configurator

- **Creating and saving a decision model in the DAI-DSS Knowledge Base**:
  A Key element in centrally configuring decentralized AI services, is to provide a DAI-DSS Knowledge Base that enables the configuration of services mentioned in AI-Enrichment Services. In this sample we use a decision model. The decision model is created as a combination of different model types using suitable model environments. To guarantee that the decisions formulated in the model can be trusted an assessment phase is needed. The assessment can be done using questionnaires to get feedback from the involved entities. If the decision model passes this assessment process the model can be stored in the DAI-DSS Knowledge Base, where the various components can access it Figure 17.

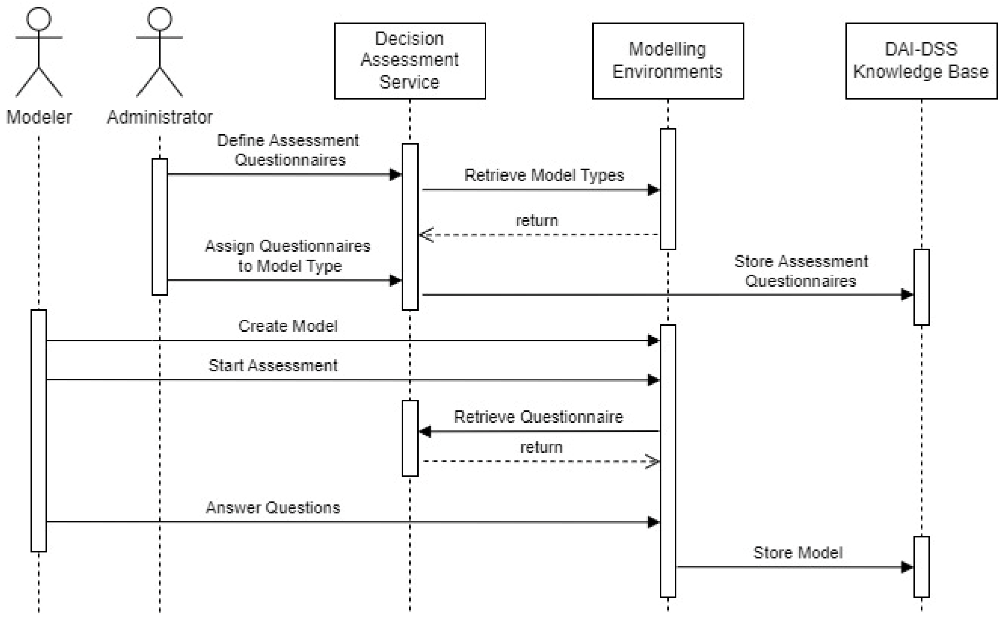


Figure 17 Sequence Diagram Decision Assessment

3.2.4 Dependencies

The DAI-DSS Configurator defines the settings for various components. Because this phase is repeated after each evaluation phase to ensure continuous improvement, the configurator is used not only at the start of the project, but also whenever changes are required. Because these changes affect other components of the decision support system, they are heavily reliant on the configurator.

3.2.4.1 DAI-DSS User Interfaces

The DAI-DSS User Interfaces are dependent on the DAI-DSS Configurator because the concept and layout for the user interface is derived from decision models, which include information about what front end Microservices are suitable for the use case scenario. The configuration data is extracted from the decision model using the GUI Configurator component. The result are configuration files for each widget and the whole layout of the user interfaces.

3.2.4.2 DAI-DSS Orchestrator

The DAI-DSS Orchestrator handles the communication between the Microservices. All these services must be configured prior to execution to meet the use case and technical requirements. Because the configuration files for each service and workflow are defined using the Service Configurator during the configuration phase, the DAI-DSS Orchestrator is dependent on it. After the services have been configured, they must be deployed. The DAI-DSS Orchestrator receives this deployment data in the form of a configuration file created with the Deployment Configurator.

3.2.4.3 Model environments and platforms

The DAI-DSS Configurator highly depends on the model environments and platforms used for creating the various models. This could include the interaction with Process Modelling Environments, Dashboard Modelling Environments and Meta Modelling Environments. A decision model created by the DAI-DSS Configurator can consists of different types of models. This model types may also include information relevant for the configuration of AI services and Agents to support the decision making.

3.2.5 Interfaces and Data

Different kinds of models are created using Dashboard-, Process-, and Meta-modelling environments. These models can be retrieved from the different model tools using REST APIs. The Configurator Framework combines them forming hybrid decision models and its subcomponents create the configuration files for a specific purpose in the needed format. From the configuration file a Microservice instance can be built and provided over a REST API, using a specific platform. A connector is a function offered by the OLIVE Microservice framework and oversees carrying out a certain task. OLIVE incorporates 24 connectors right out of the box. The following Table 3 DAI-DSS Configurator Interfaces Overview gives an overview of all interfaces relevant for the configurator and the corresponding input and output data types.

Table 3 DAI-DSS Configurator Interfaces Overview

Interface	Input components	Output component	Input Data Type	Output Data Type
Dashboard Model Environment REST API	Dashboard Model Environment	Configuration Integration Framework	JSON with model request	Dashboard Model
Process Model Environment REST API	Process Model Environment	Configuration Integration Framework	JSON with model request	Process Model
Multi-Agent Configuration REST API	Meta Modelling Environment ADOxx	Configuration Integration Framework	JSON with model request	Multi Agent Model
AI Configuration Model Interface	Meta Modelling Environment ADOxx	Configuration Integration Framework	JSON with model request	AI Configuration Model
UI Configuration Interface	GUI Configurator	DAI-DSS User interfaces	Decision Model	JSON containing layout information
Data Flow and Service Orchestration Interface	Deployment Configurator, Service Configurator	DAI-DSS Orchestrator	Decision Model	JSON containing service combination information
Knowledge Base Configuration Interface	Knowledge Base Data Flow Configurator	DAI-DSS Knowledge Base	Decision Model	JSON containing decision model information

3.2.6 Development Focus Area

The DAI-DSS Configurator is built using the OLIVE framework, that allows to create model aware web applications through configuration of existing components, both for the backend and for the frontend side. For the backend side such components are named Connectors, and their configuration results in ready to use REST Microservices. Because these parameters are sufficiently abstract to be expressed as models and because OLIVE comes pre-integrated with the ADOxx modelling environment, you may use models to design your web application's whole appearance and behaviour. For this project, the OLIVE framework needs to be expanded by developing Microservices that can connect to AI services and Multi-Agents Systems based on configuration files generated from decision models.

3.3 DAI-DSS Orchestrator

3.3.1 Purpose

The DAI-DSS Orchestrator is critical to the overall execution of the decision support system because it manages and coordinates the configured services and workflows needed to providing decision-making options for the end user. Services can be standalone or linked in the form of a workflow. While in the first case, the DAI-DSS Orchestrator is responsible for simply calling the individual service and retrieving the necessary data from the DAI-DSS Knowledge Base, the second option is more complex due to the involvement of more services. In this case, the DAI-DSS Orchestration is handled by a workflow engine or by a Multi-Agent System. The former is used when the task to be performed does not allow so much flexibility in rearranging the components of the system to be optimized, for example. When the relevant data model is more adequate to a workflow-based orchestrator, it will perform the orchestration of the components. Such configuration data is sent by the DAI-DSS Configurator via the Dataflow and Service Orchestration Model. When the task requires flexibility to rearrange its structure, the Multi-Agent System is a more adequate approach to perform the task, as through communication between autonomous entities that can negotiate interests among their peers, it allows an intelligent and distributed control and monitoring of the task execution.

In a summary, the DAI-DSS Orchestrator has a multitude of interactions with its surrounding DAI-DSS modules that seek to streamline the way information flows through the system, to organize the way communication can happen so that the competencies of each module are properly appreciated. It has the challenge of harmonizing the characteristics of each sector.

3.3.2 Internal Architecture

The DAI-DSS Orchestrator consists of several subcomponents and interacts with its surrounding components using different interfaces. Figure 18 High Level internal Architecture of DAI-DSS Orchestration gives an overview of the high-level architecture of the orchestrator and the multi agent system. Each subcomponent is described in more detail below.

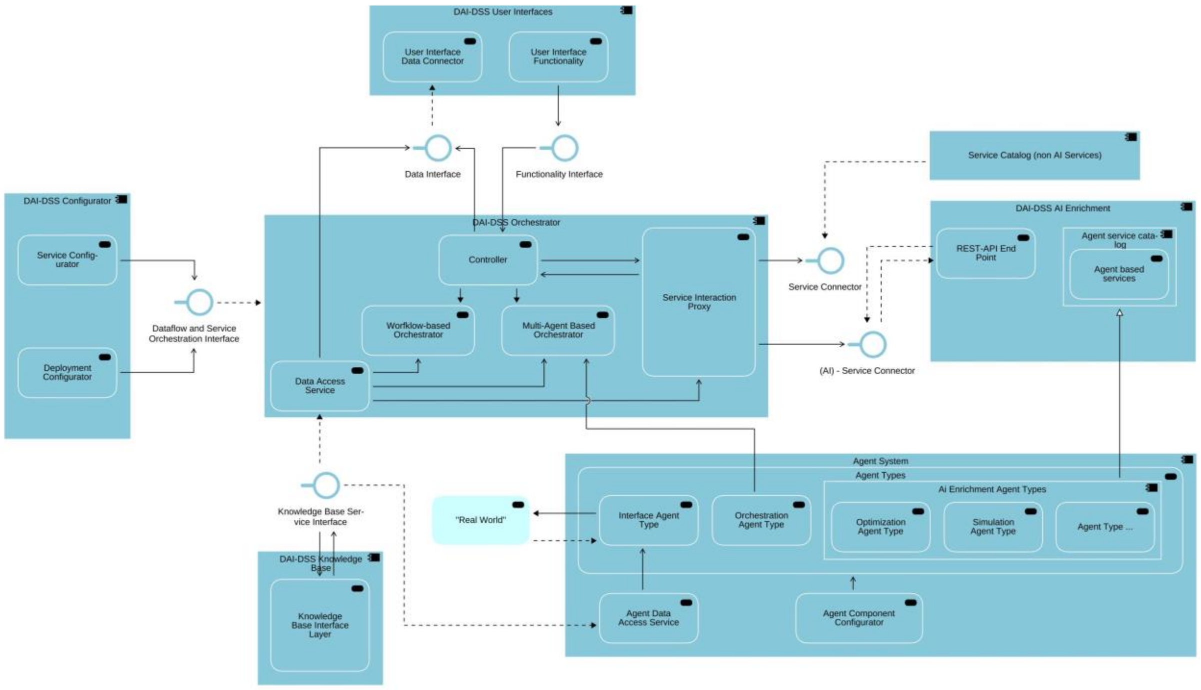


Figure 18 High Level internal Architecture of DAI-DSS Orchestration

3.3.2.1 Controller

The Controller allows to manage Microservices and controlling their whole lifecycle. The lifecycle management can include the following functions and activities: by using a configuration file coming from the configurator the controller can generate an instance of the Microservice. In addition, the Controller component allows to start a Microservice, to keep it running in an isolated environment, to stop it, and dismiss it. To handle the different type of orchestration the Controller rely on the Workflow Based Orchestrator and on the Multi-Agent based Orchestrator. It provides data to the User Interface Functionality Microservice via the Functionality Interface connector, receives data from the User Interface Functionality and triggers the Microservice orchestrator modules. In addition to that, it is connected to AI and non-AI service catalogs by the Service Interaction Proxy.

3.3.2.2 Service Interaction Proxy

The Service Interaction Proxy component enables the access to the services in the catalogue and to AI services provided by the DAI-DSS AI Enrichment. The controller can request the required service via this interface and feed the Workflow based Orchestrator with algorithms (AI or non-AI).

3.3.2.3 Data Access Service

The Data Access Service exchange data with the DAI-DSS Knowledge Base for storing historical data on decision-making recommendations provided to the end user and retrieval of data that can be necessary as input for the services. It also gives the recommendations to the DAI-DSS User Interface via the Data Interface connector.

3.3.2.4 Workflow based Orchestrator

Once the Microservices have been defined, they can be combined to accomplish the business logic task using the Workflow based Orchestrator component. This component is in charge of bringing together existing Microservices using the Enterprise Integration Pattern notation. To provide a higher level of freedom, the orchestrator also supports the use of the JavaScript scripting language to combine Microservices in a more programmatic manner.

A Microservice can be set up to serve as an orchestrator for other Microservices and, for more streamlined settings, it can rely on third-party services like Netflix Conductor[3].

3.3.2.5 Multi–Agent based Orchestrator

The DAI-DSS Orchestrator is also linked to a Multi-Agent System, which means that agents could be used for orchestration as well. The Multi-Agent system can be used for a variety of purposes, including digitizing physical entities, and providing agent-based AI approaches to assist in decision making in form of services. The Agent System module comprises the Microservices that define the agent types that will be needed in the project and how they interact with the DAI-DSS modules around them. Among the agent types, Interface Agent Type are defined, responsible for the deployment of agents in the real world, that is, building the multi-agent system based on the machines, robots and humans that are part of the manufacturing line. They interface with the assets of the production line. The Orchestration Agent Type works directly in orchestration. It interacts with the other agents to provide the Orchestrator module with the possible outputs that represent the decision support. The more details about Multi-Agent Systems are described in section 3.5.7.1 of this deliverable.

3.3.3 Functions

The DAI-DSS Orchestrator has a core role. It is a fundamental piece in the project architecture that unites the different DAI-DSS components while harmonizing each of their competencies in a central module. It could conduct the decision-making process based on the knowledge acquired and stored in the DAI-DSS Knowledge Base, on the improvements processed by the AI Enrichment component while following the configuration parameters of the DAI-DSS Configurator. Possible use cases are described as follows:

- Orchestration by a Multi-Agent Systems approach for scenarios where there is flexibility in the optimization possibilities.
- Workflow-based Orchestration where the possibilities are stricter.
- Storing agents in the DAI-DSS Knowledge Base and retrieving trained agents from AI Enrichment module.

3.3.4 Dependencies

The DAI-DSS Orchestrator module depends on the DAI-DSS Knowledge Base, DAI-DSS Configurator, and DAI-DSS AI Enrichment modules to function effectively. The DAI-DSS User Interface receives data processed by the DAI-DSS Orchestrator so it can offer support for a decision-making situation in a way that the system user can interact with the process. The modules work together to ensure the success of the decision-making support process.

The DAI-DSS Orchestrator dependencies are described as follows.

3.3.4.1 DAI-DSS Configurator

The DAI-DSS Configurator sets up the specific meta modelling data to the DAI-DSS Orchestrator depending on the service required. It provides the adequate metadata based on a meta modelling tool for the Orchestrator.

3.3.4.2 Multi-Agent System

The Multi-Agent System is responsible for providing agents for three different application scenarios. First, the digitalization of physical artefacts, second the multi-dimensional simulation and optimization and third an alternative way to orchestrate services. In conjunction with the DAI-DSS Orchestrator the latter is relevant. A workflow-based orchestration centralizes the orchestration within the workflow-engine, the Multi-Agent based orchestration decentralizes the orchestration by providing each agent – in our case Microservices – the capability to proactively

[3] https://conductor.netflix.com/

ask for invocation. The orchestrator is then no longer centrally invoking the different Microservice but distributes a registry of aspects or events, which the different agents – in our case Microservices – listen and start negotiation if an invocation is – according to the distributed registry – reasonable. The distribution of the registry and how the invocation is observed by the controller and to what extent needs to be worked out in detail to clarify this dependency.

3.3.4.3 DAI-DSS Knowledge Base

The DAI-DSS Knowledge Base stores the agents of the Multi-Agent System after configuration. The DAI-DSS Knowledge Base provides the necessary data to the DAI-DSS Orchestrator, so it processes and offer an intelligent solution as output. The Orchestrator access the Knowledge Base to get and update data when necessary.

3.3.4.4 DAI-DSS AI Enrichment

The DAI-DSS AI Enrichment is responsible providing the list of services that are orchestrated. Here we distinguish between non-AI services like traditional collaboration tools, search- query- or analysis tools, project management and knowledge management services as well as ticketing systems and the like. We assume that a reasonable combination of available tools that are correctly allocated to our use case scenarios will solve a reasonable set of user requirements. The AI services enriches the configured solution with AI. Here we distinguish between the symbolic AI that learn from human experts, and machine learning AI that learns from data and observations. A third category are the Multi-Agent Systems that introduce the distributed character and hence can be run as an AI service or as a combination of several AI services.

3.3.4.5 DAI-DSS User Interfaces

The DAI-DSS User Interfaces receives data outputted by the DAI-DSS Orchestrator. It represents the direct human interaction with the system where the decision options will arrive.

3.3.5 Interfaces and Data

The communication between the modules takes place via a REST-API. The queries carried out by each belonging module are directed to the desired endpoint so that the answer is direct and objective. Thus, to unify the communication process between the modules, the entities belonging to the data model are constructed based on the JavaScript Object Notation format. JSON allows the transfer and storage of data in text format and allows direct interpretation when used by any programming language. In this way, it is possible to guarantee that the answers to the queries are objects containing all the desired information, directly allocated in an array or in the form of metadata.

In this context, since communication between the modules belonging to the system can happen directly, for example between service from AI Enrichment and the DAI-DSS Knowledge Base, or indirectly, as when a request is made by AI Enrichment and the response comes from the DAI-DSS Orchestrator – i.e. workflow based or Multi-Agent System based - passing through the DAI-DSS Knowledge Base, the configuration must ensure that the response is deciphered.

3.3.6 Development Focus Area

The basic version of the Orchestrator is the controller that moderates between individual registered services. The Workflow based Orchestrator is based on the OLIVE Framework that will be extended to support EIP notation to model the service behavior.

Integration of external workflow engine will be also performed, and the focus will be on the open-source project Netflix Conductor[4]. Additionally, the integration with the DAI-DSS interfaces will be integrated in the system in form of configurable Microservices with specific connectors for each related DAI-DSS component. The Multi-Agent based Orchestrator will be connected and worked out as elaborated in the Multi Agent section.

3.4 DAI-DSS Knowledge Base

3.4.1 Purpose

The DAI-DSS Knowledge Base is the main data repository used in the DAI-DSS architecture. It is where all data required to be processed or analyzed is imported to and stored. It is also where results from decision makings are collected and stored. Other components within the framework uses the interfaces offered by the DAI-DSS Knowledge Base to retrieve or add data. The Knowledge Base is implemented with the EDMtruePLM application. EDMtruePLM has functionality to implement digital twins and digital shadows for physical assets where the asset's life-cycle data is stored and can be accessed by other DAI-DSS components. This is for example service of the DAI-DSS AI Enrichment which may use machine learning algorithms that uses data stored in the Knowledge Base for training, then performs prediction calculations and stores the results back into the Knowledge Base. Similarly, agents (of the Multi-Agent System) have digital representations in the DAI-DSS Knowledge Base. These representations have references to their configuration data, as well as agent actions and decisions. This data can be used for further optimizing the agents' performance. The DAI-DSS Configurator stores model information in the Knowledge Base. The purpose to have the Knowledge Base is to complement the traditional legacy systems and soiled data sources in manufacturing environment which cannot communicate with each other and establish a single source of truth for the DAI-DSS components which relies on the data from for providing decision support.

To store data in the DAI-DSS Knowledge Base a data collection framework will be implemented. This framework identifies three functional requirements:

- *collection identification*,
- *data collection* and
- *data pre-processing*.

The *collection identification* identifies and maps data source requirements that are required for DAI-DSS components and services for providing decision support. The *data collection* uses different hardware and software procedures to collect these identified data. The collected data is pre-processed by data wrangling techniques to produce high quality consistent data that is stored into the knowledge base that enables other DAI-DSS components to use this data for analysis, monitoring, visualization, AI/ML, etc.

[4] https://conductor.netflix.com/

3.4.2 Internal Architecture

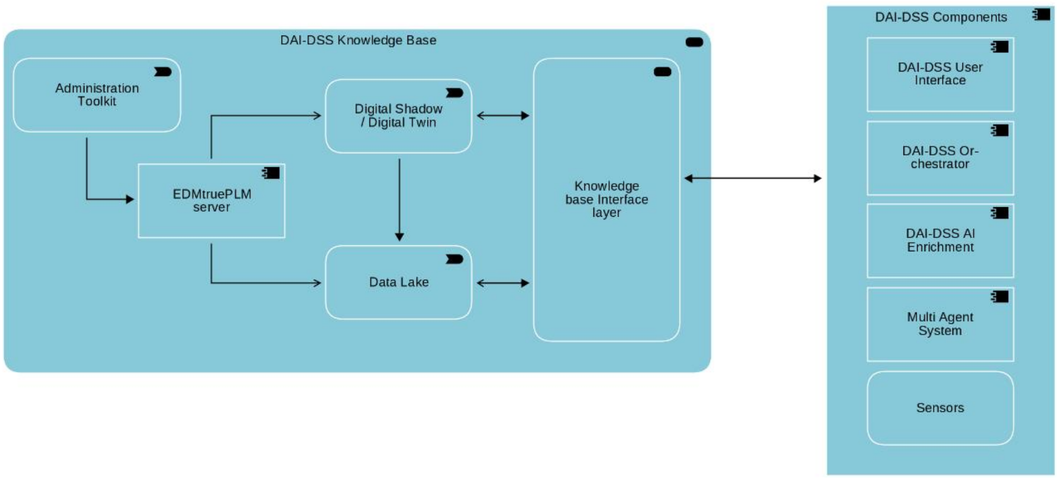


Figure 19 High level architecture of DAI-DSS Knowledge Base

The DAI-DSS Knowledge Base uses EDMtruePLM (TruePLM) to provide the data storage and interfaces required in the overall DAI-DSS Architecture refer Figure 19. TruePLM provides a server application which holds the Digital Twin/Digital Shadow representations of the physical assets in decision-making processes. The Data Lake is implemented as an EDM database. EDM (EXPRESS Data Manager), which is independent of EDMtruePLM, is a modular database management system designed specifically for managing EXPRESS schemas and product data following the ISO 10303 standard (STEP). The communication between a TruePLM server (and application servers) and an EDM database is done through an internal TCP protocol using the *EDMconnect* module.

This system implements, among others, ISO 10303-11[5], the EXPRESS language and ISO 10303-26[6], the Standard Data Access Interface (SDAI). An important aspect of EDMtruePLM is the PLCS (Product Life Cycle Support) data model, ISO 10303-239[7]. This is an ISO standard that enables data exchange in a world of heterogeneous systems and long-term archiving. TruePLM uses the PLCS concept of reference data to classify objects, relationships, and properties. A reference data class in PLCS points to an entry in a reference data library (RDL) as shown in Figure 20.

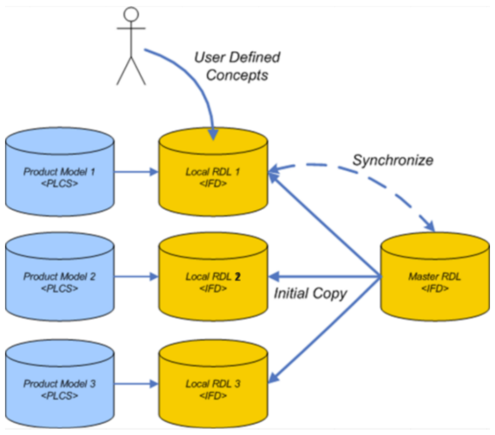


Figure 20 The use of reference data in the PLM Module

[5] ISO 10303-11:2004. ISO. (2019, December 15), https://www.iso.org/standard/38047.html
[6] ISO/TS 10303-26:2011. ISO. (2023, January 15), https://www.iso.org/standard/50029.html
[7] ISO 10303-239:2005. ISO. (2012, November 13), https://www.iso.org/standard/38310.html

PLCS requires two data elements to refer to a concept in a reference data library, an external class library id, and a class name. These two items should form a unique reference to a concept or a Uniform Resource Identifier (URI). IFD uses generated global unique identifiers as the unique reference id. The advantages of the PLCS approach are that the references are human-readable data, which makes debugging and data-browsing easier. This restricts that two different concepts cannot be given the same name within the same reference data library.

3.4.3 Functions

3.4.3.1 Data storage

The DAI-DSS Knowledge Base acts as data storage hub where all the product life cycle data is collected and stored using EDMTruePLM. Instead of interfacing with different IoT devices, external services, manual import/export, etc., all other components in DAI-DSS retrieves and stores data, results, and information in the DAI-DSS Knowledge Base. The EDM database can store both structured and unstructured data. Thus, the database will serve as a data lake to archive all relevant data for using ISO 10303 standards for data exchange, sharing and archiving (file exchange, APIs, and web-services).

The PLM module applies the following methods:

- Data interoperability: ISO 10303, STEP
- Data dictionary: ISO 10303-239
- Data import/export:
  - ISO 10303-242 AIM P21
  - ISO 10303-242 BOM XML
  - ISO 10303-239
- DBMS: EXPRESS Data Manager accessed securely using REST API methods.

3.4.3.2 REST API services and IoT integration

The REST API of EDMtruePLM provides management functionality for setting up projects, defining breakdown structures, uploading data or file, downloading data or files, uploading data to properties of data objects as single values or aggregates, and more. The REST API methods are also used for IoT integration.

IoT adds a new category of data to the domain of PLM product data. Such data originate in sensors and are streamed to a repository. EDMtruePLM not only receive such sensor data, but also links such data streams to relevant components in a product structure. Aggregate types of properties hold records of time-stamped sensor data. The user assigns such properties to the sensors that produce corresponding types of data and that are integral parts of a larger product structure in TruePLM.

REST API methods can be used to send streamed sensor data that may be sampled at a specific frequency. After storage, sensor data may be filtered for being browsed, retrieved via other REST API methods, or manually into downstream processing by, for example, AI or ML applications.

3.4.3.3 EDMTruePLM client

EDMtruePLM comes with a web-client interface. The web TruePLM application contains a front-end, developed in JavaScript with the VUE.js[8] framework, and a back-end, developed in Java with the Spring Boot[9] framework. Front-end and back-end communicate with each other through the REST API of EDMtruePLM. The descriptions of the REST API functions are on the EDMtruePLM swagger page[10]. The web application itself does not provide additional

[8] https://vuejs.org/
[9] https://spring.io/projects/spring-boot
[10] https://demo.jotne.com/EDMtruePLM/swagger.html

data storage; all data is stored on the server in the EDM database. So, each call from the front-end goes through the back end to the EDM server and back. The back end may use several calls to EDM for one REST API call. The back end and the EDM server communicate with each other through an internal TCP protocol, that is, the Java implementation EDMconnect. The web application, that is, the EDMtruePLM GUI may be executed by any of the modern web-browser refer Figure 21.

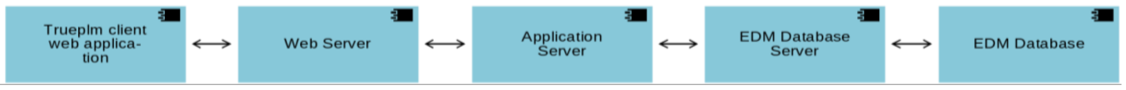


Figure 21 Interfaces between EDM TruePLM client to EDM database

The available client functionality depends on the type of user who is logged in. For details, see the EDM User Manual[11].

3.4.3.4 Digital Twin / Shadow representation

By holding the virtual representation of physical assets in EDMtruePLM and connecting properties of such assets to sensors or any other system producing real world measurements, giving a state of the items, we get a Digital Twin view. In this sense, EDMtruePLM acts as a Digital Twin for other application to harvest data from, perform processes, ML/AI calculations and the like, and feedback results into the repository. The solution is also standard based, more specifically, based on the open ISO 10303 standard, meaning that the whole repository can be exchanged to other systems, without having any complications of proprietary formats.

A complete digital twin is the result of combining several technologies into a system to acquire the required functionality. In the case of a DT for a product that will be modelled from prototype to operation, the following constituents would be required for full functionality.

- The physical part/product
- A sensor package collecting relevant parameters
- A gateway or similar hardware to collect and potentially filter and process sensor data
- A network for transmitting the collected data from the gateway
- An IoT framework
- A CAD or PLM-model of the part/product
- A data repository/platform for hosting the model and the data
- An analysis module that can run simulations on the model data
- Potential AI/ML capability for recommending actions

A *Digital Shadow* is a subset the Digital twin concept. The physical assets of a *Digital Shadow* have digital representations in the system where relevant data is streamed to and stored. This data can be used for getting current and previous states of the system but without any feedback to the physical assets from the digital representation (as opposed to a complete *Digital Twin*).

3.4.4 Dependencies

The DAI-DSS Knowledge Base is the place where all the data is stored, and it is from here the other DAI-DSS components access the data about the physical assets. This data is used for modelling, optimizing, and making predictions and decisions. It is therefore dependent on, and a dependency to multiple DAI-DSS components.

[11] https://jotne.atlassian.net/wiki/spaces/EDM/pages/3471409196/User+Manual+-+3.3

3.4.4.1 DAI-DSS Configurator

The DAI-DSS Configurator defines the decision models which are stored in the Dai-DSS Knowledge Base, that is dependent on how the models are exchanged and mapped. This is achieved either by manually mapping and replicating the models inside EDMtruePLM, storing model configuration files, or setting up automatic mapping between the two components via the available interfaces.

3.4.4.2 DAI-DSS Orchestrator

The DAI-DSS Orchestrator is responsible for setting up the Agents of the Multi-Agent System. These so-called agents are stored in the DAI-DSS Knowledge Base with their configuration. When the agents require optimizations or predictions the DAI-DSS AI Enrichment models access the agents stored in knowledge base.

3.4.4.3 DAI-DSS AI Enrichment

The Services from DAI-DSS AI enrichment uses the data from DAI-DSS Knowledge Base for model training and for predictions. The data exchange will be done using REST-API methods and the trained models are archived in the DAI-DSS Knowledge Base for future reference and optimization.

3.4.4.4 Data Sources

In the DAI-DSS Framework, data will be collected from many different sources; IoT devices, legacy systems, human bio sensors, maintenance data, and more. As mentioned previously, it's the Knowledge Base's role to handle these data, and it can be stated that the Knowledge Base is dependent of all these data sources and their interfaces.

3.4.5 Interfaces and data

3.4.5.1 REST API

The documentation for the EDMtruePLM REST API interface can be found in the EDMtruePLM Swagger page[12]. The main concepts covered by the API are:

- **Admin**
  - The common functions for administrative functionality
  - i.e., creation, edit and deletion of users, projects
- **Authorization**
  - The common authorization functions
  - i.e., access token generation for REST API credentials
- **Baseline**
  - The functions for baseline (also called snapshots) functionality
  - i.e., creation, update, deletion of baselines
- **Breakdown**
  - The common functions for breakdown structure functionality
  - i.e., create, edit, delete breakdown elements (folders/parts), modify breakdown element properties, append data to aggregate properties.
- **Data**
  - The common functions for documents functionality
  - i.e., Search, download, upload documents.

[12] https://demo.jotne.com/EDMtruePLM/swagger.html

- **Exchange**
    - The import and export functions for different types of data
    - i.e., export project as STEP package, zip package, or structure as text file.

This will be the main interface used to communicate with the DAI-DSS Knowledge Base by other DAI-DSS components.

3.4.5.2 EDMtruePLM web client

The EDMtruePLM web client acts as Administration toolkit in the DAI-DSS Knowledge Base which allows the users to create projects and access them. The web client enables the users to access the PLM server where they can set up projects and create required product break down structures for storing data. Furthermore, the GUI allows the users to model properties of physical assets as break down elements with reference data values. Aggregated data structures can also be defined, especially for initiating IoT device REST API end points.

3.4.6 Development Focus Area

3.4.6.1 IoT interfaces

EDMtruePLM supports the use of the *Arrowhead Framework* for streaming sensor data in addition to the mentioned REST API methods. The Arrowhead Framework provides an architecture for building IoT based Automation systems. The architecture features a local cloud approach where real time automation, IT security, system scalability and engineering simplicity are critical requirements. This framework can also be used for secured cloud-to-cloud data exchange, and this will be one of the focus areas to use this framework for streaming the IoT data into EDMtruePLM.

3.4.6.2 Notifications

The EDMtruePLM web client has the functionality to provide alerts in the form of notifications to the users. These are defined by specifying a set of logical rules when creating a break down element structure for sensor data. When sensor data is uploaded to these structures, checks are performed on the server to see if the rules are broken. This can for example be that a value is over or under some specific threshold (or more complicated rules that can be defined using arithmetic's and depend on multiple data sources). When a rule is broken, a notification to one or many users is raised.

An envisioned development area is to extend this functionality so that not only internal notifications and e-mail notifications may be raised, but also more generic notifications such as calling a specific REST-API method. This can be used to better create systems that automatically invokes some methods in external components, for example in other DAI-DSS components.

3.4.6.3 3D viewer in web-client

It is envisioned that the EDMtruePLM web client will be extended with a 3D viewer, specifically for viewing CAD STEP files. It is possible these integrations will allow links between the breakdown structures elements in a project to actual CAD parts and assemblies in a CAD model.

3.5 DAI-DSS AI Enrichment

3.5.1 Purpose

The AI Enrichment in DAI-DSS is a collection of services, which consists of various AI algorithms and models that contribute to decision-making by being connected to a Multi-Agent System (MAS). The main role of the AI Enrichment in the DAI-DSS environment is to provide a collection of AI models. The principle is to use explainable, traditional AI-algorithms as well as solutions that are more advanced for representation of the agents. Since each agent is unique, the selection of the model and its development is handled individually. Each of such models contributes to the decision-making process bringing closer the decentralized and fair approach.

3.5.2 Internal Architecture

The internal architecture of DAI-DSS AI Enrichment as well as its dependencies on other components is presented in Figure 22.

The first module in the AI Enrichment architecture is Data Validation. In this phase, all the information about the use case and involved interface agents is collected to enable data selection. This is also done under consideration of the objective, which should be met in decision-making. The database forms the basis for any data-driven modelling; therefore, the quantity and quality of the collected data have a considerable influence on the final model quality. Furthermore, the available data (e.g., process parameters, product quality, and human knowledge) is validated against the first model concept. If required, the model concept can be readjusted based on the available data, its type, quality, and size.

After the model has been selected, the model development stage is required. In this module, data and knowledge engineering is one of the most important tasks. The previously collected data may require customized preparation to be used for model development. This stage, depending on the needs, may require data cleaning, normalization, scaling as well as feature engineering. Such model may use capabilities from Multi Agent Systems, hence integrate their capabilities to predict or optimize systems. This enables the improvement of internal parameters of the model through the training. Furthermore, the validation of the model is supported. The resulting model is prepared to be applied on actual, real-time data where a request to receive the prediction for individual agent is communicated via REST-API endpoint and is being processed in AI Service module. The solutions are ultimately delivered and stored in a database for reference in future decision-making processes.

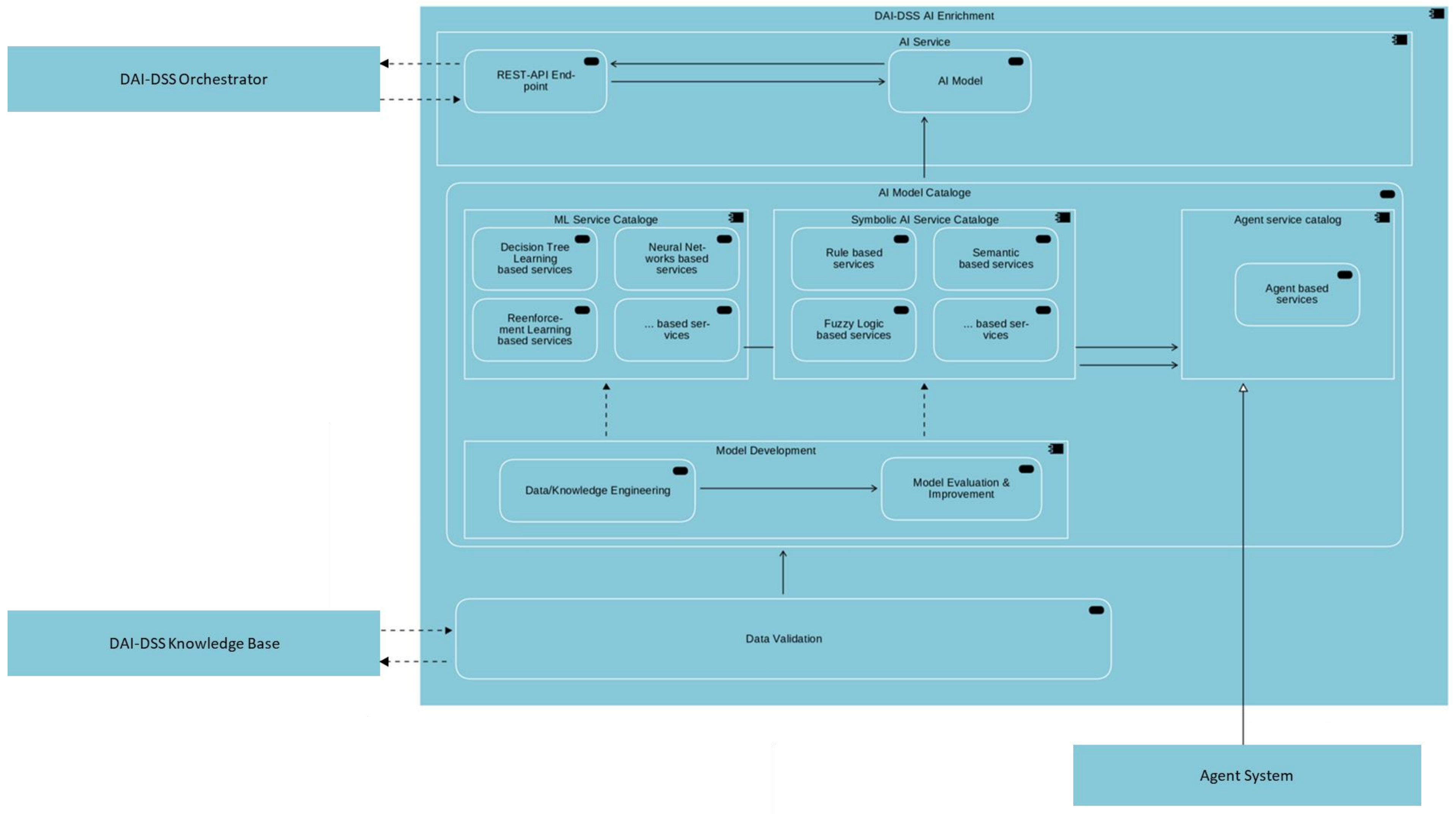


Figure 22 Architecture of DAI-DSS AI Enrichment

3.5.3 Functions

The various AI algorithms and models collected in the DAI-DSS AI enrichment module are connected to the multi-agent system (MAS) and provide predictions about the behavior of the agents or calculated suggestions for system optimization. Some of the possible methods that can be used for this purpose are described below.

Reinforcement Learning (RL) is about an agent interacting with the environment over time, learning an optimal policy, by trial and error, for successive decision-making. The agent therefore is responsible to find a policy that maximizes a reward it receives by interacting with its environment. The reward is derived from a problem specific reward function. In the context of industrial manufacturing, reward functions can be derived from process KPIs. In this approach, the agents can be modelled as neural networks and the RL method is used to optimize them. (Li 2018[13], Kaelbling 1996[14]) RL has been already successfully applied in various research fields like natural and social sciences and engineering. The capabilities of RL were also used in AI-based process optimization on the shop-floor level of a production (Samsonov 2021)[15] as well as on the machine level (Samsonov 2020)[16].

Decision Tree (DT) is a model that can be used for both classification and regression with a flowchart-like tree structure. It is built using two categories of elements: nodes and branches. Each internal node denotes a test on an attribute, following a branch represents an outcome of the test, and each leaf node holds a class label. During each test, a new numeric value is compared against a threshold or a set of values against a nominal attribute. This approach, consisting of logical rules, has an advantage over other models, such as artificial neural networks (ANNs), in terms of a more interpretable result. (Kotsiantis 2013)[17] With the DT structure, it is possible to logically trace the process to the final decision. However, it should be considered that as the dimensionality of the data increases, the interpretability of the model decreases.

Artificial Neural Networks (ANNs) are broadly applied to cope with challenges like pattern recognition, prediction, optimization, associative memory, and control. The ANN structure consists of an input layer, multiple hidden layers, and an output layer. Each layer internally consists of multiple nodes. The nodes of the individual layers are connected to one another and from a computational graph. Each input variable is represented by a neuron or node in the input layer. Likewise, the output layer consists of one neuron for each output. During the training, the network learns the weights between nodes from given dataset and following the performance of the model improves by adjusting the weights in the network with each iteration. One of the biggest advantages of ANNs over traditional systems is the ability to learn the underlying input-output relationships rather than following a set of rules specified by human experts. (AK. Jain 1996)[18] However, the development and training of ANNs require a great volume of data.

Semantic data is data that has been structured to add meaning to the data. This is done by creating data relationships between the data entities to give truth to the data and the needed importance for data consumption. Semantic data helps with the maintenance of the data consistency relationship between the data. Semantics are used to establish the integration of models, data, and AI, to infer domain knowledge in the data and formalizing it within AI. Integrating the semantics into an upper-level ontology allows for an abstraction of the tasks from both

[13] Li, Y. (2018): Deep Reinforcement Learning. Online: https://arxiv.org/pdf/1701.07274.pdf
[14] Kaelbling, L. P.; Littman, M. L.; Moore, A. W. (1996): Reinforcement Learning: A Survey. In: Journal of Artificial Intelligence Research, pp. 237–285
[15] Samsonov, V.; Kemmerling, M.; Paegert, M., Luetticke, D.; Sauermann, F.; Guetzlaff, A.; Meisen, T. (2021): Manufacturing Control in Job Shop Environments with Reinforcement, Learning. In: *ICAART*, pp. 589–597
[16] Samsonov, V.; Enslin, C.; Koepken, H. G.; Baer, S.; Luetticke, D. (2020). Using Reinforcement Learning for Optimization of a Workpiece Clamping Position in a Machine Tool. In: *ICEIS*, pp. 506-514
[17] Kotsiantis, S. B. (2013): Decision trees: a recent overview. In: *Artificial Intelligence Review*, Vol. 39, pp. 261–283
[18] Jain, A. K.; Mao, J.; Mohiuddin, K. M. (1996): Artificial Neural Networks: A Tutorial. In: *IEEE Computer Society*, Vol. 29, No. 3, pp. 31-44

use-cases and thus enhancing interoperability of models and operations in production. The semantic modelling found already the usage in the domain like product assembly planning (Wang 2007)[19] as well as for detecting and interpreting damage in an automatic process, where the predefined rules based on expert knowledge, the detected anomalies were classified and assessed automatically. (Hamdan 2021)[20]

3.5.4 Dependencies

The DAI-DSS AI Enrichment module is a collection of services, which consists of various AI algorithms. The AI algorithms cannot be developed without digitalized data and therefore, the AI Enrichment strongly depends on the format, quality, and quantity of available data for given use case. A dependency also occurs with the DAI-DSS Knowledge Base, where data is collected and stored to be further linked to the DAI-DSS AI Enrichment module.

In addition, the type of individual model depends on the use case scenario and available data.

3.5.5 Interfaces and Data

Once the model is developed, it needs to be deployed and integrated into the overall System. This is typically realized via a REST-APIs. In principle, one could write the code and define all REST-API endpoints manually with a REST-API Framework like Flask[21] or FastAPI[22]. However, TensorFlow Serving[23] allows to generate all REST-API endpoints with a single command and has some additional benefits that are useful in a production setting. An important benefit of TensorFlow Serving is that multiple requests can be aggregated in batches and evaluated by the model at once. This yields better hardware resource utilization and faster response times. Additionally, TensorFlow Serving allows easy versioning of Machine Learning Models. It can deploy multiple versions of the same model, which is useful for migrating from one version to another without any downtime.

3.5.6 Development Focus Area

The DAI-DSS AI Enrichment component targets the research and development of digital and intelligent agents as well as their application in selected industrial use cases. Therefore, the focus is on applying and combining multiple AI algorithms and models in the DAI-DSS environment with the objective of achieving a decentralized, fair decision. The motivation is a desire to incorporate a more democratic decision support system for all agents involved in a complex and dynamic environment such as production lines.

3.5.7 Other Details

3.5.7.1 Multi-Agent System

3.5.7.1.1 Definition of an Agent

One of the most widespread agent definitions is: “an autonomous component that represents physical or logical objects in the system, capable to act in order to achieve its goals, and being able to interact with other agents, when it doesn’t possess knowledge and skills to reach alone its objectives” (Leitão, 2009)[24].

[19] Wang H.; Xiang D.; Duan G.; Zhang L. (2007): Assembly planning based on semantic modeling approach. In: *Computers in Industry*, Vol. 58, No. 3, pp. 227-239
[20] Hamdan, Al-H.; Taraben, J.; Helmrich, M.; Mansperger, T.; Morgenthal, G.; J. Scherer, R. (2021): A semantic modeling approach for the automated detection and interpretation of structural damage. In: Automation in Construction, Vol. 128
[21] https://flask.palletsprojects.com/
[22] https://fastapi.tiangolo.com/
[23] https://www.tensorflow.org/tfx/guide/serving
[24] Leitão, P. (2009): Agent-based distributed manufacturing control: A state-of-the-art survey. In: Engineering Applications of Artificial Intelligence, Volume 22, Issue 7, Pages 979-991.

The agent is an entity capable of cooperation with its peers performing its own choices in decisive matters without human interference. They interact with each other to achieve goals that are beyond their individual capabilities, however such goals are efficiently and robustly achieved with their social competencies.

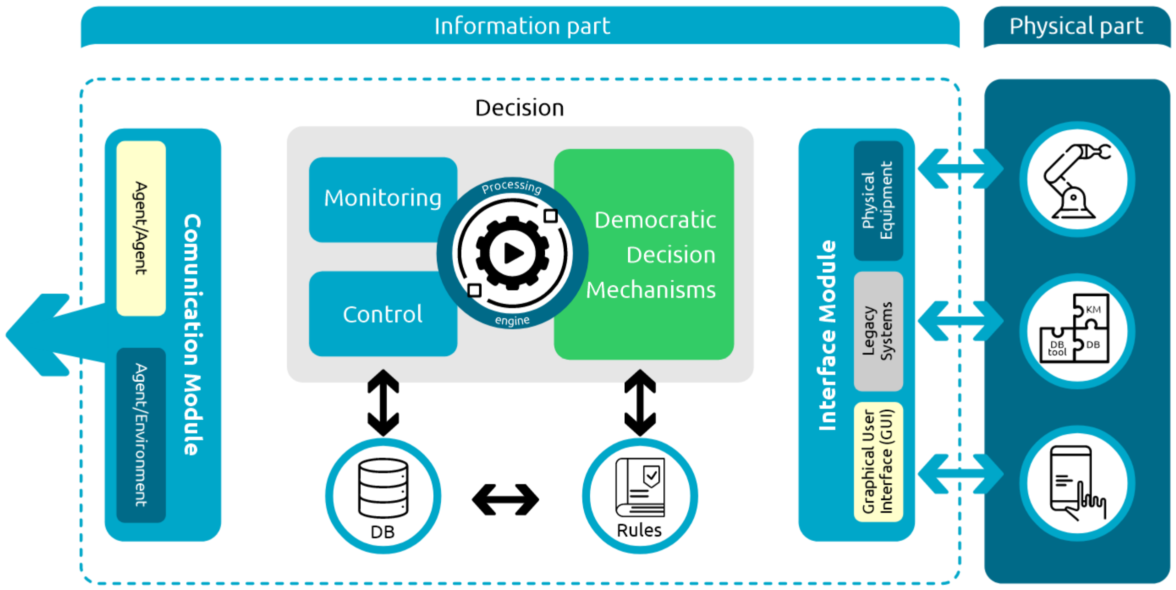


Figure 23 Multi-Agent System: Internal Architecture

In Figure 23 is described the internal architecture for a generic agent as. It is basically composed by two sections: information and physical sectors. In the information part, the decision is taken. The main section of the agent internal structure is the decision module. The Processing Engine will process data from the database where data collected from assets with embedded agents will monitor the system components and control their behavior based on those inputs. From the database it is possible to exchange references to adjust functional parameters of the robots/machines to set rules that will guide the decision-making while considering the human factor in the production chain sustainably. The decision is taken as democratically as the system can achieve cooperation through the Multi-Agent System and as fairly as it can orchestrate with AI optimization and prevision capabilities.

There is a communication module responsible for connection to other agents and the environment. The agent capacity on processing information itself and regulating internal and external communication is based on how the architecture is designed. The interface module provides a connection to physical devices where each agent, representing an asset, e.g., a conveyor robot, a milling machine, a robot arm, can be interfaced with legacy systems.

3.5.7.1.2 Communication between agents

The society of agents must exchange information as messages and for this it is necessary to use standardized mechanisms or protocols such as FIPA (Foundation for Intelligent Physical Agents). It is a standardization organization that develops specifications for Multi-Agent Systems. FIPA specifications define a common framework for the development and deployment of agents in the manufacturing industry. This framework includes specifications for agent communication and agent management. Through FIPA Agent Communication Language (ACL), it is possible to deploy and embed agents in a large scale and properly promote a distributed monitoring and control system. The communication between agents can be direct, broadcasted or from an agent to a specific group of agents while also being synchronous or asynchronous. Another aspect is that the communication can be based on the content of the messages that are being exchanged between agents or it can be based on the context of the negotiation which considers system's state and agent's goals now.

3.5.7.1.3 Communication between agents and other DAI-DSS components

The interaction between the Multi-Agent System and the big data tool is done through the REST API protocol. Every time the agents, need to exchange information with the tool, a REST method is used.

The big data tool provides a collection of data from the distributed Multi-Agent System. This data represents the situation of the system with respect to the occupancy rates and status of its machinery and availability of operators for each task. The tool can analyze historical data analytically to create normal time/workload models and then be able to develop an optimized system model where workloads can be divided as fairly and efficiently as possible under this approach.

The REST API service represents the entry point to the big data pool. The detection and training of the data occurs through it. The data representing the entities that will be analyzed needs to be uploaded in JSON format.

Some steps are necessary to integrate the big data pool with the agents, among them are:

- the configuration of the project to define the parameters concerning the pre-processing and training of the data.
- the periodic creation of models based on the trained data.
- and setting parameters for the detection of optimization opportunities based on the trained data models (e.g., algorithms and variances).

The following is a sample API used by agents to upload training and detection data:

- Training data: training data represents a set of files that are uploaded in JSON format using, for example, the HTTP POST request. A file is uploaded per request while a subset of training data is used for training in a previously specified time window where only the data that belongs to that time window is used.
- Detection of changes in the occupancy rate: after training the data it is possible to start offering possible outputs due to changes in the system's occupancy rate, be it due to a machine breaking down or the need to reconfigure the production line structure for a new product to be produced. To achieve the desired goal, it is still necessary some interactions with the system through the tuning of parameters that produce the result closest to what is expected after the training. The more trained the system is, the richer the dataset is, it will also be necessary to fine-tune the parameters that establish the desired boundaries and tolerances for each product, and thus achieve the optimal system state according to the existing workloads and self-organization needs that may become effective soon.

It is important to specify the technological perspective after defining the interaction between the ResourceAgent and the assets. One of the possible protocols to be chosen for being widely consolidated in the industry is OPC-UA (Open Platform Communications - Unified Architecture).

The agents that are part of the inspection mesh of the system occupancy parameters (number of actors / workload) will subscribe to items that will allow them to be notified about the availability of new data, usually after performing quality tests and before accepting the parameters coming from the ResourceAgent.

The ResourceAgent will be responsible for defining the scenario that will compose the product based on the data received from the FAIRWorkAgent (e.g., tolerances such as maximum and minimum heating times for milk in a coffee setting) and will also receive data from the OrderAgent about the product being manufactured in parallel (in the present time). The occupancy inspection results will be sent back to the FAIRWorkAgent after the completion of each product which will then provide the data to the big data pool. Self-adaptation occurs every time OrderAgent finishes executing a product. The response provided by FAIRWorkAgent will not take effect immediately because of the time required to process the data, but the delay does not destructively impact the optimization goal of the system. Each inspection data will act as an OPC-UA Server providing data to the associated ResourceAgents that will act as an OPC-UA Client.

The product ID is needed for proper association between ResourceAgent and OrderAgent which is associated with the product currently being produced; quality measurement results need to be stored locally in the ResourceAgent database for future data analysis; and other data such as production time and its boundaries can be modified by the ResourceAgent according to data obtained from other parts of the system such as that coming from the OrderAgent which is particularly interesting for maintaining execution parameters for the purpose of occupancy analysis and that coming from the FAIRWorkAgent which has more general/global system data.

3.5.7.1.4 Agent descriptions

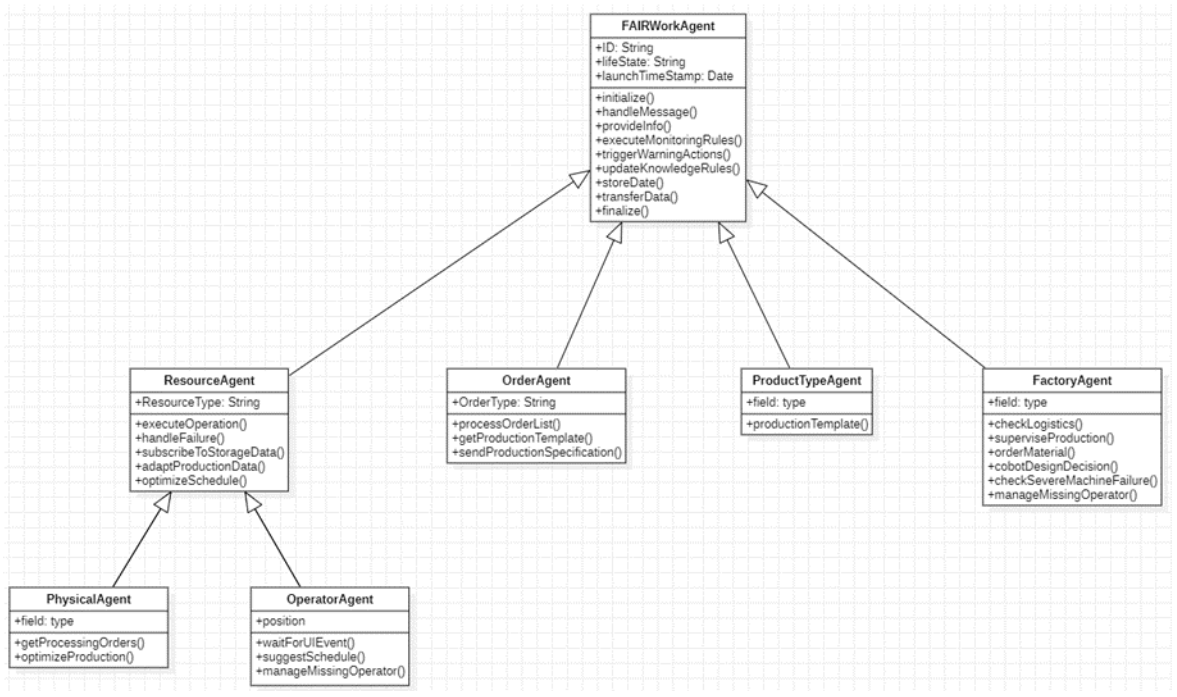


Figure 24 Current agent UML Class Diagram

FAIRWork Agent: Is the central actor in this perspective of production system management. It enables the fair distribution of work among the system members through its ability to use the data locally collected by the agents and process them to optimize the configuration and production times of each member of the production flow. This adjustment occurs in a strategic way for the system improvement, since the data accessed comes from the various parts of the production line and provides a global view of the system state, which provides the generation of new knowledge to the system refer Figure 24.

Order Agent: The OrderAgent is responsible for managing the order placed by customers. Collection and storage of the production data linked to the execution of the product during the manufacturing process, among others, is a function of this type of Agent.

- Create and process the order list based on the customers' requirements monitoring the product along the production line and adapting, if necessary, the process according to the available resources following local knowledge and historical data.
- Accesses the database where all de production plans are stored. For each type of product ordered, the OAs get the specific step-by-step of the manufacturing process.

- Send to the Physical Agent the product production specification based on the order list previously processed.

Resource Agent: Manages the production execution along the lune and is directly associated with the assets. These assets are represented by physical devices such as robots and automated workstations as well as the human operators that are part of the production system. This agent type has specializations that can be considered such as Physical Agent and Operator Agent.

These agents are responsible for the:

- Adaptation of the production parameters, according to the local and knowledge and historical data from the resource perspective.
- Collection and storage of data related to the process station, using proper Internet of Things technologies and other transport protocols. In case of Operator Agents, the events created by the user interface can be also interconnected using the same technology.
- Optimize the product production schedule based on the pre-processing of the collected data.
- Monitoring resource performance to detect potential failure points avoiding quality of service degradation.

PhysicalAgent: Represents exactly the robotic automation system. These agents are responsible for:

- Processes the production order created by the OrderAgent.
- Optimize the production process over time based on the line resources.
- Check for machine failure solutions and report to the ResourceAgent the current work status.

ProductTypeAgent: This type of agent contains all the necessary product information that can be produced in the production line. The detailed process and the characteristic that clearly define each product are catalogued in the system and can be accessed by the Order Agent. These agents are responsible for:

- Collection and storage of data related to the production process and specific characteristics for each product, e.g., amount of coffee, milk, and sugar among others.

FactoryAgent: Has the function of supervising the production status based on several inputs such as logistics situation, production line necessitates, and supplies product needed. The main responsibilities are discriminated bellow:

These Agents are responsible for the following main functions:

- Traceability of the produced products in the production line.
- Monitoring the ongoing production with the capability of re-structure the schedule in case of missing OperatorAgent or if notified of a machine failure. This capability is based on the pre-processing of collected data provided for the other agents.
- Issues the material requisition in case of not enough supplies to produce the orders.
- Act as a collaborative agent in cases that is necessary to make decisions about processes on the production line crossing the system inputs and the human interface actions.

3.5.7.1.5 Development focus area

To begin with the development of the Multi-Agent Systems, it was decided to approach the development with one Multi-Agent System for each scene of the use cases. These scenes represent each sequence of production execution flow that might require optimization by the Multi-Agent System. Each Multi-Agent System will be prototyped to validate which agent and skills are the most adequate for each scene. They will compose a catalogue of Multi-Agent Systems that after validation of each set will constitute a single Multi-Agent System capable of achieving desired goals through skill full negotiations between agents refer Figure 25.

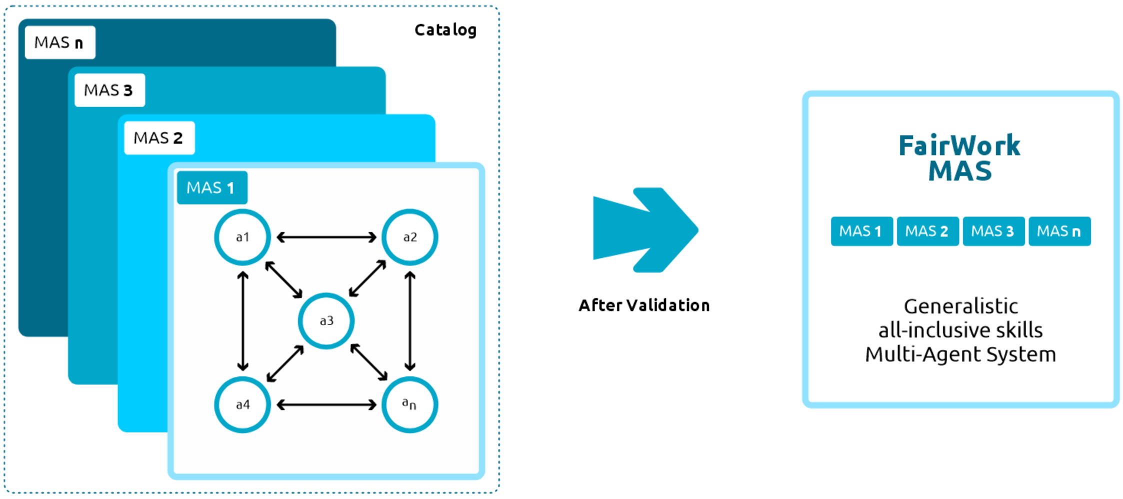


Figure 25 Multi-Agent System development and validation.

The platform used for agent deployment is the JADE Framework. It is at the middleware layer and implements the agent software abstraction over the Java object-oriented language. It consists of agent containers that are distributed throughout the system that provide the services needed for running and hosting the agents. Compliant with FIPA specifications, each agent runs in its own thread, and it can send or receive messages at any given time. The work focuses on developing a distributed system populated by different types of agents that can negotiate between themselves to reach a common goal (even if the goal doesn't consider agreement between them, e.g., competition) and JADE is the tool that enables this.

The Workflow based Orchestrator is based on the OLIVE Framework that will be extended to support EIP notation to model the service behavior. Integration of external workflow engine will be also performed, and the focus will be on the open-source project Netflix Conductor[25].

Additionally, the integration with the DAI-DSS interfaces will be integrated in the system in form of configurable Microservices with specific connectors for each related DAI-DSS component.

3.6 DAI-DSS External Data Asset Marketplace

3.6.1 Purpose

The purpose of this component is to provide a data catalogue. It can be seen as an extensible metadata platform to support for data practitioners to leverage the value of data within their organization and developers to tame the complexity of their rapidly evolving data ecosystems. First, it simplifies the management of metadata. Second, it makes data sets more easily searchable and usable. This is achieved by collecting and indexing metadata from different source systems in data catalogues. In this way, the suitable data stocks become more easily reusable and usable for business and research applications. Finally, it is also about helping data experts to quickly find the most suitable data e.g., for training an AI algorithm, for testing an algorithm, for demonstrators and labs to make data-driven technology more usable and presentable.

3.6.2 Internal Architecture

The architecture of a data catalogue typically involves several functional blocks refer Figure 26

[25] https://conductor.netflix.com/

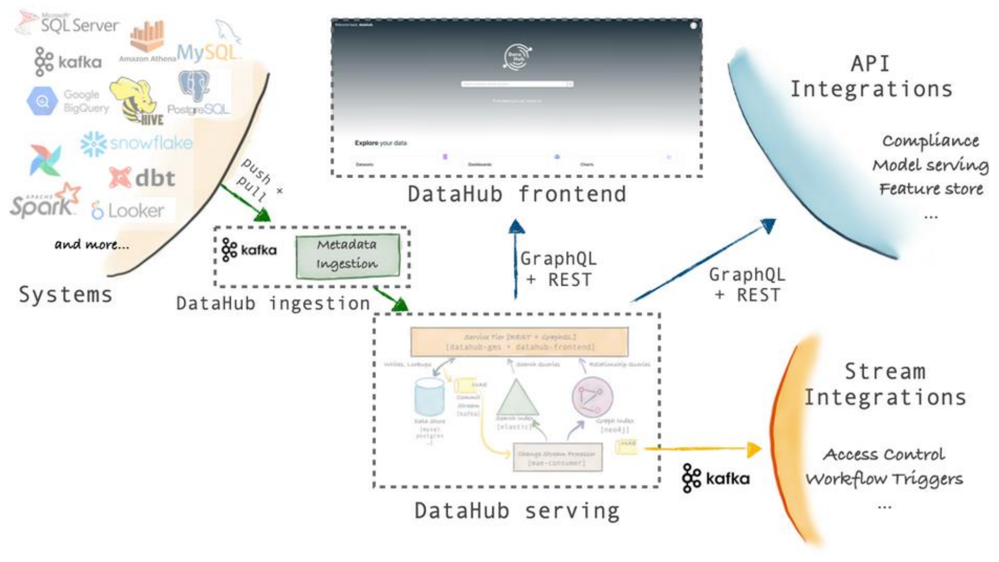


Figure 26 Building Blocks of the data catalogue of the data market

Frontend component: With this user interface, you can add, search, edit and modify data assets within the data catalogue. An important part is the search bar. From the search bar, you can find datasets, dashboards, Data pipelines, and more. Furthermore, it has also and administration part where you can create user, groups and define the access policies.

Datahub servicing component: can work a single metadata service as part of the open source repository. However, it also supports federated metadata services, which can be owned and operated by different. The federated services communicate with the central search index, to support global search and discovery while still enabling decoupled ownership of metadata.

Ingestion component: This is the component where you can create, configure, schedule, & execute batch metadata ingestion using the DataHub user interface. This makes getting the right metadata into DataHub easier by minimizing the overhead required to operate other integration pipelines.

3.6.3 Functions

This component is based on a data assets catalogue, where items can be provided from external resources in a service-based manner to be used by various service-based infrastructures. The component provides several functionalities.

3.6.3.1 User Interfaces

A set of features to make discovering and managing your data assets. E.g., user registration, search of data assets, ingest and describe data asset etc.

3.6.3.2 Search & Filters

The search function supports several stand features. For example, the search terms will match against different aspects of a data assets. This includes asset names, descriptions, tags, terms, owners, and even specific attributes like the names of columns in a table.

3.6.3.3 Metadata enrichment

It enables a powerful way to annotate entities within FAIRWork data asset marketplace. This function ensures that end-users have quick access to for a given entity, such as:

- Ownership: who owns the data?
- Description: what is the intended use case for this data set.
- Application Domain: how is it associated with the application domain? Do which area it belongs?

3.6.3.4 Metadata transformation

The tool also allows the metadata to be enriched in an automated way.

3.6.4 Dependencies

Data Market place is a standalone service that provides the ability to store and maintain a data set. It has its own interface. Furthermore, the component can be used via API.

3.6.5 Interface and Data

Data catalogue provides several APIs to manipulate metadata on the platform:

- The GraphQL API is our recommended option for querying and manipulating the metadata graph. It is a data query language and API with the following properties. It enables validate specification; it is strongly types and document oriented.
- The REST API is a much more powerful, low-level API intended only for advanced users.

3.6.6 Development Focus Area

The component is most likely built on an available open-source tool such as DataHub[26]. Additional functionality will be provided within the FAIRWork project.

3.6.7 Other Details

The data asset can be used for AI based services development.

3.7 DAI-DSS Intelligent Sensor Box

3.7.1 Purpose

In the DAI-DSS there is the specific need to capture human centered data to represent the worker and decision-making operator. In general, these data can be gathered by qualitative methods, such as, providing questionnaires or applying interviews, but also by quantitative methods, such as, by streaming data originating from sensors that are mounted either directly on the human body or at the workplace to capture the context of the worker's task. The focus of the Intelligent Sensor Box (ISB) is primarily to collect the sensor data from the human as well as from the direct environment where the worker is operating and further to interpret these data in terms of edge analytics. The ISB needs to encapsulate the personalized data due to privacy reasons and only transfer anonymized data to

[26] https://datahub.io/

another component in the system architecture, i.e., the DAI-DSS Knowledge Base. The ISB is therefore situated between the main sensor network and the factory infrastructure.

Due to the requirements in the production industry and the privacy aspect of the human aspect, the ISB also needs the ability to analyze data. Edge analytics is a model of data analysis where incoming data streams are analyzed at a non-central point near the sensor network or other data sources. Data can be prepared and filtered on the edge layer (e.g., within the factory). This reduces the amount of data for the centralized decision service layer. Finally, specific visualizations for parts of the sensor or infrastructure within a factory could be provided in this layer. This software component – herein it is referred to as an Intelligent Sensor Box – supports local data management and data analysis. Depending on the use case, different forms of data analytics such as time-aware graph, logical and statistical reasoning as well as edge AI methods, are supported if necessary. The ISB will assist users in local decision-making. This component should also be capable of dealing with limited resources (e.g., network bandwidth, data storage). In such cases, the overall functionality may be decrease.

3.7.2 Internal Architecture

The internal architecture of the DAI-DSS Intelligent Sensor Box supports the data flow from human- and workplace-mounted sensor data to higher abstractions of human factors, i.e., dominantly ergonomic, and psychophysiological constructs that determine the mental state and behaviors of the human, and from this the sociotechnical systems within the production process.

The integration of human-centered data into the overall DAI-DSS decision-making process needs anonymization of the highly vulnerable psychophysiological data. However, personalized data are of interest to the individual worker or decision-maker and are available for individual insight on a security- and privacy-preserving basis.

The internal architecture of DAI-DSS Intelligent Sensor Box as well as its dependencies on other components is presented in Figure 27.

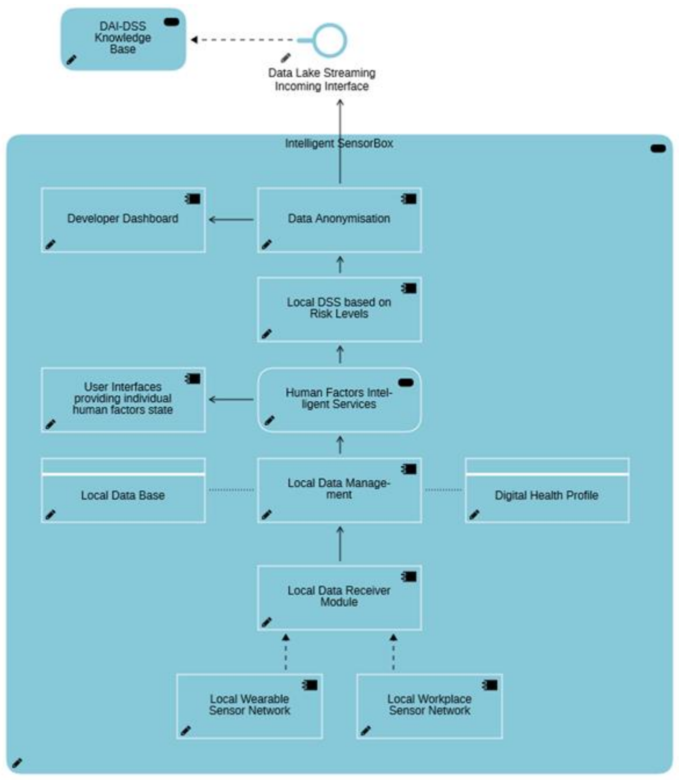

DAI-DSS Knowledge Base
Data Lake Streaming Incoming Interface
Intelligent SensorBox
Developer Dashboard
Data Anonymisation
Local DSS based on Risk Levels
User Interfaces providing individual human factors state
Human Factors Intelligent Services
Local Data Base
Local Data Management
Digital Health Profile
Local Data Receiver Module
Local Wearable Sensor Network
Local Workplace Sensor Network

Figure 27 Architecture of DAI-DSS Intelligent Sensor Box

3.7.2.1 Local Wearable Sensor Network

Non-obtrusive wearable sensors are the basis to measure human factors during the execution of tasks at the workplace. The proposed digital sensor network consists of wearables that are specifically suited to measure psychophysiological variables, such as, cognitive-emotional stress, physiological strain, and fatigue. Figure 28 depicts the sensors for human factors state estimation from wearable data streams and digital data analytics. (a) Wearable (smart band; Garmin, Switzerland) for cardiovascular data capture, (b) biosensor shirt (QUS, Austria) for vital data monitoring, (c) eye tracking glasses (Pupil Labs, Germany) for recording eye movements and derive from the digital sensor data stream the mental state analytics. The Pupil Labs Invisible (c) are very lightweight calibration-free eye tracking glasses with detachable scene-camera and cable connection to a mobile recording unit.

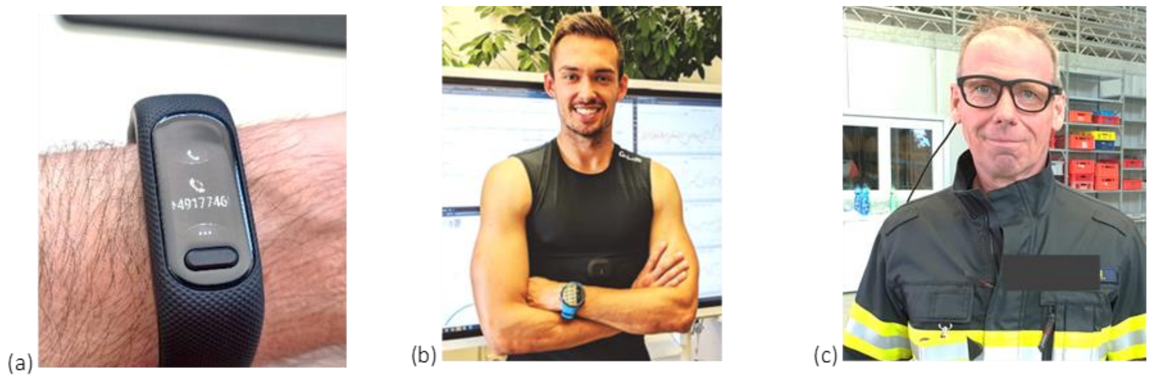


Figure 28 Sensors for human factors

3.7.2.2 Local Workplace Sensor Network

The local environment of the workplace can impact the human factors state of the individual workers. Several sensors are used to represent the state of stressors and their impact on human profiles. The sensor network might include sensors for air quality, temperature, noise, etc. The data can be transferred to the Local Data Receiver Module where the sensor data are pre-framed for the storage in the local database.

3.7.2.3 Local Data Receiver Module

This component implements the communication between sensor devices and a local server component. This module enables either data from the stationary care or to represent the DSS by another care organization.

3.7.2.4 Local Data Management

Local data management consists of a local data lake that receives data from the sensor networks and receiver module and stores the data for real-time data management. It is a MongoDB[27] that enables rapid access and storage. The component also initiates specific data managing operations (see digital health profile).

3.7.2.5 Digital Health Profile

The digital health profile includes biometric data and indicators about well-being and health from individual workers. It needs specific conditions on security and privacy since it contains the most vulnerable data. The local data management (above) might operate to anonymize in an early stage several data to make them available for further feature-managing computing.

[27] https://www.mongodb.com/

3.7.2.6 Human Factors Intelligent Services

Based on the data of the local data management component, several intelligent services will operate in real-time. These services are at the core of the ISB and include methods that were developed in earlier projects but also newly developed estimators of psychophysiological variables, such as, stress, concentration, fatigue, and motivation. In addition, environmental variables - e.g., air quality depending on percentage of oxygen, operating temperature at the workplace, etc. – will be considered. Furthermore, optimization methods will be available to construct novel measures that particularly refer to the requirements of socio-technological systems. A typical example would be the computation of individual duration of time periods or extents of time shifts that would enable work with high motivation and highest safety prediction score. This component generates the dynamic human profiles that can be used, after anonymization, for further processing in the DAI-DSS Knowledge Base.

3.7.2.7 Local DSS based on Risk Levels

The component of the local decision support system (DSS) will predict certain risk levels with relevance for the socio-technical system that arise from the sensor networks. The physiological strain index (PSI) will provide risk levels for cardiovascular cause-based risk of dropout. The cognitive-emotional risk level (CE) will provide safety-oriented risk levels due to estimated scores for mental fatigue. Other risk levels will be discussed and applied according to the requirements of the DAI-DSS architecture and the use cases.

3.7.2.8 User Interfaces Providing Individual Human Factors State

These individual interfaces can display individually relevant information to the worker upon her interest. These data are particularly personalized and need specific concern on security and privacy encapsulation. However, this service might be relevant from an ethical point of view to integrate the motivation of the worker into the DAI-DSS framework.

3.7.2.9 Data Anonymization

Before transferring the resulting risk level data to the DAI-DSS Knowledge Base, these must be anonymized. Several algorithms are under consideration, with the common objective to render the vulnerable data without reference to the individual but at the same time enable further integration of the data into the DAI-DSS framework.

3.7.2.10 Developer Dashboard

The objective of the Developer Dashboard is to support the developers of novel intelligent human factors services and to demonstrate and represent on-going sensor and analytics data. The dashboard Figure 29 is depicting instantaneous vital parameters (left), live bio-signal information and scores (upper right graph), risk levels (middle right graph) and recommendations (lower right graph).

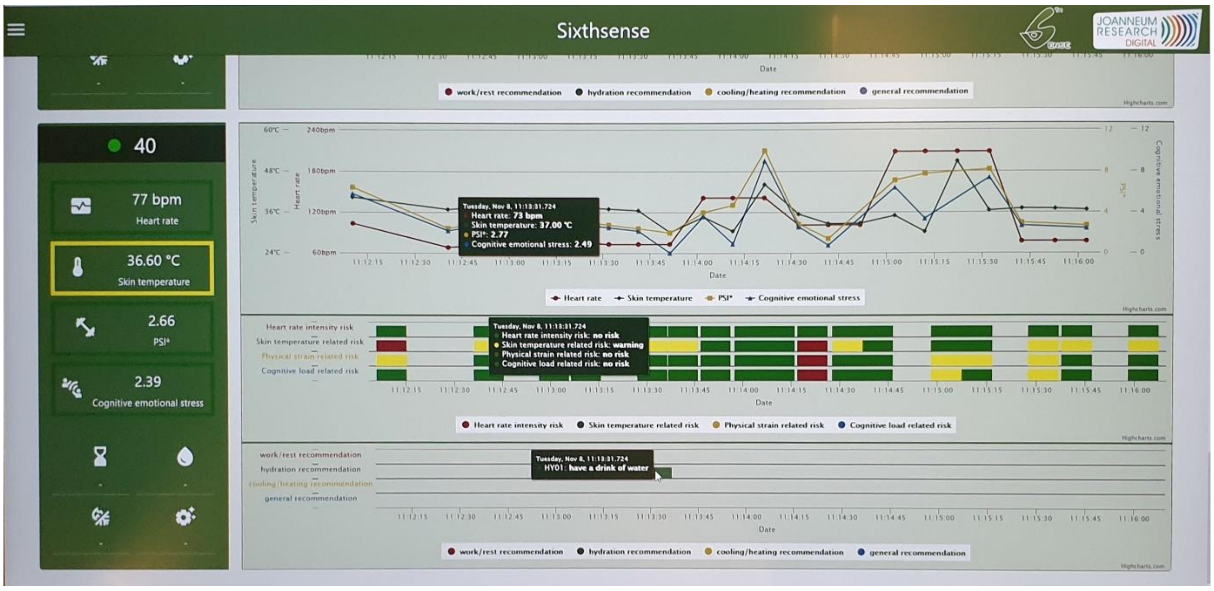

Sixthsense
JOANNEUM RESEARCH DIGITAL
40
77 bpm
Heart rate
36.60 °C
Skin temperature
2.66
PSI*
2.39
Cognitive emotional stress
Tuesday, Nov 8, 11:13:31.724
Heart rate: 73 bpm
Skin temperature: 37.00 °C
PSI*: 2.77
Cognitive emotional stress: 2.49
Heart rate
Skin temperature
PSI*
Cognitive emotional stress
Heart rate intensity risk
Skin temperature related risk
Physical strain related risk
Cognitive load related risk
Heart rate intensity risk: no risk
Skin temperature related risk: warning
Physical strain related risk: no risk
Cognitive load related risk: no risk
work/rest recommendation
hydration recommendation
cooling/heating recommendation
general recommendation
HY01: have a drink of water
Date
Highcharts.com

Figure 29 describes the developer dashboard from another Horizon project (SIXTHSENSE) that might be adapted. The dashboard is depicting instantaneous vital parameters (left), live bio-signal information and scores (upper right graph)

3.7.3 Functions

The main individual subcomponents can be described as follows:

- The DAI-DSS Intelligent Sensor Box provides services for supporting device management on the edge.
- With the registry, a list of devices and services can be added. For devices and services, the functions “add”, “modify”, and “delete” are supported.
- Data storage – especially the saving of time series – should be supported. Therefore, a suitable edge storage database is used.
- Data can be processed in different ways (e.g., filtered, aggregated, summarized) and diverse algorithm applied to the personal profiles.
- RESTful APIs are used to integrate the user interface. Sending data to the ISB is supported in the form of a streaming interface or a dedicated RESTful API.
- The resource manager handles basic resources such as hard disk, space in the data storage, etc. The resource manager is used to estimate whether sufficient resources are still available for basic functions. If the resource manager runs out of resources, it can report this event to a neighboring edge analytics box, or another connected software component.
- Furthermore, a monitoring service (including logging information) is supported.

3.7.4 Dependencies

The DAI-DSS Intelligent Sensor Box is a very important part for providing sensitive data for the digital twin.

3.7.4.1 DAI-DSS Knowledge Base

The DAI-DSS Knowledge Base will receive biosensor data as well as risk level-based metadata. These data are mandatory to be able to construct measures based on concrete experience in the Human Factors Lab or in the use case-specific Labs, finally, to develop meaningful persona constructs.

3.7.4.2 DAI-DSS AI Enrichment

In ISB we apply local intelligent services to comply with privacy concerns, In the DAI-DSS AI Enrichment component we develop the persona models. Further optimization algorithms will be implemented by using the more dynamic human profiles (human factors service results).

As described in the development focus area, the development of persona will be further generated in the DAI-DSS AI Enrichment Module. Risk levels for working groups or typical worker's representatives are designed. Ultimately, the Digital Twin of Digital Human Sensor (DHS) will be operated there.

3.7.5 Interfaces and Data

There will two interfaces available to access data from the ISB: a REST based and an MMQT application programming interface (API).

3.7.5.1 REST Interface

The Intelligent Sensor Box will provide a REST interface enabling various (Micro) services. The interface will offer several GET, POST, PUT and DELETE methods to access (derived) sensor data as well as psychophysiological scores and risk-levels, to adjust sensor parameter settings, maintain digital health profile related user data or manage internal ISB data tables. The REST interface is therefore mainly used to retrieve post-processed (offline) data, manage settings, and maintain the internal database. To access real-time data (streams), the MQTT interface, which is described in the next chapter, is the more suitable choice.

3.7.5.2 MQTT Interface

The MMQT interface of the ISB offers the possibility to subscribe for specific data and data streams of interest, which are then forwarded (pushed) to the subscriber, without the latter having to request new data each time (polling). This interface makes it, for example, very easy for the DAI-DSS to receive real-time psychophysiological scores and risks from the ISB.

3.7.5.3 Data format

Both interfaces provide data in JSON (JavaScript Object Notation) format, which is widely used for data exchange between applications. This format is independent on platforms and programming languages, compact, and due to its easy-to-read text form very intuitive.

3.7.6 Development Focus Area

There will be three main development focus areas, as follows,

- Mapping environment and machine-relevant data to digital Human Factors estimators (services mapping the local context)
- Development of local decision support components and services from validated digital human factors estimators (HF-DSS) that are based on biosensor data (Digital Shadow). Development of specific risk levels for well-being of the worker that will be transferred to the DAI-DSS Knowledge Base.
- Research and development on specific optimization variants by using human profiles and other use case-relevant resources (for application specific local services).
- The development of persona profiles (see description above).

4 APPLICATION SERVICE CATALOGUE FOR DECISION MAKING

This section provides a list of AI services that can be used for decision support as mentioned in chapter 2. The list of services acts as initial starting point which evolved by the number of services and their nature. The current list is indicative to introduce the idea of DAI-DSS.

4.1 Structure of component descriptions

For each service detailed technical implementation details are described and follows the below structure:

1. **Purpose**
   - The purpose of the AI service
2. **Service component**
   - UML diagram describing the service, also showing how it connects to the other components.
3. **Interfaces and data**
   - Description of all external and internal interfaces, including mechanism for communication, and required import and export functionalities.
     - Input: Type of data passed through the interfaces, such as data structures, file formats, etc.
     - Output: Type of data passed through the interfaces, such as data structures, file formats, etc.
4. **Functions**
   - An overview of functional capabilities of AI service that provides to either the user or to the other components within the framework.
5. **Dependencies**
   - How the service relates to, depend on, or is a dependency to other components.

4.2 AI services list

The list of services provided here are few ideas to show how different AI technologies can be used to provide decision support for the use case scenarios outlined in section 2.1 of this deliverable, where in for each use case, the potential areas of application for AI services are presented. Nonetheless, there is no definitive guide for implementing the listed services with the mentioned technologies. It also feasible to have more or fewer services than those listed in this section, and the number of services implemented is depended upon the research output produced form WP3. Several of the mentioned AI services may be attempting to address the same problem, but these services will be evaluated in a laboratory environment to determine the optimal one based on the required performance measures.

The following are the AI services that are described in this section:

1) Order to Production Line with Fuzzy rules
2) Order to Production Line with Neural Networks
3) Similarity Matching with Knowledge Graphs
4) Resource mapping with Decision Trees
5) Worker to Production Line Mathematical Optimization/heuristic
6) Reinforcement Learning Based Resource Allocation Service
7) Sematic Resource Allocation Service

8) Pattern Based Resource Allocation Service
9) Rule Based Resource Allocation service
10) Operator Stress Estimation with Neural Networks
11) Annotation Support Service
12) Production Process Simulation with Agents
13) Configuration Support Service
14) Parameter Optimization Service with Machine Learning
15) Operator persona development with Machine Learning
16) Community Detection with Knowledge Graphs
17) Production Process Simulation
18) Similarity Matching with Semantics
19) Impact Assessment with Knowledge Graphs
20) Multi Agent-based Resource Allocation

4.2.1 Order to Production Line with Fuzzy rules

4.2.1.1 Purpose

Order allocation to the production line with Fuzzy Logic rules allows to include rules made through experience. Experts who are working with the machines daily have a profound understanding of processes and are often able to intuitively provide answers for many problems. This human-like reasoning can solve many real issues in a more efficient way and is thus relevant for decision-support efforts. With Fuzzy Logic informal, often not mathematical describable know-how can be captured and used for better supporting decision making. This knowledge-based approach provides benefits for finding solutions in uncertain, ambiguous environments and even decisions that are not restricted to exactly known input values can be assisted. Due to the concept of fuzziness, problems that would otherwise need a lot of computing power can be easily solved, thus Fuzzy Logic is especially interesting when assistance for highly complex decisions with many input parameters should be provided. Figure 30 shows an UML-Use-Case diagram of the Fuzzy Logic Based Order Allocation Service.

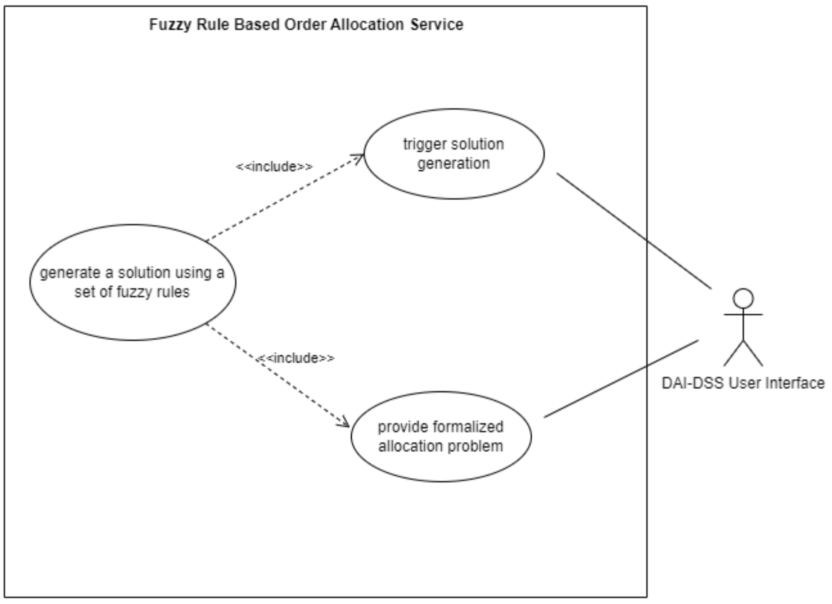

Fuzzy Rule Based Order Allocation Service
trigger solution generation
<<include>>
generate a solution using a set of fuzzy rules
<<include>>
provide formalized allocation problem
DAI-DSS User Interface

Figure 30 UML Use Case Diagram for Fuzzy Rules Order Allocation Service

4.2.1.2 Service component

The Fuzzy Rule Based is an allocation service that is not subdivided into other service components. Figure 31 illustrates the interfaces between it and other components of the overall system using a component diagram. The service accesses the DAI-DSS User Inface via a REST interface.

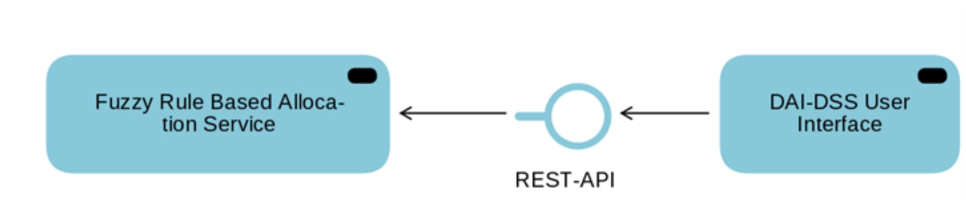


Figure 31 Service Component for Rule based Allocation Service

4.2.1.3 Interface and Data

Suitability of a concrete order for a specific type of machine:

- Interface: REST-API.
- Input: concrete decision relevant details of a specific order provided by the user
- Output: Based on the input data, the suitability of the order for a machine assignment is indicated and. illustrated in a graph.

4.2.1.4 Functions

The function of the Fuzzy Logic based decision support is to include informal, experience-based knowledge for order assignments made by the machine operators. The service can assess and predict whether an order-to-machine allocation will be suitable or not. It is especially interesting for providing solutions on greyish problem-solving areas. As the fuzzy Logic is dependent on the formulation and coding of internal known experts' knowledge to create the underlying rules for the fuzzy system, they must be adapted specifically for each use case. Due to the knowledge gathering and storing actives, the fuzzy system can be seen as a knowledge management tool within the company.

4.2.1.5 Dependencies

The Fuzzy Rule Based Resource Allocation Services does not depend on other components.

4.2.2 Order to Production Line with Neural Networks

4.2.2.1 Purpose

With the neural network the ability to classify and predict the order to machine allocation is enabled. It is a data-driven approach that can be trained by providing historical allocation data to the system. As the system learns and adapts the weight according to the provided training dataset, the quality of the results and its accuracy relies heavily on proper historical machine allocation activities. The relation within the input and output data of the training materials are learned and then applied to predict the output on new inputs. The proposed order allocation can be used as a decision aid and can additionally impact other decision and planning dimensions within the company such as expected machine utilization. In general, the application potentials of neural networks are far-reaching due to their adaptive learning features refer Figure 32

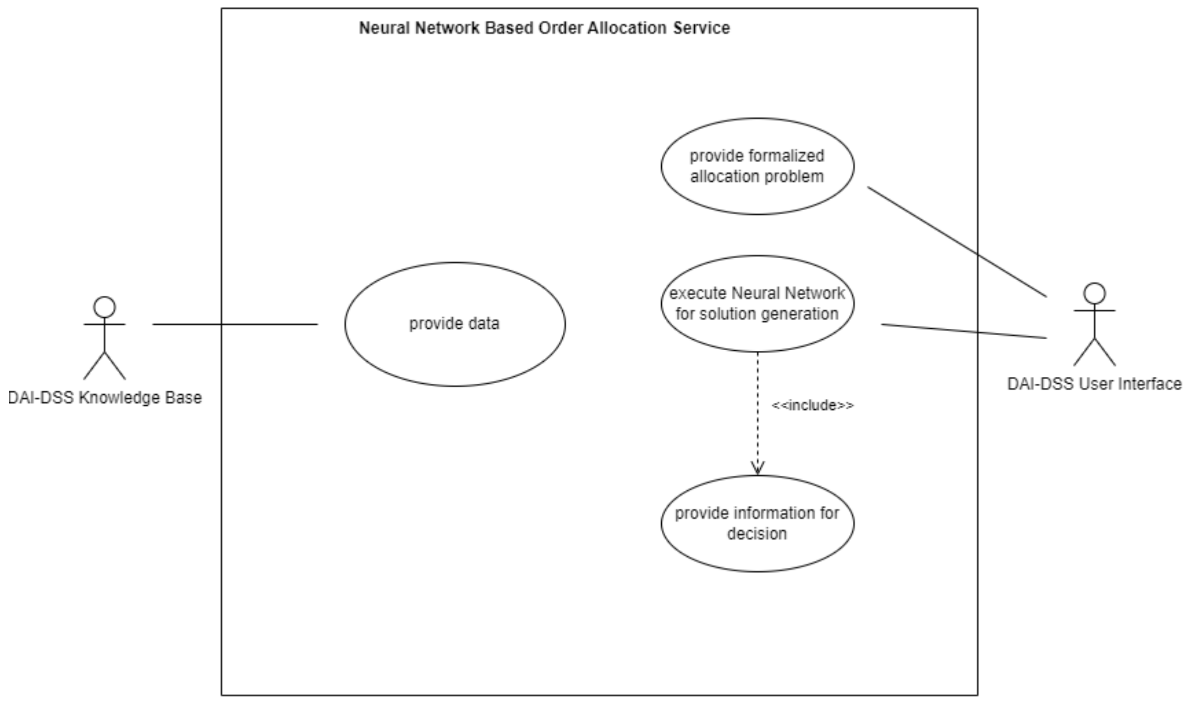

Neural Network Based Order Allocation Service
provide formalized allocation problem
provide data
execute Neural Network for solution generation
<<include>>
provide information for decision
DAI-DSS Knowledge Base
DAI-DSS User Interface

Figure 32 UML Use Case Diagram for Neural Network Order Allocation Service

4.2.2.2 Service component

The Neural Network Based Allocation Service is a service that is not subdivided into other components. Figure 33 illustrates the interfaces between it and other components of the overall system using a component diagram. The service accesses the DAI-DSS Knowledge Base via a REST-API. The DAI-DSS Knowledge Base includes historical order-to-machine allocations activities.

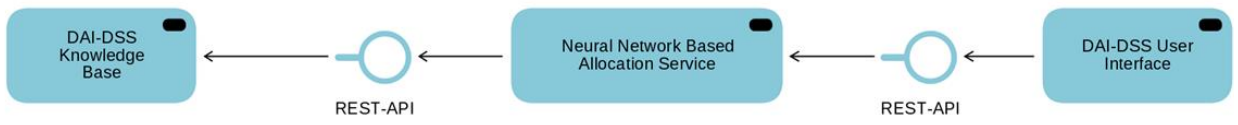


Figure 33 Service Component for Neural Network based Allocation Service

4.2.2.3 Interface and Data

Assignment of orders to a machine:

- Interface: REST-API.
- Input: concrete details of the order in an excel sheet.
- Output: Based on the input a machine assignment with a certain accuracy is suggested.

4.2.2.4 Functions

The sole function of the Neural Network is to predict the allocation of orders to a machine. The service is specified on suggesting whether an order should be produced on a specific type of machine or not. This information is especially relevant for operational production planning tasks.

4.2.2.5 Dependencies

The Neural Network Service depends on the DAI-DSS Knowledge Base. For proper allocation suggestions, historical data for training purposes is utilized.

4.2.3 Similarity Matching with Knowledge Graphs

4.2.3.1 Purpose

This service focuses on the allocation of resources (e.g., workers) to specific tasks which must be achieved within a production line. Goal of the allocation is a good matching between the capabilities of the resources to the needs of the tasks. To support the allocation, this service will use a knowledge graph that will contain the needed information and will be the input for the similarity matching algorithm. Figure 34 shows a use case diagram for this service.

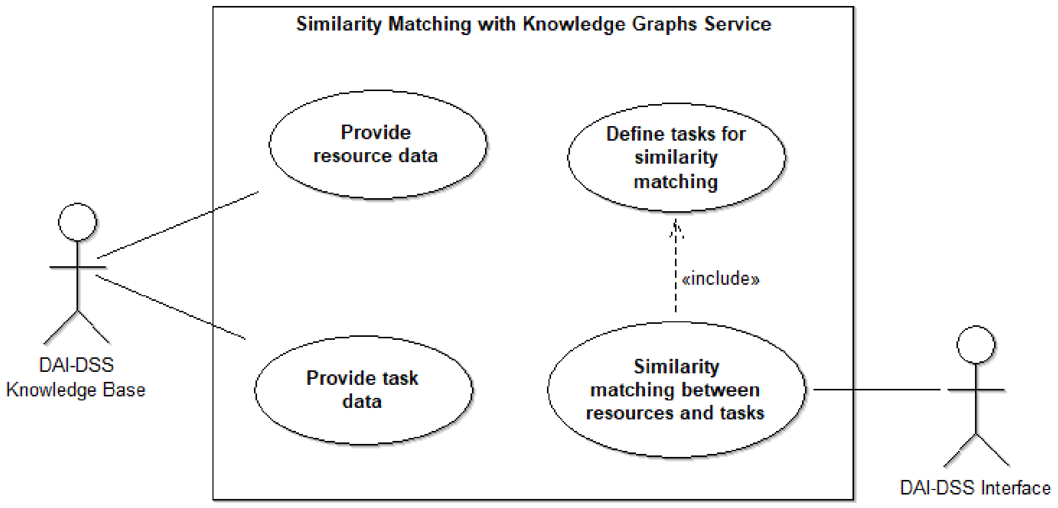

Similarity Matching with Knowledge Graphs Service
Provide resource data
Define tasks for similarity matching
«include»
DAI-DSS Knowledge Base
Provide task data
Similarity matching between resources and tasks
DAI-DSS Interface

Figure 34 UML Use Case Diagram for the Similarity Matching with Knowledge Graphs service

4.2.3.2 Service Component

The "similarity matching with knowledge graphs" service component will be provided as a singular service component and therefore will not be divided into subcomponents. The service component will gather the needed information from the user and the DAI-DSS Knowledge Base, calculates the similarity matching and returns the result. An overview of the service component can be seen in Figure 35.

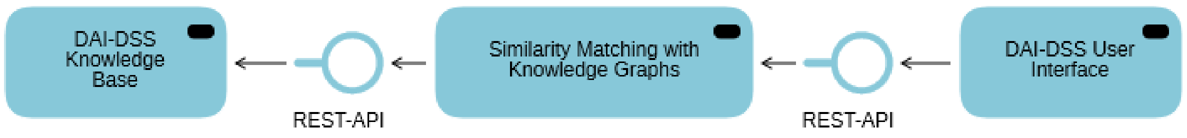


Figure 35 Service Component for Similarity Matching with Knowledge Graphs service

4.2.3.3 Interface and Data

Similarity matching for resource allocation.

- Interface: REST.
- Input: task and resource data in specific data structure, e.g., JSON or RDF-based data-structure.
- Output: list of matchings with similarity measure.

4.2.3.4 Functions

The service offers functions to trigger the similarity matching between a set of tasks and possible resources. Therefore, the possible resources and the tasks must be available in the knowledge base in a knowledge graph structure. Then an actor defines the tasks, for which possible resources should be found, as an input for the similarity matching. For example, if a worker cannot come to work due to illness, the “similarity matching with knowledge graphs” service component can be used to identify other works which can fulfil the task.

4.2.3.5 Dependencies

This service depends on the availability of the data about the tasks and the resources. For tasks the requirements for the resources must be defined and for the resources their capabilities must be described. The data should be available in a knowledge graph structure, so that the algorithms of the service can be applied. Therefore, it depends on the data of the DAI-DSS Knowledge Base and how they are structured.

The set of tasks, for which resources should be allocated must come from an external source. This could be a human actor or another application/service.

4.2.4 Resource Mapping with Decision Trees

4.2.4.1 Purpose

The purpose of this service is to allocate resources to task in a production line based on a decision tree. The tasks in a production line have certain needs for the execution, which can also be seen as requirements. The resources, e.g., human workers, must therefore fulfil these requirements. In this services decision trees will be used to propose mapping from resources to tasks. A use case diagram for this service can be seen in Figure 36.

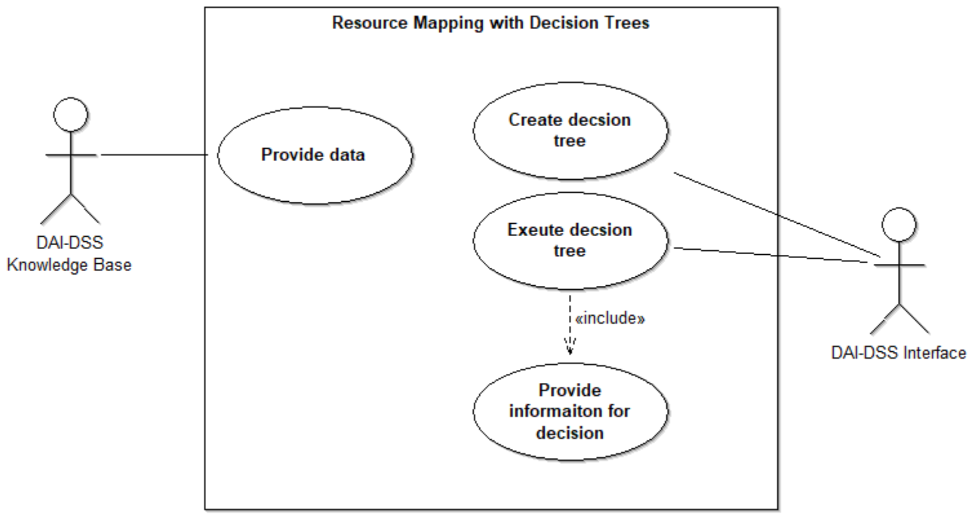


Figure 36 UML Use Case Diagram for the Resource Mapping with Decision Trees Service

4.2.4.2 Service Component

The "resource mapping with decision trees" component will be provided as a singular service component and therefore will not be divided into subcomponents. The service component will gather needed information from the user and the DAI-DSS Knowledge Base, where needed. The service can help to define the decision trees and execute the decision trees to provide possible solutions for a resource allocation problem. An overview of the service component can be seen in Figure 37.

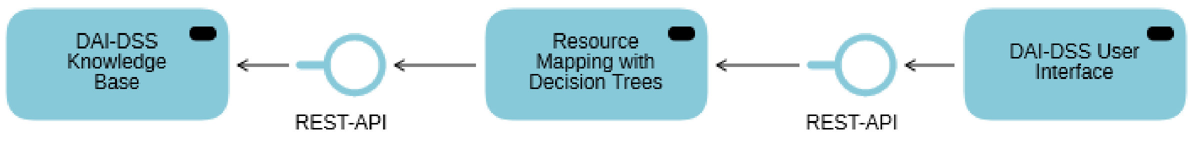


Figure 37 Service Component Resource Mapping with Decision Trees Service

4.2.4.3 Interface and Data

Creation of a decision tree:

- Interface: REST.
- Input: data for decision in defined data structure, e.g., JSON.
- Output: defined decision tree.

Execute of a decision tree:

- Interface: REST.
- Input: data for the decision in defined data structure, e.g., JSON.
- Output: suggestion solution for the decision.

4.2.4.4 Functions

This service component possesses two main functionalities. One is the definition of a decision tree for the support of a certain decision. The other is to execute a decision tree to support actors in their decision-making process. To allow the creation of a decision tree the data underlying the decision must be provided and the tree structure must be created. Once it is created it can be used to propose solutions to decisions. To achieve this, the context data of a certain decision must be provided to the service.

4.2.4.5 Dependencies

The service component depends on different types of data, which can either be provided by the DAI-DSS Knowledge Base, by an actor through the interface (e.g., a human user through the user interface), or through a combination of both. One type of data is the input for the decision-making process, based on a concrete situation. The second type of data is needed to create a decision tree.

4.2.5 Worker to Production Line Mathematical Optimization/Heuristic

4.2.5.1 Purpose

In this service, suitable workers are assigned to the tasks. Part of the service creation is the creation of detailed technical and strategy implementation, considering the framework conditions required for the problem, e.g., in terms of human resources, profiles of the workers, the tasks to be performed in the scenario etc. In most cases, different approaches with very different abstractions are possible. However, the choice of the model for optimal resource allocation by considering the time constraints, constraints on the worker profiles (e.g., stress levels, etc.), and available time for making the decisions have a relevant impact on the conceptual solvability of the final resource optimization problem, so that the flexible model building is a key contribution in service implementation refer Figure 38.

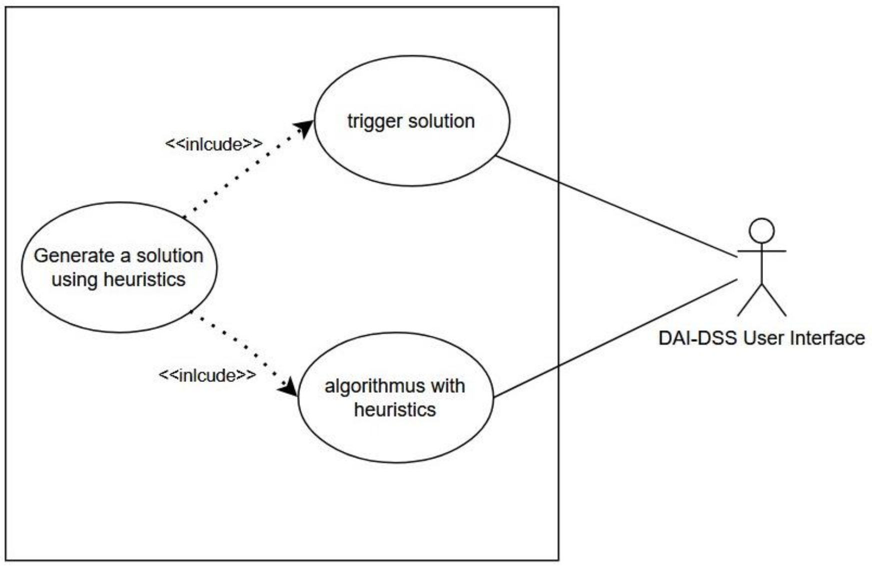

trigger solution
<<inlcude>>
Generate a solution using heuristics
<<inlcude>>
algorithmus with heuristics
DAI-DSS User Interface

Figure 38 UML Use Case Diagram for Worker assignment to production line by using mathematical optimization/heuristics

4.2.5.2 Service component

The worker assignment is not subdivided into other components. The service accesses the DAI-DSS Knowledge Base via a REST interface. The service is provided to the end user via the DAI-DSS User Interface refer Figure 39. It should be noted that the worker assignment does not create any visualizations. It only offers a solution for the specified problem if one exists.

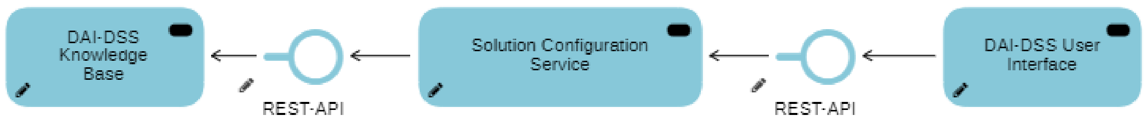


Figure 39 Service Component for Worker assignment to production line by using mathematical optimization/heuristics

4.2.5.3 Interface and Data

Assignment of a resource for a concrete problem

- Interface: REST-API.
- Input: concrete problem instance in JSON format.
- Output: concrete mapping of the allocation task for the given problem instance.

4.2.5.4 Functions

The sole and only function of the workers assignment to the production line. Allocation Service is to allocate one worker resource to one task. The worker resources allocation need depends on the use case and the resource allocation strategy. It will perform a resource allocation for a specific set of use case group and not for a generic resource allocation problem. However, the algorithms can be used a suitable configuration for the doing the proper resource allocation.

4.2.5.5 Dependencies

The DAI-DSS Knowledge Base stores configuration and resources which include scheduling plan and different information about the required resources for a proper resource allocation.

4.2.6 Reinforcement Learning Based Resource Allocation Service

4.2.6.1 Purpose

The allocation of resources is usually a recurring problem in the sense that resources must be allocated to different tasks in a sequential manner. Moreover, allocating a resource to a particular task generally affects subsequent resource mappings, so the allocation of multiple resources to multiple tasks results in a combinatorial optimization problem. These kinds of problems are referred to as Scheduling Problems.

Scheduling Problems come in many different forms. For example, some companies manufacture very similar products that all go through the same machines in the same order. Others are to manufacture many different products whose manufacturing processes are multifarious. However, one can characterize all scheduling problems arising in this context by three parameters as follows, according to Graham et al. (1979)[28] [29].

A machine scheduling problem $\alpha|\beta|\gamma$ is described by the specification of the resources α, predefined instance characteristics and constraints β and an objective function γ that is to be minimized. (Jaehn and Pesch 2014)[30] cover a large variety of possible parameterizations and discuss the resulting problems and classic solution methods like heuristics and branch and bound algorithms. The Job Shop Problem modelling approach offers one of the most flexible description forms in resource scheduling. It is of high practical relevance since it can aim to find a schedule for various tasks. Widely used solution methods, such as heuristics or branch and bound methods, can provide good solutions with respect to individual criteria.

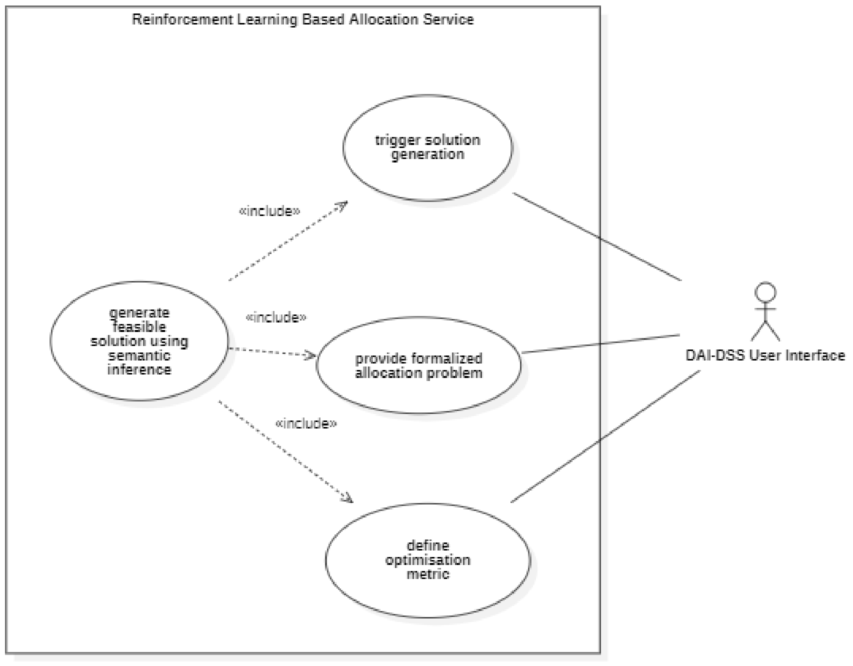
Often optimization variables contradict each other.[1] For example, it is frequently the case that quick machining of a work piece leads to lower product quality. On the one hand, one wants to have the shortest possible manufacturing times, on the other hand, the highest possible product quality. These two goals conflict and may not be resolved or only with great effort using classical solution methods. Reinforcement learning can optimize multiple metrics by

[28] Graham, Ronald Lewis, et al. "Optimization and approximation in deterministic sequencing and scheduling: a survey." Annals of discrete mathematics. Vol. 5. Elsevier, 1979. 287-326

[2929] Jaehn, Florian, et al. "Maschinenumgebungen, Ablaufeigenschaften, Ziele." Ablaufplanung: Einführung in Scheduling (2014): 11-19

[30] Jaehn, Florian, et al. "Maschinenumgebungen, Ablaufeigenschaften, Ziele." Ablaufplanung: Einführung in Scheduling (2014): 11-19

design through the appropriate design of the reward function. Therefore, a major advantage of the reinforcement learning approach is that different metrics for the quality of a schedule can be unrestrictedly combined without having to adapt the underlying algorithm. Furthermore, new constraints can be introduced relatively easily in a reinforcement learning setup. In practice, this has the advantage that it is easier to adapt or react to changing circumstances. Figure 40 describes the interaction of the user with the service.



Reinforcement Learning Based Allocation Service
trigger solution generation
«include»
generate feasible solution using semantic inference
«include»
provide formalized allocation problem
«include»
define optimisation metric
DAI-DSS User Interface

Figure 40 UML Use Case Diagram of the Reinforcement Learning Based Resource Allocation Service

4.2.6.2 Service component

The Reinforcement Learning Based Allocation Service is a component that is not subdivided into other components. Figure 41 illustrates the interfaces between it and other components of the overall system using a component diagram.

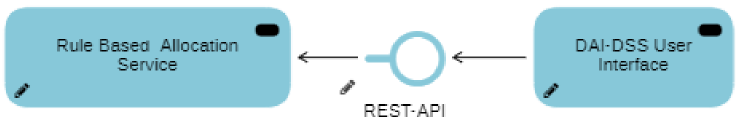


Figure 41 Service Component for interfaces between the Reinforcement Learning Based Resource Allocation Service and DAI-DSS Architecture

4.2.6.3 Functions

The sole and only function of the Reinforcement Learning Based Allocation Service is to find a solution of a Job Shop Problem Instance. The Reinforcement Learning Based Allocation Service can generate Solutions for arbitrary Job Shop Instances agnostic of the instance size.

4.2.6.4 Interface and Data

Generation of a resource schedule for a Job Shop Instance:

- Interface: REST-API.
- Input: concrete problem instance in JSON format.
- Output: concrete mapping of tasks to resources for the given problem instance. the schedule is formatted so that a Gantt chart can easily be constructed.

4.2.6.5 Dependencies

The Rule Based Resource Allocation Services does not depend on other components.

4.2.7 Semantic Resource Allocation Service

4.2.7.1 Purpose

The Semantic Resource Allocation Service provides a solution to a concrete instance allocation problem identified in Deliverable 2.1. In the FLEX Use Case, a concrete instance may consist, for example, of assigning a specific person to the programming of a robot. Figure 42 shows a UML-use-case diagram of the Semantic Resource Allocation Service.

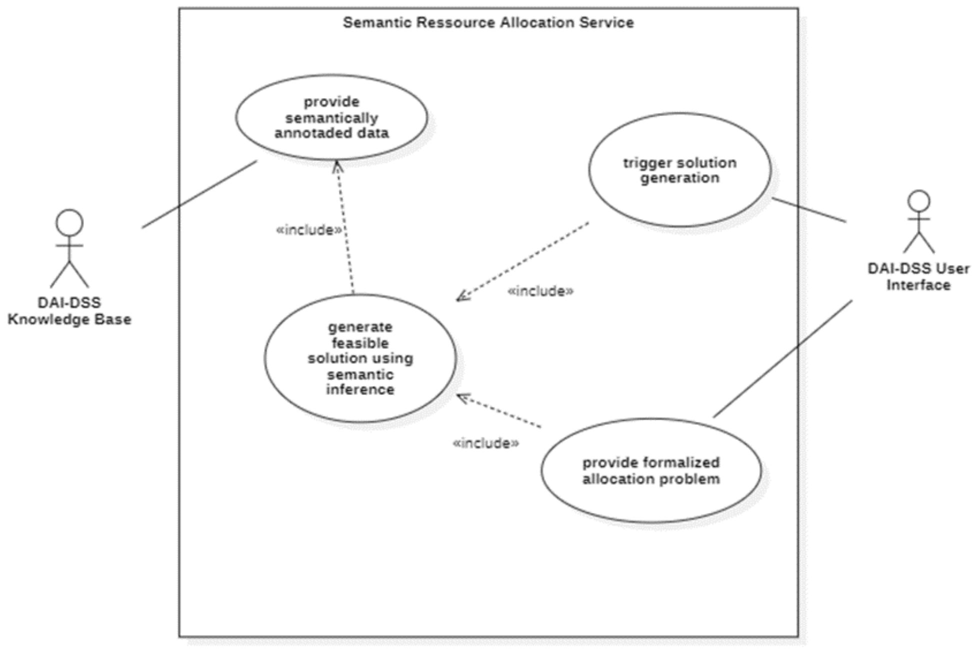

Semantic Ressource Allocation Service
provide semantically annotaded data
trigger solution generation
«include»
«include»
generate feasible solution using semantic inference
«include»
provide formalized allocation problem
DAI-DSS Knowledge Base
DAI-DSS User Interface

Figure 42 UML Use Case Diagram of the Semantic Resource Allocation Service

4.2.7.2 Service component

The Semantic Resource Allocation Service is a component that is not subdivided into other components. Figure 43 illustrates the interfaces between it and other components of the overall system using a component diagram. The service accesses the DAI-DSS Knowledge Base via a REST interface. The DAI-DSS Knowledge Base forms the ontological basis by which resource mapping can be derived utilizing semantic inference. The service is provided to the end user via the DAI-DSS user interface. It should be noted that the Semantic Resource Allocation Service does not create any visualizations. It only offers a solution for the specified problem if one exists.

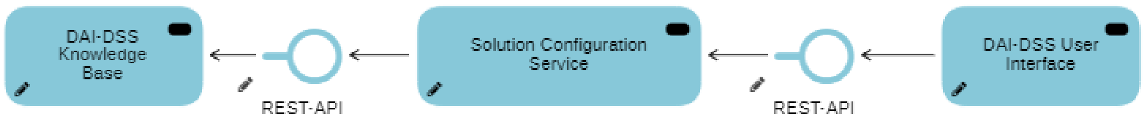


Figure 43 Service Component of interfaces between Semantic Resource Allocation Service and DAI-DSS Architecture

4.2.7.3 Interface and Data

Assignment of a resource for a concrete problem

- Interface: REST-API.
- Input: concrete problem instance in JSON format.
- Output: concrete mapping of the allocation task for the given problem instance.

4.2.7.4 Functions

The sole and only function of the Semantic Resource Allocation Service is to allocate one resource to one task. The Semantic Resource Allocation Service can only perform a resource allocation for a specific use case and not for a generic resource allocation problem.

4.2.7.5 Dependencies

The Knowledge Base stores semantically annotated data. Its acts as an ontology and enables the services to perform semantic inference.

4.2.8 Pattern Based Resource Allocation Service

4.2.8.1 Purpose

The Pattern Based Resource Allocation Service provides a solution to a concrete instance allocation problem identified in Deliverable 2.1. In the FLEX Use Case, a concrete instance may consist, for example, of assigning a specific person to the programming of a robot. Figure 44 shows a UML-use-case diagram of Pattern Based Resource Allocation Service. The DAI-DSS Knowledge Base provides historical data of resource allocations. Assuming that the past allocations are optimal or at least good allocations, machine learning methods can be applied to generate allocations for newly emerging resource allocation problems. Technologies such as neural networks or decision tree learning can be employed to solve this task.

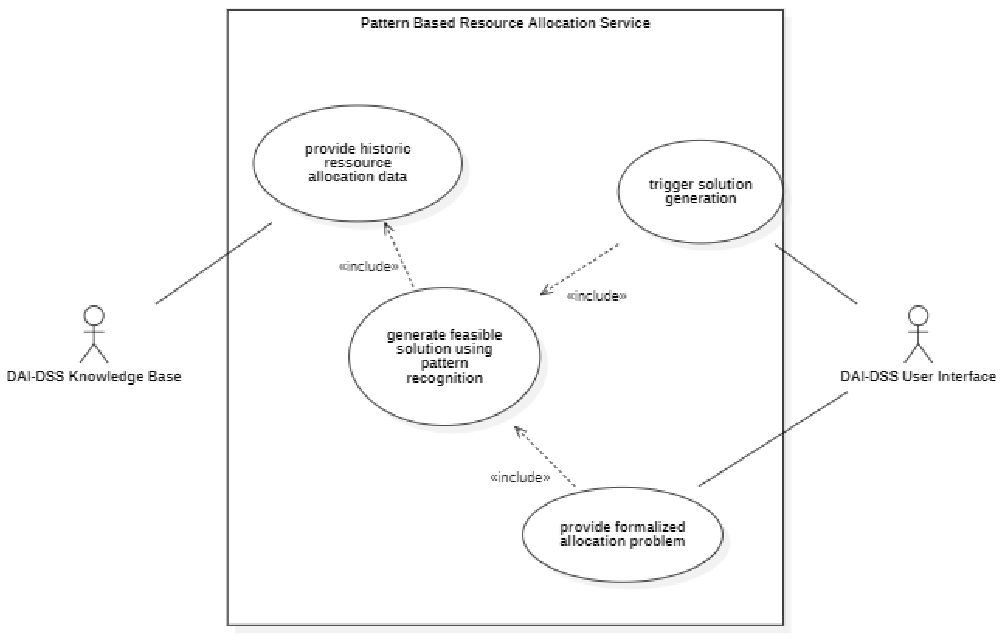

Pattern Based Resource Allocation Service
provide historic ressource allocation data
trigger solution generation
«include»
«include»
generate feasible solution using pattern recognition
DAI-DSS Knowledge Base
DAI-DSS User Interface
«include»
provide formalized allocation problem

Figure 44 UML Use Case Diagram of Pattern Based Resource Allocation Service

4.2.8.2 Service component

The Pattern Based Allocation Service is a component that is not subdivided into other components. Figure 45 illustrates the interfaces between it and other components of the overall system using a component diagram. The service accesses the DAI-DSS Knowledge Base via a REST interface. The DAI-DSS Knowledge Base provides historical data of resource allocations.

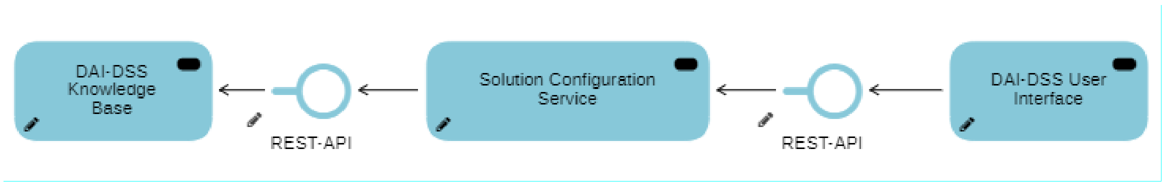


Figure 45 Service Component for interfaces between Pattern Based Resource Allocation Service and DAI-DSS Architecture

4.2.8.3 Functions

The sole and only function of the Pattern Based Allocation Service is to allocate one resource to one task. The Pattern Based Allocation Service can only perform a resource allocation for a specific use case and not for a generic resource allocation problem.

4.2.8.4 Interface and Data

Assignment of a resource for a concrete problem

- Interface: REST-API.
- Input: concrete problem instance in JSON format.
- Output: concrete mapping of the allocation task for the given problem instance.

4.2.8.5 Dependencies

The DAI-DSS Knowledge Base provides historical data of resource allocations. And enables the services to utilize machine learning methods.

4.2.9 Rule Based Resource Allocation Service

4.2.9.1 Purpose

The Rule Based Resource Allocation Service provides a solution to a concrete instance allocation problem identified in Deliverable 2.1. In the FLEX Use Case, a concrete instance may consist, for example, of assigning a specific person to the programming of a robot. Figure 46 shows a UML-use-case diagram of the Rule Based Resource Allocation Service. Rule-based mappings can be realized by (meta-) heuristics or by fuzzy rules. It should be noted that generic rule-based problem-solving methods, also known as metaheuristics, do not exist by construction according to the No free lunch theorem by David Wolpert and William Macready[31]. It may be that a concrete heuristic provides optimal solutions in one use-case but will not perform optimally in another.

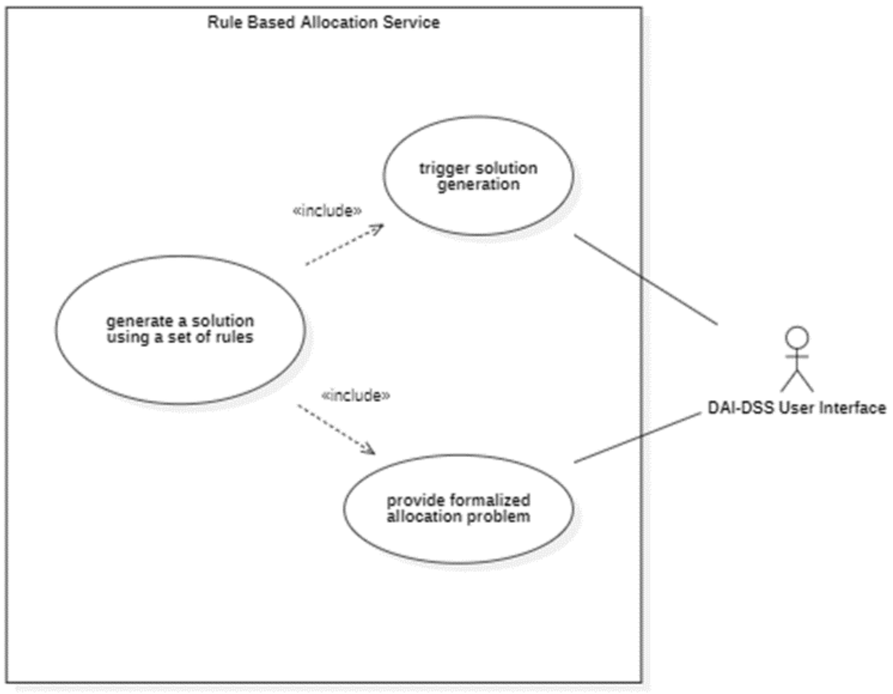

Rule Based Allocation Service
trigger solution generation
«include»
generate a solution using a set of rules
«include»
provide formalized allocation problem
DAI-DSS User Interface

Figure 46 UML Use Case Diagram of the Rule Based Resource Allocation Service

4.2.9.2 Service component

The Rule Based Allocation Service is a component that is not subdivided into other components. Figure 47 illustrates the interfaces between it and other components of the overall system using a component diagram.

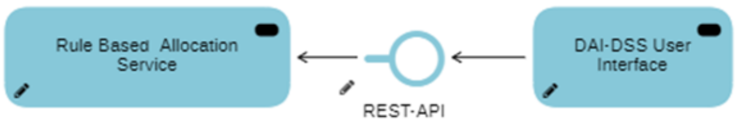


Figure 47 Service Component for interfaces between the Rule Based Resource Allocation Service and DAI-DSS Architecture

[31] D. Wolpert and W. Macready. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation, 1(1):67–82, 1997

4.2.9.3 Functions

The sole and only function of the Pattern Based Allocation Service is to allocate one resource to one task. The Pattern Based Allocation Service can only perform a resource allocation for a specific use case and not for a generic resource allocation problem.

4.2.9.4 Interface and Data

Assignment of a resource for a concrete problem:

- Interface: REST-API.
- Input: concrete problem instance in JSON format.
- Output: concrete mapping of the allocation task for the given problem instance.

4.2.9.5 Dependencies

The Rule Based Resource Allocation Services does not depend on other components.

4.2.10 Operator Stress Estimation with Neural Networks

4.2.10.1 Purpose

Stress directly affects human decision-making and might lead to numerous undesirable consequences, including a restriction or narrowing of attention, increased distraction, increases in reaction time and deficits in the person's working memory. Therefore, the consideration of levels of cognitive-emotional stress is important to harmonize man-machine production processes. A neural network (e.g., Support Vector Machine) is defined that map specific vital parameters (such as, heart rate, heart rate variability, breathing rate, eye movement features, etc.) to a score for cognitive-emotional stress refer Figure 48

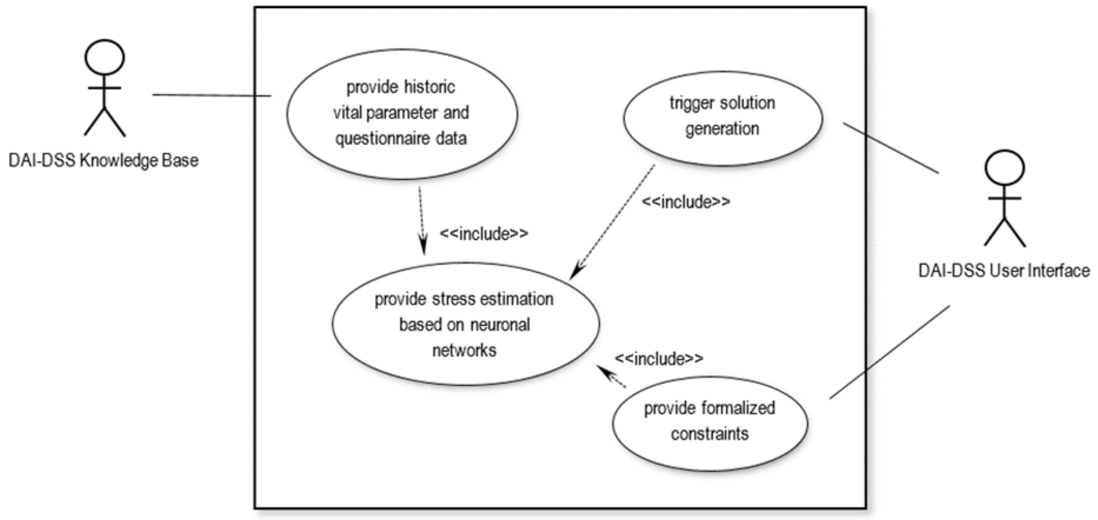

DAI-DSS Knowledge Base
provide historic vital parameter and questionnaire data
trigger solution generation
<<include>>
<<include>>
provide stress estimation based on neuronal networks
<<include>>
provide formalized constraints
DAI-DSS User Interface

Figure 48 UML-use-case diagram of Operator Stress Estimation Service

4.2.10.2 Service component

The operator stress estimation service is not subdivided into other components refer Figure 49.

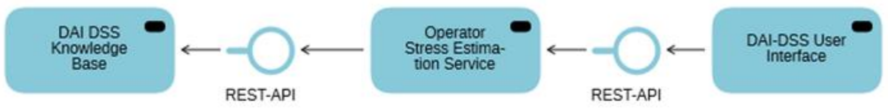


Figure 49 Service Component for Operator Stress Estimation Service

4.2.10.3 Functions

Learn a mapping between historic vital parameters and a score for cognitive-emotional stress and provide that mapping for later inference based on recorded or live data.

4.2.10.4 Interface and Data

The interfaces of the service for input and output data are based on a REST-API.

- Input: historic bio signal data provided by the DAI-DSS Knowledge Base or live data captured by the DAI-DSS Intelligent Sensor Box itself; formalized constraints and settings by the user; ISB's internal digital health profile.
- Output: inferenced stress level signal.
- Data format: json (for in and output data).

4.2.10.5 Dependencies

The operator stress estimation service depends on the DAI-DSS Knowledge base. In live mode the vital parameters are directly delivered by the DAI-DSS Intelligent Sensor Box itself. If user formalized constraints and settings are provided these data is used otherwise common default settings are applied.

4.2.11 Annotation Support Service

4.2.11.1 Purpose

Manual annotation of data is very tedious and time-consuming. Therefore, it is helpful to automate repetitive annotation as much as possible. The annotations required in the DAI-DSS Knowledge Base can be filled out automatically using the Annotation Support Service. This task is structure-wise like Active Learning in Machine Learning.[32]. It would be possible to provide information on how confident the service is when filling the annotation fields. This way, repetitive annotations can be automated, and only annotations where the service is uncertain can be manually annotated. Figure 50 shows a UML-use-case diagram of the Annotation Support Service.

[32] Settles, Burr. "Active learning literature survey." (2009)

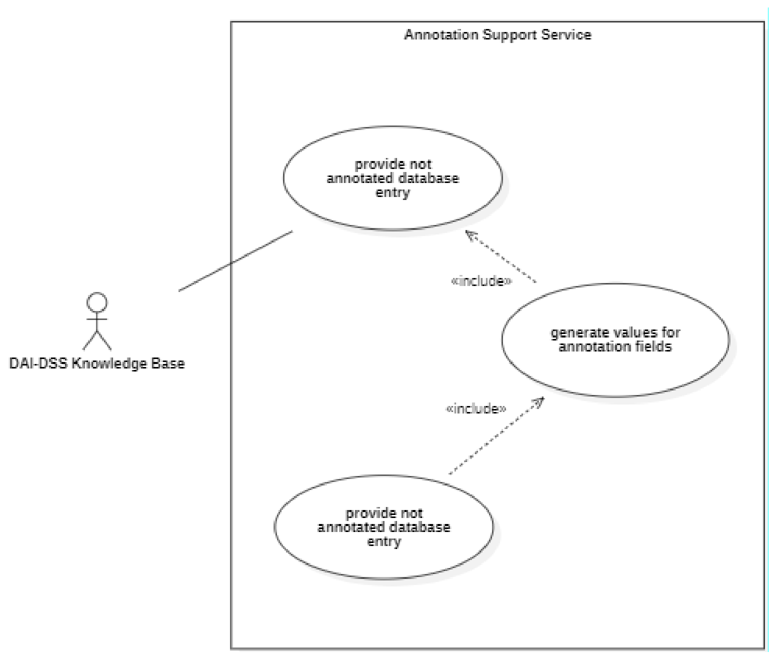

Annotation Support Service
provide not annotated database entry
«include»
generate values for annotation fields
«include»
provide not annotated database entry
DAI-DSS Knowledge Base

Figure 50 UML Use Case Diagram of the Annotation Support Service

4.2.11.2 Service component

The Configuration Support Service is a component that is not subdivided into other components. Figure 51 illustrates the interfaces between it and other components of the overall system using a component diagram.

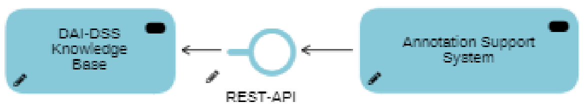


Figure 51 Service Component for interfaces between the Annotation Support Service and DAI-DSS Architecture

4.2.11.3 Functions

Fill out Annotation Fields in the DAI-DSS Knowledge Base based on historic annotation data.

4.2.11.4 Interface and Data

The Annotation Support Service does not require an interface to the outside world. The service could annotate all unannotated entries in the DAI-DSS knowledge base at a fixed time interval.

4.2.11.5 Dependencies

The DAI-DSS Knowledge Base provides unannotated data entries.

4.2.12 Production Process Simulation with Agents

4.2.12.1 Purpose

Production process simulation with agents allows the creation of a forecast that allows the anticipation of system behavior through modelling using Multi-Agent Systems. Simulation generates a representation of the system's behavior by using independent but communicating entities. These entities reproduce the simulated behavior through the interaction of agents provided with data that describe the state of the components of the production system through quantitative parameters refer Figure 52.

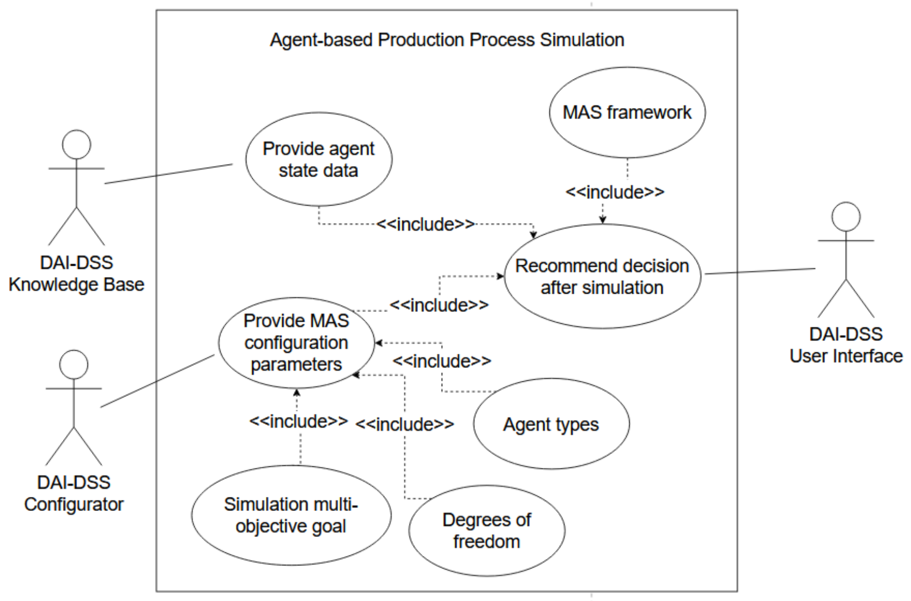


Figure 52 UML Use Case Diagram for Agent-based Production Process Simulation

4.2.12.2 Service Component

A production process simulation with agent's service can provide a forecast for the system's behavior based on how agents representing products and resources interact with each other on the shop floor. This service component realizes the simulation execution by acquiring the system current stat data in the Knowledge Base and providing a prevision to the end user by an agent-based simulation solution while receiving requirements configuration by the Configurator module. The internal components are depicted in Figure 53.

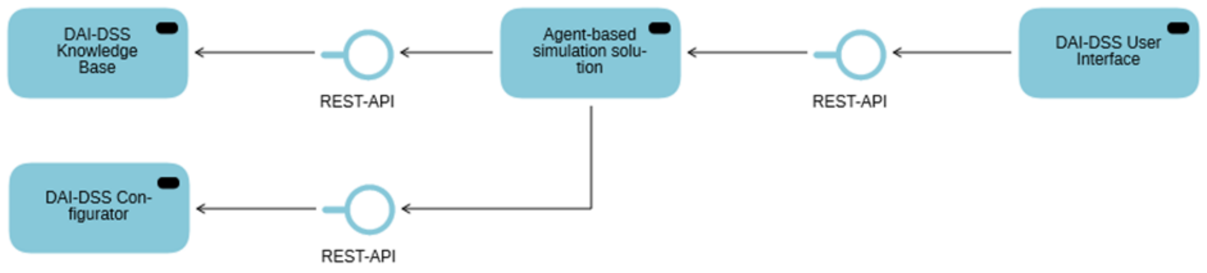


Figure 53 Service Component for production process simulation with agents

4.2.12.3 Interface and Data

The interface of the service for input and output data is based on a REST-API

- Communication: REST API.
- Input: Data structed in json format.
- Output: Comparative Agent's behavior based on the input data.

4.2.12.4 Functions

The Simulation Agents perform simulation based on the state of the assets in the field. This data is managed by the DAI-DSS AI Enrichment module that has specific agent-based services to train such data and forecast the system's behavior whenever it can ben simulated.

4.2.12.5 Dependencies

The Simulation Agent Type gets the parameters needed for simulation from the Agent Configurator and provide to the Agent based services with data that can be trained and used for behavior forecast of the system and give to the DAI-DSS User Interface an adequate system behavior prevision.

4.2.13 Configuration Support Service

4.2.13.1 Purpose

The Configuration Support Service is intended to assist in the setting up of configurations. This is realized by comparing past configurations with the present configuration task. The configurations that are most like the current configuration task should be made available to the user as an orientation. The similarity metric can be realized using machine learning techniques such as neural networks or decision tree learning refer Figure 54.

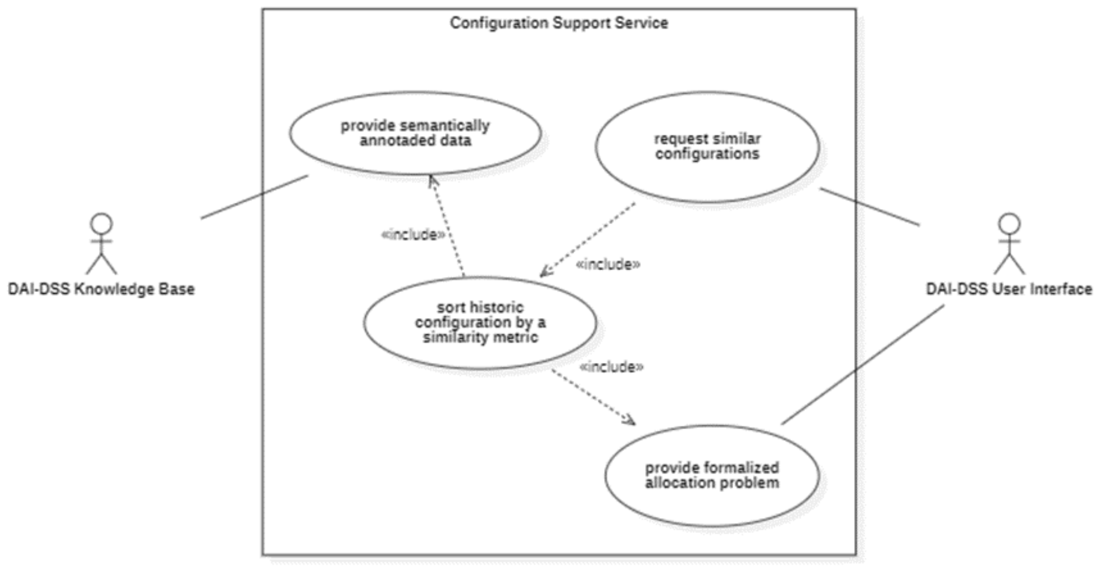

Configuration Support Service
provide semantically annotaded data
request similar configurations
«include»
«include»
DAI-DSS Knowledge Base
DAI-DSS User Interface
sort historic configuration by a similarity metric
«include»
provide formalized allocation problem

Figure 54 UML Use Case diagram of the Configuration Support Service

4.2.13.2 Service Component

The Configuration Support Service is a component that is not subdivided into other components. Figure 55 illustrates the interfaces between it and other components of the overall system using a component diagram.

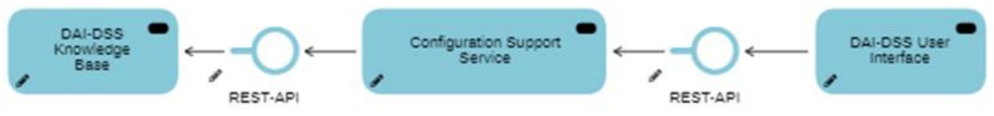


Figure 55 Service Component for interfaces between the Configuration Support Service and DAI-DSS Architecture

4.2.13.3 Functions

The only function of the Configuration Support Service is to provide a list of similar configurations to assist the user in a configuration task. The list of similar configurations is supposed to be viewed as examples. The Configuration Support Service only provides the data. Presentation, visualizations, and the like are not part of the service.

4.2.13.4 Interface and Data

Request similar configurations:

- Interface: REST-API.
- Input: concrete problem instance in JSON format.
- Output: a list of similar configurations.

4.2.13.5 Dependencies

The DAI-DSS Knowledge Base provides historical data of configurations.

4.2.14 Parameter Optimization Service with Machine Learning

4.2.14.1 Purpose

Optimizing the parameters is a time intensive task that requires a series of experiments with different parameter combinations. The complexity of this challenge increases with the number of parameters. In such a case, a well-known MLOps - Wandb [33]- can be used for a specific, selected use case. Wandb is an MLOps platform that aids in performance visualization and tracking of experiments for machine learning models. It supports automation, tracking, training, and improvement of ML models refer Figure 56.

[33] https://wandb.ai/site

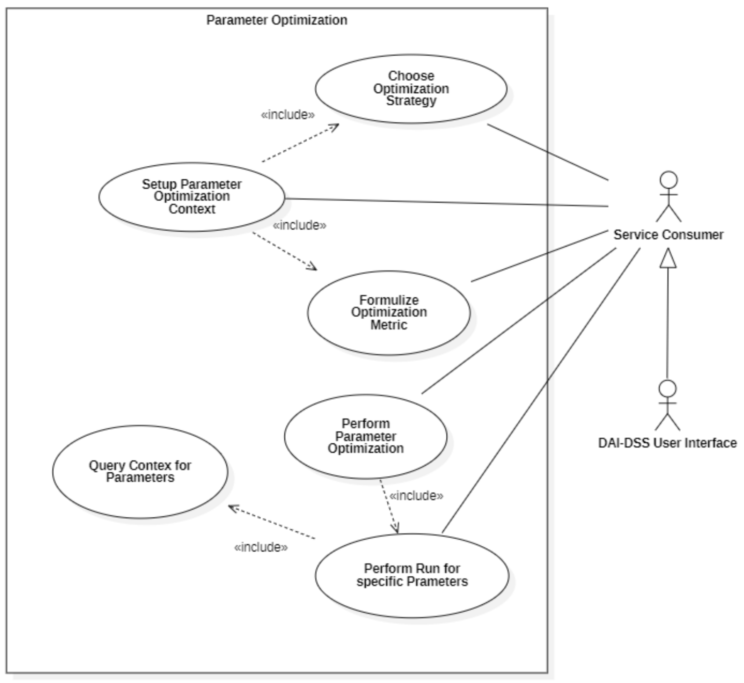

Parameter Optimization
Choose Optimization Strategy
«include»
Setup Parameter Optimization Context
«include»
Formulize Optimization Metric
Perform Parameter Optimization
«include»
Query Contex for Parameters
«include»
Perform Run for specific Prameters
Service Consumer
DAI-DSS User Interface

Figure 56 UML-use-case diagram for Parameter optimization service with machine learning

4.2.14.2 Service component

The Parameter Optimization Service is a component that is not subdivided into other components. Figure 57 illustrates the interfaces between it and other components of the overall system using a component diagram.

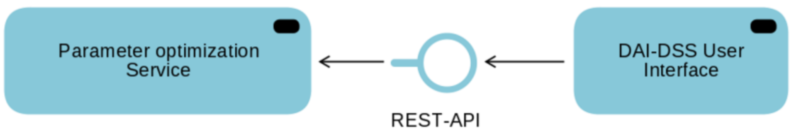


Figure 57 Service Component for interfaces between the Parameter Optimization Service and DAI-DSS Architecture

4.2.14.3 Functions

The function of the Parameter Optimization Service is parameter search (Bayesian, grid search or random) and model optimization.

4.2.14.4 Interface and Data

Parameter optimization in manufacturing:

- Interface: REST-API.
- Input: concrete problem instance in JSON format.
- Output: set of parameters.

4.2.14.5 Dependencies

No dependencies on other components of the DAI-DSS Architecture.

4.2.15 Operator Persona Development with Machine Learning

4.2.15.1 Purpose

Personas represent a group or population of real human operators in terms of their Human Factors features cognitive, affective, social, and motivational parameters. In the context of FAIRWork, we also refer to definition or even rule base-like characterization of risk levels with respect to physiological strain, cognitive-emotional stress as well as fatigue. Specifically, based on a set of human profiles, a clustering methodology, such as, via a machine learning (ML) driven stochastic process, will determine a mathematical representation of the entire group with minimal global distances in the feature space. The description of the persona can in principle also contain data from questionnaire-based scoring to include subjective data Figure 58.

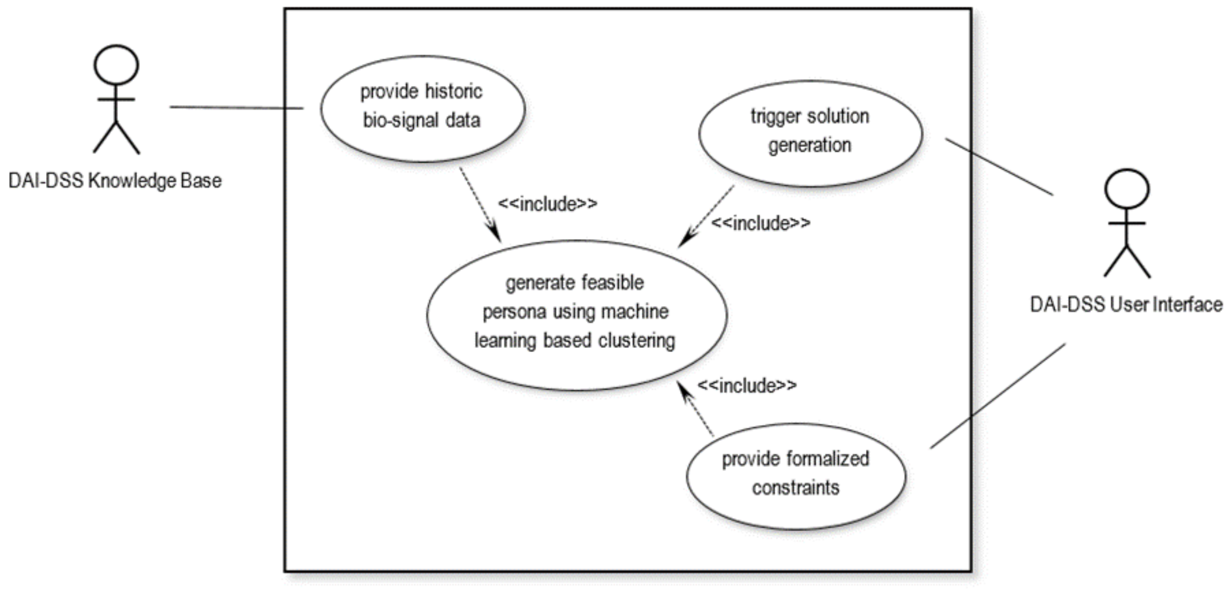

DAI-DSS Knowledge Base
provide historic bio-signal data
trigger solution generation
<<include>>
<<include>>
generate feasible persona using machine learning based clustering
<<include>>
provide formalized constraints
DAI-DSS User Interface

Figure 58 UML Use Case Diagram of Operator Persona Development Service

4.2.15.2 Service component

The operator persona development service is atomic and thus not subdivided into other components Figure 59.

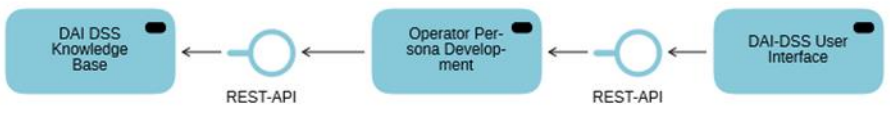


Figure 59 Service Component for Operator Persona Development Service

4.2.15.3 Functions

Learn and provide personas based on a set of human profiles with minimal global distances in the feature space.

4.2.15.4 Interface and Data

The interfaces of the service for input and output data are based on a REST-API.

- Input: formalized persona related constraints provided by the user and historic bio signal data.
- Output: Persona profile.
- Data format: json (for in and output data).

4.2.15.5 Dependencies

The operator persona development service depends on user formalized constraints and settings via the DAI-DSS User Interface and on the DAI-DSS Knowledge base providing historic data.

4.2.16 Community Detection with Knowledge Graphs

4.2.16.1 Purpose

The purpose of this service is to find groups of concepts in the data base, which can be used to allocate resources to tasks in a production line. The assumption is that the data of resources (e.g., human workers) and tasks in a production line are available in a knowledge graph structure. This graph structure is then used to identify groups of concepts (e.g., resources or tasks) that are highly linked together. The identified groups or communities (as they are also called), can then be used to find similar concepts (e.g., similar workers) and support in this way the decision-making process. A use case diagram of this service can be seen in Figure 60.

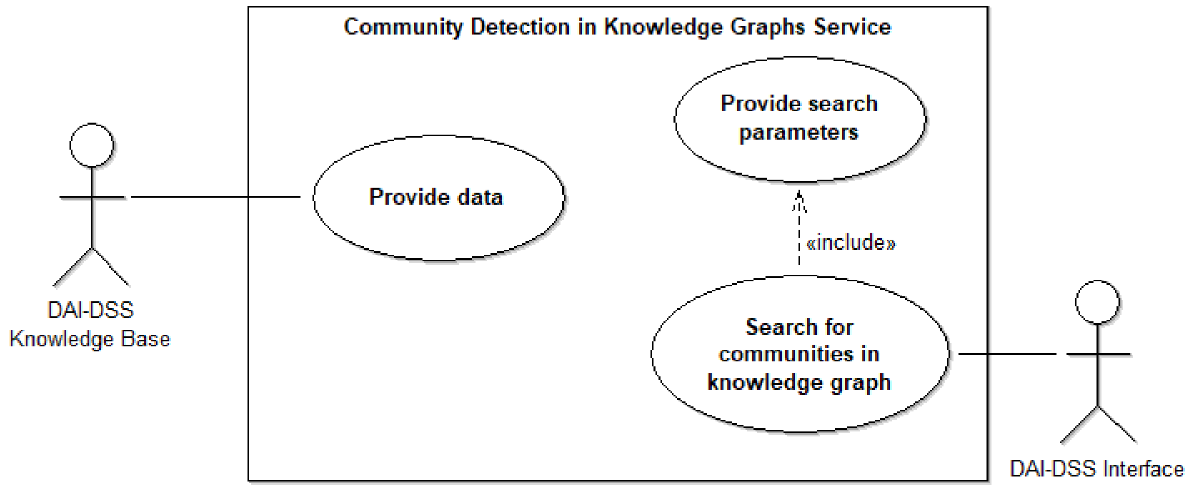

Community Detection in Knowledge Graphs Service
Provide search parameters
Provide data
«include»
Search for communities in knowledge graph
DAI-DSS Knowledge Base
DAI-DSS Interface

Figure 60 UML Use Case Diagram for the community detection in knowledge graphs service

4.2.16.2 Service Component

The "community detection in knowledge graphs" component will be provided as a singular service component and therefore will not be divided into subcomponents. The service component will gather the needed information from the knowledge base and the user. From the DAI-DSS knowledge base it needs the knowledge graph data and from the DAI-DSS User Interface the search must be triggered and parameters for defining the searched community must be provided. The service will then return the concepts in the community. An overview of the service component can be seen in Figure 61.

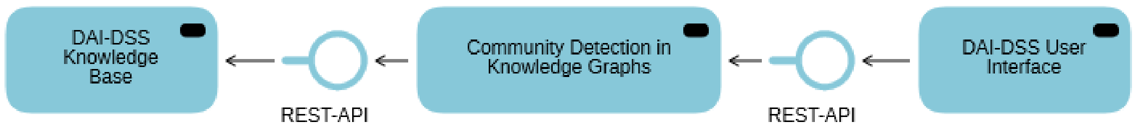


Figure 61 Service Component for community detection with knowledge graphs service

4.2.16.3 Interface Data

Search for communities in the knowledge graph:

- Interface: REST.
- Input: search parameters for the community, e.g., JSON or RDF-based data-structure.
- Output: found communities and their concepts.

4.2.16.4 Functions

The main function of this service component is the searching of communities in a knowledge graph representation of the environment (e.g., the production line). The communities must not explicitly be defined but can be found from the general structure of the saved information. The service then returns a group of similar concepts that can be further utilized for the support of the decision-making process.

4.2.16.5 Dependencies

The service component depends on the available information. This must be on the one hand be in a knowledge graph structure and additionally enough information must be available so that meaningful communities can be found.

4.2.17 Production Process Simulation

4.2.17.1 Purpose

The purpose of this service is to provide additional information for the decision of which configuration of the production line should be used. Here not only static information of the production process of a configuration will be used, but also dynamic aspects should be considered to allow better insight in the production process, before it is established in the real production line. To do this the different production process configuration should be simulated to create more information for making the decision for a certain configuration. A use case diagram for this service component can be found in Figure 62.

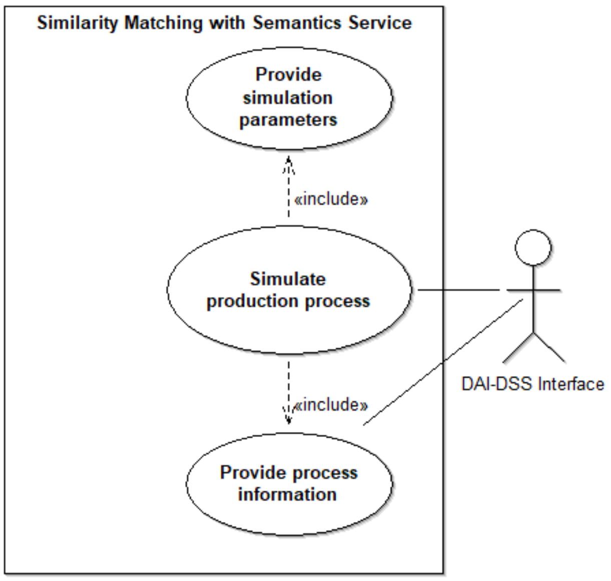

Similarity Matching with Semantics Service
Provide simulation parameters
«include»
Simulate production process
«include»
Provide process information
DAI-DSS Interface

Figure 62 UML Use Case Diagram for the production process simulation service

4.2.17.2 Service Component

A production process simulation will be provided as a singular service component and therefore will not be divided into subcomponents. The service allows to simulate a production process, which correspondence to a configuration of the production line. Therefore, the service component allows to provide the information about the process and the configuration of the simulation. Then the process can be simulated, and the simulation results are returned to support the overall decision-making process. An overview of the service component can be seen in Figure 63.

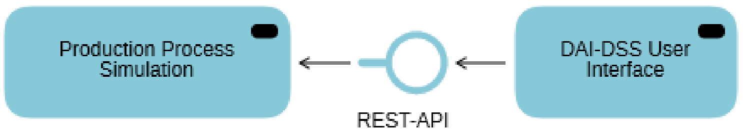


Figure 63 Service component for production process simulation service

4.2.17.3 Interface Data

Production process simulation:

- Interface: REST.
- Input: process and simulation data, e.g., JSON or XML.
- Output: simulation results.

4.2.17.4 Functions

This service components offers the functionality to simulate production process of a certain configuration of the production line. Therefore, the process and the simulation parameters can be provided to the service, the service then executes the service and returns the simulation results.

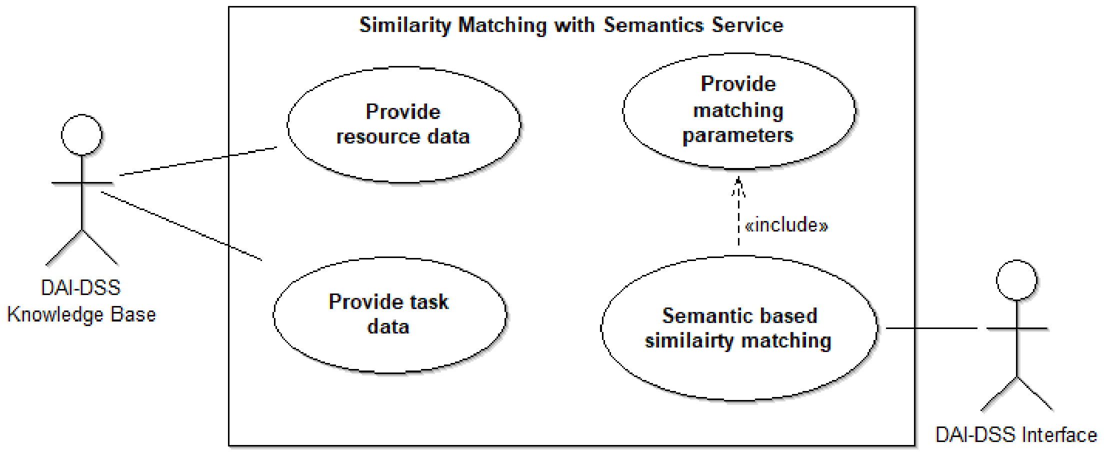
4.2.17.5 Dependencies

This service component depends on the availability of the production process configuration information and the data to create to define the production process.

4.2.18 Similarity Matching with Semantics

4.2.18.1 Purpose

This service focuses on the allocation of resources (e.g., human workers) to tasks in a production line. Both are described with their properties. Resources have capabilities which must match the requirements of the tasks. This service focuses on semantic rich matching of these two sides, by not only allowing a one-to-one matching between capabilities and requirements, but to allow the service to understand them and match them based on their semantics. A use case diagram for this service component can be found in Figure 64.



Similarity Matching with Semantics Service
Provide resource data
Provide matching parameters
«include»
Provide task data
Semantic based similairty matching
DAI-DSS Knowledge Base
DAI-DSS Interface

Figure 64 UML Use case diagram for the similarity matching with semantics service

4.2.18.2 Service Component

The "similarity matching with semantics" service component will be provided as a singular service component and therefore will not be divided into subcomponents. The service component offers a semantic based similarity matching which can be triggered if resources must be allocated to tasks. Then the needed data is gathered from the knowledge base and the similarity matching will be performed. The trigger will contain the parameter to guide the matching, for example for which task a resource should be allocated. The matching will not be based on a direct matching of saved data, but the service component will use the semantic of the task and resource data. For example, this can be done by applying *Natural Language Processing (=NLP)* approaches to better understand the data. An overview of the service component can be seen in Figure 65.

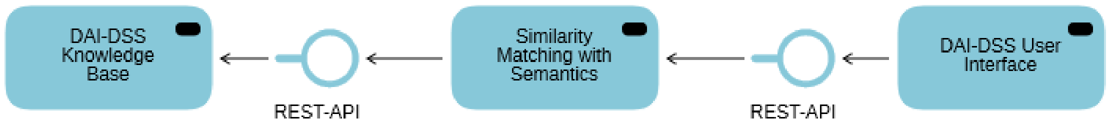


Figure 65 Service component for similarity matching with semantics service

4.2.18.3 Interface Data

Semantic based similarity matching between resources and tasks:

- Interface: REST.
- Input: matching parameters and matching data (tasks and resources), e.g., JSON or RDF-based data-structure.

- Output: list of matchings with similarity measure.

4.2.18.4 Functions

The function this service component offers, is a matching between resources and tasks, which supports the decision-making process by providing matches between resources and tasks, based on their semantics.

4.2.18.5 Dependencies

This service components depends on a description of the task requirements and the resource capabilities in a format, so that they can be consumed and processed with the semantic based similarity matching.

4.2.19 Impact Assessment with Knowledge Graphs

4.2.19.1 Purpose

This service allows to analyze the impact that certain events will have on the overall configuration of a production line. This can support the decision-making process by showing how certain events influence the overall system. The system will be described in a knowledge graph and this graph structure will be exploited to find the influenced components of the system. This analysis allows to better understand configurations within a system and how certain events (like the missing of resources) influence it. Based on this insight, the decision-making process can be improved. A use case diagram for this service component can be found in Figure 66.

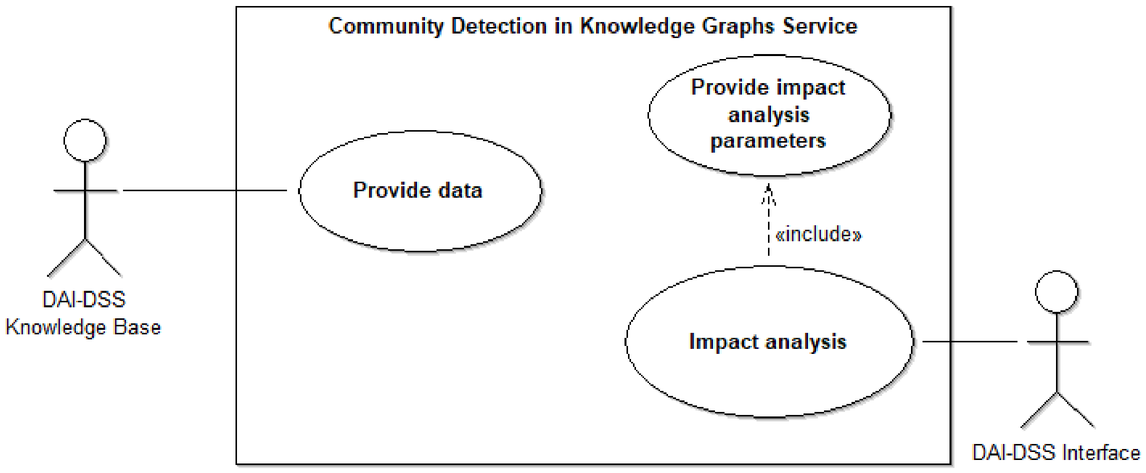

Community Detection in Knowledge Graphs Service
Provide impact analysis parameters
Provide data
«include»
Impact analysis
DAI-DSS Knowledge Base
DAI-DSS Interface

Figure 66 UML Use Case diagram for the impact assessment with knowledge graphs service

4.2.19.2 Service Component

The "impact assessment with knowledge graphs" service component will be provided as a singular service component and therefore will not be divided into subcomponents. The service will analyze the impact a certain event makes on a system. The system itself will be described in a knowledge graph and an actor a trigger the analysis by providing the parameters of the analysis, e.g., the event. An overview of the service component can be seen in Figure 67.

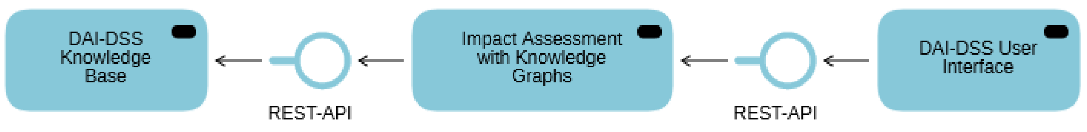


Figure 67 Service component for impact assessment with knowledge graphs service

4.2.19.3 Interface Data

Production process simulation:

- Interface: REST.
- Input: system description and analysis configuration, e.g., as RDF base data structure or JSON.
- Output: analysis results.

4.2.19.4 Functions

This service component offers the functionality to analyze the impact an event has on a system (e.g., a production line). Therefore, the system description must be provided in knowledge graph structure and the analysis must be triggered. Afterwards the components or parts of the system will be returned, which are influenced by a certain event. These identified components will then be used support the decision-making process.

4.2.19.5 Dependencies

The service depends on the availability of a knowledge graph description of the system.

4.2.20 Multi Agent-based Resource Allocation

4.2.20.1 Purpose

Multi-agent-based resource allocation serves to manage the allocation of scarce resources using the intelligence provided by the agents in a competitive setting. A recommended decision on resource allocation is expected as an outcome from the iterations on Multi-Agent System configuration and recollection of historical data when available refer Figure 68.

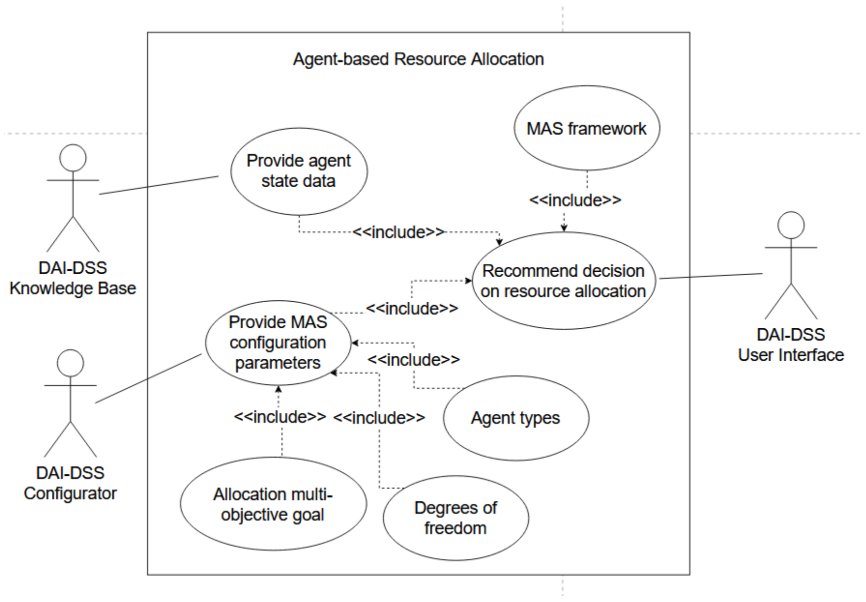


Figure 68 UML Use Case Diagram for Agent-based Resource Allocation

4.2.20.2 Service Component

A resource allocation solution is provided to the end user whenever an optimization process is regarding resource scarcity can be managed by a multi-Agent approach. Data about resource shortage can be accessed in the Knowledge Base so a solution can be provided to the end user via this multi-Agent solution and parameters regarding current task requirements can be accessed by the Configurator setup refer Figure 69.

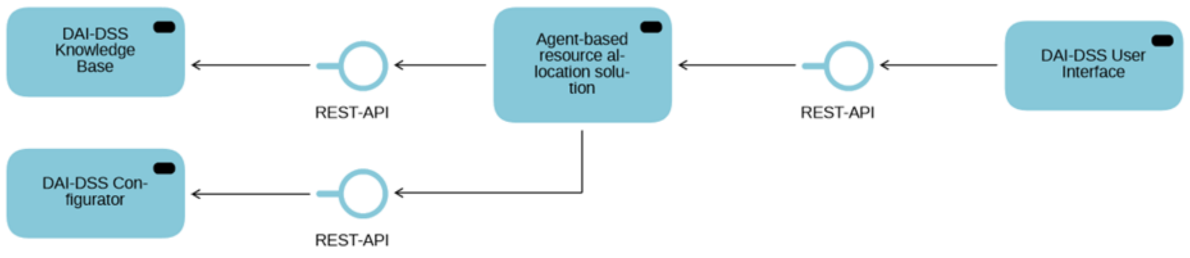


Figure 69 Service Component for production process simulation with agents

4.2.20.3 Interface and Data

The interfaces of the service for input and output data are based on an REST-API

- Communication: REST API.
- Input: Agent’s process requirements.
- Output: Product optimization based on scarce resource negotiation.

4.2.20.4 Functions

It can be realized in an auction manner where each resource can be allocated to the most interested bidder. Agents compete against each other so that most desire the resource will receive if they can bid against their peers. Auctions can happen in a multi-agent system for promoting a competitive environment where decision are taking on individual interests that reflect the overall system’s desire.

4.2.20.5 Dependencies

Resource allocation is associated with the resources available in the Real World and how the system wants to allocate these resources. Therefore, the Multi-Agent System is dependent on knowing which configuration setting is desired the end user and it can provide an intelligent solution based on competitive negotiation among agents deployed on the field taking advantage of the previous data collected by the DAI-DSS Knowledge Base. It can provide an optimized outcome for resourced that need to be better addressed along the production chain.

5 FUNCTONAL CAPABILTIES

This chapter contains a list of identified functional capabilities. These are the initial capabilities which are identified for the DAI-DSS components which is expected to evolve during the project duration. The next version of the deliverable in M20 will update the list.

The following aspects are described:

- ID	: A unique identification code
- Component	: Component of the FAIRWork architecture
- Requirement	: Name of the requirement
- Description	: The description of the requirement

ID	Component	Requirement	Description
R-001	Knowledge base	User and role management	Creation and management of users with assigned roles that give specific data access in Knowledge Base.
R-002	Knowledge base	Digital representation of physical assets	The system shall be able to represent the physical assets in a scenario as digital models and store data and documents related to those assets.
R-003	Knowledge base	Properties	The system shall be able to store user defined properties to any breakdown element (folder, document, data set, part, etc.).
R-004	Knowledge base	Sensor data	The system shall be able to represent sensors and store measured readings.
R-005	Knowledge base	Aggregate data	The system shall be able represent aggregates of user defined data structures and store these as property data of breakdown elements.
R-006	Knowledge base	Data storage	The system shall be able to store structured and unstructured data files also with associated properties
R-007	Knowledge base	Machine learning results	The system shall be able to store results from machine learning algorithms.
R-008	Knowledge base	Data exchange	All data in the system shall be retrievable through a REST API interface.
R-009	User interface	Standardized user interface	Using well known MS teams
R-010	Orchestrator	Execution across clouds	It must capably execute services that are deployed across different clouds
R-011	Orchestrator	Multi-agent/ Workflow Orchestration	The system shall recognize the best approach for processing the optimization/simulation task.

R-012	Orchestrator	Support to decision-making	The system shall provide a plural number of recommendations to decision-making whenever those options are available.
R-013	Orchestrator	Multi-agent resource allocation	The system shall be able to provide a valid and democratic approach for proper allocation of processing tasks.
R-014	Service Catalogue	Standardized Interface Structure	The interfaces of the services should follow a common logic and structure to support easier integration and reuse.
R-015	Service Catalogue	Common Deployment Environment	There should be a common deployment procedure and environment to support an easy and automated reuse of the services
R-016	Service Catalogue	Common Service Framework	A common framework for offering and using the services should be used to ease the reuse of the system.
R-017	Orchestrator	Multi-agent operational environments	The Multi-Agent System shall be able to perform control over hardware and software aiming at orchestrating the execution of processing tasks
R-018	User interface	Modularity	The use of front end Microservices should enable the possibility to create use case specific user interfaces
R-019	User interface	Usability	The user face should be easy to use and have a clear layout by using well known design features
R-020	User interface	Communication with Orchestrator	The user should be able to retrieve data and call services via the orchestrator
R-021	User interface	Configurable	The widgets displayed in the user interface should configurable using configuration files created by the configurator component
R-022	User interface	Deployable on different devices and platform	The web application should be deployable on different well-known platforms (e.g., MS Teams, Confluence etc.) and devices (phone, laptop etc.)
R-023	Configurator	Model-awareness	Enables the creation of decision models using different model environments
R-024	Configurator	Communication with user interface	Enables creation of configuration files for user interfaces

R-025	Configurator	Communication with orchestrator	Enables creation of configuration files containing information about services or workflows and the deployment of them
R-026	Configurator	Communication with knowledge base	Configuration files should be storable in the knowledge base for further use
R-027	Configurator	Provide Assessment service	Feedback should be collectable from the “real world” and evaluated to adjust models and services when needed
R-028	AI Enrichment	Providing AI services	The component shall provide data-driven models for chosen agents in given use cases
R-029	AI Enrichment	Execution of models	The component shall be able to execute developed models
R-030	AI Enrichment	Data exchange	The component shall be able to provide results of developed models in JSON format
R-031	External Data Asset Market	Provide external data set	The component provides a searchable data catalog with external data sets.
R-031	External Data Asset Market	Provide external data set	This component provides a user interface to find, search and described data assets.
R-032	External Data Asset Market	Provide external data set	The component provides an API for integrating data sets in in 3rd party tools
R-033	Intelligent Sensor Box	Provide personal data	The components provide data from human-centred worker profiles in an anonymized way.
R-034	Intelligent Sensor Box	Provide personal data	The components provide data from human-centred worker profiles including provide risk levels (e.g., health related risk levels and safety risk levels for human worker profiles).

6 SECURITY

In this section, the result of the initial analysis of the required security functionality or services is described. To this aim, several minimum-security controls are outlined below that should be adopted in the use cases, describing them in standard terms that have their roots in international standards such as those proposed by ENISA for Industry 4.0 systems and the Cloud Security Alliance's IoT controls, as explained below refer Figure 70. This taxonomy includes the list of affected assets per threat such as industrial internet of things devices, ICS, or Cloud computing services.

In the first area of interest, we must care about securing the sensor networks. First, it is about the security of the sensors. These sensors, or sensor boxes, can be attacked by hackers and thus represent vulnerabilities for networks. Since such devices are often visible and thus easily found, this makes them accessible and vulnerable to attack. For this reason, sensors in industrial environments must be tested for proven attack patterns to maintain their secure functionality in an emergency. However, this also means that it is possible to check whether sensor devices meet existing security requirements before they go into operation.

Second, ensuring communication in sensor systems and the sensor of the machinery is highly relevant. In the context of Industry 4.0, large and sometimes confidential amounts of data are exchanged between sensors and other systems in real time. For this reason, securing communication channels and the underlying computing units is of utmost importance. Data security between interfaces must therefore be systematically checked and any defensive measures installed.

Third, it is also about functional resilience: the functionality of sensors must be maintained as far as possible even in case of physical impacts, failures, dynamic environments, and misinformation. For this purpose, the respective sensor devices must be tested and safeguarded against various unexpected scenarios. As another very relevant international reference, the Cloud Security Alliance (CSA) has recently released a complete IoT Security Controls Framework, which supports security experts assessing the security controls required in a specific IIoT system. This framework has been defined taking into consideration cybersecurity standards, guidelines and reports from well-regarded international organizations leading cybersecurity standardization such as NIST, ISO, ENISA, OWASP and ISA. In such a service-oriented working area are the following typical security issues around Web/Service applications.

Based on the performed analysis about the cyber threats on our environment, the specified principles when deciding the list of security controls for the decision environment are mainly the following ones:

1. Device, network, and gateway are the most critical layers of the DAI-DSS framework.

2. System impact levels from higher to lower are Availability > Integrity > Confidentiality. Availability is the cyber security attribute with more impact in the FAIRWork decision system; then, integrity and finally, confidentiality. It is crucial that all assets of the decision environment system have high availability for the correct data and models in operation. Furthermore, the data gathered and processed by all the parts of the system must keep integrity and shall not be tampered with or modified on purpose or accidentally, so the system or the operators themselves can make correct data analysis and decisions in the next steps of the business process workflow. Lastly, underlying data can be subject to data breaches and espionage if confidentiality is broken but these threats seem to be less harmful at least in the short-term.

3. The scope of the security perspective in DAI-DSS *platform* is focused on both automatic and semi-automatic technical mechanisms, leaving aside manual mechanisms or processes.

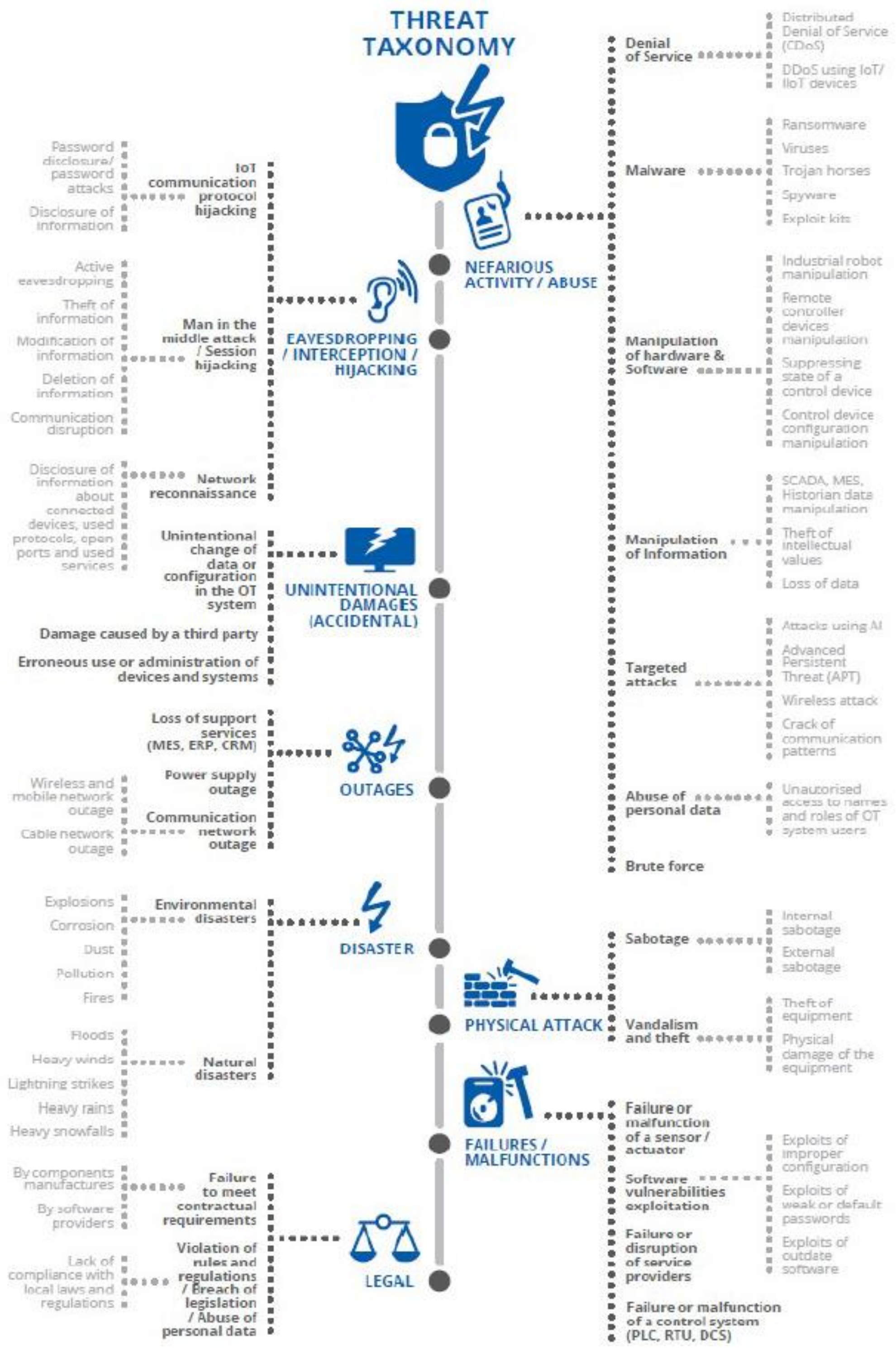

THREAT TAXONOMY
NEFARIOUS ACTIVITY / ABUSE
Denial of Service
Distributed Denial of Service (CDoS)
DDoS using IoT/ IIoT devices
Malware
Ransomware
Viruses
Trojan horses
Spyware
Exploit kits
Manipulation of hardware & Software
Industrial robot manipulation
Remote controller devices manipulation
Suppressing state of a control device
Control device configuration manipulation
Manipulation of Information
SCADA, MES, Historian data manipulation
Theft of intellectual values
Loss of data
Targeted attacks
Attacks using AI
Advanced Persistent Threat (APT)
Wireless attack
Crack of communication patterns
Abuse of personal data
Unautorised access to names and roles of OT system users
Brute force
EAVESDROPPING / INTERCEPTION / HIJACKING
IoT communication protocol hijacking
Password disclosure/ password attacks
Disclosure of information
Man in the middle attack / Session hijacking
Active eavesdropping
Theft of information
Modification of information
Deletion of information
Communication disruption
Network reconnaissance
Disclosure of information about connected devices, used protocols, open ports and used services
UNINTENTIONAL DAMAGES (ACCIDENTAL)
Unintentional change of data or configuration in the OT system
Damage caused by a third party
Erroneous use or administration of devices and systems
OUTAGES
Loss of support services (MES, ERP, CRM)
Power supply outage
Communication network outage
Wireless and mobile network outage
Cable network outage
DISASTER
Environmental disasters
Explosions
Corrosion
Dust
Pollution
Fires
Natural disasters
Floods
Heavy winds
Lightning strikes
Heavy rains
Heavy snowfalls
PHYSICAL ATTACK
Sabotage
Internal sabotage
External sabotage
Vandalism and theft
Theft of equipment
Physical damage of the equipment
FAILURES / MALFUNCTIONS
Failure or malfunction of a sensor / actuator
Software vulnerabilities exploitation
Exploits of improper configuration
Exploits of weak or default passwords
Exploits of outdate software
Failure or disruption of service providers
Failure or malfunction of a control system (PLC, RTU, DCS)
LEGAL
Failure to meet contractual requirements
By components manufactures
By software providers
Violation of rules and regulations / Breach of legislation / Abuse of personal data
Lack of compliance with local laws and regulations

Figure 70 ENISA threat taxonomy focused on Industry 4.0

There are three new categories, four categories with naming and scoping changes, and some consolidation in the Top 10 for 2021.

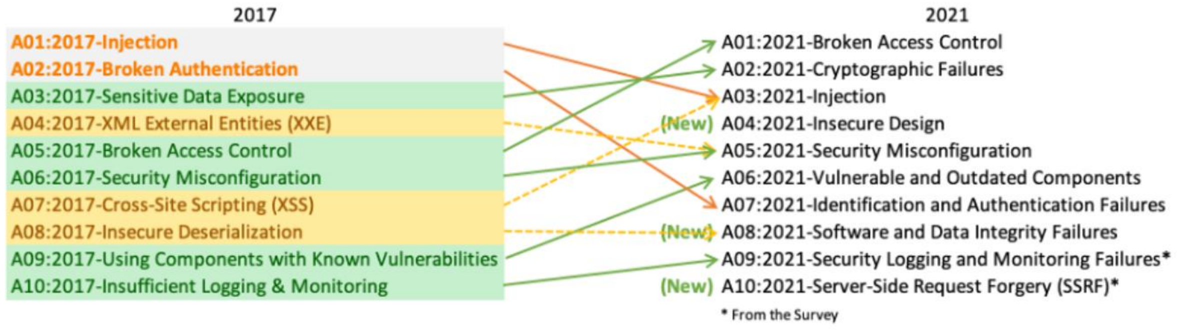

2017
A01:2017-Injection
A02:2017-Broken Authentication
A03:2017-Sensitive Data Exposure
A04:2017-XML External Entities (XXE)
A05:2017-Broken Access Control
A06:2017-Security Misconfiguration
A07:2017-Cross-Site Scripting (XSS)
A08:2017-Insecure Deserialization
A09:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring
2021
A01:2021-Broken Access Control
A02:2021-Cryptographic Failures
A03:2021-Injection
(New) A04:2021-Insecure Design
A05:2021-Security Misconfiguration
A06:2021-Vulnerable and Outdated Components
A07:2021-Identification and Authentication Failures
(New) A08:2021-Software and Data Integrity Failures
A09:2021-Security Logging and Monitoring Failures*
(New) A10:2021-Server-Side Request Forgery (SSRF)*
* From the Survey

Figure 71 Changing of threat scenarios from 2017 till 2022.

7 ARCHITECTURE TESTING AND REPORTING

In this section an initial plan for testing and reporting the DAI-DSS Architecture is discussed. This is section acts as reference for testing DAI-DSS architecture and reporting the results and a more detailed testing processes, methodologies, environments, and reporting will be discussed in next version of deliverable D4.1 in M30. The next version will describe details of testing environment, configuration and format of test reports and their storage and exchange in more detail.

7.1 Architecture Testing

Test architecture is the discipline of looking at a stream of delivery and working out what, how and when to test to achieve the best possible outcome. It produces a framework of test activities that can then be scoped, prepared, and executed. In this project a V model of testing will be referred at every phase of the development of DAI-DSS architecture refer Figure 72. V-Model also referred to as the Verification and Validation Model. In this, each phase of software development life cycle (SDLC) must complete before the next phase starts. It follows a sequential design process same as the waterfall model (Hamilton, T)[34].

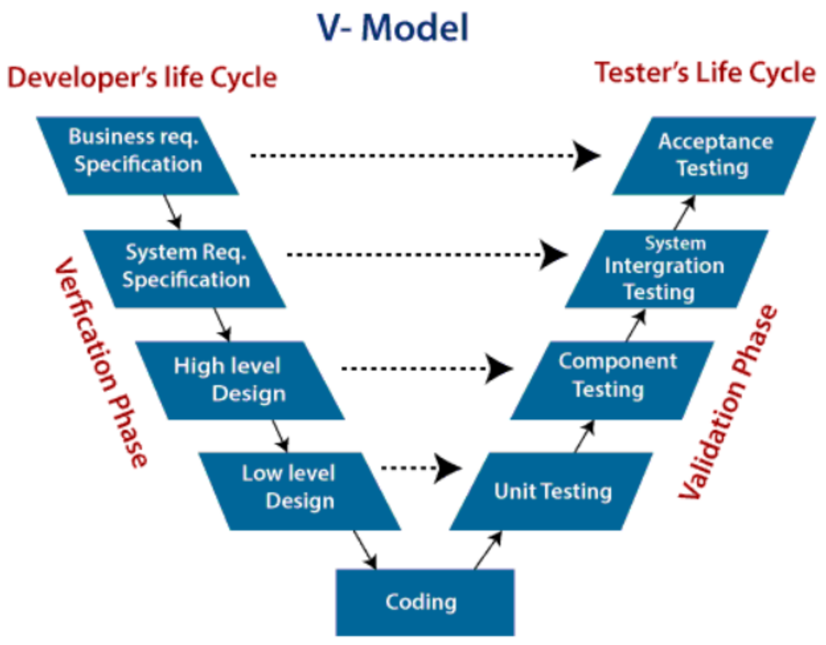


Figure 72 V model Architecture testing

Verification: It involves a static analysis method (review) done without executing code. It is the process of evaluating the product development process to find whether required conditions are met.

Validation: It involves dynamic analysis method (functional, non-functional), testing is done by executing code. Validation is the process to classify the software after the completion of the development process to determine whether the software meets the requirements. So, V-Model contains Verification phases on one side of the

[34] Hamilton, T. (2022, December 31). V-model in software testing. Guru99. Retrieved February 21, 2023, from https://www.guru99.com/v-model-software-testing.html

Validation phases on the other side. Verification and Validation process is joined by coding phase in V-shape. Thus, it is known as V-Model[35].This will be used a reference for planning and testing of DAI-DSS architecture.

7.2 Architecture Test Reporting

The DAI-DSS architecture test report will focus on including the details of the environment like where the code was tested, who tested it, when it was tested and how it was tested. The document serves as a physical log that records exactly what code was tested, in what system configuration the code was tested on and any bugs that came up during testing. The overall goal of the test report summary is to record the actions and results of a test.

The test reports will be determined based on the target audience and why the report is needed. For example, if the application requires an audit trail for regulatory reasons, the test report's likely need to include more specific data. If the target audience is FAIRWork project members, they want to understand what the team has tested and then the report can include a summary that outlines the main functions tested. So, in general the test report tries to cover the following details.

- **Test objective**: The test objective should describe the type of testing like unit or functional testing and which component or service is tested and why it has been tested and expected result of the testing.

- **Test execution details:** The next component of a test report is an explanation of the test suite. This comprises the details of type of test that was executed, where it was stored, and when it was performed.

- **Defects:** Another critical part is recording the defect or bugs discovered during testing. The test report should include more detailed description of defects encountered during testing.

- **Platform and test environment configuration details:** The test report should also need to have the information about the type of configuration and environments used for the testing. This enables to identity the cause of defects during configuration changes.

[35] V-model (software engineering) - javatpoint. www.javatpoint.com. (n.d.). Retrieved February 21, 2023, from https://www.javatpoint.com/software-engineering-v-model

8 SUMMARY AND CONCLUSION

This deliverable introduces the DAI-DSS Architecture and its relation to the project use cases scenarios as CRF and FLEX. The use cases have a complex interaction of human, AI, robots, and machines working together in manufacturing environment. These interactions need to be optimized for better overall production rate. To address issues in the use cases, we propose a Democratic AI-based Decision Support System (DAI-DSS), which provides decision support using AI. Furthermore, a simple coffee making scenario was presented, to introduce how decision-making processes can be divided in humans and AI based decisions.

Following the project introduction, an initial overview of the high-level architecture is presented. It includes the key components of DAI-DSS as well as their interface and connections to the other DAI-DSS components and services. This high-level architecture was then mapped to the individual use case scenarios.

The high-level architecture is mapped to the use case scenarios that are for FLEX "Automated Test Building", "Worker Allocation" and "Machine Maintenance After Breakdown" as well as for CRF "Workload Balance", "Delay of Material" and "Quality Issues". Based on the process analysis described in more detail in "D2.1 - Specification of FAIRWork and Initial DAI-DSS Architecture" we identified generic challenges to the aforementioned use case scenarios. This ensures that the architecture and the implemented solutions will not exclusively fit to those use cases but also serve similar use cases that are out of the project.

The challenges are: "finding similar projects", "find relevant experts", "simulate production process", "allocate worker", "map workers with profiles", "find similar problems", "reschedule production line", "allocate order to production line", "assess the impact". Such challenges are targeted by Microservices that may use Artificial Intelligence (AI) when appropriate. Hence, we propose a list of services that either individually or in a cooperative manner target the aforementioned list of challenges.

In the subsequent section, a detailed description of each component and their technical implementation details are presented, including the purpose of the component in the DAI-DSS architecture, as well as their internal architecture, functions, dependencies, and supported types of interfaces and data to other components.

The (a) cores components are (i) the user interface that is a framework enabling user interface widgets to be deployed in environment like web-pages of MS TEAMS, (ii) the orchestrator that is a framework enabling the controlling of individual services or the orchestration of services, (iii) the knowledge base that integrates data from legacy applications, sensor data streams that require no protection and sensor data stream that require protection in form of an Intelligent Sensor Box (e.g. for human-related information); (iv) the externa data asset marketplace complements the data coming from inside the use case with available data form outside, (v) the configurator that enables the selection and configuration of services to a use-case specific solution and finally (vi) the so-called AI-enrichment which is a service catalogue providing different AI-based services that fulfill the end users' needs.

The (b) list of (AI) services is a collection of available services, commercial products and research prototypes that are partly used from outside the consortium where feasible and partly created during the project. Hence, we currently propose an initial list of services that target the aforementioned challenges and propose different AI realizations to demonstrate the flexibility of our core components and to enable a selection of appropriate AI solutions. This list is therefore seen as indicative, and it is expected that in the duration of the project it will evolve. The final set of services will also be provided as projects results in the so-called innovation shop.

Chapter 4 described the proposed catalogue of AI services in the DAI-DSS architecture that are used to provide decision support in the use cases. The descriptions include purpose, functions, and dependencies.

1) Order to Production Line with Fuzzy Logic
2) Order to Production Line with Neural Networks
3) Similarity Matching with Knowledge Graphs
4) Resource mapping with Decision Trees
5) Worker to Production Line Mathematical Optimization/heuristic
6) Reinforcement Learning Based Resource Allocation Service
7) Sematic Resource Allocation Service
8) Pattern Based Resource Allocation Service
9) Rule Based Resource Allocation service
10) Operator Stress Estimation with Neural Networks
11) Annotation Support Service
12) Production Process Simulation with Agents
13) Configuration Support Service
14) Parameter Optimization Service with Machine Learning
15) Operator persona development with Machine Learning
16) Community Detection with Knowledge Graphs
17) Production Process Simulation
18) Similarity Matching with Semantics
19) Impact Assessment with Knowledge Graphs
20) Multi Agent-based Resource Allocation

The functional capabilities of DAI-DSS components were summarized in chapter 5. These will provide the summary of requirements for each component based on the sources listed, ensuring that components can interact with each other and to the services seamlessly within the DAI-DSS architecture. The security section in chapter 6 introduces potential security vulnerabilities and explains required areas to implement control measures to avert data breaches. This section will be updated in the next version of this deliverable. The initial testing plan and reporting methodologies for the DAI-DSS architecture has been introduced, which will be filled in the next version of this deliverable.

Overall, the document provides a baseline for the initial design and implementation of the DAI-DSS architecture along with the proposed application services and a plan for testing and reporting the architecture. In the final version of the document D4.1 the implemented architecture details along with updated application services and functional capabilities will be updated. In addition, the implemented security measure and the detailed architecture testing methods and reporting will be complemented.

9 REFERENCES

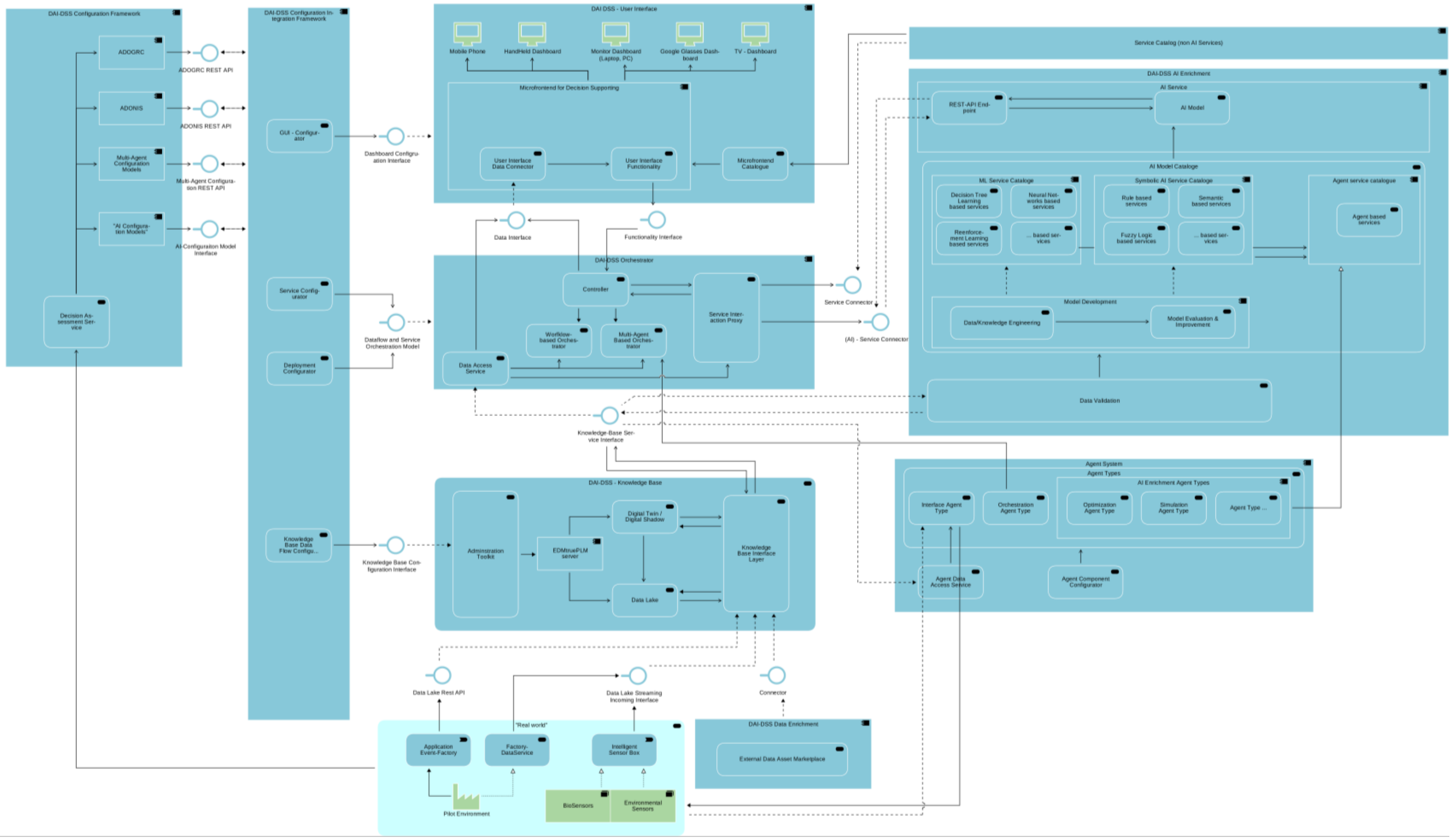
References are included as footnote within the text.

ANNEX A: LIST OF TOOLS

ADONIS®	Business Process modelling tool, http://www.boc-eu.com,
ADOxx	Metamodelling Platform, https://www.adoxx.org
Scene2Model	Digital Design Thinking Tool, https://omilab.org
OLIVE	Model-Aware Microservice Environment, https://www.adoxx.org/

ANNEX B: LIST OF HIGH-QUALITY FIGURES

Annex B.1 FAIRWork High Level Architecture


DAI-DSS Configuration Framework
ADOGRC
ADONIS
Multi-Agent Configuration Models
"AI Configuration Models"
Decision Assessment Service
ADOGRC REST API
ADONIS REST API
Multi-Agent Configuration REST API
AI-Configuraiton Model Interface
DAI-DSS Configuration Integration Framework
GUI - Configurator
Service Configurator
Deployment Configurator
Knowledge Base Data Flow Configu...
Dashboard Configuration Interface
Dataflow and Service Orchestration Model
Knowledge Base Configuration Interface
DAI DSS - User Interface
Mobile Phone
HandHeld Dashboard
Monitor Dashboard (Laptop, PC)
Google Glasses Dashboard
TV - Dashboard
Microfrontend for Decision Supporting
User Interface Data Connector
User Interface Functionality
Microfrontend Catalogue
Data Interface
Functionality Interface
DAI-DSS Orchestrator
Controller
Workflow-based Orchestrator
Multi-Agent Based Orchestrator
Service Interaction Proxy
Data Access Service
Service Connector
(AI) - Service Connector
Knowledge-Base Service Interface
Service Catalog (non AI Services)
DAI-DSS AI Enrichment
AI Service
REST-API End-point
AI Model
AI Model Catalogue
ML Service Cataloge
Decision Tree Learning based services
Neural Networks based services
Reenforcement Learning based services
... based services
Symbolic AI Service Cataloge
Rule based services
Semantic based services
Fuzzy Logic based services
... based services
Agent service catalogue
Agent based services
Model Development
Data/Knowledge Engineering
Model Evaluation & Improvement
Data Validation
Agent System
Agent Types
AI Enrichment Agent Types
Interface Agent Type
Orchestration Agent Type
Optimization Agent Type
Simulation Agent Type
Agent Type ...
Agent Data Access Service
Agent Component Configurator
DAI-DSS - Knowledge Base
Adminstration Toolkit
EDMtruePLM server
Digital Twin / Digital Shadow
Data Lake
Knowledge Base Interface Layer
Data Lake Rest API
Data Lake Streaming Incoming Interface
Connector
"Real world"
Application Event-Factory
Factory-DataService
Intelligent Sensor Box
Pilot Environment
BioSensors
Environmental Sensors
DAI-DSS Data Enrichment
External Data Asset Marketplace