



DAI-DSS ARCHITECTURE AND INITIAL DOCUMENTATION AND TEST REPORT

D4.1

Editor Name	Rishyank Chevuri (Jotne Connect)
Submission Date	April 30, 2024
Version	2.0
State	Final
Confidentially Level	PU



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

EXECUTIVE SUMMARY

This deliverable “D4.1.1 – FAIRWork Architecture and initial Documentation and Test Report” is the second iteration of the deliverable D4.1 which has been submitted in the month M6 of the project. This iteration of the deliverable builds upon it and provides a comprehensive overview of the current status of the implementation of DAI-DSS components and services required for the decision support.

This iteration outlines key performance indicators for FLEX, covering "Automated Test Building," "Worker Allocation," and "Machine Maintenance After Breakdown," as well as for CRF, focusing on "Workload Balance," "Delay of Material," and "Quality Issues." These key performance indicators (KPIs) have been revised to align with the FAIRWork goals and objectives. These metrics will be used as benchmarks for evaluating the solutions proposed by the DAI-DSS for the described challenges.

The list of AI services proposed to support the use case challenges are revised in this iteration and further classified into three categories based on their development status: "Initial Prototype", "Development in Progress" and "Planned for Implementation" indicating their status of implementation. In addition to this the prerequisites and usage descriptions for the services have been added. These additional descriptions outline the specific circumstances under which the service is designed to operate, which aids in the selection process, ensuring that end users choose the most appropriate service for their specific scenarios.

This iteration further specifies the software and hardware requirements essential for the operation of DAI-DSS components and provides the status of the functional capability's implementations, including User Interface, Configurator, Orchestrator, Knowledge Base, AI Enrichment, and Sensor Boxes.

Furthermore, the information related to the security considerations critical to the DAI-DSS architecture, highlighting mechanisms for data protection, authentication, and authorization has been described. This includes employing robust security practices such as SSL certificates, two-factor authentication, role-based access control, and secure data transmission protocols, which ensures the system is safeguarded against unauthorized access and data breaches, ensuring the protection of sensitive information. Additionally, an outlook on the necessary security measures and implementations aimed at enhancing the DAI-DSS's security has been described.

Finally, the iteration provides an overview of the testing methods employed to evaluate the integrity and functionality of the DAI-DSS architecture. Describing the methodologies for assessing the interactions between both internal and external components, the functional evaluation of each distinct component, and the integration testing of the components. This ensures that the DAI-DSS operates seamlessly, maintaining high standards of quality and reliability in its performance. The contents of this deliverable will contribute to the “D4.3 Final DAI-DSS Prototype, Documentation and Test Report”, scheduled for M30.

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
Rishyank (Jotne) Herwig Zeiner, Lucas Paletta, Michael Schneeberger (JR) Robert Woitsch, Magdalena Dienstl (BOC) Gutavo Vieira (MORE) Sylwia Olbrych, Lauwigi Johanna, Alexander Nasuta (RWTH) Christian Muck (OMiLAB) Roland sitar (FLEX)	Wilfrid Utz (OMiLAB) Damiano Falcioni (BOC) Marco Kemmerling (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
2.0	April 30, 2024	Rishyank (Jotne) and Contributors	All

Copyright Statement – Restricted Content

This document does not represent the opinion of the European Community, and the European Community is not responsible for any use that might be made of its content.

This is a restricted deliverable that is provided to the community under the license Attribution-No Derivative Works 3.0 Unported defined by creative commons <http://creativecommons.org>

You are free:

	to share within the restricted community — to copy, distribute and transmit the work within the restricted community
Under the following conditions:	
	Attribution — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
	No Derivative Works — You may not alter, transform, or build upon this work.

With the understanding that:

Waiver — Any of the above conditions can be waived if you get permission from the copyright holder.

Other Rights — In no way are any of the following rights affected by the license:

- Your fair dealing or fair use rights.
- The author's moral rights;
- Rights other persons may have either in the work itself or in how the work is used, such as publicity or privacy rights.

Notice — For any reuse or distribution, you must make clear to others the license terms of this work.

This is a human-readable summary of the Legal Code available online at:

<http://creativecommons.org/licenses/by-nd/3.0/>

TABLE OF CONTENT

LIST OF ABBREVIATIONS AND ACRONYMS	17
1 Introduction	19
1.1 Purpose of the Document	19
1.2 Document Structure	19
2 HIGH LEVEL architecture	20
2.1 High Level Architecture Overview	20
2.2 Use Case Mapping with the DAI-DSS High Level Architecture.....	22
2.2.1 Flex Automated Test Building Process	24
2.2.2 Flex Worker Allocation Process	25
2.2.3 FLEX Machine Maintenance After Breakdown Process	26
2.2.4 CRF Workload Balance Process	28
2.2.5 CRF Delay of Material Process	30
2.2.6 CRF Quality Issue Process.....	31
3 Architecture components	34
3.1 DAI-DSS User Interfaces	35
3.1.1 Purpose	35
3.1.2 Internal Architecture.....	35
3.1.2.1 Back End for Decision Supporting Dashboards	35
3.1.2.2 User Interface Data Connector	36
3.1.2.3 User Interface Functionality	37
3.1.3 Functions	37
3.1.4 Dependencies.....	38
3.1.5 Interfaces and Data	38
3.1.6 Development Focus Area	39
3.1.7 Hardware and Software requirements.....	39
3.2 DAI-DSS Configurator.....	39
3.2.1 Purpose	39
3.2.2 Internal Architecture.....	39
3.2.2.1 GUI Configurator.....	40
3.2.2.2 Service Configurator	40
3.2.2.3 Deployment Configurator	40
3.2.2.4 Knowledge Base Data Flow Configurator	41
3.2.2.5 Decision Assessment Service.....	41
3.2.3 Functions	41

3.2.4	Dependencies.....	43
3.2.4.1	DAI-DSS User Interfaces	43
3.2.4.2	DAI-DSS Orchestrator	43
3.2.4.3	Model environments and platforms	44
3.2.5	Interfaces and Data	44
3.2.6	Development Focus Area	45
3.2.7	Hardware and Software requirements	45
3.3	DAI-DSS Orchestrator	45
3.3.1	Purpose	45
3.3.2	Internal Architecture.....	46
3.3.2.1	Controller	46
3.3.2.2	Service Interaction Proxy	46
3.3.2.3	Data Access Service	47
3.3.2.4	Workflow based Orchestrator	47
3.3.2.5	Multi-Agent based Orchestrator	47
3.3.3	Functions	47
3.3.4	Dependencies.....	47
3.3.4.1	DAI-DSS Configurator.....	48
3.3.4.2	Multi-Agent System.....	48
3.3.4.3	DAI-DSS Knowledge Base	48
3.3.4.4	DAI-DSS AI Enrichment.....	48
3.3.4.5	DAI-DSS User Interfaces	48
3.3.5	Interfaces and Data	48
3.3.6	Development Focus Area	49
3.3.7	Hardware and Software requirements	49
3.4	DAI-DSS Knowledge Base.....	50
3.4.1	Purpose	50
3.4.2	Internal Architecture.....	51
3.4.3	Functions	52
3.4.3.1	Data storage	52
3.4.3.2	REST API services and IoT integration.....	53
3.4.3.3	EDMTruePLM client.....	53
3.4.3.4	Digital Twin / Shadow representation	53
3.4.4	Dependencies.....	54
3.4.4.1	DAI-DSS Configurator.....	54
3.4.4.2	DAI-DSS Orchestrator	54

3.4.4.3	DAI-DSS AI Enrichment.....	54
3.4.4.4	Data Sources	54
3.4.5	Interfaces and data	55
3.4.5.1	REST API	55
3.4.5.2	EDMtruePLM web client	55
3.4.6	Development Focus Area	55
3.4.6.1	IoT interfaces	55
3.4.6.2	Notifications	56
3.4.6.3	3D viewer in web-client.....	56
3.4.7	Hardware and Software requirements	56
3.5	DAI-DSS AI Enrichment.....	56
3.5.1	Purpose	56
3.5.2	Internal Architecture.....	57
3.5.3	Functions	59
3.5.4	Dependencies.....	60
3.5.5	Interfaces and Data	60
3.5.6	Development Focus Area	60
3.5.7	Other Details.....	61
3.5.7.1	Multi-Agent System.....	61
3.5.7.1.1	Definition of an Agent	61
3.5.7.1.2	Communication between agents	62
3.5.7.1.3	Communication between agents and other DAI-DSS components	62
3.5.7.1.4	Agent descriptions	63
3.5.7.1.5	Development focus area.....	65
3.5.8	Hardware and Software requirements	65
3.6	DAI-DSS External Data Asset Marketplace	66
3.6.1	Purpose	66
3.6.2	Internal Architecture.....	66
3.6.3	Functions	67
3.6.3.1	User Interfaces.....	67
3.6.3.2	Search & Filters	67
3.6.3.3	Metadata enrichment	67
3.6.3.4	Metadata transformation	67
3.6.4	Dependencies.....	67
3.6.5	Interface and Data	67
3.6.6	Development Focus Area	67

3.6.7	Other Details.....	68
3.6.8	Hardware and Software requirements.....	68
3.7	DAI-DSS Intelligent Sensor Box.....	68
3.7.1	Purpose	68
3.7.2	Internal Architecture.....	68
3.7.2.1	Local Wearable Sensor Network	69
3.7.2.2	Local Workplace Sensor Network.....	70
3.7.2.3	Local Data Receiver Module.....	70
3.7.2.4	Local Data Management.....	70
3.7.2.5	Digital Health Profile	70
3.7.2.6	Human Factors Intelligent Services	70
3.7.2.7	Local DSS based on Risk Levels.....	71
3.7.2.8	User Interfaces Providing Individual Human Factors State.....	72
3.7.2.9	Data Anonymization.....	72
3.7.2.10	Developer Dashboard	72
3.7.3	Functions	73
3.7.4	Dependencies.....	74
3.7.4.1	DAI-DSS Knowledge Base	74
3.7.4.2	DAI-DSS AI Enrichment.....	74
3.7.5	Interfaces and Data	74
3.7.5.1	REST Interface	74
3.7.5.2	MQTT Interface.....	74
3.7.5.3	Data format	75
3.7.6	Development Focus Area	75
3.7.7	Hardware and Software requirements.....	75
4	Application Service catalogue for decision making.....	76
4.1	Structure of component descriptions.....	76
4.2	AI services list	76
4.2.1	Worker to Production Line Mathematical Optimization/Heuristic	78
4.2.1.1	Purpose	78
4.2.1.2	Service component	79
4.2.1.3	Interface and Data	79
4.2.1.4	Functions	79
4.2.1.5	Dependencies.....	79
4.2.1.6	Prerequisites.....	80
4.2.2	Resource Allocation Service with Linear Sum Assignment Solvers.....	80

4.2.2.1	Purpose	80
4.2.2.2	Service Component	80
4.2.2.3	Interface and Data	81
4.2.2.4	Functions	81
4.2.2.5	Dependencies	81
4.2.2.6	Prerequisites	81
4.2.2.7	Usage	81
4.2.3	Pattern Based Resource Allocation Service	81
4.2.3.1	Purpose	81
4.2.3.2	Service Component	82
4.2.3.3	Functions	82
4.2.3.4	Interface and Data	82
4.2.3.5	Dependencies	82
4.2.3.6	Prerequisites	82
4.2.3.7	Usage	83
4.2.4	Operator Stress Estimation with Neural Networks	83
4.2.4.1	Purpose	83
4.2.4.2	Service Component	83
4.2.4.3	Functions	83
4.2.4.4	Interface and Data	84
4.2.4.5	Dependencies	84
4.2.4.6	Prerequisites	84
4.2.5	Multi Agent-based Resource Allocation	84
4.2.5.1	Purpose	84
4.2.5.2	Service Component	85
4.2.5.3	Interface and Data	85
4.2.5.4	Functions	85
4.2.5.5	Dependencies	85
4.2.5.6	Prerequisites	85
4.2.5.7	Usage	86
4.2.6	Rule-based Service based on Conceptual Models	86
4.2.6.1	Purpose	86
4.2.6.2	Service Component	86
4.2.6.3	Interface and Data	87
4.2.6.4	Functions	88
4.2.6.5	Dependencies	88

4.2.6.6	Prerequisites	88
4.2.6.7	Usage	89
4.2.7	Rule Based Resource Allocation Service	89
4.2.7.1	Purpose	89
4.2.7.2	Service Component	89
4.2.7.3	Functions	90
4.2.7.4	Interface and Data	90
4.2.7.5	Dependencies	90
4.2.7.6	Prerequisites	90
4.2.8	Worker Efficiency Estimation	90
4.2.8.1	Purpose	90
4.2.8.2	Service Component	91
4.2.8.3	Interface and Data	91
4.2.8.4	Functions	91
4.2.8.5	Dependencies	92
4.2.8.6	Prerequisites	92
4.2.9	Resource Mapping with Decision Trees	92
4.2.9.1	Purpose	92
4.2.9.2	Service Component	92
4.2.9.3	Interface and Data	93
4.2.9.4	Functions	93
4.2.9.5	Dependencies	93
4.2.9.6	Prerequisites	94
4.2.10	Schedule Optimizer with Linear Programming	94
4.2.10.1	Purpose	94
4.2.10.2	Service Component	94
4.2.10.3	Interface and Data	95
4.2.10.4	Functions	95
4.2.10.5	Dependencies	95
4.2.10.6	Prerequisites	95
4.2.11	Similarity Matching with Knowledge Graphs	95
4.2.11.1	Purpose	95
4.2.11.2	Service Component	96
4.2.11.3	Interface and Data	96
4.2.11.4	Functions	97
4.2.11.5	Dependencies	97

4.2.11.6	Prerequisites	97
4.2.12	Reinforcement Learning Based Resource Allocation Service	97
4.2.12.1	Purpose	97
4.2.12.2	Service Component	98
4.2.12.3	Functions	98
4.2.12.4	Interface and Data	98
4.2.12.5	Dependencies	98
4.2.12.6	Prerequisites	99
4.2.12.7	Usage	99
4.2.13	Annotation Support Service	99
4.2.13.1	Purpose	99
4.2.13.2	Service Component	100
4.2.13.3	Functions	100
4.2.13.4	Interface and Data	100
4.2.13.5	Dependencies	100
4.2.13.6	Prerequisites	100
4.2.14	Production Process Simulation with Agents	100
4.2.14.1	Purpose	100
4.2.14.2	Service Component	101
4.2.14.3	Interface and Data	101
4.2.14.4	Functions	101
4.2.14.5	Dependencies	102
4.2.14.6	Prerequisites	102
4.2.15	Operator Persona Development with Machine Learning	102
4.2.15.1	Purpose	102
4.2.15.2	Service Component	102
4.2.15.3	Functions	103
4.2.15.4	Interface and Data	103
4.2.15.5	Dependencies	103
4.2.16	Community Detection with Knowledge Graphs	103
4.2.16.1	Purpose	103
4.2.16.2	Service Component	104
4.2.16.3	Interface Data	104
4.2.16.4	Functions	104
4.2.16.5	Dependencies	105
4.2.16.6	Prerequisites	105

4.2.17	Production Process Simulation	105
4.2.17.1	Purpose	105
4.2.17.2	Service Component	106
4.2.17.3	Interface Data	107
4.2.17.4	Functions	107
4.2.17.5	Dependencies	107
4.2.17.6	Prerequisites	107
4.2.18	Similarity Matching with Semantics	107
4.2.18.1	Purpose	107
4.2.18.2	Service Component	108
4.2.18.3	Interface Data	108
4.2.18.4	Functions	108
4.2.18.5	Dependencies	109
4.2.18.6	Prerequisites	109
4.2.19	Impact Assessment with Knowledge Graphs	109
4.2.19.1	Purpose	109
4.2.19.2	Service Component	109
4.2.19.3	Interface Data	110
4.2.19.4	Functions	110
4.2.19.5	Dependencies	110
4.2.19.6	Prerequisites	110
4.2.20	Semantic Search with vector embeddings	110
4.2.20.1	Purpose	110
4.2.20.2	Service Component	111
4.2.20.3	Interface and Data	111
4.2.20.4	Functions	111
4.2.20.5	Dependencies	111
4.2.20.6	Prerequisites	111
4.2.21	Automated Test Building Support with Hybrid Filtering	112
4.2.21.1	Purpose	112
4.2.21.2	Service Component	112
4.2.21.3	Interface and Data	112
4.2.21.4	Functions	112
4.2.21.5	Dependencies	113
4.2.21.6	Prerequisites	113
5	FUNCTIONAL CAPABILITIES	114

6	Security.....	118
6.1	DAI-DSS Knowledge Base.....	118
6.2	DAI-DSS Orchestrator	121
6.3	DAI-DSS AI -Enrichment infrastructure.....	121
6.4	Outlook.....	121
7	Architecture Testing and reporting.....	122
7.1	DAI-DSS Knowledge Base.....	122
7.2	DAI-DSS User Interface	124
7.3	DAI-DSS Orchestrator	124
7.4	DAI-DSS Configurator.....	125
7.5	DAI-DSS AI Enrichment.....	126
7.6	DAI-DSS Intelligent Sensor Box.....	126
7.7	Outlook.....	127
8	Summary and Conclusion.....	128
9	References	129
Annex A:	List of Tools	130
Annex B:	List Of HIGH-QUALITY Figures.....	131
	Annex B.1 FAIRWork High Level Architecture	131

LIST OF FIGURES

Figure 1 Overview of high-level architecture with key components (HQ figure in Annex B.1)	20
Figure 2 describes the model-based alignment and configuration of AI Services to address the use case specific need.	23
Figure 3 FLEX Automated Test Building Process	24
Figure 4 FLEX Worker Allocation Process	25
Figure 5 FLEX Machine Maintenance After Breakdown Process	27
Figure 6 Workload Balance Process	28
Figure 7 CRF Delay of Material Process	30
Figure 8 CRF Quality Issues Process	32
Figure 9 High level internal architecture of DAI-DSS User interfaces	35
Figure 10 Mock-up of customizable User Interface	36
Figure 11 Use Case Dashboard View	37
Figure 12 Use Case Alternative View	38
Figure 13 High level internal Architecture of DAI-DSS Configurator	40
Figure 14 Sequence Diagram UI Configuration	42
Figure 15 Sequence Diagram Orchestrator Configurator	42
Figure 16 Sequence Diagram Decision Assessment	43
Figure 17 High Level internal Architecture of DAI-DSS Orchestration	46
Figure 18 High level architecture of DAI-DSS Knowledge Base	51
Figure 19 The use of reference data in the PLM Module	51
Figure 20 ISO 10303 AP239 PLCS Data model	52
Figure 21 Interfaces between EDM TruePLM client to EDM database	53
Figure 22 Architecture of DAI-DSS AI Enrichment	58
Figure 23 Multi-Agent System: Internal Architecture	61
Figure 24 Current agent UML Class Diagram	63
Figure 25 Multi-Agent System development and validation	65
Figure 26 Building Blocks of the data catalogue of the data market	66
Figure 27 Architecture of DAI-DSS Intelligent Sensor Box	69
Figure 28 Sensors for human factors	69
Figure 29 describes the FAIRWork Resilience Monitor	71
Figure 30 Risk Stratification System of the FAIRWork Intelligent Sensor Box.	72
Figure 31 describes the developer dashboard on the “Physical Status”. The dashboard is depicting instantaneous vital parameters (left), live biosignal information and scores (upper right graph)	73
Figure 32 describes the developer dashboard on the “Cognitive Status”. The dashboard is depicting instantaneous vital parameters (left), live biosignal information and scores (upper right graph)	73
Figure 33 UML Use Case Diagram for Worker assignment to production line by using mathematical optimization/heuristics.	79
Figure 34 Service Component for Worker assignment to production line by using mathematical optimization/heuristics	79
Figure 35: UML Use Case Diagram of the Resource Allocation Service with Linear Assignment Solver	80
Figure 36: Service Component for interfaces between the Resource Allocation Service with CP-Solvers and DAI-DSS Architecture	81
Figure 37 UML Use Case Diagram of Pattern Based Resource Allocation Service	82
Figure 38 Service Component for interfaces between Pattern Based Resource Allocation Service and DAI-DSS Architecture	82
Figure 39 UML-use-case diagram of Operator Stress Estimation Service	83

Figure 40 Service Component for Operator Stress Estimation Service.....	83
Figure 41 UML Use Case Diagram for Agent-based Resource Allocation	84
Figure 42 Service Component for production process simulation with agents	85
Figure 43 UML Use Case Diagram for the Model-enabled Rule Service	86
Figure 44 Service Component Model-enabled Rule Service.....	87
Figure 45 UML Use Case Diagram of the Rule Based Resource Allocation Service	89
Figure 46 Service Component for interfaces between the Rule Based Resource Allocation Service and DAI-DSS Architecture	90
Figure 47 UML Use Case Diagram for Fuzzy Rule-based Worker Efficiency Estimation Service	91
Figure 48 Service Component for Fuzzy Rule based Worker Efficiency Estimation	91
Figure 49 UML Use Case Diagram for the Resource Mapping with Decision Trees Service	92
Figure 50 Service Component Resource Mapping with Decision Trees Service.....	93
Figure 51 UML Use Case Diagram for Schedule Optimizer with Linear Programming	94
Figure 52 Service Component for Schedule Optimizer with Linear Programming.....	95
Figure 53 UML Use Case Diagram for the Similarity Matching with Knowledge Graphs service	96
Figure 54 Service Component for Similarity Matching with Knowledge Graphs service	96
Figure 55 UML Use Case Diagram of the Reinforcement Learning Based Resource Allocation Service	98
Figure 56 Service Component for interfaces between the Reinforcement Learning Based Resource Allocation Service and DAI-DSS Architecture	98
Figure 57 UML Use Case Diagram of the Annotation Support Service	99
Figure 58 Service Component for interfaces between the Annotation Support Service and DAI-DSS Architecture	100
Figure 59 UML Use Case Diagram for Agent-based Production Process Simulation	101
Figure 60 Service Component for production process simulation with agents	101
Figure 61 UML Use Case Diagram of Operator Persona Development Service	102
Figure 62 Service Component for Operator Persona Development Service	103
Figure 63 UML Use Case Diagram for the community detection in knowledge graphs service	104
Figure 64 Service Component for community detection with knowledge graphs service	104
Figure 65 UML Use Case Diagram for the production process simulation service.....	106
Figure 66 Service component for production process simulation service.....	106
Figure 67 UML Use case diagram for the similarity matching with semantics service	108
Figure 68 Service component for similarity matching with semantics service	108
Figure 69 UML Use Case diagram for the impact assessment with knowledge graphs service.....	109
Figure 70 Service component for impact assessment with knowledge graphs service	110
Figure 71 UML Use Case Diagram Semantic Search with vector embeddings.....	111
Figure 72 UML Use Case Diagram for Support of Hybrid Filtering.....	112
Figure 73 Service Component for Automated Test Building Support with Hybrid Filtering.....	112
Figure 74 Monitoring the website using umami.	119
Figure 75 2FA Authentication for Knowledge Base.....	120
Figure 76 Role based access control in the Knowledge Base.....	120
Figure 77 REST Method for token generation.....	120

LIST OF TABLES

Table 1 Use Case Scenarios.....	22
Table 2 DAI-DSS User Interfaces API Overview	38
Table 3 DAI-DSS Configurator Interfaces Overview.....	44
Table 4 AI Services List.....	77
Table 5 Testing information related to Knowledge Base	122
Table 6 Testing information related to User Interface.....	124
Table 7 Testing information related to Orchestrator.	124
Table 8 Testing information related to Configurator.	125
Table 9 Testing information related to AI Enrichment.....	126
Table 10 Testing information related to Intelligent Sensor Box	126

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 Lifecycle 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

1.1 Purpose of the Document

The purpose of the document, "D4.1.1 – FAIRWork Architecture and Initial Documentation and Test Report," is to provide a detailed update on the progression and current state of the Democratic AI-based Decision Support System (DAI-DSS) Architecture as part of the FAIRWork project. It outlines the changes made since the previous iteration (D4.1), highlighting improvements in components architecture, functionalities, and capabilities as well as AI services. The document aims to serve as a detailed guide for understanding the use case scenarios and mapping the scenarios with potential AI services that can be utilized to provide the required decision support. Additionally, it details the architecture and interfaces of the components required for the Decision Support System, alongside their functional capabilities. The document not only reviews the AI services that have been implemented but also those that are in the pipeline for future integration. Moreover, it delves into the security measures adopted and the methodologies employed for testing the functionality of architectural components.

1.2 Document Structure

This deliverable is structured as follows:

Chapter 1: this chapter provides the information related to the purpose and structure of the document.

Chapter 2: contains a high-level architecture overview and description. It shows how the different components within the architecture are arranged, without going into the details of each individual component. It further maps the high-level architecture with use cases and corresponding key performance indicators.

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 and implementation 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 and outlook.

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 following 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, a loosely coupled approach of independent services that offer required functionalities will be utilized. In addition, a Micro-Frontend Framework will be used 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 as well as by the provision of various AI services. The Figure 1 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.

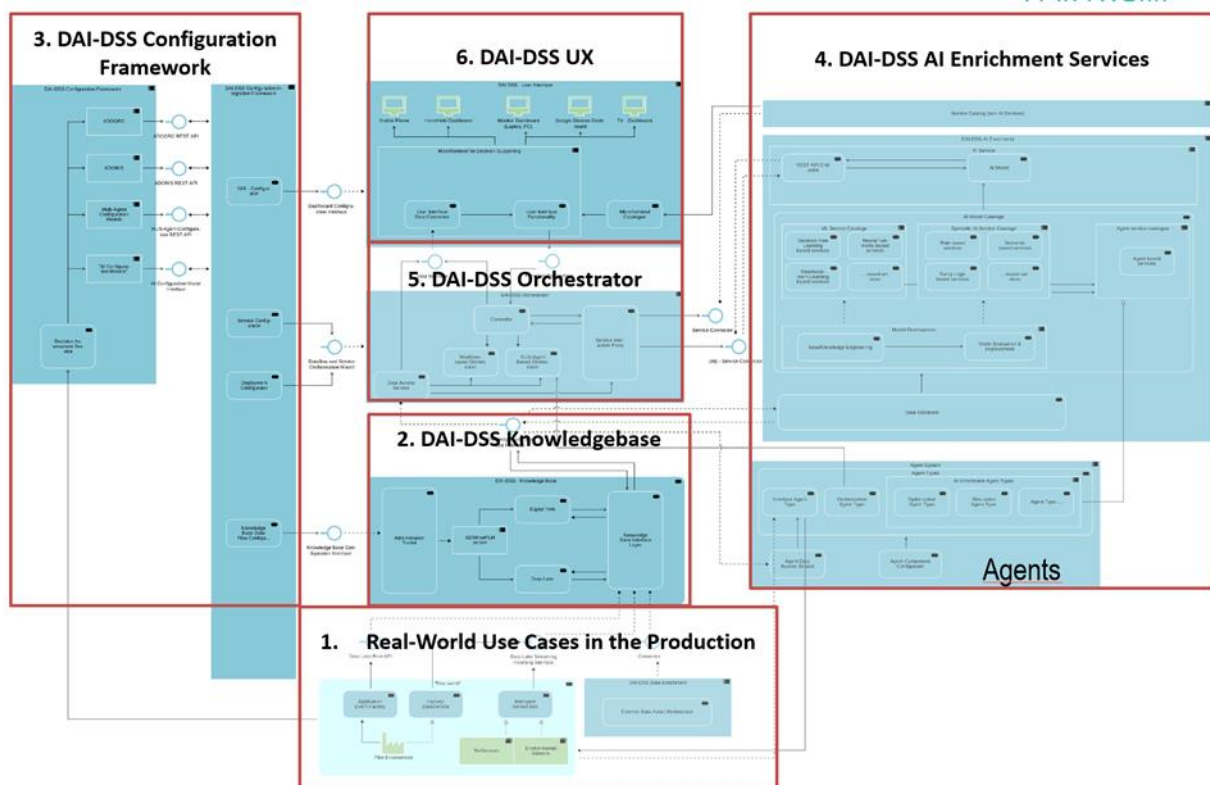


Figure 1 Overview of high-level architecture with key components (HQ figure in Annex B.1)

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, which is accessible in the form of an AI model catalogue. How these 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 abstracting on 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

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

USE CASE SCENARIOS	FLEX		CRF	
	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.

We identify and map process requirements with the appropriate AI services as shown in Figure 2 for providing decision support.

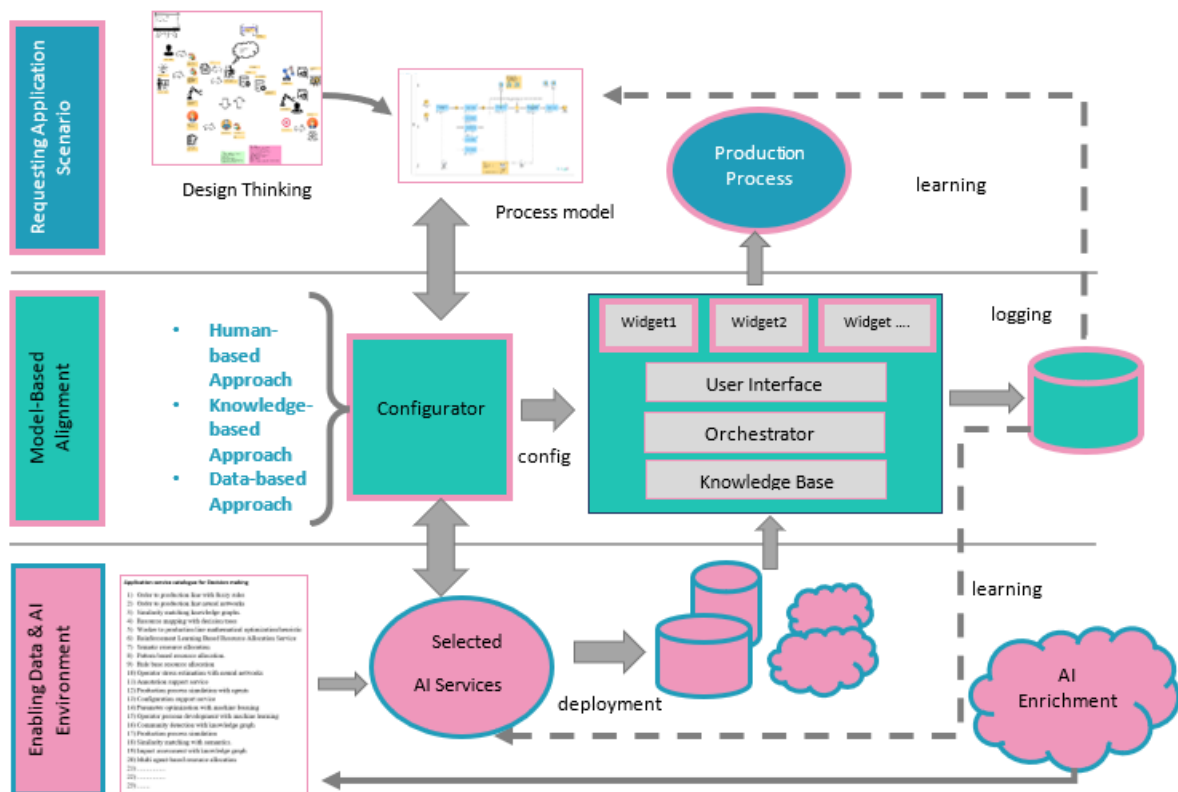


Figure 2 describes the model-based alignment and configuration of AI Services to address the use case specific need.

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 of 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 3 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.

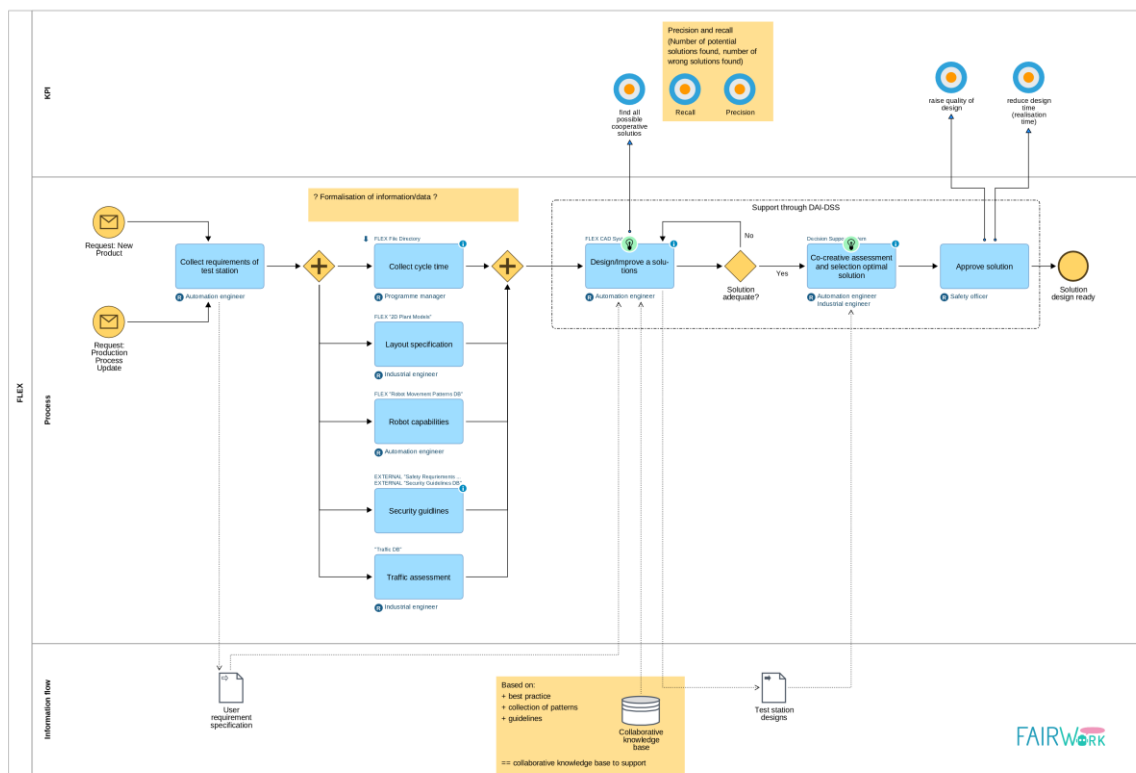


Figure 3 FLEX Automated Test Building Process

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 from 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”.

Measuring the successful implementation by usage of a Key Performance Indicator (KPI) is not foreseen at the moment for this use case as it will improve the overall development process, make it leaner and add transparency. The present process is still new in the flex enterprise and in optimization. Therefore, it is not rated as sufficiently mature to use it as a baseline for challenging or as a base for a KPI. The improved DAI-DSS process will be used to measure continuous improvement on the development of future systems. Main parameter to measure the performance will be the processing time of an automation request.

2.2.2 Flex Worker Allocation Process

The second scenario concerns with worker allocation are shown in Figure 4. 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.

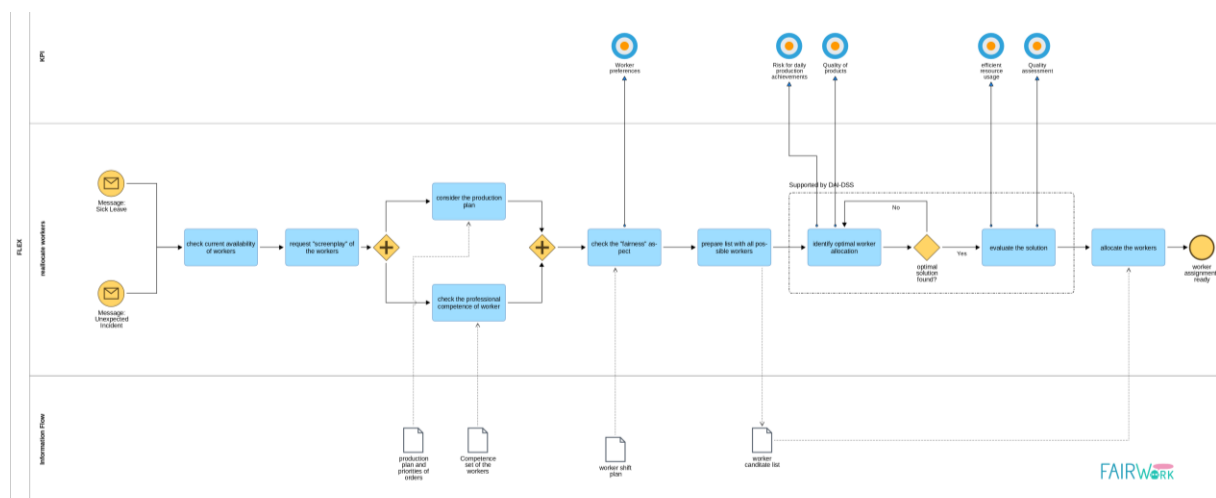


Figure 4 FLEX Worker Allocation Process

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
 Copyright © 2023 Jotne and other members of the FAIRWork Consortium
 www.fairwork-project.eu

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”.

Measuring the successful implementation of this use-case will be performed by usage of a Key Performance Indicator (KPI) measuring the reaction time or time to deliver a solution for a given problem compared to a decision made without support the DAI-DSS. Currently, the manager must acquire information about available and applicable substitutes over various interfaces. By feeding the correct information to the DAI-DSS system, potential options should be presented to the operator within seconds. Especially at initial tests, the quality of the systems feedback shall be evaluated additionally, whether the information is useable for the operator or not.

- The desired improvement is to eliminate (100%) the gap when missing personnel from a certain production line, due to various reasons, with the best solution available on other production line (ex.: When maintenance in place, no production plan, reduce activities, personnel can be reassigned on the line with needed personnel).
- Improve decision making for production managers when they miss personnel from a shift.

2.2.3 FLEX Machine Maintenance After Breakdown Process

The machine maintenance scenario is shown in Figure 5. 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 fulfil 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.

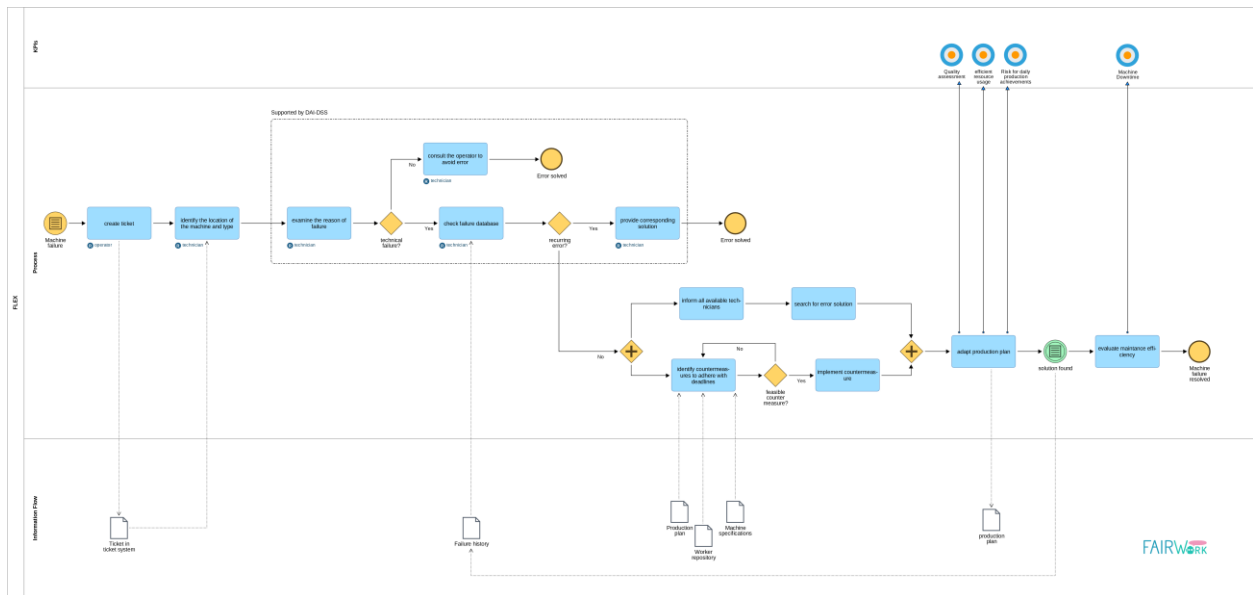


Figure 5 FLEX Machine Maintenance After Breakdown Process

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 the best production rescheduling plan by considering quality assessment, efficient resource usage, and 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”.

In the final evaluations, it is planned to measure the effectiveness of the improved DAI-DSS process by acquisition of the maintenance interventions divided by the total errors requiring maintenance intervention per week. Currently, this value is 100%, as we have the need of maintenance support at every error, which will be reduced if the interventions by the maintenance team can be reduced. By support of operators through the DAI-DSS system, in the best case no notification for the maintenance team is necessary in the future. This KPI is mainly influenced by the usability of the DAI-DSS interface and the completeness and quality of information in the database.

An optional KPI would describe the improvement of component maintenance intervals focused on single events with a high probability of occurrence. This KPI is still in definition and might occur as supplement to the use-case.

- Reduce the intervention time by 30% using the breakdown data base (history of previous incidents)

- Reduce the learning curve of new technical personnel by 30% (they can learn by doing, based on the history breakdown of the equipment)

2.2.4 CRF Workload Balance Process

The workload balance scenario is shown in Figure 6 and 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.

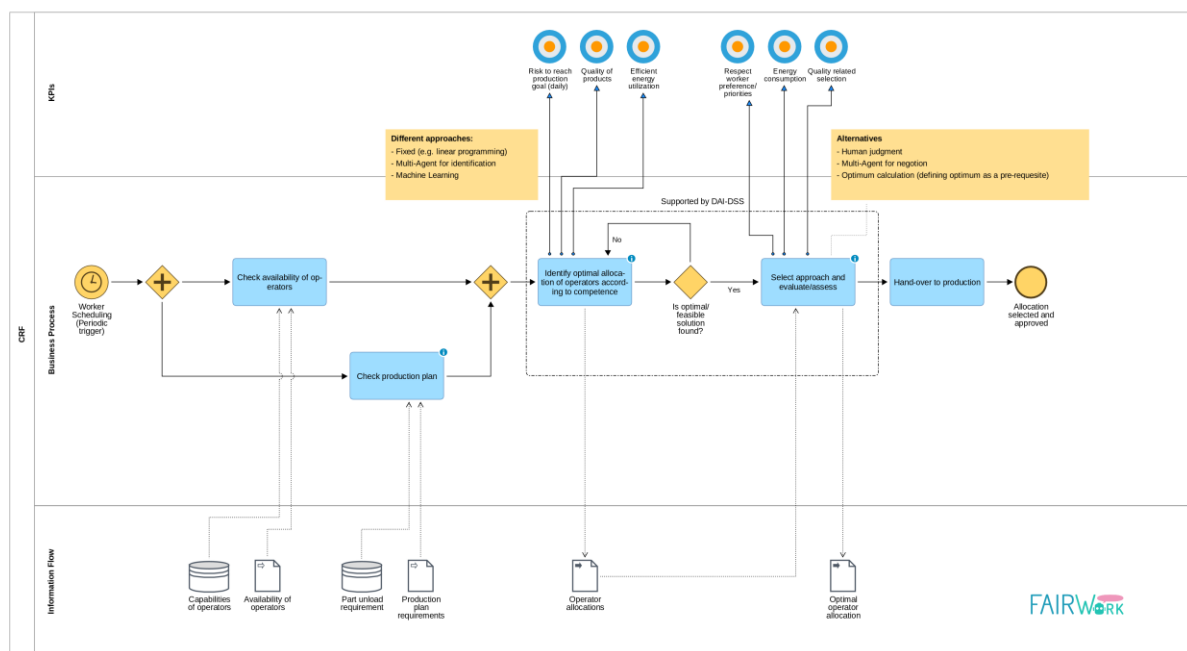


Figure 6 Workload Balance Process

In 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 from 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”.

In the Deliverable 2.1, section 4.4.2 (KPI used in the CRF/Stellantis Scenario), official KPIs identified for the press shop by Stellantis were listed, updated on 29 June 2022. They consider six macro aspects: Safety/People, Quality, Delivery, Cost, Productivity, Environment/Energy.

These indicators are general guidelines, which concern the entire molding process. Every intervention must aim to improve some of these indicators or at least not worsen them. In this section, we will define the KPIs that are applicable using a democratic decision-making model in the Workload Balance Process scenario.

The analysis must be conducted consistently with the "impacts" envisaged in the work program, applicable to the case study. From this point of view, the most significant expected impacts are:

- Improve optimization: raise efficiency through faster and more precise decision-making (raise operating efficiency by 10%).
- Improve optimization: improve co-creation capabilities and compliance with 100% coverage and reduction of non-compliant decision by 95%.
- Strengthening the perception of trust and fairness: improve workforce wellbeing, reduce stress level and raise competence (higher employee satisfaction, sense of belonging raised, physical and mental health) reduction of safety incidents by 25%.

Regarding the objectives set by FAIRWork, the expectations applicable for this scenario, are:

- Improve flexibility: at least 5 alternatives for decisions.
- Raising the competence of decision support: decrease the time to enact decisions by 70% (under 1 minute) and decrease the numbers of decision revisions (down to 1).
- Strengthening the perception of trust and fairness: decrease the time for understanding, checking, and approving the output of decision-making by 90%.
- Reduce stress situation: increase number of alternative evaluations by 1000% and reduce decision revision to 1.
- Improve reliability and compliance: reduce number of non-compliant decisions by 95%.

Given these premises, the 3 KPIs that will be applied to the case study are listed below.

1. **DECISION TIME [sec]:** Regarding the first "impact" listed, the way to raise the efficiency, through faster and more precise decision-making, is to measure the time needed to take a decision. Time will be measured from the instant in which all the data necessary to allocate workers on the lines will be available. The input data will therefore be the presence of people in the plant for the work shift, the database of the operators' features, the desired production (components and quantities). This KPI applies not only for the initial production planning, at the beginning of the shift, but also in the case of replanning due to unforeseen events during production. The target value is: **UNDER 1 MINUTE**
2. **ALTERNATIVE PLANS [n]:** the model must be able to provide multiple alternative solutions, clearly assessable by the line manager. This does not mean that a solution is always possible, but if the personnel present in the plant is not suitable to meet the planned production, this must be clearly evident. In the best-case scenario, where scheduled production is possible, the model must provide **5 ALTERNATIVES**.

3. **UNPLANNED ABSENTEEISM [%]**: Regarding the third "impact" listed (improve workforce wellbeing), according with the official KPIs identified by Stellantis, the way to measure worker well-being is linked to the level of unplanned absenteeism. The formula to calculate it is reported into the Deliverable 2.1, section 4.4.2. and is defined as the total person-hours (paid, unpaid) where the hourly worker was not at work as expected. The acceptable target in this case is a **NO WORSENING**, in the best case a **2.5% REDUCTION**.

2.2.5 CRF Delay of Material Process

The scenario delay of material delivery is illustrated in Figure 7. 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.

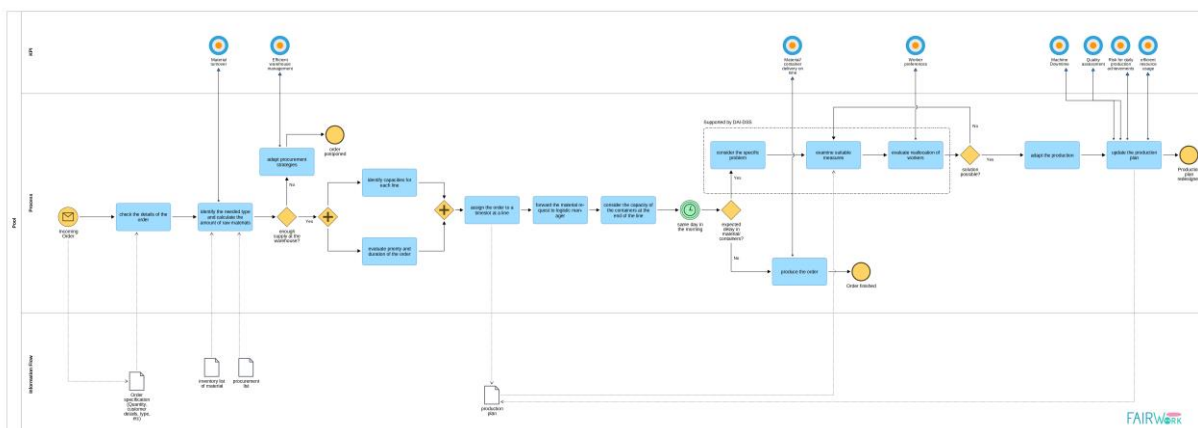


Figure 7 CRF Delay of Material Process

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”.

The second scenario relates to “delay of materials”, where the word “materials” is understood in its broadest sense. It concerns all the material resources necessary to produce the components: raw materials, molds, containers, etc. The delay of materials generates inefficiency from multiple points of view. For example, the line can remain in standby waiting for the materials. The standby of a molding line is not zero energy consuming, so in this situation energy is wasted. The lines must be used 100% during working shifts. Sometimes delays also cause the need to reschedule work, working outside the normal and ideal schedule (overtime or working on Saturdays). Obviously, even in this case additional resources (energy) are consumed.

In the Stellantis document cited in the previous paragraph (official KPIs identified by Stellantis for the press shop), attention is paid to Energy Consumption. It is therefore believed that, for the scenario examined in this section, a KPI relating to energy efficiency can be considered.

Also in this case, with reference to the “impacts” expected in the work program, in addition to those previously mentioned, we highlight:

- Raise energy efficiency by 20-30%

Furthermore, referring to the objectives of FAIRWork, we consider:

- Strengthening the effect of energy efficiency: Reduce energy consumption by 10-20 %

Regarding this scenario, in addition to the first 2 KPIs previously defined, also valid in this case (**DECISION TIME** and **ALTERNATIVE PLANS**), we add:

- **ENERGY CONSUMPTION [kwh/Ton]:** according with the official KPIs identified by Stellantis, the formula used to measure the energy consumption is reported in the Deliverable 2.1, section 4.4.2. It is a measure of the total energy consumed by a facility to produce components. The values that will be taken into consideration will be the ones of the molding lines in relation to the quantities produced. The desirable target in this case is a **5% REDUCTION**.

2.2.6 CRF Quality Issue Process

The scenario “quality issues” is a recurring process of quality checks and is illustrated in Figure 8. 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 is not possible or feasible the products must be discarded.

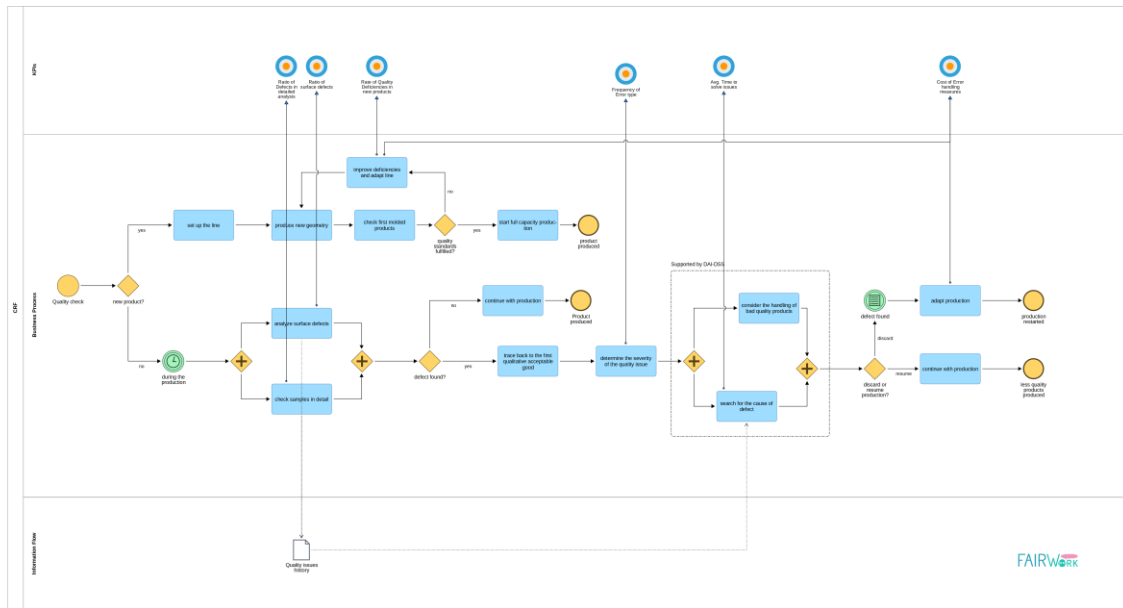


Figure 8 CRF Quality Issues Process

In this scenario, we propose an AI service which looks 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 the 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”.

In the "impacts" expected in the work programme, as well as in the objectives of the project, there is no reference to an improvement in the quality of the product. In fact, the quality of the product depends on other factors that go beyond the objectives of the democratic decision-making model.

As seen in the description of the KPIs identified by Stellantis for the press shop, an aspect relating to quality is the quantity of components that must be reworked. The rework rate measures how much rework takes place in a process. It is therefore an indicator of internal operational efficiency. It is important to quickly identify the quality drift of the process, to avoid producing non-compliant components, the rework of which causes inefficiency and energy consumption.

Regarding this scenario, in addition to the first 2 KPIs previously defined, also valid in this case (**DECISION TIME** and **ALTERNATIVE PLANS**), we add a specific one:

- **REWORK RATE [%]**: it is related to the number of repaired appearance parts divided by the total number of produced appearance parts. Also, in this case the formula is explicated into the Deliverable 2.1, section 4.4.2. The desirable target in this case is a **5% REDUCTION**.

3 ARCHITECTURE COMPONENTS

Each of the next sub-sections of this chapter describes the different DAI-DSS components introduced in chapter 2.1. Each sub-section is structured using the following template:

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.

8. Hardware and software requirements (optional)

- Description of required software and hardware infrastructure.

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 for retrieving 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 configured before there deployment. The configuration files are created using the DAI-DSS Configuration Integration Framework and are provided through the UI Configuration Interface. Figure 9 gives an overview of the internal high-level architecture, and each component is described in more detail below.

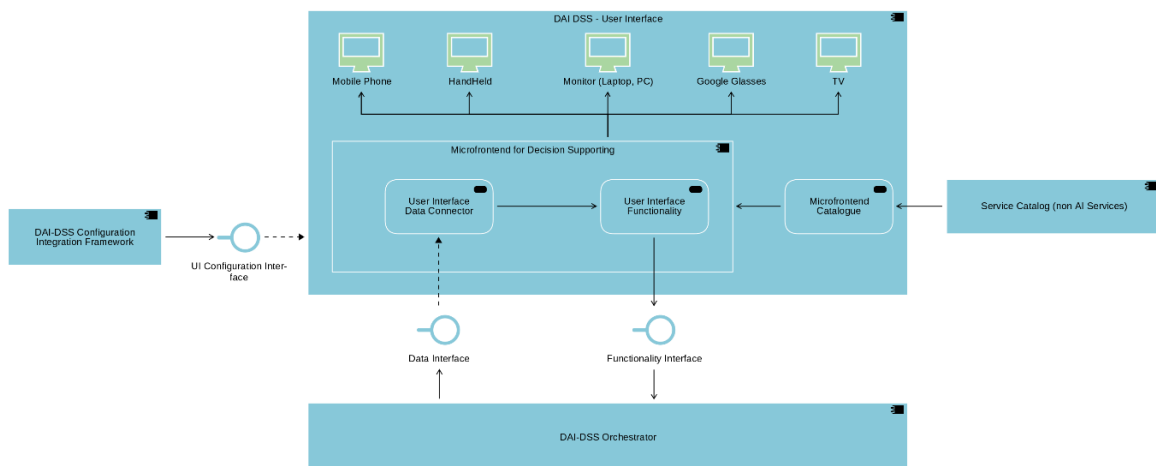


Figure 9 High level internal architecture of DAI-DSS User interfaces

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 10 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 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 configured 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 a JavaScript file (for widgets targeting Web Browsers and MS Teams pages) and the widget configuration in form of a 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 10. The micro frontend consists of two subcomponents, which are described in more detail below.

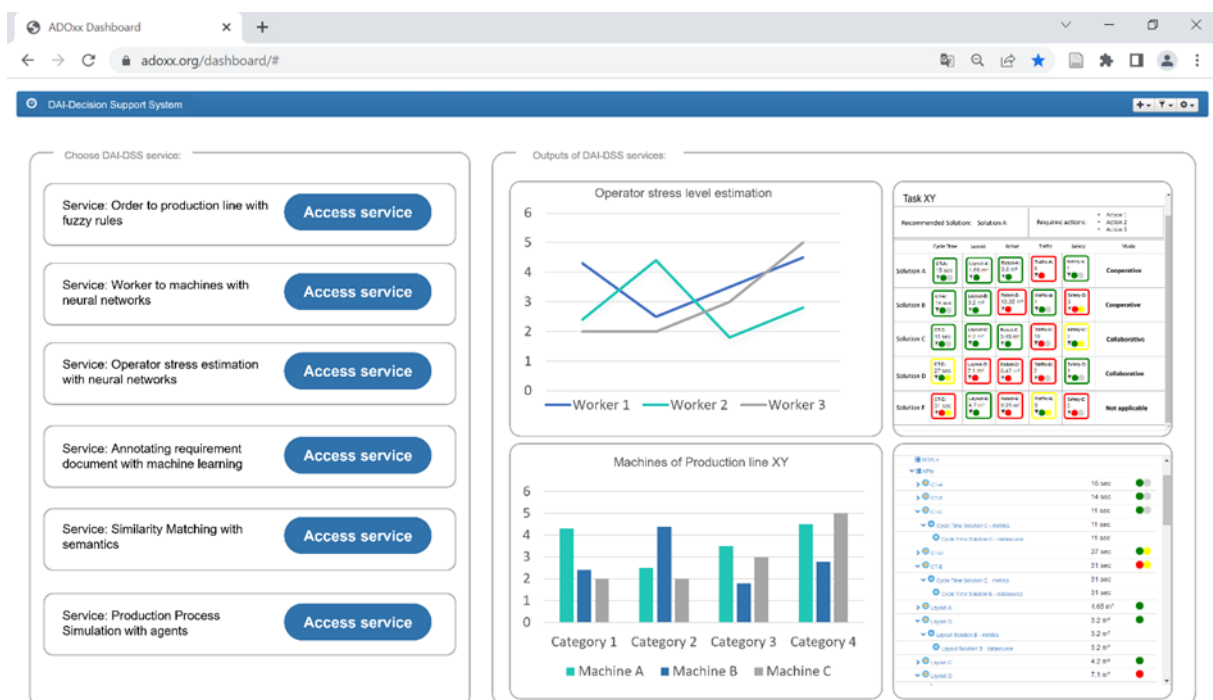


Figure 10 Mock-up of customizable User Interface

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 configured 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 configured. 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 11).

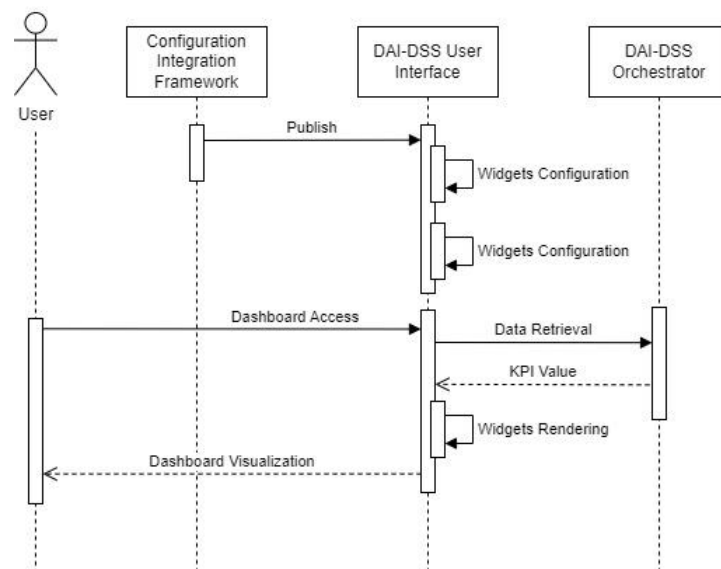


Figure 11 Use Case Dashboard View

- **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 trigger 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 12).

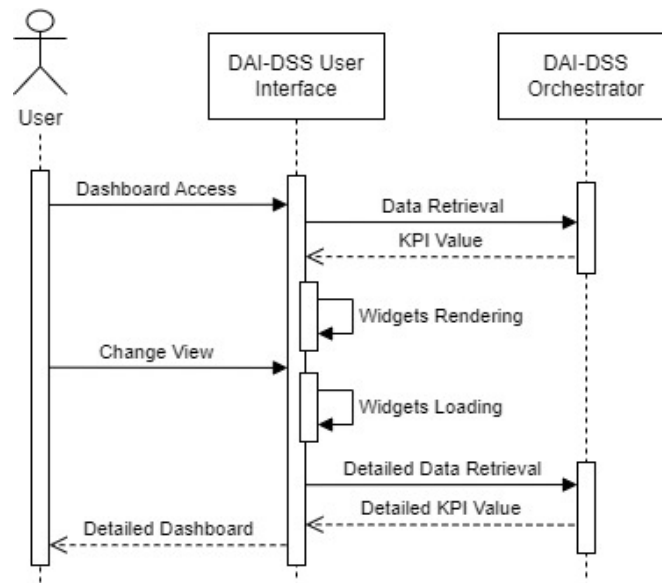


Figure 12 Use Case Alternative View

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 or results coming from different services.

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.

3.1.7 Hardware and Software requirements

For creating widgets from scratch the use of React, which is a JavaScript library, is recommended and was also used for the initial prototype (see Deliverable D4.2). Additionally, a good understanding of foundational technologies for web development like HTML, CSS, and JavaScript is preferable. For setting up a React development environment, Node.js need to be installed. The finalized widgets can then be uploaded to the Micro-Frontend catalogue in form of zip-files following a specific format. To configure pre-defined widgets the DAI-DSS Configurator is used. The corresponding hardware and software requirements are described in section 3.2.7.

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 ADOxx¹, 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 configuring 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 OLIVE² 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

¹ <https://www.adoxx.org/>

² <https://www.adoxx.org/live/olive>

Configurator, and the Knowledge Base Data Flow Configurator. The following Figure 13 gives an overview of the internal high-level architecture, and each subcomponent is described in more detail below.

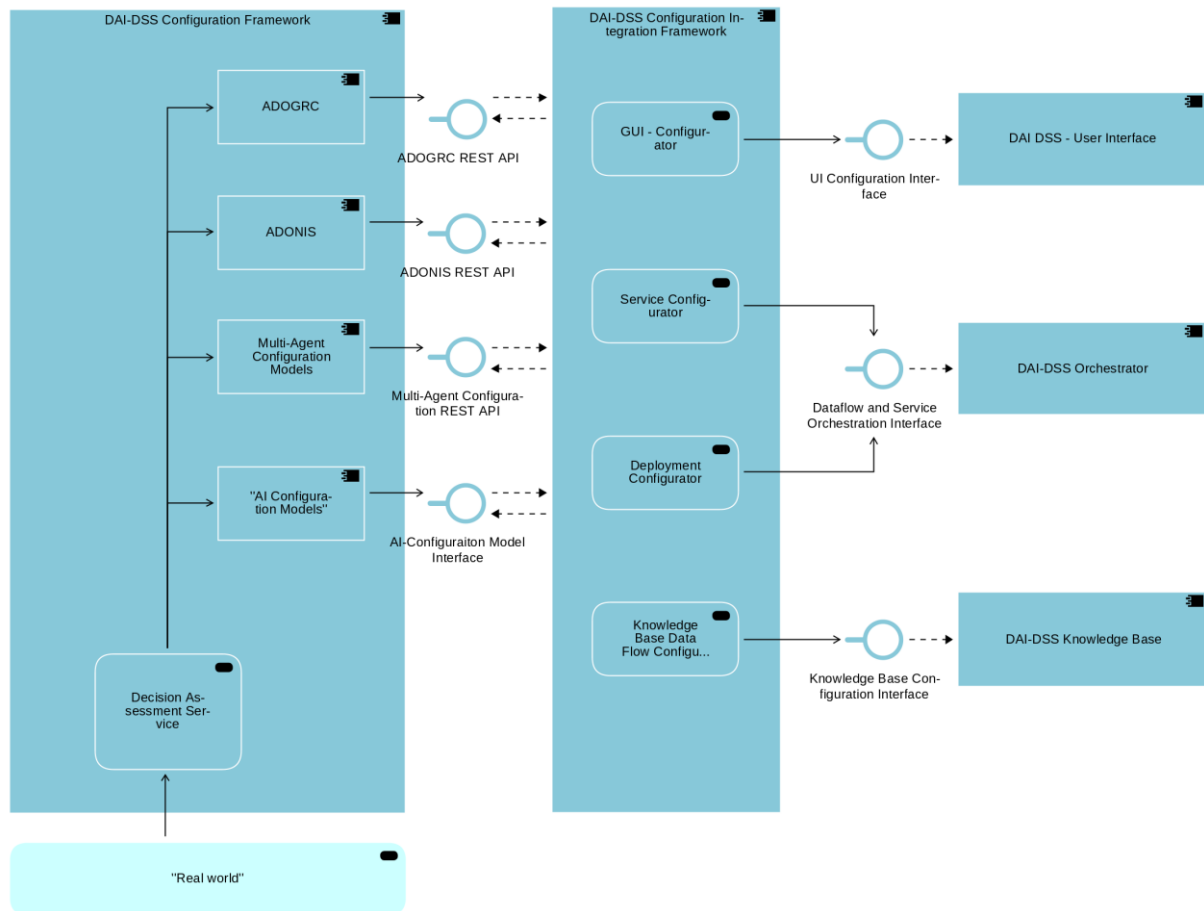


Figure 13 High level internal Architecture of DAI-DSS Configurator

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 colours 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 configured 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 configured 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 the 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 configured. 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 14 provides 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 data from the knowledge base. This type of flexibility enables the evolution of a use-case specific decision support system.

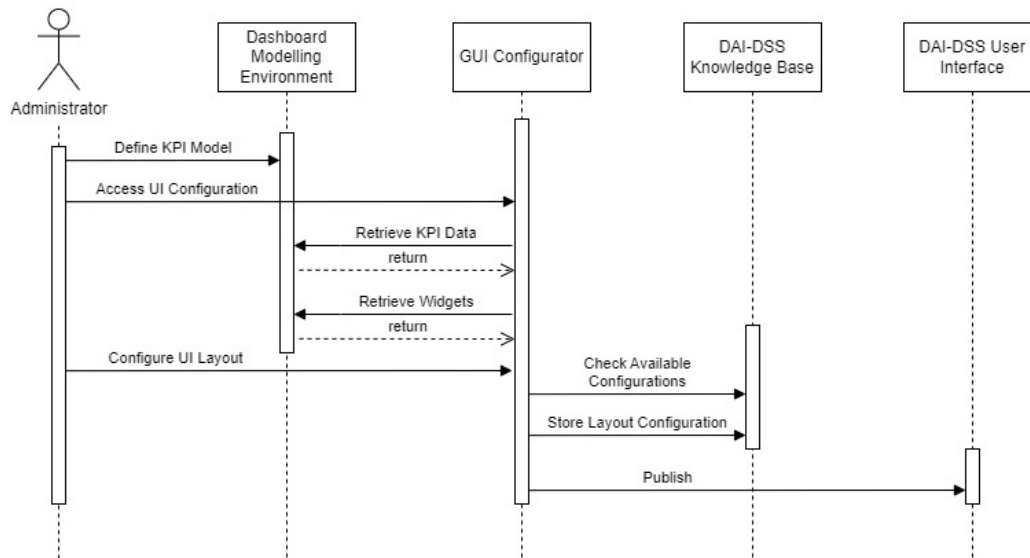


Figure 14 Sequence Diagram UI Configuration

- **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 configured using the service configurator. If services are dependent on each other, the whole workflow needs to be configured, so that service inputs and outputs fit together. The configuration files are stored in the DAI-DSS Knowledge Base. If all services are configured, the configuration files are forwarded to the orchestrator component, which oversees the execution (see Figure 15). 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 not only combines the User Interface, the AI Enrichment services and the DAI-DSS Knowledge Base and hence is a central element of the DAI-DSS architecture, but also enables to build complex solutions by combining several AI services.

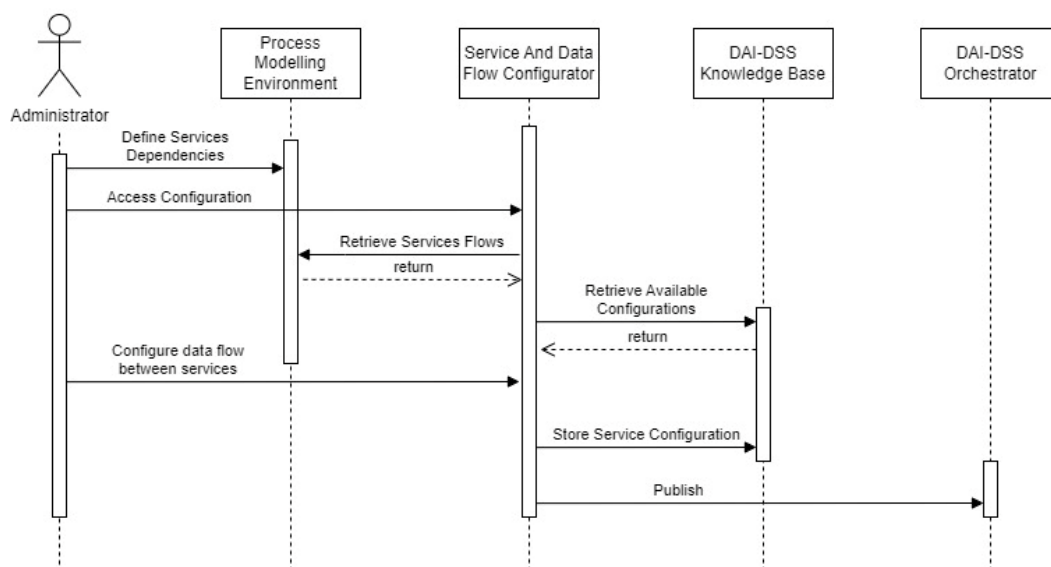


Figure 15 Sequence Diagram Orchestrator Configurator

- **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 16.

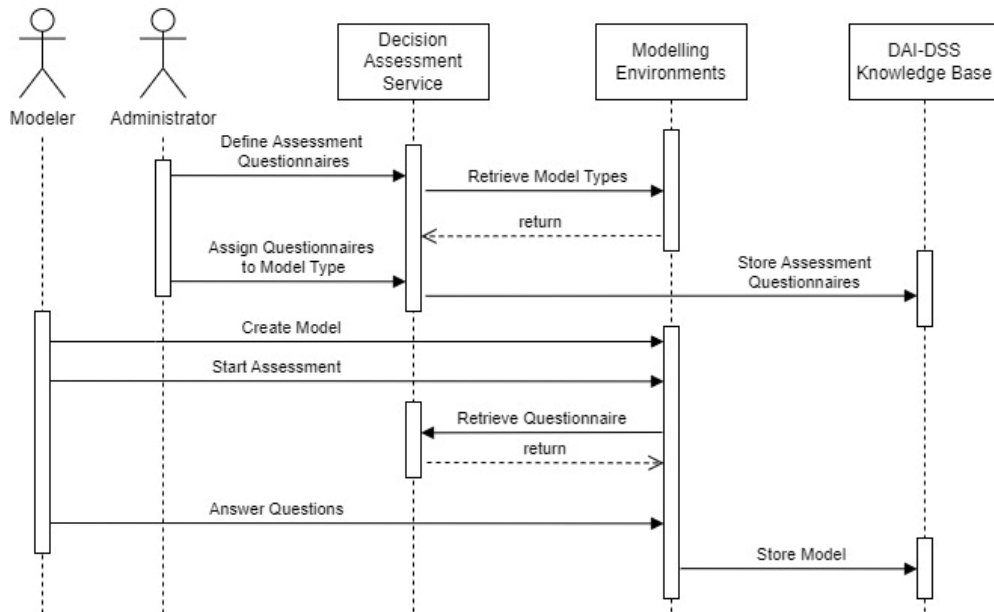


Figure 16 Sequence Diagram Decision Assessment

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 has been expanded by developing Microservices that can connect to AI services and Multi-Agents Systems based on configuration files generated from decision models.

3.2.7 Hardware and Software requirements

The DAI-DSS Configuration Framework is realized using different modelling environments. For the initial prototype the Bee-up³ modelling tool, which is built on ADOxx, was used, and adapted to configure services based on models in an experimental setting. The meta-modelling platform ADOxx and the Bee-up Tool can be installed locally on different operating systems (Windows 10, Windows 11, Ubuntu and macOS). For creating decision models an instance of the process modelling software ADONIS⁴ running in the cloud was set up. The ADONIS web server can be accessed using popular desktop browsers like Microsoft Edge, Mozilla Firefox, Google Chrome, and Safari 9 or higher. Additional hardware and software requirements can be found under the following link in the footnotes.

The DAI-DSS Integration Framework is built on the microservice framework OLIVE. The OLIVE framework operates within the Amazon Web Services (AWS) cloud environment, leveraging AWS's serverless computing resources and EC2 (Elastic Compute Cloud) instances. This means that for anyone to operate and manage the OLIVE framework, an AWS account is a requirement. The necessary computing hardware and resources are set up in the AWS cloud using deployment scripts, which automate the process of provisioning the required infrastructure.

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

³ <https://bee-up.omilab.org/activities/bee-up/download-details/>

⁴ <https://docs.boc-group.com/adonis/en/docs/14.0/hw-sw-regs/>

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 17 gives an overview of the high-level architecture of the orchestrator and the multi agent system. Each subcomponent is described in more detail below.

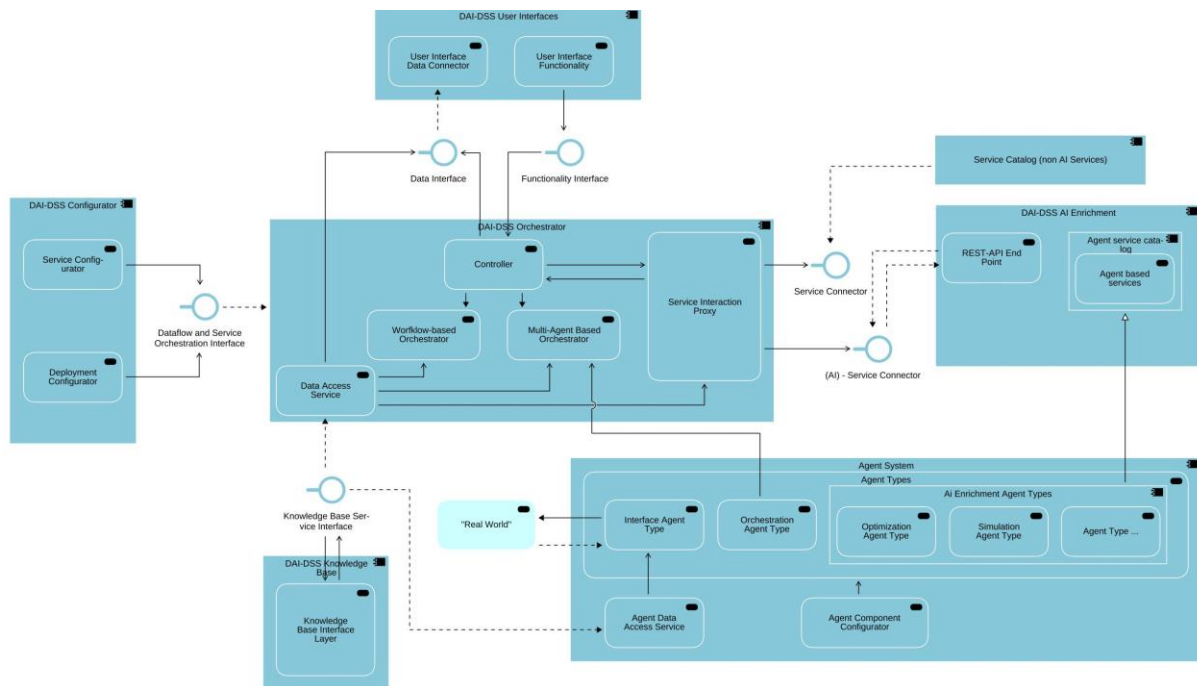


Figure 17 High Level internal Architecture of DAI-DSS Orchestration

3.3.2.1 Controller

The Controller allows to manage Microservices and control 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 types of orchestration the Controller relies 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 catalogues 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 Domain Specific Language model 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 Conductor⁵.

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.

⁵ <https://conductor-oss.org/>

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 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 setup, modules within the system can communicate either directly or indirectly. For instance, the AI Enrichment service can directly interact with the DAI-DSS Knowledge Base, or it can communicate indirectly via the DAI-DSS Orchestrator. In the latter scenario, a request from AI Enrichment passes through the DAI-DSS

Knowledge Base before reaching the Orchestrator. It's crucial that the system configuration allows for the responses to be properly decoded, whether communication is direct or follows a multi-agent or workflow-based path

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 Enterprise Integration Pattern (EIP) notation to model the service behaviour.

Integration of external workflow engine will be also performed, and the focus will be on the open-source project Conductor⁶. 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.

3.3.7 Hardware and Software requirements

The workflow-based orchestration is currently realized using different docker images:

Conductor Engine

Conductor is a microservices orchestration engine designed by Netflix. It helps manage workflows across various microservices by dictating the order of operations, handling retries, and tracking the state of each task within a workflow. Conductor makes it easier to create and manage complex processes that span multiple microservices in a distributed system.

Conductor UI

The UI component of Conductor provides a visual interface for users to interact with the Conductor engine. Through this UI, users can define workflows, tasks, and monitor the execution of these workflows in real-time. It offers a more user-friendly way to interact with Conductor compared to using API calls or scripting.

Elasticsearch

Elasticsearch is a powerful search and analytics engine. In this context, it's used to index and search through the logs and execution histories of the workflows managed by Conductor. This allows for quick searching, filtering, and analysis of the workflow data, which can be critical for debugging, monitoring, and optimizing performance.

Redis (NoSQL Database)

Redis is an in-memory data structure store, used as a database, cache, and message broker.

In the following the execution and usage of these four resources is explained in more detail:

A user defines a workflow using the Conductor UI, which then gets stored and managed by the Conductor engine. This definition includes the order of tasks, conditions for transitions, and any failure handling logic. As the workflow is executed, tasks might be distributed across different microservices. Conductor orchestrates these tasks, ensuring they are executed in the correct order, retried upon failure, and successfully completed. Elasticsearch indexes logs and execution data from the workflows. This indexed data is then available for searching and analysis, helping users understand the execution flow, identify bottlenecks, and troubleshoot issues. Redis plays a key role in

⁶ <https://conductor-oss.org/>

ensuring high availability and fast access to workflow and task data. It can cache frequently accessed data, store execution states, and facilitate messaging between different components or services involved in a workflow.

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 analysed 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.

3.4.2 Internal Architecture

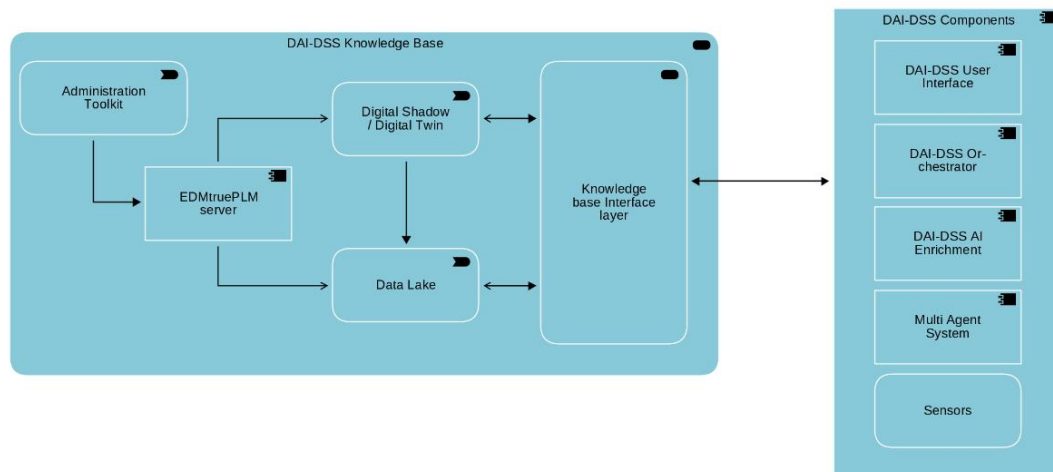


Figure 18 High level architecture of DAI-DSS Knowledge Base

The DAI-DSS Knowledge Base uses EDMtruePLM (TruePLM) to provide the data storage and interfaces required in the overall DAI-DSS Architecture refer Figure 18. 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 10303-11⁷, the EXPRESS language and ISO 10303-26⁸, the Standard Data Access Interface (SDAI). An important aspect of EDMtruePLM is the PLCS (Product Life Cycle Support) data model, ISO 10303-239⁹. 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 19.

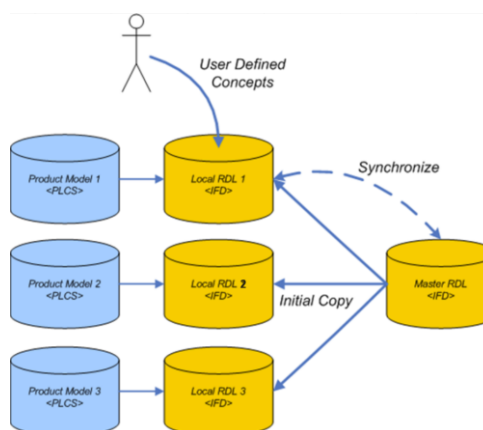


Figure 19 The use of reference data in the PLM Module

⁷ ISO 10303-11:2004. ISO. (2019, December 15), <https://www.iso.org/standard/38047.html>

⁸ ISO/TS 10303-26:2011. ISO. (2023, January 15), <https://www.iso.org/standard/50029.html>

⁹ ISO 10303-239:2005. ISO. (2012, November 13), <https://www.iso.org/standard/38310.html>

To represent the physical entities in TruePLM the PLCS provide the flexibility to use breakdown elements for creating and associate the corresponding properties. TruePLM manages the primary structure of nodes known as Breakdowns, which primarily interact with fundamental PLCS concepts at a low level. In essence, TruePLM operates with a single Breakdown decomposition represented as a sequence of versions, with a formal part serving as the primary component. Additionally, breakdown elements can be endowed with property values and document attachments, and users have the capability to establish organizational links with these breakdown elements Figure 20 ISO 10303 AP239 PLCS Data model.

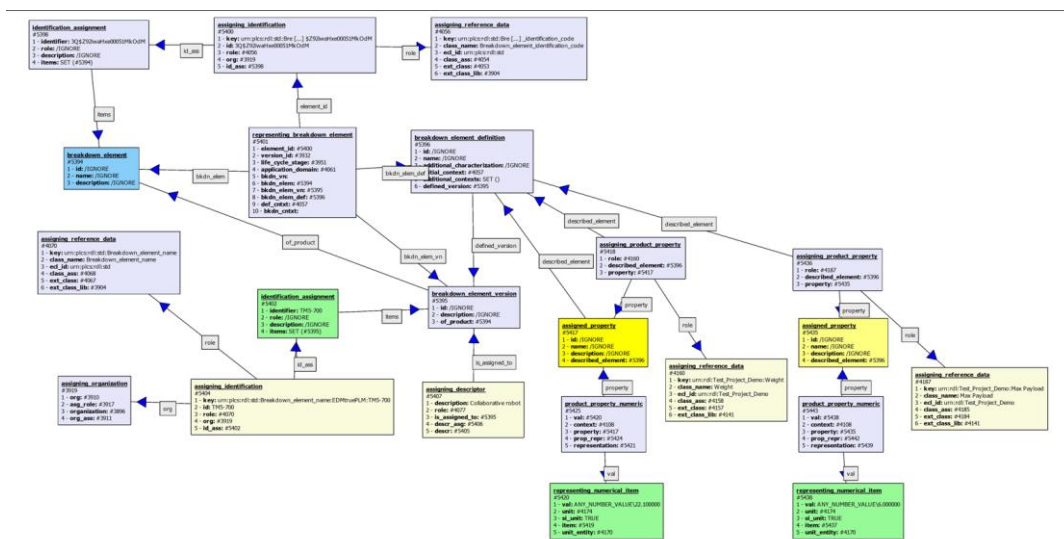


Figure 20 ISO 10303 AP239 PLCS Data model.

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.js¹⁰ framework, and a back-end, developed in Java with the Spring Boot¹¹ 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 page¹². The web application itself does not provide additional 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 Manual¹³.

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 applications 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.

¹⁰ <https://vuejs.org/>

¹¹ <https://spring.io/projects/spring-boot>

¹² <https://demo.jotne.com/EDMtruePLM/swagger.html>

¹³ <https://jotne.atlassian.net/wiki/spaces/EDM/pages/3471409196/User+Manual+-+3.3>

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.

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 is the Knowledge Base's role to handle these data, and it can be stated that the Knowledge Base is dependent on 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 page¹⁴. 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, and append data to aggregate properties.
- **Data**
 - The common functions for documents functionality
 - i.e., Search, download, and upload documents.
- **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.

¹⁴ <https://demo.jotne.com/EDMtruePLM/swagger-ui.html>

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.

3.4.7 Hardware and Software requirements

To successfully install EDMtruePLM, the target system must meet specific prerequisites and configurations:

- Java Requirements: The system should have the Java SE Development Kit 11 or higher (64-bit version) installed.
- .NET Framework: It is expected that the system already has .NET Framework version 4.5 or above.
- Operating System Compatibility: Installation is supported on Windows Server operating systems, with expanded compatibility to includes Linux systems.
- Storage Requirements: A minimum of 1 Terabyte of storage space is required on the hard drive.
- Additional Software Components:
 - Visual C++ Redistributable for Visual Studio 2017
 - Visual C++ Redistributable for Visual Studio 2013
 - IKVM.NET components

The EDMtruePLM Server is composed of several integral modules, which include:

1. EXPRESS Data Manager™: This module serves as the core for data management.
2. EDM6ServerConsole: Provides a console for server management and operations.
3. Apache Tomcat web server: A vital component for hosting the web application.
4. EDMtruePLM web application: The interface through which users interact with the system.
5. EDMtruePLM database: Stores all the data and information managed by EDMtruePLM.

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 the DAI-DSS Orchestrator. 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 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.

The AI Model Catalogue holds different AI services that the user can choose from. The services are categorized into three different types: the Machine Learning (ML) Service Catalogue, the Symbolic AI Service Catalogue, and the Agent Service Catalogue. While the ML Services use data-based ML algorithms to offer their services the symbolic AI services are based on rules and semantics that are described as symbolic AI. The services of both, the ML services as well as the symbolic AI services, are based on models that are developed in the Model Development. In the Model Development, 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. Furthermore, the validation of the model is supported. The services in the Agent Service Catalogue on the other hand need a connection the Agent System to function.

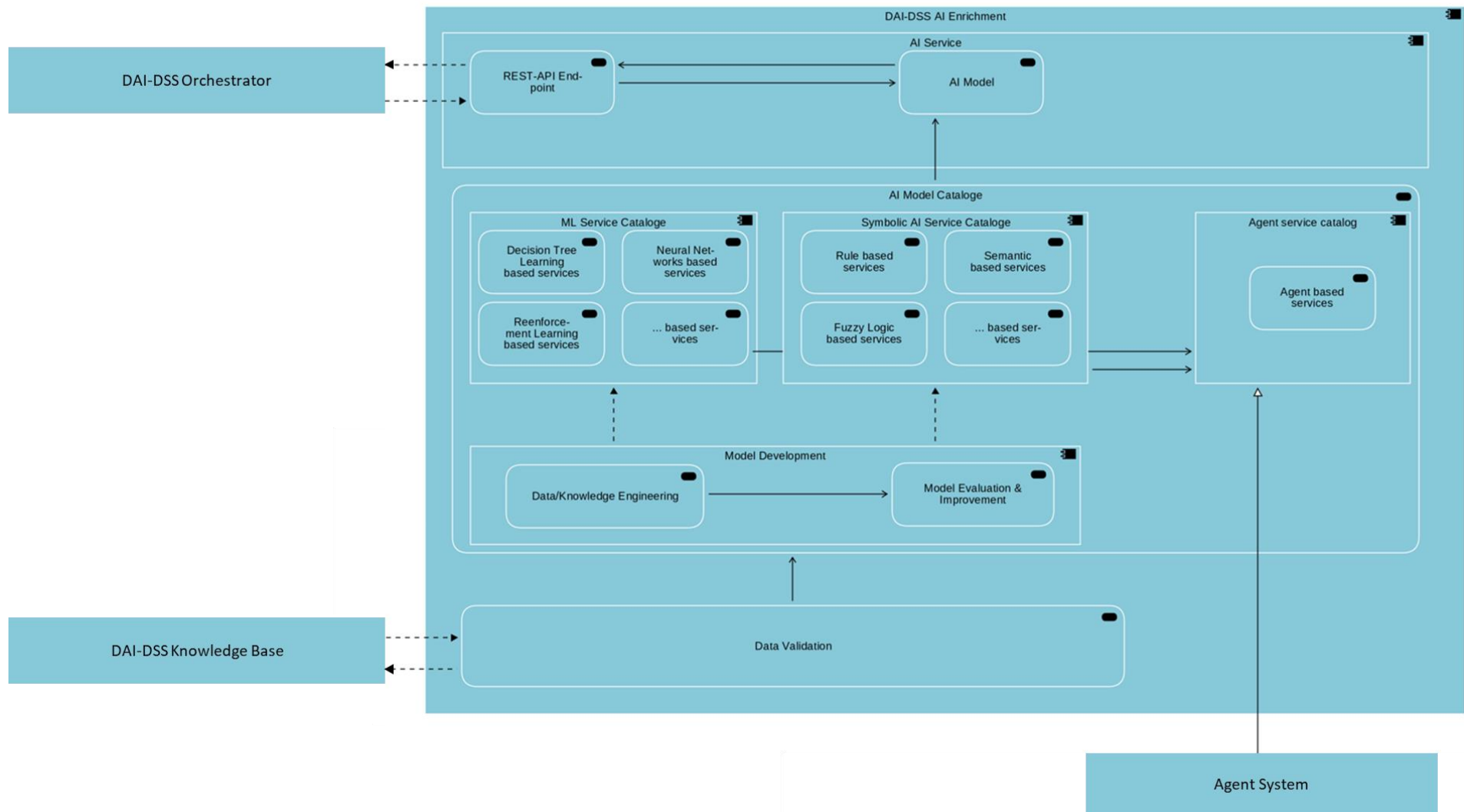


Figure 22 Architecture of DAI-DSS AI Enrichment

3.5.3 Functions

The various AI algorithms and models collected in the DAI-DSS AI enrichment module are connected to the DAI-DSS Orchestrator and provide predictions about the behaviour of the agents or calculated suggestions for system optimization. Some of the possible methods that can be used for this purpose as AI models in the AI model catalogue 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 2018¹⁵, Kaelbling 1996¹⁶) 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)¹⁷ as well as on the machine level (Samsonov 2020)¹⁸. Job shop problems can be a standard use case in most production environments and RL is therefore a useful method for the FAIRWork environment.

Decision Tree Learning (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)¹⁹ With the DTL 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. As the results of the models need to be understandable to build trust and understanding, the interpretable results of DTL models can be very useful.

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 form 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)²⁰, The development and training of ANNs require a great volume of data. But with the possibility to understand patterns beyond the human expert knowledge, ANNs enable insights to problems

¹⁵ Li, Y. (2018): Deep Reinforcement Learning. Online: <https://arxiv.org/pdf/1701.07274.pdf>

¹⁶ Kaelbling, L. P.; Littman, M. L.; Moore, A. W. (1996): Reinforcement Learning: A Survey. In: *Journal of Artificial Intelligence Research*, pp. 237–285

¹⁷ 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

¹⁸ 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

¹⁹ Kotsiantis, S. B. (2013): Decision trees: a recent overview. In: *Artificial Intelligence Review*, Vol. 39, pp. 261–283

²⁰ 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

that offer new solutions. Dependent on the problem definition, this method can bring an advantage to solving complex problems.

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 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)²¹ 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)²²

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 Flask²³ or FastAPI²⁴. However, TensorFlow Serving²⁵ 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 involved in a complex and dynamic environment such as production lines.

²¹ 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

²² Hamdan, Al-H.; Taraben, J.; Helmrigh, 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

²³ <https://flask.palletsprojects.com/>

²⁴ <https://fastapi.tiangolo.com/>

²⁵ <https://www.tensorflow.org/tfx/guide/serving>

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)²⁶.

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.

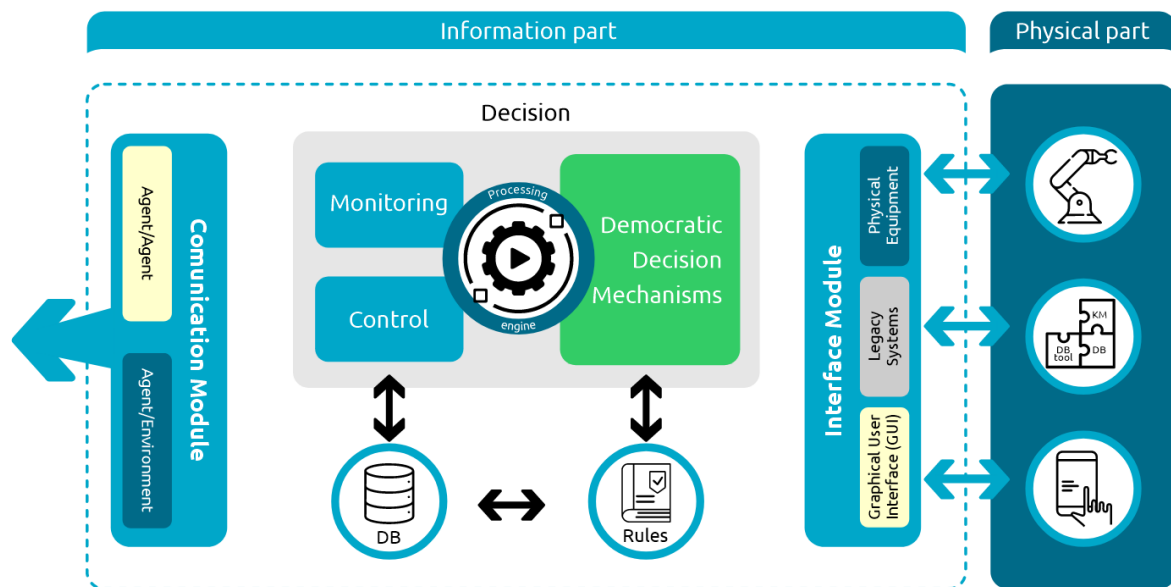


Figure 23 Multi-Agent System: Internal Architecture

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 behaviour 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.

²⁶ 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.

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 analyse 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 analysed 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 crucial for correctly linking the ResourceAgent with the OrderAgent, which manages the currently produced product. Additionally, the ResourceAgent needs to store quality measurement results in its local database for future analysis. The ResourceAgent can also adjust production times and their limits based on information received from other system components. Notably, data from the OrderAgent is used to refine execution parameters for occupancy analysis, while information from the FAIRWorkAgent provides broader system-wide data. This integrated approach ensures efficient resource management and production optimization.

3.5.7.1.4 Agent descriptions

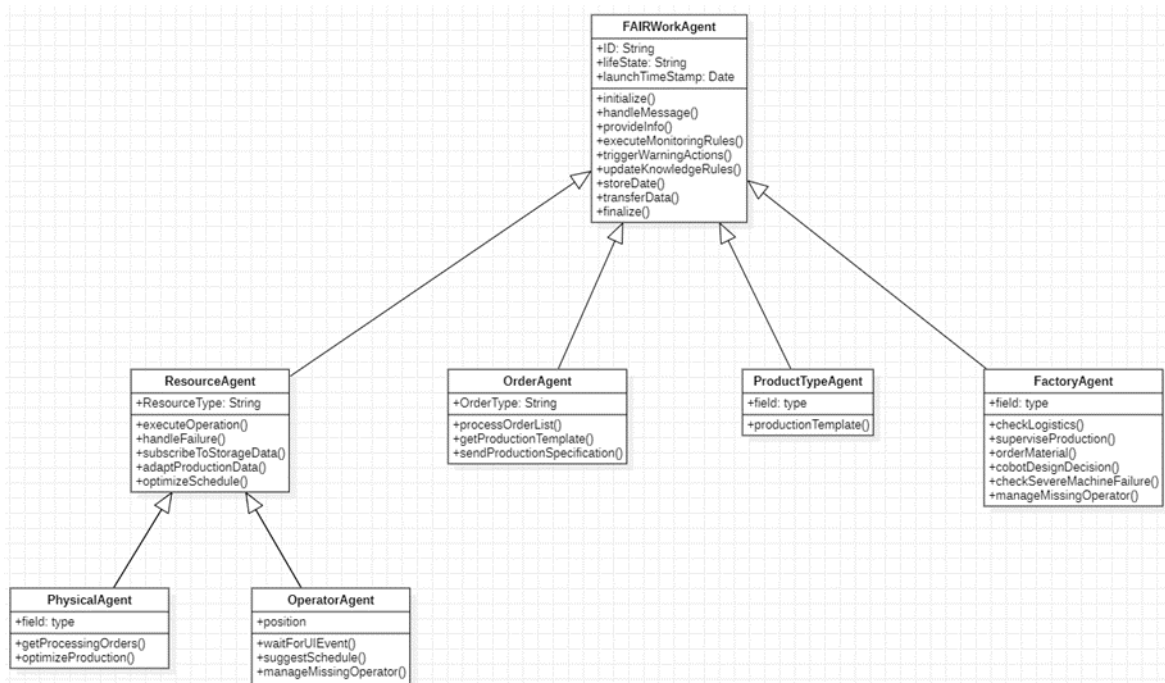


Figure 24 Current agent UML Class Diagram

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 the 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 line 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 necessities, and supplies product needed. The main responsibilities are discriminated below:

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.

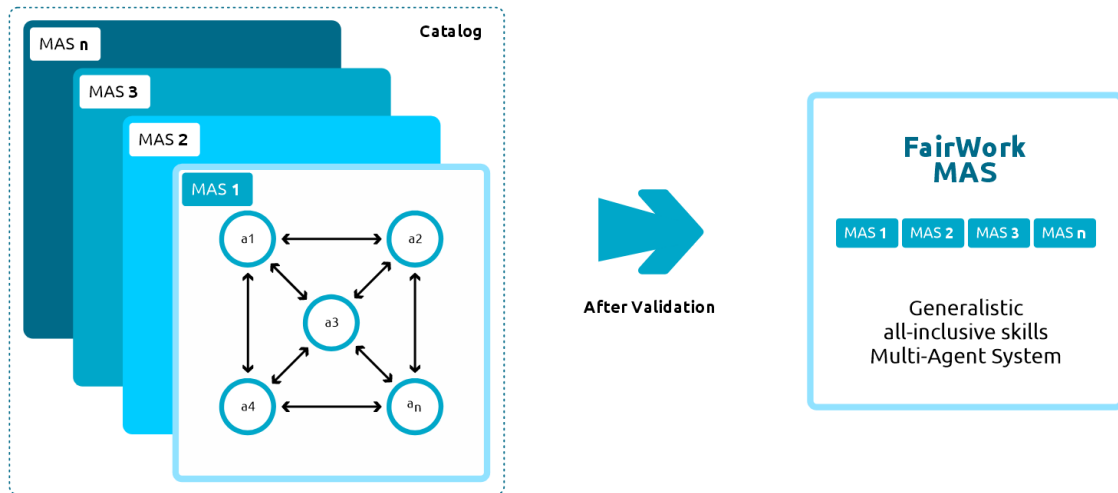


Figure 25 Multi-Agent System development and validation.

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.

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.

3.5.8 Hardware and Software requirements

The DAI-DSS AI Enrichment requires a server tailored to the specific demands of its models. The computing power of the server, particularly CPU performance, should be scaled according to the requirements of the models to ensure smooth and efficient operation. To support the DAI-DSS AI Enrichment, the server must be equipped with an operating system such as Windows or Linux. It should also be capable of running the chosen AI frameworks and libraries, which are expected to be open-source and thus straightforward to install. Additionally, the server must have the necessary REST API framework and other dependencies installed and properly integrated to facilitate the functionality of the models.

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

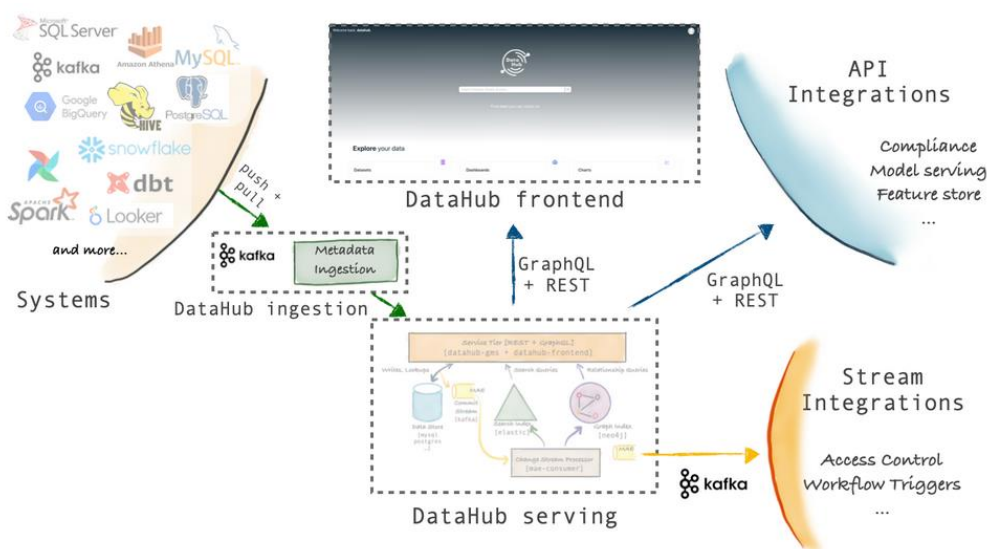


Figure 26 Building Blocks of the data catalogue of the data market

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 an administration part where you can create user, groups and define the access policies.

Datahub servicing component: This component is capable of functioning as a standalone metadata service²⁷ within an open-source repository. Additionally, it supports federated metadata services, which may be owned and operated independently. These federated services interact with a central search index to facilitate global search and discovery capabilities, while still allowing for independent management 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.

²⁷ <https://github.com/datahub-project/datahub/blob/master/metadata-service>

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 assets 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 asset. 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 does it belong?

3.6.3.4 Metadata transformation

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

3.6.4 Dependencies

Data Marketplace 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 DataHub²⁸. Additional functionality will be provided within the FAIRWork project.

²⁸ <https://datahub.io/>

3.6.7 Other Details

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

3.6.8 Hardware and Software requirements

It requires the installation of docker and docker compose. Python 3.8+ is also required. It is important to have sufficient hardware resources for the Docker engine. The software has been tested on 2 CPUs, 8GB of RAM, 2GB of swap space and 10GB of disk space.

3.7 DAI-DSS Intelligent Sensor Box

3.7.1 Purpose

In the DAI-DSS there is the specific need to capture human centred 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 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 analyse data. Edge analytics is a model of data analysis where incoming data streams are analysed 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 behaviours of the human, and from this the sociotechnical systems within the production process.

The integration of human-centred 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.

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 MongoDB²⁹ 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.

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.

AI-based Physiological Strain Estimation

The physiological strain index is particularly relevant for the estimation of stress that is accumulating with respect to the workplace where physiologically intensive tasks have to be accomplished. This service computes, firstly, the core body temperature based on recent skin temperature sensor data using Gaussian Progress Regression. In a further step, it estimates the Physiological Strain Index PSI* using data on recent heart rate-based information as well. The outcome of this component directly feeds into the service for Resilience Risk Stratification. The input raw sensor data are collected from the bio signal smart shirt, alternatively, a smartwatch, as well as by a skin temperature sensor.

Heuristic Cognitive-emotional Stress Estimation

²⁹ <https://www.mongodb.com/>

Cognitive-emotional stress particularly arises when time pressure, cognitively demanding task, or any emotionally arousing situation occurs. This service computes a heuristic index based on input raw sensor data that are collected from the bio signal smart shirt, alternatively, a smartwatch, as well as by a skin temperature sensor.

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.

Resilience Score

This service derives from the input about physiological strain index (PSI) as well as the cognitive-emotional strain score (CES) a resilience score that can itself be used to determine the risk level for critical resilience dynamics. The resilience score is based on a time window of recent Daily Stress Scores (DSC) that are computed from the daily profiles of PSI and CES data. A more detailed description is provided in Deliverable D3.2.

Resilience Monitor

The Resilience Monitor refers to the component that provides a visualization of the development of strain as well as resilience scores (as a basis for resilience risk stratification) over time, i.e., the chosen time window of working days. In this first stage of the development, we refer to a time window of 20 working days. In the top sub-window of Figure 29 the sample collection of PSI and CES scores is represented, for the example of decreasing stress figures. Conversely, the resilience score is increasing over time with a certain inertia and delay.

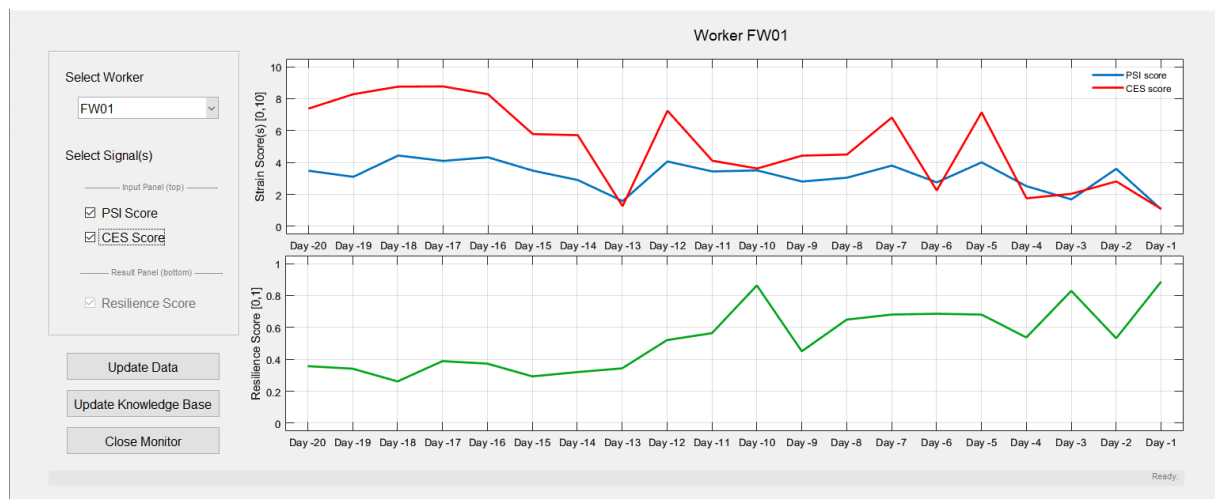


Figure 29 describes the FAIRWork Resilience Monitor

The user can trigger the update of the Knowledge Base by means of an interaction unit (see Figure 29) so that the Resilience Monitor transfers the current resilience score profile within the designated time window of 20 days to the pre-determined directory within the Knowledge Base.

Risk Stratification System

Within the Intelligent Sensor Box there is a specific support module (see Figure 30) with the communication handler and the engine for a research and development dashboard. The raw signals are grabbed, filtered and forwarded to several major processing streams. One sample processing stream is related to the risk stratification for physiological strain, providing a strain index and a rule base to determine the risk level and recommendations on

pause management. Another processing stream is related to the cognitive-emotional strain, again with risk level management. Further processing streams can be identified with respect to the resilience risk stratification and any further future service.

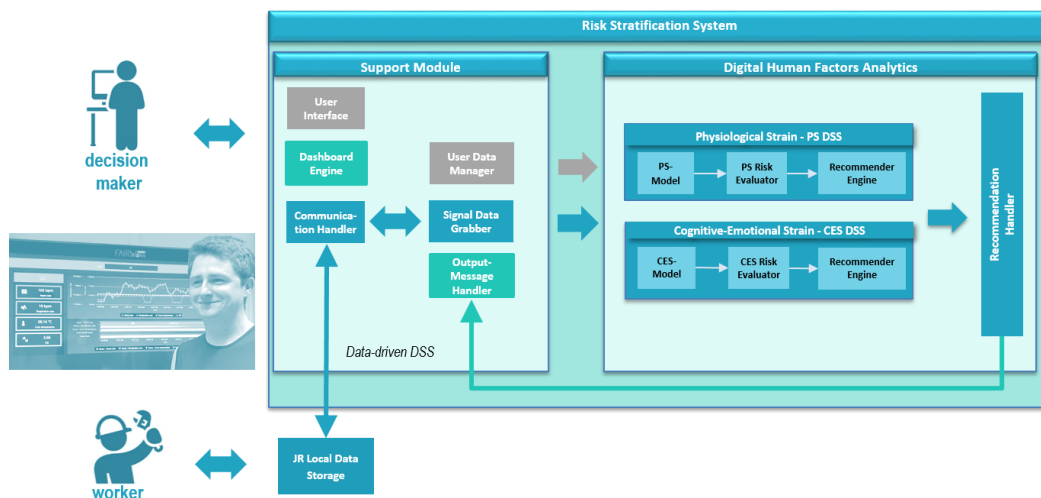


Figure 30 Risk Stratification System of the FAIRWork Intelligent Sensor Box.

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.

Physical Status Monitor

The dashboard Figure 31 is depicting instantaneous vital parameters (left), live biosignal information and scores (upper right graph), risk levels (middle right graph) and recommendations (lower right graph), for the example of the physiologically stressed use case, depicting the “Physical Status” of the worker.

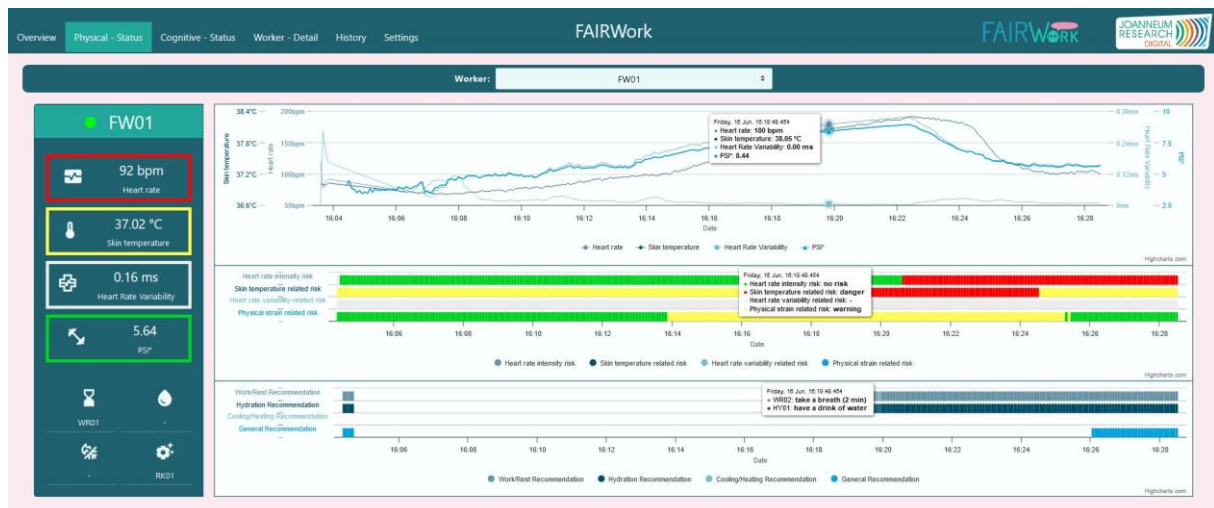


Figure 31 describes the developer dashboard on the “Physical Status”. The dashboard is depicting instantaneous vital parameters (left), live biosignal information and scores (upper right graph)

Cognitive Status Monitor

The dashboard in Figure 32 describes the “Cognitive Status” that is determined by a heuristic cognitive-emotional strain score (CES) that is in turn dependent on input sensor data from skin temperature, heart rate and heart rate variability. Here the raw data (top subfigure), the risk levels (mid subfigure) as well as the recommendations (bottom subfigure) are depicted in a synchronized way. To the left are the current precise data including a semaphore-based visualisation (green, yellow, red; gray) of the current risk level associated with the data.

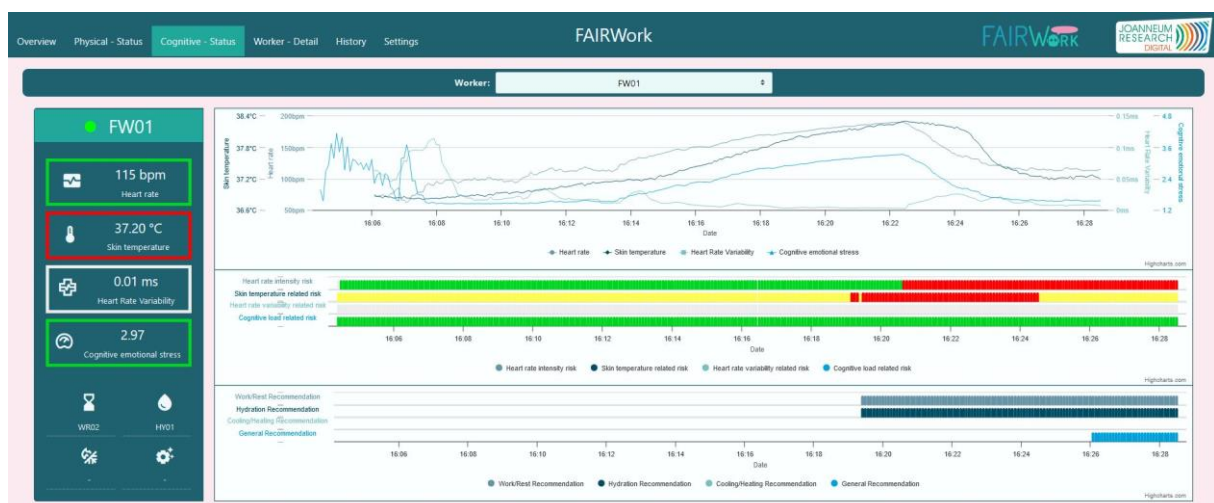


Figure 32 describes the developer dashboard on the “Cognitive Status”. The dashboard is depicting instantaneous vital parameters (left), live biosignal information and scores (upper right graph)

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).

3.7.7 Hardware and Software requirements

At the current stage of development, the hardware requirements are determined by the input bio signal sensors and respective interfaces, such as, from the bio signal smart shirt of QUS, the smartwatch of Garmin (Vivosmart 5) as well as the Pupil Labs “Invisible” or “Neon” devices. For the real-time monitoring from the smartwatch, a license for the use of the wearable data platform Fitrockr (c/o Digital Rebels GmbH) is required. For the computing of the resilience score as well as for the Resilience Monitor, Matlab functionality is required.

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 technical implementation details are described following 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.
- 6. Prerequisites**
 - This section outlines the prerequisites necessary for a use case or decision scenario to effectively utilize the service. It provides users of the DAI-DSS with the criteria needed to determine if this service is suitable for their specific needs. For instance, it specifies the minimum size of the data set, the required data structure, and the expert knowledge that must be available.
- 7. Usage**
 - Describe the Applicable FAIRWork use case scenarios where this service can provide decision support.

4.2 AI services list

The list of services provided in this section showcases various AI technologies designed to support decision-making for the use case scenarios outlined in section 2.1 of this deliverable. Each use case highlights potential applications for AI services, though this is not an exhaustive guide for implementing these technologies. The actual number of services developed may vary, influenced by the research outcomes from WP3. It's possible for multiple services to address the same problem; these will be assessed in a lab setting to determine the most effective based on performance metrics. The services are organized into categories as follows:

- **Initial Prototype**
 - Includes all AI services equipped with REST API endpoints.
- **Development in Progress**
 - Comprises services that are under development and do not yet offer API endpoints.
- **Planned for Implementation**
 - Currently in the research phase and not yet in development.

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

Table 4 AI Services List

S.NO	AI Service name	State of Development
1	Worker to production line mathematical formula or Heuristic	Initial Prototype
2	Resource Allocation Service with Linear Sum Assignment Solver	Initial Prototype
3	Pattern based resource allocation	Initial Prototype
4	Operator stress estimation with neural networks	Initial Prototype
5	Multi agent-based resource allocation	Initial Prototype
6	Rule-based Service based on Conceptual Models	Initial Prototype
7	Rule base resource allocation service	Initial Prototype
8	Worker Efficiency estimation	Development in Progress
9	Resource mapping with decision trees	Development in Progress
10	Schedule optimizer with Linear Programming	Planned for Implementation
11	Similarity matching knowledge graphs	Planned for Implementation
12	Reinforcement Learning Based Resource Allocation Service	Planned for Implementation
13	Annotation support service	Planned for Implementation

S.NO	AI Service name	State of Development
14	Production process simulation with agents	Planned for Implementation
15	Operator persona development with machine learning	Planned for Implementation
16	Community detection with knowledge graph	Planned for Implementation
17	Production process simulation	Planned for Implementation
18	Similarity matching with semantics	Planned for Implementation
19	Impact assessment with knowledge graph	Planned for Implementation
20	Semantic search with vector embeddings	Planned for Implementation
21	Automated Test Building Support with Hybrid Filtering	Planned for Implementation

4.2.1 Worker to Production Line Mathematical Optimization/Heuristic

4.2.1.1 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 33.

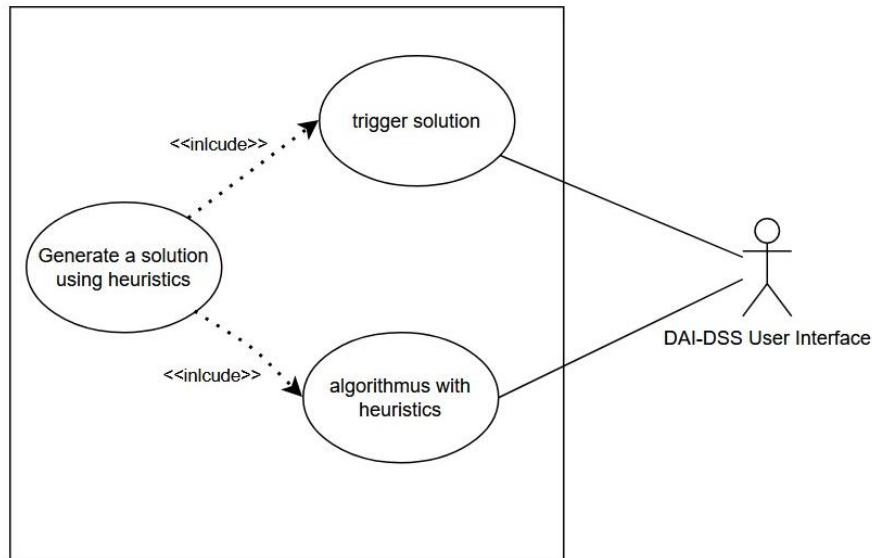


Figure 33 UML Use Case Diagram for Worker assignment to production line by using mathematical optimization/heuristics.

4.2.1.2 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 34. 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.

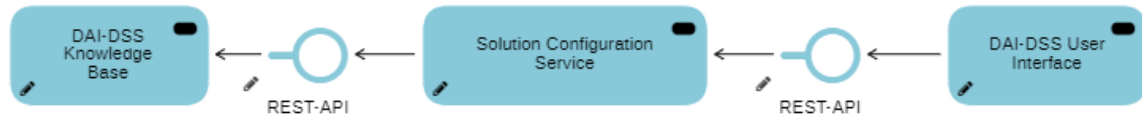


Figure 34 Service Component for Worker assignment to production line by using mathematical optimization/heuristics

4.2.1.3 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.

4.2.1.4 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.

4.2.1.5 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.

4.2.1.6 Prerequisites

To use this service, it requires a description of the input parameters and configuration, which is sent via the REST API from the orchestrator. The input parameters and configuration get the information from the data inserted into the user interfaces and return the result with the resource assignments.

4.2.2 Resource Allocation Service with Linear Sum Assignment Solvers

4.2.2.1 Purpose

The Linear Sum Assignment Resource Allocation Service efficiently assigns available workers to specific machines based on predefined criteria and constraints. Each machine requires a fixed number, N , of workers to operate effectively. Moreover, certain workers may possess specialized skills or qualifications, limiting their availability to work only on specific lines. The allocation service considers these constraints and optimally assigns workers to machines while adhering to the requirement of N workers per machine and accommodating the restrictions on worker-line compatibility. Figure 35 shows the UML-use-case diagram of the Linear Sum Assignment Solver.

Additionally, the service considers the preferences of individual workers, ensuring that assignments align with their skills, expertise, and personal inclinations whenever possible. By leveraging linear sum assignment techniques, this service performs worker allocations, maximizing efficiency and productivity while satisfying the various constraints and preferences inherent in the workforce and worker to machine allocation problem.

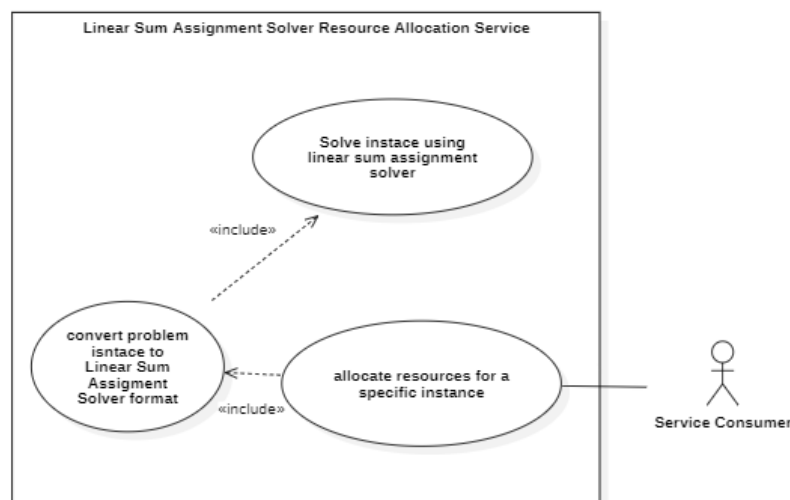


Figure 35: UML Use Case Diagram of the Resource Allocation Service with Linear Assignment Solver

4.2.2.2 Service Component

The Linear Sum Assignment Resource Allocation Service is a component that is not subdivided into other components. Figure 36 illustrates the interfaces between it and other components of the overall system using a component diagram.



Figure 36: Service Component for interfaces between the Resource Allocation Service with CP-Solvers and DAI-DSS Architecture

4.2.2.3 Interface and Data

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

4.2.2.4 Functions

The function of the Linear Sum Assignment Resource Allocation Service is to allocate a set of available workers to a set of machines.

4.2.2.5 Dependencies

The Linear Sum Assignment Resource Allocation Service does not depend on other components.

4.2.2.6 Prerequisites

To use this service, the algorithm needs a description of the problem instance that will be sent via the REST-API from the orchestrator. The problem instance gets the information from the data inserted in the user interface like the available resources and the jobs to be done. That way the algorithm can map resources.

4.2.2.7 Usage

This service is developed with synthetic data based on the situation at CRF. In the current status it can be used to allocate workers to machines for the CRF use-case.

4.2.3 Pattern Based Resource Allocation Service

4.2.3.1 Purpose

The Pattern Based Resource Allocation Service provides a solution to a concrete instance of an allocation problem. Figure 45 shows a UML-use-case diagram of the Pattern Based Resource Allocation Service. The Pattern Based Resource Allocation Service uses machine learning to perform the allocation. The neural network mimics the allocation from historical data. 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.

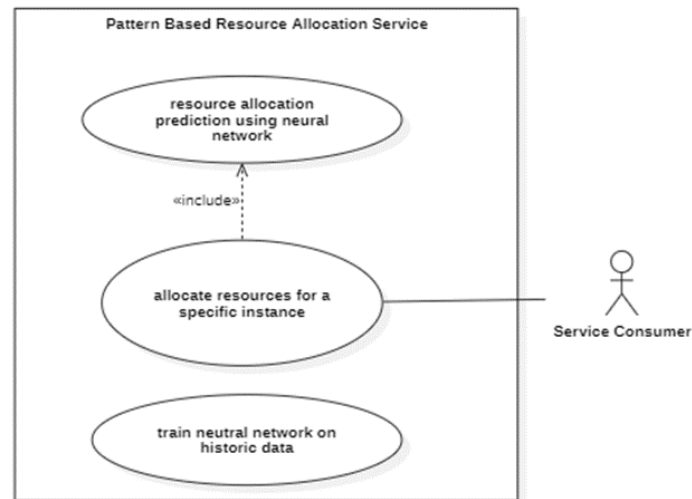


Figure 37 UML Use Case Diagram of Pattern Based Resource Allocation Service

4.2.3.2 Service Component

The Pattern Based Allocation Service is a component that is not subdivided into other components. Figure 38 illustrates the interfaces between it and other components of the overall system using a component diagram.



Figure 38 Service Component for interfaces between Pattern Based Resource Allocation Service and DAI-DSS Architecture

4.2.3.3 Functions

The 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.

4.2.3.4 Interface and Data

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

4.2.3.5 Dependencies

The pattern Resource Allocation Service does not depend on other components.

4.2.3.6 Prerequisites

To use this service, the algorithm needs a description of the problem instance that will be sent via the REST-API from the orchestrator. The problem instance gets the information from the data inserted in the user interface like the available resources and the jobs to be done. That way the algorithm can map resources.

4.2.3.7 Usage

This service is developed with synthetic data based on the situation at CRF. In the current status it can be used to allocate workers to machines for the CRF use-case. This algorithm is especially useful for more complex data when historical allocation data is available.

4.2.4 Operator Stress Estimation with Neural Networks

4.2.4.1 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 39).

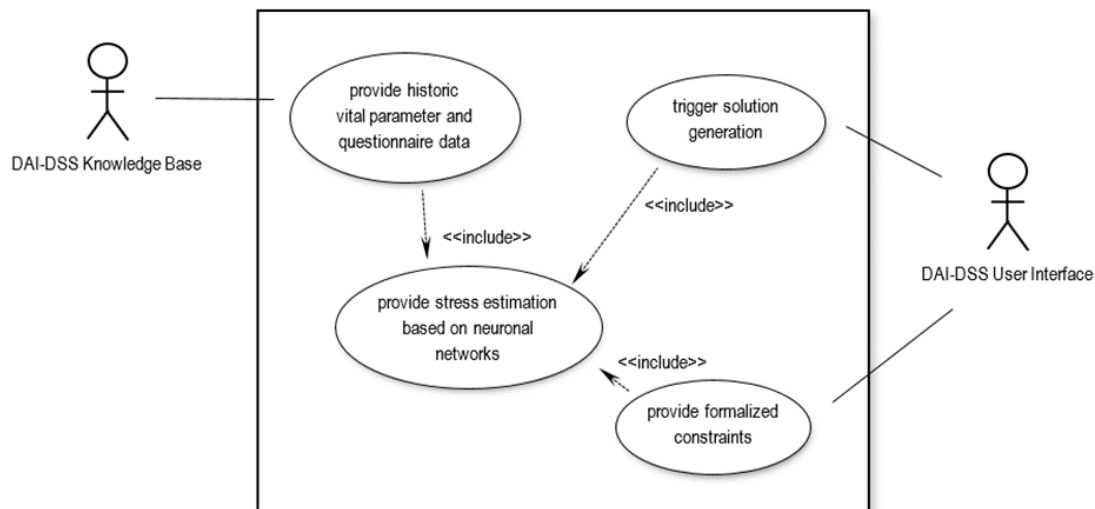


Figure 39 UML-use-case diagram of Operator Stress Estimation Service

4.2.4.2 Service Component

The operator stress estimation service is not subdivided into other components (refer Figure 40).



Figure 40 Service Component for Operator Stress Estimation Service

4.2.4.3 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.

4.2.4.4 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: inferred stress level signal.
- Data format: Json (for in and output data).

4.2.4.5 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.

4.2.4.6 Prerequisites

To use this service, employees' historical biosignal data from the last few working days (ideally the last 20 days) have to be available. The data is recorded by the Intelligent Sensor Box itself which retrieves data from eye-tracking glasses, sensor shirts (QUS-Shirt) and/or wrist bands (Garmin vivosmart 5). Also the workers metadata like age, sex, weight, height, and HR_{max} (if available) have to be present in the ISB' worker health profiles.

4.2.5 Multi Agent-based Resource Allocation

4.2.5.1 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 41).

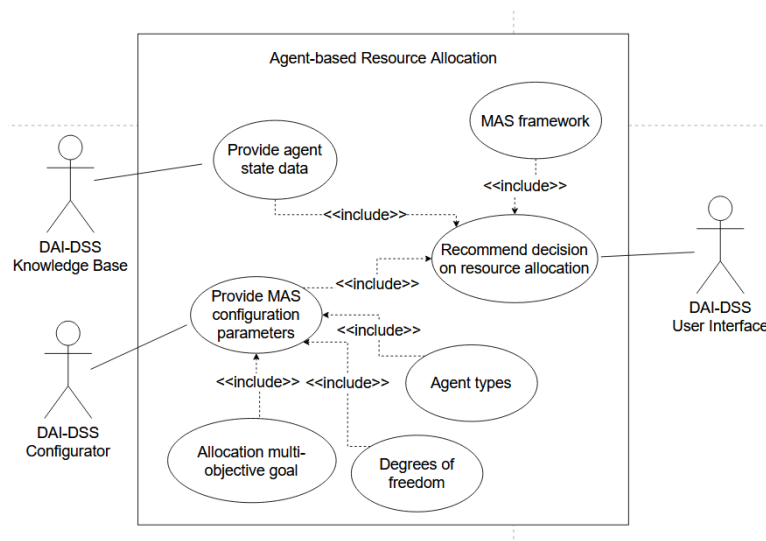


Figure 41 UML Use Case Diagram for Agent-based Resource Allocation

4.2.5.2 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 42).

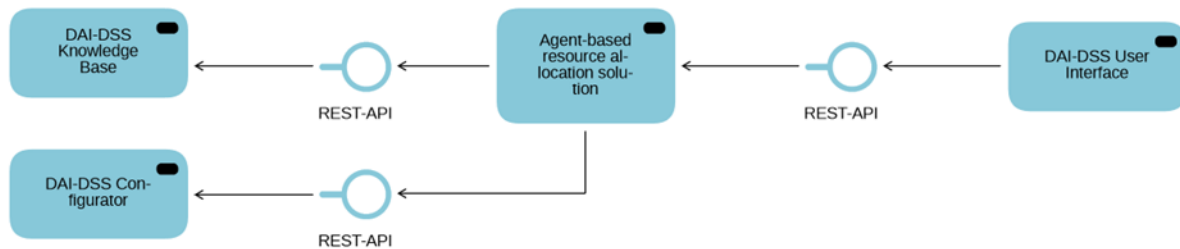


Figure 42 Service Component for production process simulation with agents

4.2.5.3 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.

4.2.5.4 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.

4.2.5.5 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 resources that need to be better addressed along the production chain.

4.2.5.6 Prerequisites

For the Multi-Agent Resource Allocation service, the prerequisite is the data relevant for executing resource allocation strategies effectively according to each resource allocation case. The service requires parameters that guide the allocation decisions, such as resource availability, production demands, and priority constraints for example. These parameters can be derived from historical data on resource usage or projected requirements in evolving operational contexts.

4.2.5.7 Usage

The current service is being applied to the CRF Workload Balance Use Case Scenario where the Multi-Agent System access via Knowledge Base data such as Worker Availability, Medical Condition, Worker Experience, Worker Resilience, Production Priority, Due Date and Workers Required to perform the resource allocation service. The current application demonstrates an advanced integration of diverse data types to optimize workforce management. These data are crucial for ensuring that the resource allocation not only meets production demands but also aligns with the well-being and capabilities of the workforce.

4.2.6 Rule-based Service based on Conceptual Models

The service described here was prototypically implemented in an experiment. More details of this implementation can be found in FAIRWork's deliverable D4.2. In this chapter a generic version of the design and implementation, agnostic from a concrete case is described. The service itself was integrated with the prototype of the DAI-DSS orchestrator to test the connectivity.

4.2.6.1 Purpose

The purpose of this service is to enable an adaptable way to assign values or groups to the provided input parameter. The core of the service uses a rule-engine to provide a fitting result to the provided data. The goal is to not only create one service, containing all rules to make all possible decisions within the DAI-DSS, but to have a generic service, which can be configured to make concrete decisions. Each configuration is instantiated in its own endpoint, making it usable by the orchestrator for different decisions. The first prototype of this service was used for the first prototype of the DAI-DSS³⁰. Figure 43 shows the use case diagram for the service.

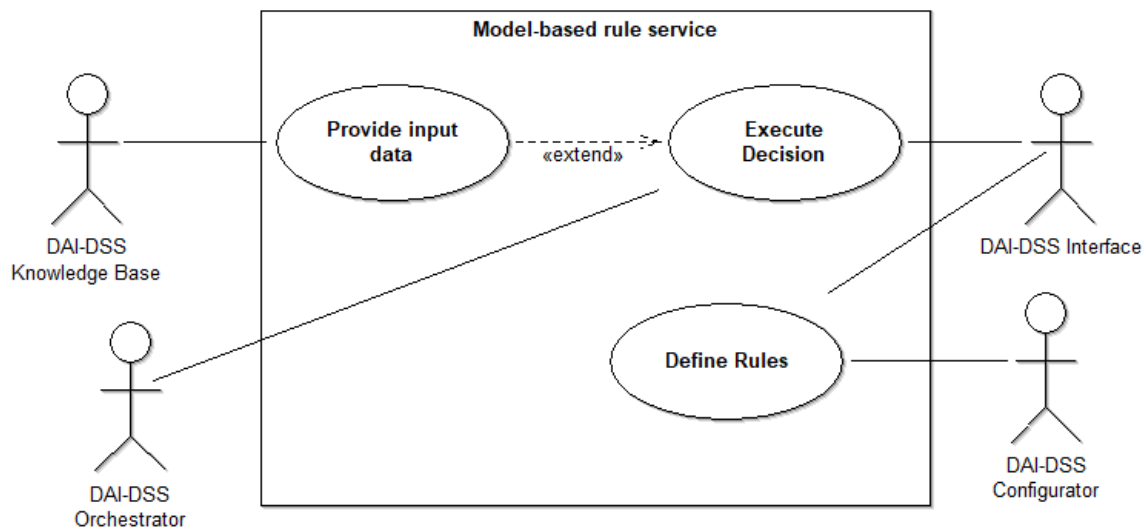


Figure 43 UML Use Case Diagram for the Model-enabled Rule Service

4.2.6.2 Service Component

This *Model-enabled Rule Service* component will be provided as a singular service, which can be configured to support a specific decision problem. For each problem a separately configured service instance must be created.

³⁰ Vieira, G. (2023). Initial DAI-DSS Prototype D4.2.

To achieve this, two interfaces are offered: One interface for the rule configuration and one for the triggering the decision.

The rule configuration interface needs information about the decision logic and technical information about the endpoint. The interface for triggering a decision, must provide the input parameters in the right JSON format (see section 4.2.6.3). That a decision is made can be triggered by the DAI-DSS orchestrator or the DAI-DSS User Interface. The triggering component must provide the data needed for the decision, which means for the orchestrator, that the data will be loaded from the DAI-DSS Knowledge Base and then prepared to be used as input for the decision service.

An overview of the design of this component can be seen in Figure 44. The service component and its two interfaces can be seen in the lower left half of the figure. The other objects, represent the corresponding components from the DAI-DSS architecture.

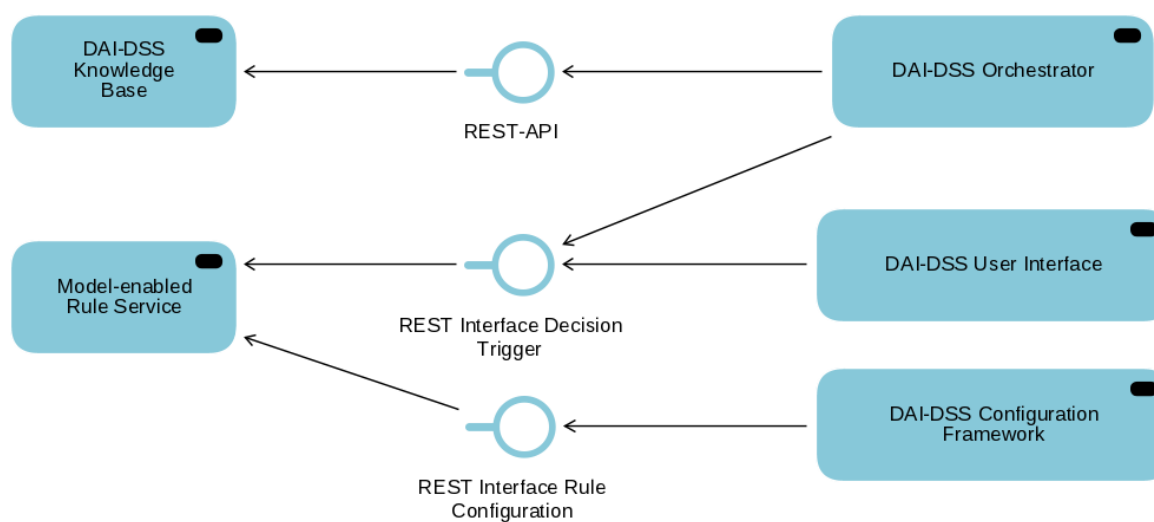


Figure 44 Service Component Model-enabled Rule Service

4.2.6.3 Interface and Data

REST Interface Decision Trigger: This interface allows to trigger decisions, based on the configured decision rules of a certain service. The input data must be provided in JSON format.

To call this endpoint the basic structure of the Olive microservice framework must be used. Therefore, a POST call must be made to base URL of a deployed Olive instance, with the two parameters *microserviceId* and *operationId* where the concrete value is also configured.

The input data for the decision, must be formatted in JSON. Thereby multiple decisions can be triggered at once. The basic structure of the JSON looks like this:

```
{
  "decision-variable-name": {
    "value": "true;false"
  },
  "decision-variable-name-2": {
    "value": "1;3", ...}
}
```

Here the decision variables, with the values must be provide. The key of the JSON file is provided through the configuration. Each of the keys is required to have a JSON object, where the key is *value*, and it contains a string with the input parameters for all the decisions which should be made at once. In the example above, two decisions will be made, one has the parameter *true* for <decision-variable-name> and 1 for <decision-variable-name-2> and the second decision has *false* for <decision-variable-name> and 3 for <decision-variable-name-2>.

In the result, the key *DecsionResultList* contains an array with all the solutions of the different decisions. This key is always the same. But the content of the arrays within is not fixed and depends on the configuration. The result from the applied rules is provide together with the key, which is defined in the configuration. For each decision send to the service, an entry is made in the *DecisoinResultList* array.

REST Interface Rule Configuration: This interface allows to configure concrete instances of the decisions service. The information for the rules and for creating the endpoint must be provide and send at the REST interface. For the prototype of the service, we use the Olive Controller web interface, which automatically creates the needed JSON file and uploads it, if the needed information is provided.

The needed information for configuring the service consists of three parts. The decision logic in the form of rules must be provided as an XML file based on the DMN standard. This file can be created out of the models, created with the Bee-Up extension³¹ implemented for the project. As this rule file can contain multiple decisions, the identifier of the decisions which should be returned must be provided as a string. Last but not least, a mapping from the endpoint parameters, used as key values for the decision trigger interface (see above) to the decision variables used in the model.

More information on how these two interfaces can be used for the prototype can is describe in the deliverable D4.2.

4.2.6.4 Functions

This service possesses two main functionalities. One is the configuration of the decision logic in the form of rules and the other is to allow to provide an answer to the sent parameters, based on the rules. To support the configuration of a concrete decision service, a user interface is offered, as it was developed with the Olive microservice framework.

4.2.6.5 Dependencies

This service depends on the modelling tool created to support the configuration, because otherwise the configuration is only possible with advanced knowledge about DMN XML standard. This would reduce its usefulness within the DAI-DSS.

It depends also on the orchestrator, which will get the data from the knowledge base, preprocess it, call this service, and then further use the provided results.

Finally, this service depends on the knowledge of the decision makers. As their knowledge must be encoded in the rules and support the decision. If this knowledge is not available or cannot be correctly encoded, the service will provide unreliable results.

4.2.6.6 Prerequisites

To use this service, knowledge on how the decision is be made must be accessible. For example, this information can come from an expert who already made the decision in the production environment. This information must then be gathered from the expert, especially the attributes on which the decision is based and the rules on how an

³¹ Bee-Up Extension created for usage with the decision service: <https://code.omilab.org/research-projects/fairwork/decision-services/bee-up-dmn-extension> (accessed: 01.03.2024)

outcome from the decisions can be derived. This knowledge must then be encoded in conceptual models, so that it can be used as input for the service.

4.2.6.7 Usage

To use this service an instance of the generic service implemented in Olive microservice framework must be made, containing the needed parameters and the decision logic. This information must be provided in a file with the fitting data format. This file can be created using the ADOxx-based Bee-Up Modelling tool and the extension that was created to work with the rule-based decision service.

After the decision logic is defined, the service must be configured using the modelled information, to create an instance with its endpoint. This endpoint is created within an Olive instance and can then be used by the orchestrator.

How this configuration and the modelling tool can be used is described in more detail on the extensions project page 34 or in our Deliverable 4.2 (Vieira, G 2023). The deliverable also contains information about the needed Olive instance. In addition, a Docker for the Olive instance with the decision service and additional information can be found on the project page.

4.2.7 Rule Based Resource Allocation Service

4.2.7.1 Purpose

The Rule Based Resource Allocation Service provides a solution to a concrete instance allocation problem. Figure 45 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.



Figure 45 UML Use Case Diagram of the Rule Based Resource Allocation Service

4.2.7.2 Service Component

The Rule Based Allocation Service is a component that is not subdivided into other components. Figure 46 illustrates the interfaces between it and other components of the overall system using a component diagram.



4.2.7.3 Functions

The 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.

4.2.7.4 Interface and Data

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

4.2.7.5 Dependencies

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

4.2.7.6 Prerequisites

To use this service, the algorithm needs a description of the problem instance that will be sent via the REST-API from the orchestrator. The problem instance gets the information from the data inserted in the user interface like the available resources and the jobs to be done. That way the algorithm can map resources.

4.2.8 Worker Efficiency Estimation

4.2.8.1 Purpose

Worker efficiency estimation with Fuzzy Logic allows to include rules made through experience, but also to avoid the "cut-off date issue" for input and output values that have no clear yes or no. Especially KPIs that are hard to measure, as they include subjective information or are dependent on changing circumstances such as daily motivation of workers, are suitable to be estimated with Fuzzy Logic. 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 not only benefits for finding solutions in uncertain, ambiguous environments but can also assist decisions with not directly measurable KPIs that include not exactly known input values. Figure 47 shows an UML-Use-Case diagram of the Fuzzy Logic Based Worker Efficiency Estimation.

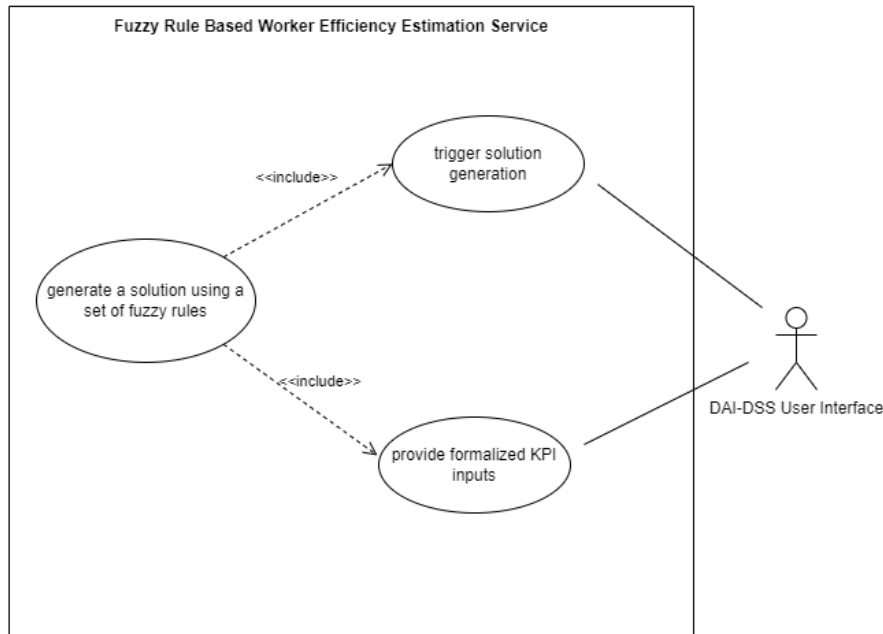


Figure 47 UML Use Case Diagram for Fuzzy Rule-based Worker Efficiency Estimation Service

4.2.8.2 Service Component

The Fuzzy Rule Based is an allocation service that is not subdivided into other service components. Figure 48 illustrates the interfaces between it and other components of the overall system using a component diagram. The service accesses the DAI-DSS User Interface via a REST interface.

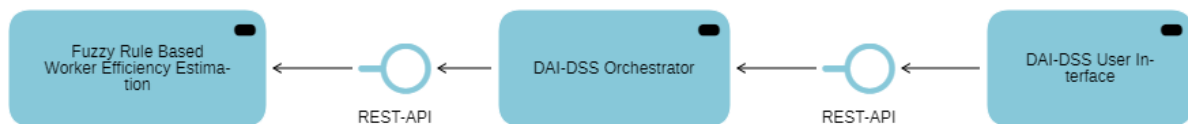


Figure 48 Service Component for Fuzzy Rule based Worker Efficiency Estimation

4.2.8.3 Interface and Data

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

- Interface: REST-API Endpoint
- Input: concrete KPI relevant details of a specific worker provided by the user
- Output: Based on the input data, the efficiency of the worker at a given line with a specific geometry indicated and illustrated in a graph.

4.2.8.4 Functions

The function of the Fuzzy Logic based decision support is to include informal, subjective, or experience-based knowledge for worker efficiency estimations. The service can assess and predict whether worker will be suitable or not for a specific geometry and line. 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 service, they must be adapted specifically for each use case. Due to the knowledge gathering and storing activities, the fuzzy system can be seen as a knowledge management tool within the company.

4.2.8.5 Dependencies

The Fuzzy Rule Based Worker Efficiency Estimation does not depend on other components.

4.2.8.6 Prerequisites

Experts are a precondition for the Fuzzy Logic Service. They are required to formulate and define the input parameters, their ranges, and the different memberships. Additionally, the connection of the input and the output parameters in the rule-based depends on expert experience. Due to the rule-based reasoning approach, there is no training or test data required. When the service should be applied to other scenarios, the prototype must be adapted with new rules and inputs derived from expert interviews.

4.2.9 Resource Mapping with Decision Trees

For this service a preliminary prototype was created, which still must be finished and adapted to allow an integration to the DAI-DSS. This section introduces the first design of the service on a generic level. The introduced design should be understood as a plan for a possible future service, which is influenced by the available prototype.

4.2.9.1 Purpose

The service was designed with the case for allocating resources to production lines in mind. 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. As result the service will return a possible matching, based on the provided input. The service should support the definition of multiple decision trees, supporting different decision scenarios. A use case diagram for this service can be seen in Figure 49.

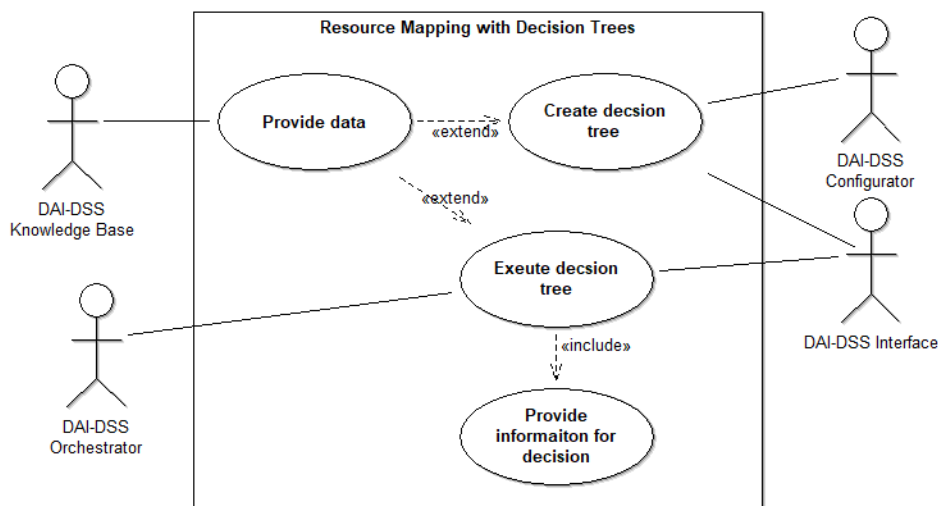


Figure 49 UML Use Case Diagram for the Resource Mapping with Decision Trees Service

4.2.9.2 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 has two interfaces, one for creating decision trees and the other to use available decision trees to execute decisions. An overview of the service component is shown in Figure 50. For this service component, the DAI-DSS can provide data input if available. This data can either be used for creating a decision tree by providing the training data or real-time data can be used as input for the decision

itself. The configuration framework and the orchestrator can use data from the knowledge base if available. Otherwise, the data must be provided from another source, e.g. through the user interface.

Therefore, both the user interface and the configuration framework are able to create decision trees via the available interface. Additionally, both the user interface and the orchestrator can trigger to decide with an available decision tree. The user interface then shows the result to the user, whereby the orchestrator can use the results for further decision making.

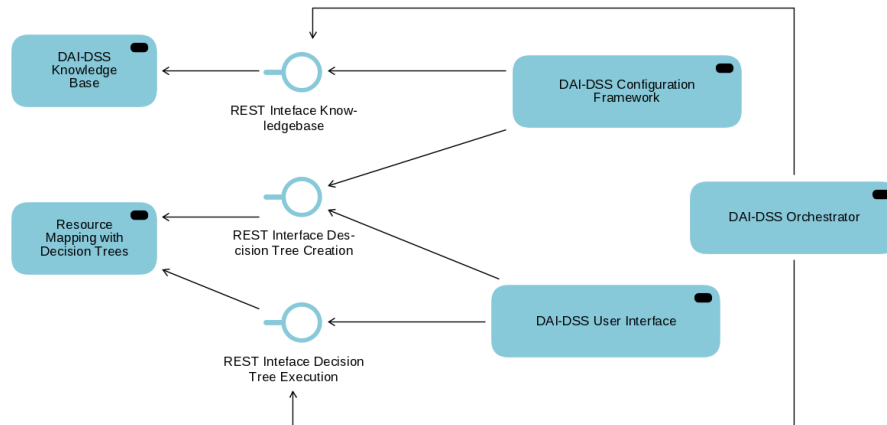


Figure 50 Service Component Resource Mapping with Decision Trees Service

4.2.9.3 Interface and Data

Decision Tree Creation Interface: This REST interface will consume data in JSON format and create a decision tree by learning from the provided data. The JSON will contain the training data and an ID for the tree, which will allow its later usage. The output of calling this service is, that the trained decision tree is available in the service and can be used for decision making.

For the parameter used for the decision, no concrete ones are defined as they must be adapted to the decision scenario. Therefore, this configuration step exists.

Decision Tree Execution Interface: This REST interface consumes data in JSON format and uses them as input for the decision tree. This data must contain the parameters as input for the decision and the decision tree identifier, which should be used. The decision parameters sent here must fit to the once used during the training of the decision trees, so that it can be used. The result of this interface call is the result of the decision tree and depends on the used decision tree itself. Therefore, no concrete decision parameters are defined for this section, as they must be configured to fit for the trained decision tree.

4.2.9.4 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 training data for 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 decision data of a concrete decision must be provided to the service.

4.2.9.5 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 and knowledge about which parameters are needed to make a decision.

4.2.9.6 Prerequisites

To use this service a decisions tree must be trained on historical data and therefore the data must be provided in a form that the decision tree service can process it. First the data must be gathered within the production environment, containing the attributes which are considered for the decisions. Here concrete instances of datasets containing values for the attributes and the expected or really applied outcome of the decision must be provided.

The gathered data must contain cases for all cases that should be covered with the decision trees in varying forms. This is needed to improve the derived decision tree.

4.2.10 Schedule Optimizer with Linear Programming

4.2.10.1 Purpose

The Schedule Optimizer with Linear Programming service aims at finding the optimal time schedule for jobs at a certain production line while considering a predefined objective and several constraints. Linear programming is a common technique for scheduling problems as constraints or objectives for scheduling problems are often generic, meaning they do not depend on concrete use case details. For example, is their formulation is often very similar, e.g., one order can only be processed at a time at one machine, or one worker can only work at one machine at a time. This service allocates efficiently N jobs to one machine with the objective to minimize the sum of jobs overdue. The feasible output is restricted by several constraints, such as that the job's "Start Time" must be after the "Date of Order" or that one machine cannot produce two jobs simultaneously. Figure 53 shows the UML-use-case diagram of the Linear Programming service for optimal production schedules.

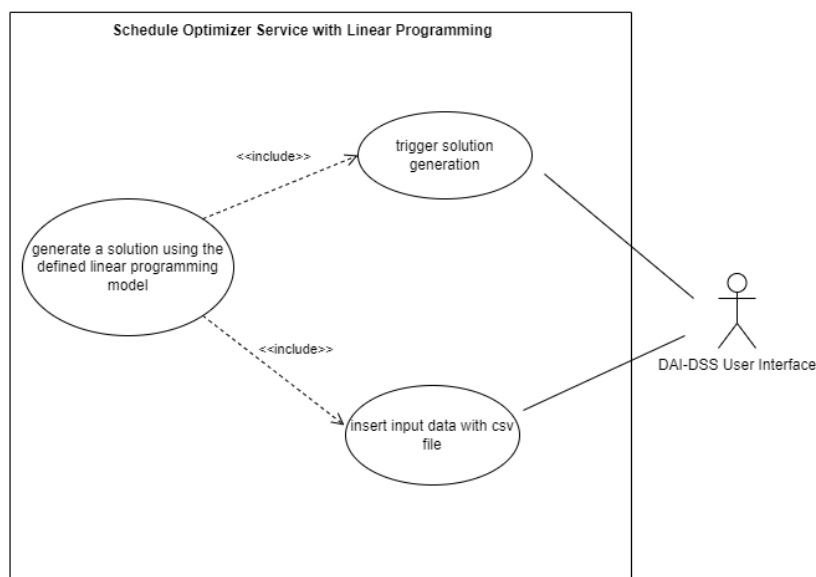


Figure 51 UML Use Case Diagram for Schedule Optimizer with Linear Programming

4.2.10.2 Service Component

The Schedule Optimizer with Linear Programming is an optimization service that is not subdivided into other service components. Figure 54 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 and it offers a visualization of the allocation to the machine. Additionally, a KPI summary of the overall linear programming model is given.



Figure 52 Service Component for Schedule Optimizer with Linear Programming

4.2.10.3 Interface and Data

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

- Interface: REST-API Endpoint
- Input: csv file including job IDs, duration of the job, date of order, target deadlines provided by the user
- Output: Based on the input data, a schedule for producing orders at a production machine is proposed and illustrated in a graph. The optimization considers the predefined objective.

4.2.10.4 Functions

The function of the Schedule Optimizer with Linear Programming is to allocate a set of incoming jobs to one machine in an optimal way while ensuring the defined objective.

4.2.10.5 Dependencies

The Schedule Optimizer with Linear Programming Service does not depend on other components.

4.2.10.6 Prerequisites

The Linear Programming Service requires the specified input data as well as a correct definition of constraints and objective for the model. If the mathematical and use case-specific requirements for formulating the model details are fulfilled, it can be applied to supporting human decision-makers by suggesting an order schedule at a single-machine bottleneck.

4.2.11 Similarity Matching with Knowledge Graphs

The service described in this section is a planned service within the FAIRWork project. Therefore, no prototype was created for it and the description contains the first introduction to the idea behind the service.

4.2.11.1 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 53 shows a use case diagram for this service. The knowledge base in this context must provide the data both for the resources and the tasks. Then the service must be able to match their similarity and allow to define the tasks for which fitting resources must be found.

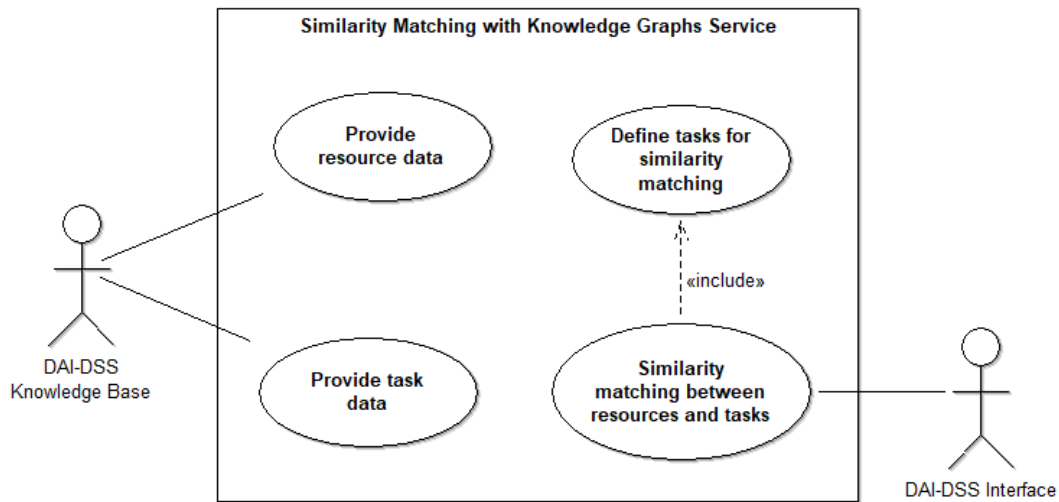


Figure 53 UML Use Case Diagram for the Similarity Matching with Knowledge Graphs service

4.2.11.2 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 54. The matching can be triggered either through the user interface directly or through the orchestrator. If used with from the orchestrator the results can be used for further decisions.

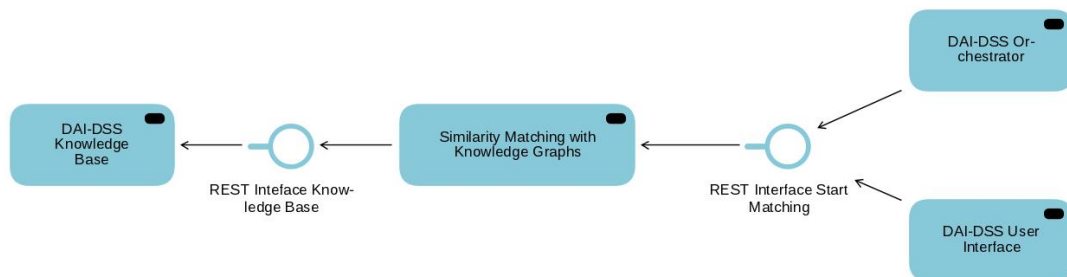


Figure 54 Service Component for Similarity Matching with Knowledge Graphs service

4.2.11.3 Interface and Data

Start Matching: As the service is in a planning phase, no concrete interface will be discussed here, but the general aspects of the needed data. The REST interface will consume data in JSON format or an RDF-based data structure. The service will need information about for which task resources should be found and a possible preselection, if possible.

The result of the interface evocation will be a list with fitting resources, which can be used for the defined task.

REST Interface Knowledge Base: Here the data set in which the communities should be found will be extracted.

As this service is still in the planned state, no concrete data points are defined here.

4.2.11.4 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.

4.2.11.5 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.

Information about for which tasks the resources should be found must be available within the system.

4.2.11.6 Prerequisites

This service is based on a knowledge graph representation of the available information about the resources' capabilities and the needs of the task. This meta information, which describes the resources and tasks is used for matching by similarity and must be created from knowledgeable persons, familiar with knowledge graphs and the production environment. A basic structure of this meta data can be provided with the service which may be adapted to concrete production environments.

To use the similarity matching for a concrete set of tasks and resources, they must be represented in the knowledge graph as instances and linked to the meta information. Through the links the service better understands the data and can use it within the similarity matching. Here the set of tasks and resources, considered for the decision must be provided.

To apply the matching the service must be enhanced with knowledge on how the matching can be made on a level of the meta information, so that it can be applied to the instances. If the meta information and instances are adapted to the concrete environment, this matching must also be adapted.

4.2.12 Reinforcement Learning Based Resource Allocation Service

4.2.12.1 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.

Reinforcement Learning presents a significant advantage in tackling resource allocation problems due to its inherent adaptability. Unlike traditional solution methods like heuristics and branch-and-bound algorithms, Reinforcement Learning allows for the easy introduction of new constraints and real-life changes without necessitating a complete overhaul of the underlying policy learning algorithm. This flexibility is especially beneficial in dynamic environments where constraints may evolve over time or where new considerations need to be incorporated into the allocation process. Figure 55 describes the interaction of the user with the service.

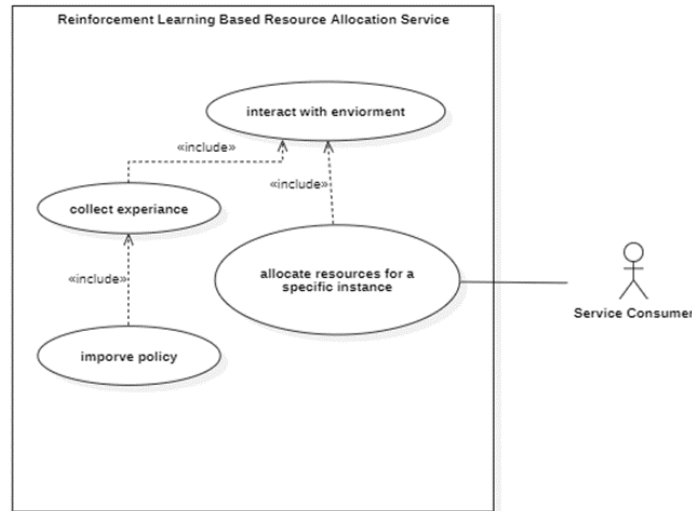


Figure 55 UML Use Case Diagram of the Reinforcement Learning Based Resource Allocation Service

4.2.12.2 Service Component

The Reinforcement Learning Based Allocation Service is a component that is not subdivided into other components. Figure 56 illustrates the interfaces between it and other components of the overall system using a component diagram.



Figure 56 Service Component for interfaces between the Reinforcement Learning Based Resource Allocation Service and DAI-DSS Architecture

4.2.12.3 Functions

The function of the Reinforcement Learning Based Allocation Service is to find a solution of a flexible Job Shop Problem Instance.

4.2.12.4 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.

4.2.12.5 Dependencies

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

4.2.12.6 Prerequisites

To use this service, the algorithm needs a description of the problem instance that will be sent via the REST-API from the orchestrator. The problem instance gets the information from the data inserted in the user interface like the available resources and the jobs to be done. That way the algorithm can map resources.

4.2.12.7 Usage

This service is trained with synthetic data based on the situation at CRF. In the current status it can be used to allocate workers to machines for the CRF use-case. It is possible to train the algorithm for further allocation problems like job-shop problems so that the algorithm can also be used for job and worker allocations.

4.2.13 Annotation Support Service

4.2.13.1 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 57 shows a UML-use-case diagram of the Annotation Support Service.

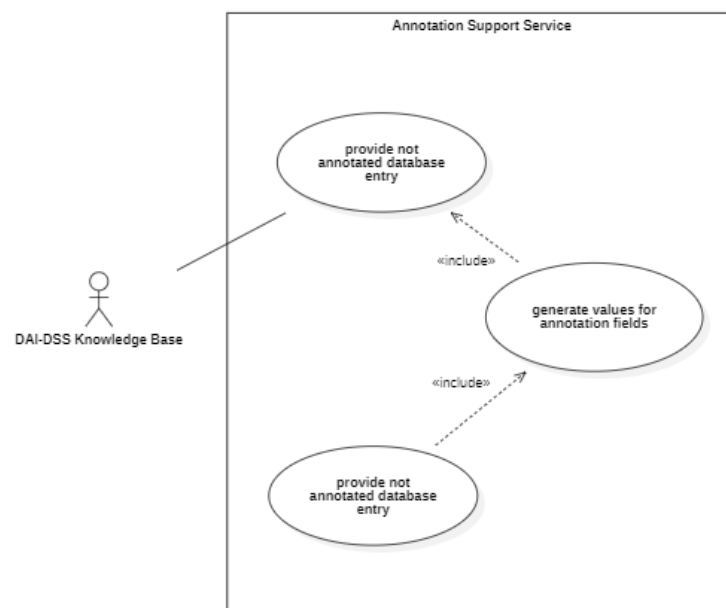


Figure 57 UML Use Case Diagram of the Annotation Support Service

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

4.2.13.2 Service Component

The Configuration Support Service is a component that is not subdivided into other components. Figure 58 illustrates the interfaces between it and other components of the overall system using a component diagram.



Figure 58 Service Component for interfaces between the Annotation Support Service and DAI-DSS Architecture

4.2.13.3 Functions

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

4.2.13.4 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.

4.2.13.5 Dependencies

The DAI-DSS Knowledge Base provides unannotated data entries.

4.2.13.6 Prerequisites

To use this service, it needs access to data to be annotated and a first set of annotated data for training. The output of the service depends on the training data of the algorithm. If the data was trained on images for a specific use-case, the algorithm can annotate images for this use-case. Thus, it must be trained for the use-case with correct data and therefore needs the annotated data.

4.2.14 Production Process Simulation with Agents

4.2.14.1 Purpose

Production process simulation with agents allows the creation of a forecast that allows the anticipation of system behaviour through modelling using Multi-Agent Systems. Simulation generates a representation of the system's behaviour by using independent but communicating entities. These entities reproduce the simulated behaviour through the interaction of agents provided with data that describe the state of the components of the production system through quantitative parameters refer Figure 59.

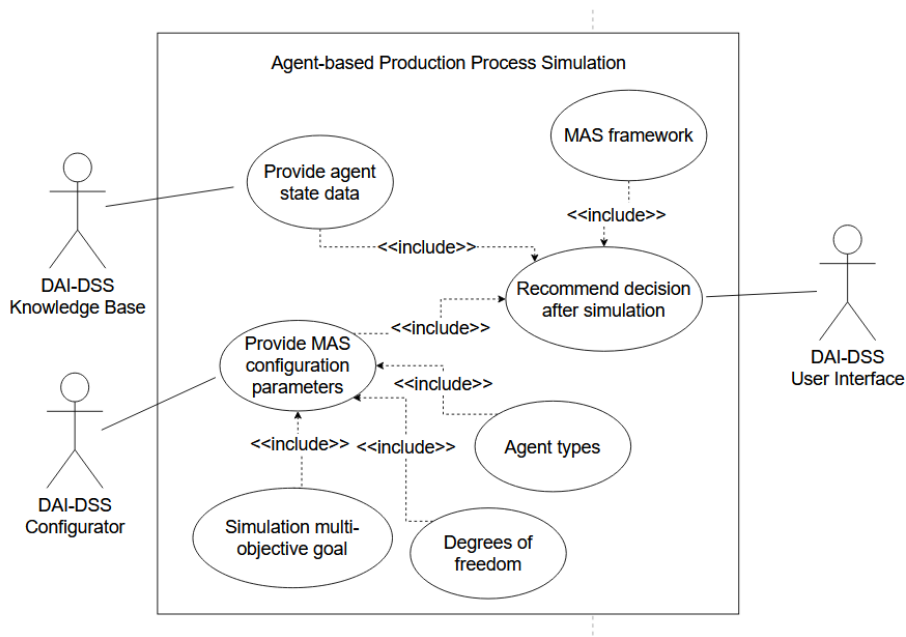


Figure 59 UML Use Case Diagram for Agent-based Production Process Simulation

4.2.14.2 Service Component

A production process simulation with agent's service can provide a forecast for the system's behaviour 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 60.

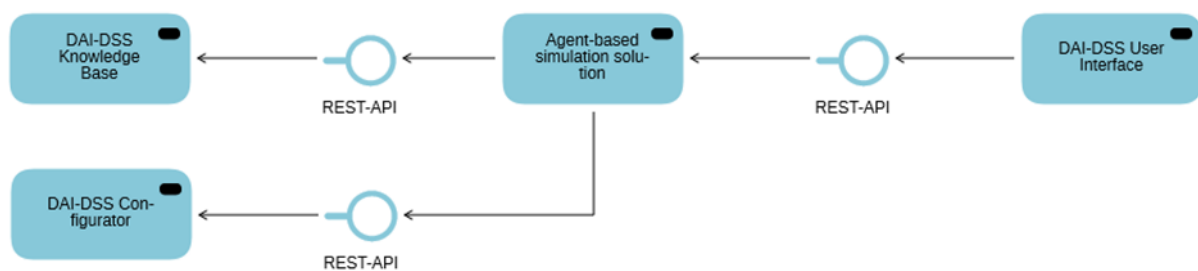


Figure 60 Service Component for production process simulation with agents

4.2.14.3 Interface and Data

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

- Communication: REST API.
- Input: Data structured in JSON format.
- Output: Comparative Agent's behaviour based on the input data.

4.2.14.4 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 behaviour whenever it can be simulated.

4.2.14.5 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 behaviour forecast of the system and provide an adequate system behaviour prevision to the DAI-DSS User Interface.

4.2.14.6 Prerequisites

For the Multi-Agent Resource Allocation service, the prerequisite is the data relevant for executing resource allocation strategies effectively according to each resource allocation case. The service requires parameters that guide the allocation decisions, such as resource availability, production demands, and priority constraints for example. These parameters can be derived from historical data on resource usage or projected requirements in evolving operational contexts.

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 61.

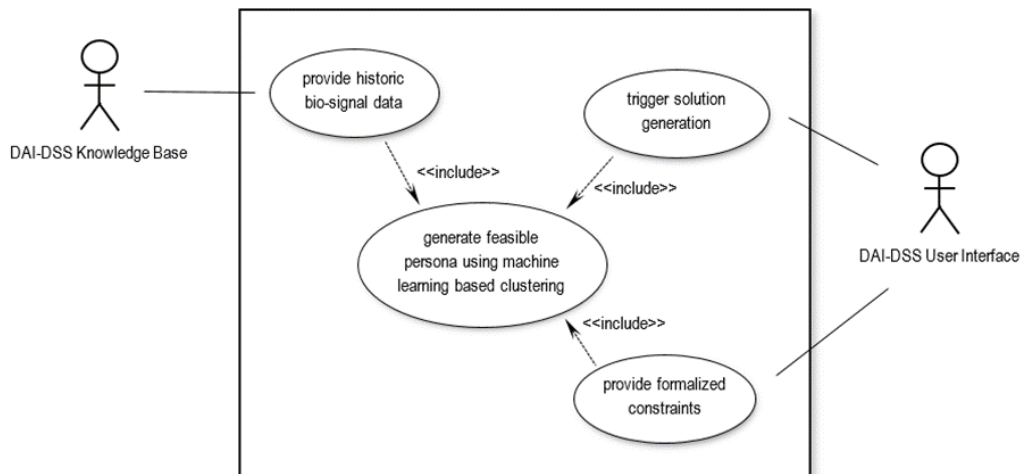


Figure 61 UML Use Case Diagram of Operator Persona Development Service

4.2.15.2 Service Component

The operator persona development service is atomic and thus not subdivided into other components Figure 62.



Figure 62 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

The service described in this section is a planned service within the FAIRWork project. Therefore, no prototype was created for it and the description contains the first introduction to the idea behind the service.

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 63.

The search for communities is either triggered by a user directly or by the orchestrator. The orchestrator is needed if the results of the community identification should be further used in decision-making. The knowledge base provided the data about the tasks and resources.

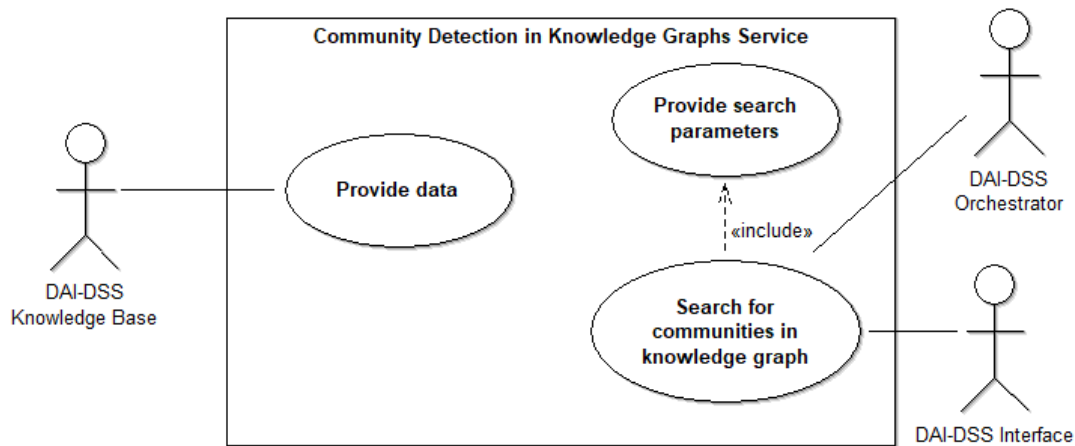


Figure 63 UML Use Case Diagram for the community detection in knowledge graphs service

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 64.

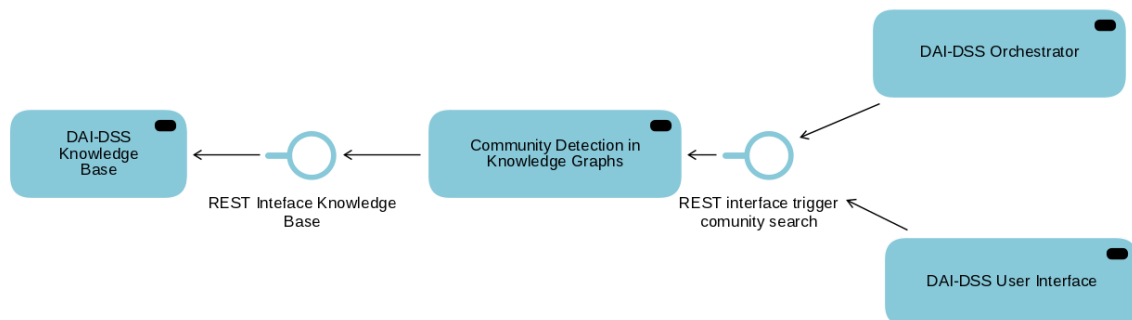


Figure 64 Service Component for community detection with knowledge graphs service.

4.2.16.3 Interface Data

Trigger Community Search: This is a REST interface which triggers the search of community in the in the available data. It consumes a JSON with the parameters needed to narrow the search. Alternatively, the data can be provided directly in an RDF-based data structure, fitting to the knowledge graph. The parameters describe the properties of the concept, based on which the communities should be build. This endpoint returns the need community with its members.

REST Interface Knowledge Base: Here the data set in which the communities should be found will be extracted.

As this service is still in the planned state, no concrete data points are defined here.

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.

4.2.16.6 Prerequisites

This service is based on a knowledge graph representation of the available information about the resources' capabilities and the needs of the task. Therefore, the metainformation regarding workers and tasks must be created within the knowledge graph, to have the fundamental structure on which communities can be detected. For using the service, a basic structure for different abstract scenarios can be created. But if the concrete production environment differs, the metainformation must be adapted.

To find similar objects the concrete tasks and resources must be added and linked to the meta information.

To find groups of similar objects, the service must be configured with a way to identify in which way the different object should be similar.

4.2.17 Production Process Simulation

The service described in this section is a planned service within the FAIRWork project. Therefore, no prototype was created for it and the description contains the first introduction to the idea behind the service.

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 65. Users can trigger the simulation of a configuration, to get the information needed to support the overall information. Additionally, the orchestrator can trigger the simulation and use the results. To use the simulation, information about the decision process must be provided, which can be done through the configuration framework and interface.

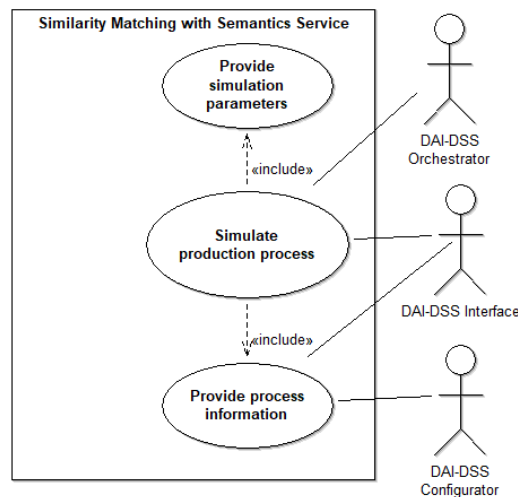


Figure 65 UML Use Case Diagram for the production process simulation service

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 corresponds 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 66.

The simulation can be triggered from the user over the interface or the orchestrator. As input for the simulation, additional real-time information, like production line configuration may be needed, which are available in the knowledge base. Additionally, the simulation and the process must be configured which can be done through the user interface, the configuration framework or a combination of both.

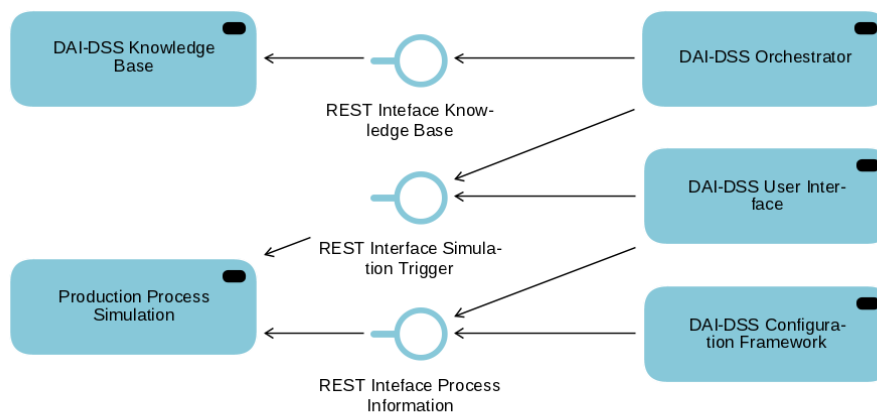


Figure 66 Service component for production process simulation service

4.2.17.3 Interface Data

Production process simulation: Here a REST interface is offered which provides information about current configuration and parameters. This information will be provided in a common format, like JSON or XML. The returned results will contain the simulation results, based on the configuration of the simulation.

Process information: This REST interface allows to provide the overall configuration and structure of the production process, so that it can then be used for the simulation.

4.2.17.4 Functions

This service component 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.

4.2.17.6 Prerequisites

The main prerequisite for simulating production processes is that a corresponding process model is created, encoded in a way that the simulation engine can execute it. This model must contain the parameters which should be evaluated within the decision. Therefore, these processes and parameters must be gathered from real production environments. Or to-be processes can be modelled with assumed parameters, to allow simulations in advance.

4.2.18 Similarity Matching with Semantics

The service described in this section is a planned service within the FAIRWork project. Therefore, no prototype was created for it and the description contains the first introduction to the idea behind the service.

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 67.

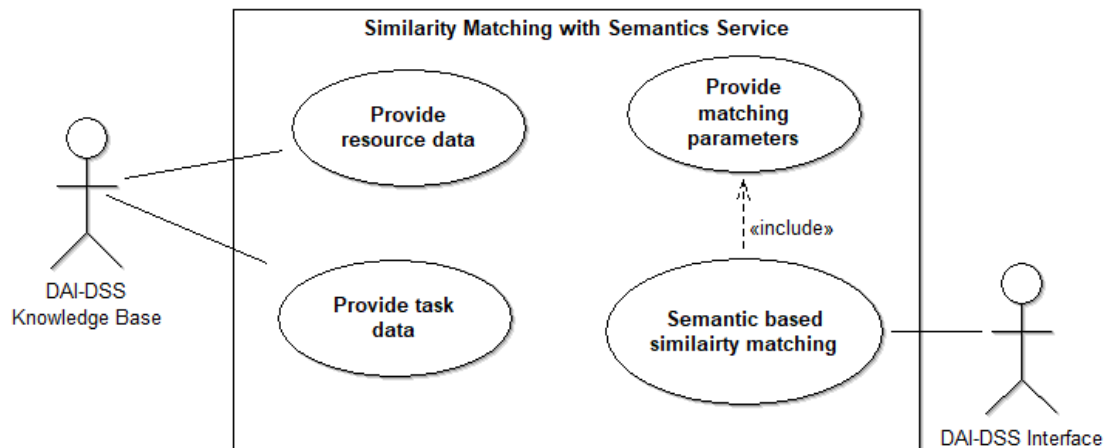


Figure 67 UML Use case diagram for the similarity matching with semantics service

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 68. The similarity matching can be triggered through the user via the interface or by the orchestrator.

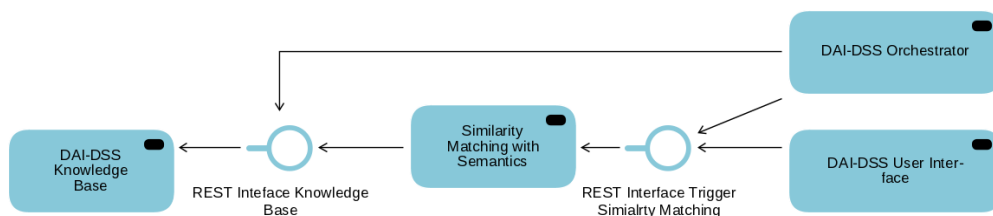


Figure 68 Service component for similarity matching with semantics service

4.2.18.3 Interface Data

Trigger Similarity Matching: This REST interface will consume its input data in an established data format, like JSON or an RDF-based data structure. For this data about for which production similar tasks should be found. Additionally, a preselection of the concepts in the production environment, which should be checked can benefit the matching. As result the list of matchings will return which can be further used in the decision-making process.

REST Interface Knowledge Base: Here the data set in which the communities should be found will be extracted.

As this service is still in the planned state, no concrete data points are defined here.

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.

4.2.18.6 Prerequisites

Information about the tasks and activities which should be matched must be provide, including their relevant attributes. This information must be collected within the production environment and encoded to fit the service.

4.2.19 Impact Assessment with Knowledge Graphs

The service described in this section is a planned service within the FAIRWork project. Therefore, no prototype was created for it and the description contains the first introduction to the idea behind the service.

4.2.19.1 Purpose

This service allows to analyse 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 69.

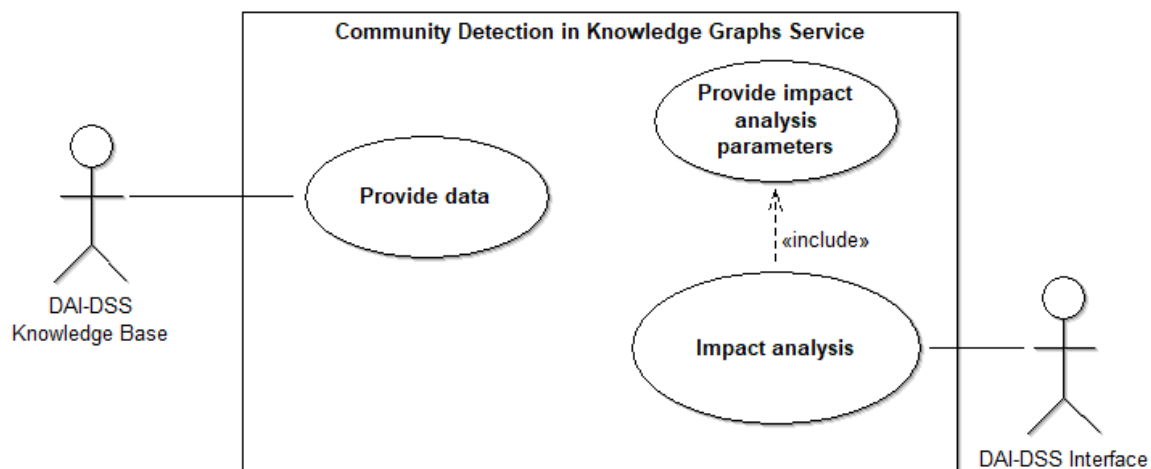


Figure 69 UML Use Case diagram for the impact assessment with knowledge graphs service

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 analyse 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 70.

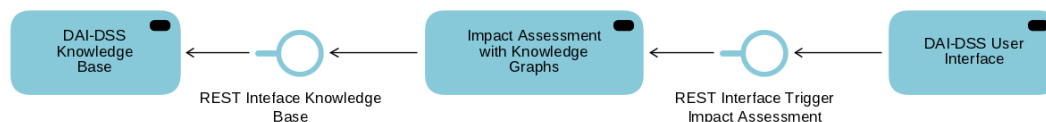


Figure 70 Service component for impact assessment with knowledge graphs service

4.2.19.3 Interface Data

Impact Assessment: The interface will be implemented as a REST interface and consume the data in JSON or an RDF-based data structure. The input parameters must contain configuration for the impact assessment and which part on the system should be analysed. The output of this service will contain the results of the analysis and depend on the provided input.

REST Interface Knowledge Base: Here the data set in which the communities should be found will be extracted.

As this service is still in the planned state, no concrete data points are defined here.

4.2.19.4 Functions

This service component offers the functionality to analyse 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.

4.2.19.6 Prerequisites

The system must be represented as a knowledge graph by a domain expert. After it is represented the impact analysis can be applied, which must be configured to groups of knowledge graph representations which are similar. Therefore, the information of the real system, which should be assessed must, must be gathered and encoded in a way that the system can understand it.

4.2.20 Semantic Search with vector embeddings

4.2.20.1 Purpose

Instead, then only searching for exact matches or variations of the query words, semantic search aims to produce more relevant results by comprehending the query's overall meaning. In the case of this service this is realized using vector embeddings. The use of this service enables the user to formulate a search using natural language and to find elements in an own knowledge base. This service can be potentially used when searching for documents and other information, where the result should be restricted to a certain document pool or database.

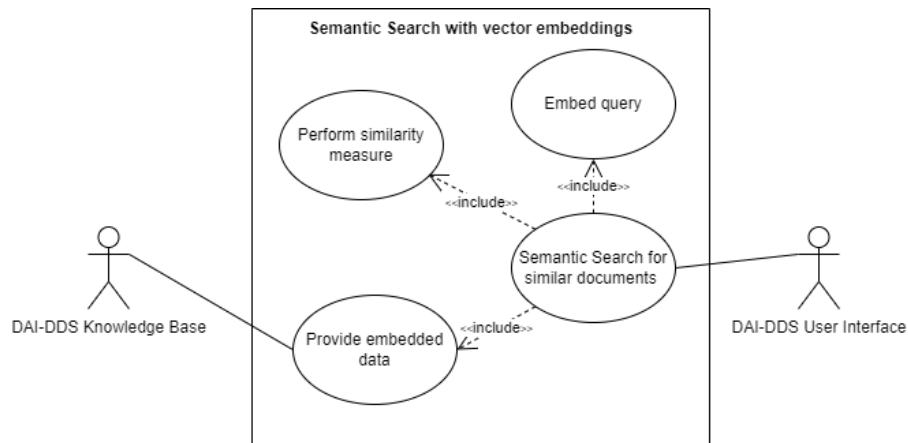


Figure 71 UML Use Case Diagram Semantic Search with vector embeddings.

4.2.20.2 Service Component

The Semantic Search with vector embedding is a query service. The user can enter a query using the DAI-DSS User Interface. The service takes care of transforming the users input into a vector with the help of an embedding model. By using similarity measures and interacting with the database (e.g. DAI-DSS Knowledge Base) this input vector is compared to other embedded documents. The DAI-DSS Orchestrator is used to manage the access and retrieval of the necessary data. The most similar vectors are presented as results using the user interface.

4.2.20.3 Interface and Data

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

- Input: User query in natural language
- Output: Based on the query the most similar documents or elements are returned to the user

4.2.20.4 Functions

The function of the semantic search is to transform a user's query (e.g. text) into a vector by using a pre-trained embedding model. The so created vector embedding is then compared with other embedded data using similarity measurements. The vectors that are closest to the query vector are returned to the user.

4.2.20.5 Dependencies

The service is dependent on third-party pre-trained embedding models to transform the data and the users query into vector representations. Furthermore, the embedded documents need to be stored to be accessed when the service is triggered, which can result in dependencies with the DAI-DSS Knowledge Base or third-party vector databases.

4.2.20.6 Prerequisites

Depending on the specific characteristics of the search task, a suitable embedding model must be chosen to generate the vectors. As this service works by comparing the vector representing the users query with the other vectors in the vector space, it is beneficial to the quality of the service results if a large number of embedded documents are available.

4.2.21 Automated Test Building Support with Hybrid Filtering

4.2.21.1 Purpose

The purpose is to compare data on finalized projects to the user-defined input parameters of ongoing or planned projects, to aid the engineer during the design phase of the production process by providing possible configurations and guidance on feasibility. It pursues a hybrid filtering approach, combining both content- and context-based approach by including available information on the ongoing or planned project and current equipment and resources. Figure 72 shows the use case diagram for the use case.

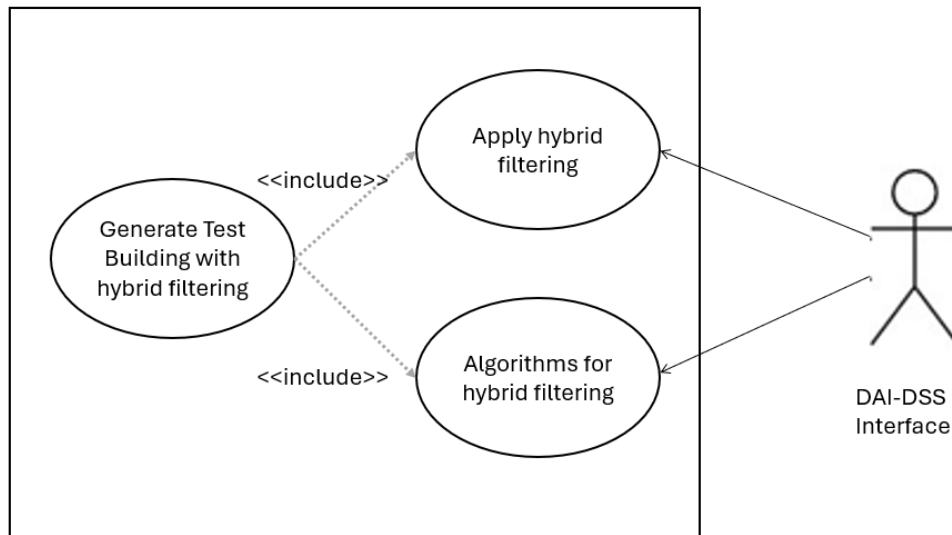


Figure 72 UML Use Case Diagram for Support of Hybrid Filtering

4.2.21.2 Service Component

The filtering 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 73. It should be noted that the worker assignment does not create any visualizations besides the ranking.

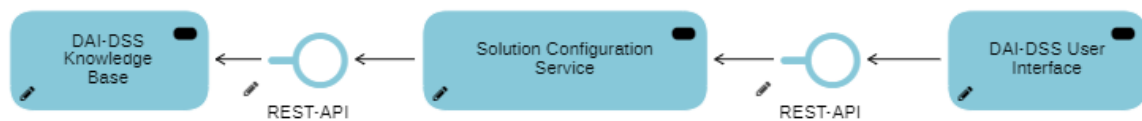


Figure 73 Service Component for Automated Test Building Support with Hybrid Filtering

4.2.21.3 Interface and Data

Filtering of comparable projects

- Interface: REST-API.
- Input: engineer's search query
- Output: concrete listing of similar projects, sorted by similarity

4.2.21.4 Functions

The service has 2 functions:

1. Process the given input and in case of textual or visual information (which is given by descriptions or plans) extract usable features.
2. Identify comparable projects by filtering depending on the input data and present them to the engineer ranked by the similarity.

4.2.21.5 Dependencies

The DAI-DSS Knowledge Base stores resources, including order details, keywords, product descriptions, listing of work steps and assembly plans.

4.2.21.6 Prerequisites

This service is based on the proper representation of available information on projects to find similar items. Therefore, the information must be gathered and encoded in a way that the system can understand it. Afterwards the similarity indices of the project pairs need to be calculated. Since this service operates by analysing and comparing the vector representation of the user query, having a large variety of successfully completed products enhances the quality of the service results.

5 FUNCTIONAL CAPABILITIES

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
- Status : Done, In progress, Not implemented

ID	Component	Requirement	Description	Status
R-001	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.	In Progress
R-002	Knowledge base	Properties	The system shall be able to store user defined properties to any breakdown element (folder, document, data set, part, etc.).	Done
R-003	Knowledge base	Sensor data	The system shall be able to represent sensors and store measured readings.	Done
R-004	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.	Done
R-005	Knowledge base	Data storage	The system shall be able to store structured and unstructured data files also with associated properties	Done
R-006	Knowledge base	Machine learning results	The system shall be able to store results from machine learning algorithms.	Done
R-007	Knowledge base	Data exchange	All data in the system shall be retrievable through a REST API interface.	Done

ID	Component	Requirement	Description	Status
R-008	Orchestrator	Execution across clouds	It must capably execute services that are deployed across different clouds	In progress
R-009	Orchestrator	Multi-agent/ Workflow Orchestration	The system shall recognize the best approach for processing the optimization/simulation task.	In progress
R-010	Orchestrator	Support to decision-making	The system shall provide a plural number of recommendations to decision-making whenever those options are available.	In progress
R-011	Orchestrator	Multi-agent resource allocation	The system shall be able to provide a valid and democratic approach for proper allocation of processing tasks.	In progress
R-012	Service Catalogue	Standardized Interface Structure	The interfaces of the services should follow a common logic and structure to support easier integration and reuse.	In progress
R-013	Service Catalogue	Common Deployment Environment	There should be a common deployment procedure and environment to support an easy and automated reuse of the services	Not implemented
R-014	Service Catalogue	Common Service Framework	A common framework for offering and using the services should be used to ease the reuse of the system.	In progress
R-015	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	In progress
R-016	User interface	Modularity	The use of front end Microservices should enable the possibility to create use case specific user interfaces	Done
R-017	User interface	Standardized user interface	Using well known MS teams	In progress

ID	Component	Requirement	Description	Status
R-018	User interface	Usability	The user face should be easy to use and have a clear layout by using well known design features	In progress
R-019	User interface	Communication with Orchestrator	The user should be able to retrieve data and call services via the orchestrator	Done
R-020	User interface	Configurable	The widgets displayed in the user interface should be configurable using configuration files created by the configurator component	In progress
R-021	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.)	Not implemented
R-022	Configurator	Model-awareness	Enables the creation of decision models using different model environments	In progress
R-023	Configurator	Communication with user interface	Enables creation of configuration files for user interfaces	In progress
R-024	Configurator	Communication with orchestrator	Enables creation of configuration files containing information about services or workflows and the deployment of them	In progress
R-025	Configurator	Communication with knowledge base	Configuration files should be storable in the knowledge base for further use	Not implemented
R-026	Configurator	Provide Assessment service	Feedback should be collectable from the “real world” and evaluated to adjust models and services when needed	Not implemented
R-027	AI Enrichment	Providing AI services	The component shall provide data-driven models for chosen agents in given use cases	In progress

ID	Component	Requirement	Description	Status
R-028	AI Enrichment	Execution of models	The component shall be able to execute developed models	In progress
R-029	AI Enrichment	Data exchange	The component shall be able to provide results of developed models in JSON format	In progress
R-030	External Data Asset Market	Provide external data set	The component provides a searchable data catalogue with external data sets.	In progress
R-031	External Data Asset Market	Provide external data set	This component provides a user interface to find, search and described data assets.	In progress
R-032	External Data Asset Market	Provide external data set	The component provides an API for integrating data sets in in 3 rd party tools	In progress
R-033	Intelligent Sensor Box	Provide personal data	The components provide data from human-centred worker profiles in an anonymized way.	In progress
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).	In progress

6 SECURITY

This section describes the outcome of the initial analysis of the required security functions or services within the FAIRWorks architecture carried out as part of WP4. To this end, a set of minimum-security controls for the decision-making system is outlined.

In the first area of interest, it is need to take care of securing the individual components. This is e.g. the knowledge base, AI enrichment infrastructure, configurator etc. The following must be implemented for these individual components:

Authentication and authorization: Strong authentication methods such as OAuth2 or OpenID Connect can be used to restrict access to the individual component. Depending on the component, role-based access control (RBAC) can also be implemented. This is used to assign users only the authorizations they need.

Data encryption: In addition, the data between individuals should be protected with a strong encryption algorithm. If there are individual web services, the data between the web service and its clients can be encrypted using TLS/SSL.

Network security: Firewalls and access control lists (ACLs) should be used to restrict access to individual components. If necessary, segmentation of the network is required. In this way, particularly sensitive data, e.g. from wearables, can be isolated from the public areas.

In the following sections highlights the implemented or planned security measures are outlined to secure the DAI-DSS components.

6.1 DAI-DSS Knowledge Base

In the digital era, where data breaches and unauthorized access can lead to significant losses and privacy violations, securing a robust data management system like EDMtruePLM is important. Knowledge Base (EDMtruePLM), standing at the core of the DAI-DSS architecture, not only stores critical lifecycle data of physical assets but also integrates with various components through its REST API capabilities. Thus, ensuring the confidentiality, integrity, and availability of the data within this platform is essential. This section discusses the approaches to information security and authentication mechanisms that are employed to safeguard data within the Knowledge Base environment.

Multi-Layered Security Approach

Knowledge Base deployed on publicly accessible server employs a multi-layered security strategy to protect against a wide array of threats. This strategy encompasses both physical and logical layers to ensure comprehensive coverage. Physically securing the server environment where knowledge base operates is the first line of defence, involving controlled access to hardware and network infrastructure with the help of Botshield. Logically, the application itself is fortified using the cybersecurity practices for regular vulnerability assessments of software. As an Intrusion-Detection / Intrusion-Prevention-System (IDS/IPS), Botshield³³ monitors the data traffic, detects unusual activities, and locks out suspicious IP addresses via firewall to protect the system against many kinds of abusive attacks.

³³ <https://botshield.de/en/>

By implementing uamami.is³⁴ for monitoring the <https://demo.jotne.com/EDMtruePLM>, provides valuable insights into visitor activity, including their geographical location, browsing behaviour, and frequency of access Figure 74. This comprehensive monitoring enables the identification of suspicious patterns or anomalies, allowing for timely intervention against potential security threats such as malicious bot attacks or unauthorized access attempts. Moreover, uamami.is facilitates the implementation of robust security protocols by providing real-time alerts and customizable access controls, empowering administrators to swiftly respond to any detected threats and fortify the website's defences.

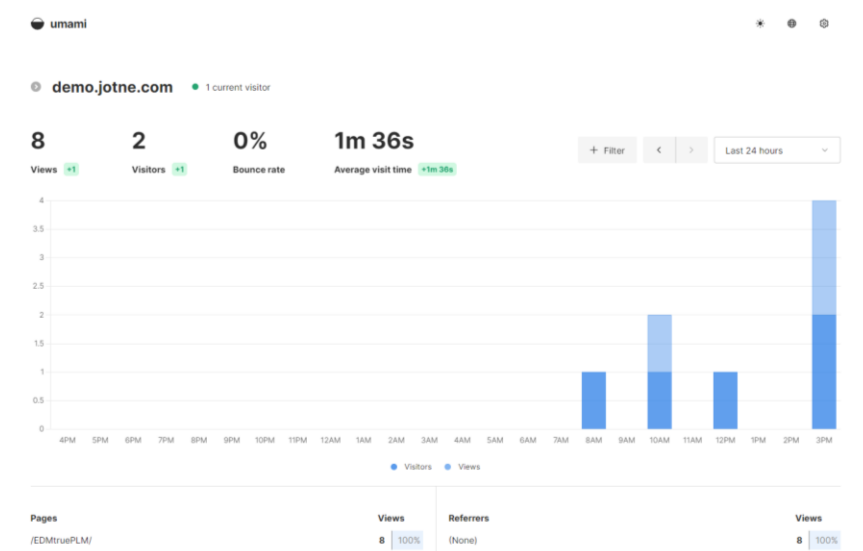


Figure 74 Monitoring the website using uamami.

Encryption and Data Protection

Data in transit and at rest within Knowledge Base are encrypted using industry-standard protocols. For data in transit, Secure Socket Layer (SSL)/Transport Layer Security (TLS) encryption is implemented, ensuring that data exchanged between the client interfaces and the server is not intercepted or tampered. For data at rest, data within the EDM database against unauthorized access. The database itself is password protected.

Authentication and Access Control

Knowledge Base integrates robust authentication mechanisms to verify the identity of users with the platform. These mechanisms include traditional username and password combinations, along with support for more secure method such as two-factor authentication (2FA) Figure 75. Access control is managed through a comprehensive role-based access control system, where permissions are granted based on the roles assigned to users or systems Figure 76. This granularity ensures that only authorized entities can access or modify data, adhering to the principle of least privilege.

³⁴ <https://umami.is/docs>

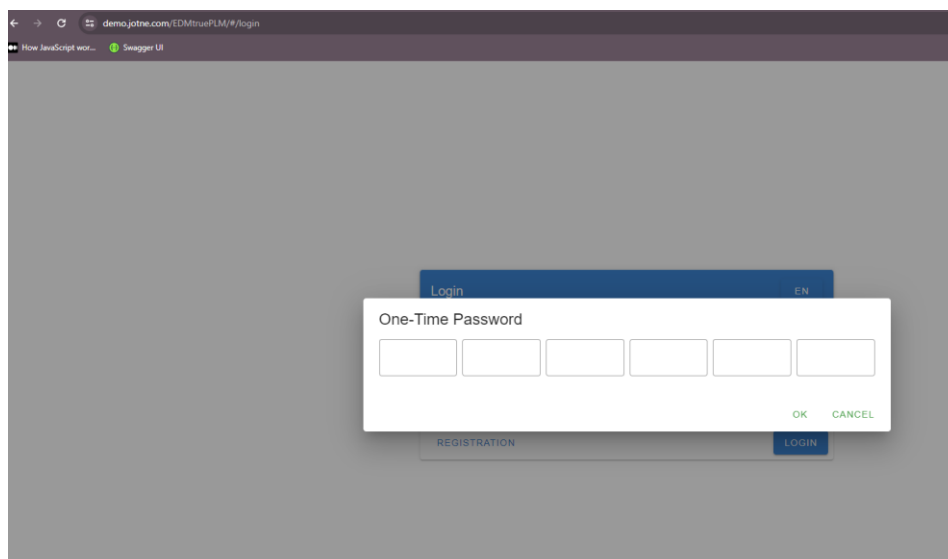


Figure 75 2FA Authentication for Knowledge Base

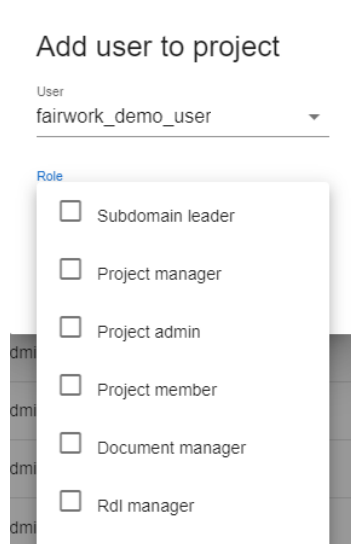


Figure 76 Role based access control in the Knowledge Base

REST API Security

The REST API of EDMtruePLM, being a critical interface for programmatic access, is secured using best practices for API security. This includes employing secure token-based Figure 77 authentication to prevent unauthorized access of the data. The API endpoints are designed to ensure that they expose only the necessary information and functionality, reducing the attack surface.

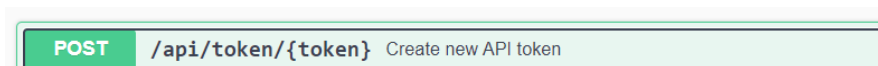


Figure 77 REST Method for token generation.

Continuous Monitoring and Incident Response

To maintain a secure Knowledge Base environment, continuous monitoring of system and network activity is vital. This involves logging and analysing access requests, system modifications, and data transactions to detect and

respond to potential security incidents promptly. This is done with the help of Apache Tomcat webserver where the API requests made to the web server are logged.

6.2 DAI-DSS Orchestrator

The DAI-DSS Orchestrator currently utilizes its individual authentication system, rather than a centralized one. To enhance security within the workflow engine, basic authentication method that require a username and password under an HTTPS connection, is used. This approach is adequate for the current prototype stage. However, as the system matures, the integration of centralized authentication may be considered to further streamline and secure access control.

6.3 DAI-DSS AI -Enrichment infrastructure

The initial version does not provide any additional security features for the individual services. The orchestrator acts as a centralized control mechanism, facilitating the invocation of all services within the architecture. It is designed so that individual services do not require separate security measures. Instead, security is consolidated at the level of the deployed virtual machine, which is comprehensively secured. This approach simplifies security management by concentrating protection efforts at the virtual machine, ensuring that services operate in a secure environment without the need for individual security configurations. However, for future use, security measures must also be implemented for the individual web services.

6.4 Outlook

This section outlines the security measures implemented for the main components in the initial phase. In the subsequent phase, the security concept will be reevaluated and enhance it with additional measures as needed. Given that the decision system is web-based, the DAI-DSS implementation will use the Web Security Testing Guide (WSTG) ³⁵ as a reference standard. The WSTG project is recognized as a leading resource in cybersecurity testing for web application developers and security experts. It represents a collective endeavour by cybersecurity professionals and volunteers to establish a comprehensive framework of best practices for assessing the security of web applications and services.

³⁵ <https://owasp.org/www-project-web-security-testing-guide/latest/>

7 ARCHITECTURE TESTING AND REPORTING

The DAI-DSS architecture testing and reporting will focus on evaluating the interfaces and connections, both internal and external, of components. The document acts as a detailed log, recording which DAI-DSS component was tested, along with the outcomes of each test, noting whether they passed or failed. The primary purpose of the test report summary is to document the procedures and outcomes of these tests comprehensively.

In the following the test details for each component is reported.

7.1 DAI-DSS Knowledge Base

In the Table 5 internal and external testing information related to the component has been outlined.

Table 5 Testing information related to Knowledge Base

Name of test	Short description about the test	Internal or External	Manual or Automatic	Passed or Failed
Start EDM Server	This test ensures that the EDM database server can be started successfully. It verifies the functionality of starting the server, which is crucial for the application to function properly	Internal	Manual	Passed
Stop EDM Server	This verifies that the server can be stopped gracefully. This to ensure that resources are released properly and to avoid any potential issues during shutdown	Internal	Manual	Passed
Configure tomcat	This test ensures that the Tomcat web service can be started and stopped successfully. It's a part of system testing, specifically testing the functionality of the web server component.	Internal	Manual	Passed
Database Backup	This involves automated testing to ensure that the process of backing up the database works as expected. It's a critical aspect of data integrity and disaster recovery planning	Internal	Automated	Passed
User creation	Creating a new user manually is a form of functional testing, ensuring that the user	Internal	Manual	Passed

Name of test	Short description about the test	Internal or External	Manual or Automatic	Passed or Failed
	management functionality works correctly.			
2FA (Two-Factor Authentication)	Testing 2FA functionality in the Microsoft Authenticator app involves verifying the security feature of the application. It ensures that the authentication process is robust and secure.	Internal	Manual	Passed
Project Creation and CRUD operations	Testing CRUD (Create, Read, Update, Delete) operations manually and via REST API involves functional testing of the application's data manipulation capabilities. It ensures that users can perform basic operations on projects effectively.	Internal	Manual	Passed
Interfacing with DAI-DSS Configurator	These tests involve integration testing. This is to ensure that the KB can interact effectively with external Olive microservice framework for exchanging data	External	Manual	Passed
Interfacing with DAI-DSS UI	Enabling Knowledge Base accessibility via Iframe involves testing the application's accessibility and integration with external systems or platforms. It ensures that the application can be embedded within other environments.	External	Manual	Passed
Interfacing with DAI-DSS AI Orchestrator	Testing the working flow for retrieving data from Knowledge base	External	Manual	Passed
Interfacing the with AI Enrichment services	Data upload and download using REST API in and out to the knowledge base	External	Manual	Passed

7.2 DAI-DSS User Interface

In the Table 6 internal and external testing information related to the component has been outlined.

Table 6 Testing information related to User Interface.

Name of test	Short description about the test	Internal or External	Manual or Automatic	Passed or Failed
Internal functionality of the UI component	Testing if UI elements (e.g. buttons) work as intended and results are displayed in a readable, understandable way	Internal	Manual	Passed
Displaying workflow responses in UI components	Services or workflows are triggered by interacting with the UI and the results are displayed in different formats	External	Manual	Passed
Uploading UI components to the component pipeline	The designed UI component is uploaded to the pipeline in form of a zip-file to use it further in the configurator	External	Manual	Passed

7.3 DAI-DSS Orchestrator

In the Table 7 internal and external testing information related to the component has been outlined.

Table 7 Testing information related to Orchestrator.

Name of test	Short description about the test	Internal or External	Manual or Automatic	Passed or Failed
Communication with Knowledge Base	APIs requests are tested as part of a workflow for retrieving information from different breakdown elements and aggregated properties in the Knowledge Base. It is expected that the workflow returns the present information at the Knowledge Base in form of a service response in JSON format	External	Manual	Passed
Communication with AI service	API endpoint for triggering a rule-based allocation service was incorporated into a workflow	External	Manual	Passed

Name of test	Short description about the test	Internal or External	Manual or Automatic	Passed or Failed
	and a errorless service response is expected			

7.4 DAI-DSS Configurator

In the Table 8 internal and external testing information related to the component has been outlined.

Table 8 Testing information related to Configurator.

Name of test	Short description about the test	Internal or External	Manual or Automatic	Passed or Failed
Configuration of UI	Test if UI components can be selected and arranged using a wizard in the configurator to form a deployable user interface	External	Manual	Passed
Deployment of UI components	After the UI components are selected and arranged, they are deployed and accessible via a link in the web browser	Internal	Manual	Passed
Connection of Configuration environment to other platforms	Creation of so called 'Tiles' in the configuration platform that allow to access external platforms (e.g. modelling environment, GUI of Knowledge base)	External	Manual	Passed
Adding service configurations and service bundles	Configuration of services and service bundles by using OLIVE specific JSON	Internal	Manual	Passed
Creation of decision models using BPMN modelling platform	Modelling of decision paths in ADONIS	Internal	Manual	Passed
Configuration Model-enabled Rule Service	The configuration of the decision knowledge in form of rules should be enabled through DMN models. Therefore, the models were created used to configure the service and the then the endpoint was tested.	Internal	Manual	Passed

7.5 DAI-DSS AI Enrichment

In the Table 9 internal and external testing information related to the component has been outlined.

Table 9 Testing information related to AI Enrichment.

Name of test	Short description about the test	Internal or External	Manual or Automatic	Passed or Failed
AI-Services-Unit-Tests	The AI services provided in the implemented with python in the Service Catalogue are subjected to unit testing to ensure their functionality, reliability, and compatibility. Unit testing is performed on each individual component or service implemented in Python. The objective of these tests is to verify the correctness of the implemented functionality, identify potential defects, and ensure that the services meet the expected specifications.	Internal	Automatic	Will be tested for each python implemented unit individually
Connectivity Test Model-enabled Rule Service	The test was done to test the connectivity between the service and orchestrator. The workflow-based orchestrator thereby called the service and used the results further.	External	Manual	Passed
Usage of the Model-enabled Rule Service	The service was deployed and tested if the parameter defined for the case can be used and if the return the expected result	Internal	Manual	Passed

7.6 DAI-DSS Intelligent Sensor Box

In the Table 10 internal and external testing information related to the component has been outlined.

Table 10 Testing information related to Intelligent Sensor Box

Name of test	Short description about the test	Internal or External	Manual or Automatic	Passed or Failed
Connectivity Test Model-enabled	Test the connectivity	Internal	Manual	Passed

API of ISB	Test the individual APIs	Internal	Manual	Passed
------------	--------------------------	----------	--------	--------

7.7 Outlook

In the initial phase of our project, testing was performed on each individual component to assess functionality and identify any issues. As project move towards the final iteration, the testing strategy will evolve to integrate a more centralized approach for monitoring and reporting. Each possible component will be configured to report its operational activities directly to the orchestrator's logging component. This will be primarily achieved using log files tailored to capture specific data relevant to each component's role.

The Orchestrator will act as the central hub within this system. It will aggregate the log data collected from all connected components, providing a comprehensive overview of the system's operational status. This centralized log data will then be managed and routed as necessary to facilitate real-time monitoring and troubleshooting.

To enable this logging mechanism, a REST API will be implemented. This API will allow for efficient communication between the components and the orchestrator, ensuring that log data is transmitted seamlessly and securely. This setup not only enhances the ability to track system performance and issues but also improves the overall reliability and maintainability of the DAI-DSS architecture.

8 SUMMARY AND CONCLUSION

The second iteration "D4.1.1 – FAIRWork Architecture and Initial Documentation and Test Report" outlines the development and enhancement of the Democratic AI-based Decision Support System (DAI-DSS) as part of the FAIRWork project. This iteration serves as an update on the progression and current state of the DAI-DSS architecture described in deliverable D4.1, elaborating on the enhancements in component architecture, functionalities, capabilities, and the AI services along with security and testing implementations.

Key Performance Indicators: The document updates Key Performance Indicators (KPIs) for use case scenarios related to FLEX "Automated Test Building", "Worker Allocation", and "Machine Maintenance after Breakdown", and similarly for CRF "Workload balance Process", "Delay of Material Process", "Quality Issues Process" aligning them with the overarching FAIRWork project goals. These KPIs serve as benchmarks for evaluating the system's effectiveness in addressing the decision challenges.

Technical Specifications: Detailed specifications of software and hardware requirements for DAI-DSS components have been outlined, ensuring that all technical partners are aware of the infrastructure needed to support the system.

Additionally, the document discusses the high-level data model (ISO 10303 AP239) utilized in the Knowledge Base for data storage. It further details the creation of Reference Data designed to encapsulate information that is specific to the business contexts of the FAIRWork cases for CRF and FLEX.

AI Services Classification: AI services have been categorized into three stages of development - Initial Prototype, Development in Progress, and Planned for Implementation. This classification helps the use case owners to understand the readiness level of each service and its applicability to their needs. In total, there are seven services which have been implemented out of which three services were integrated to DAI-DSS orchestrator to support the workload balance scenario which can be invoked with the help of UI.

Security and Data Protection: The existing security practices for DAI-DSS components have been detailed based on preliminary analyses. These practices include the implementation of SSL certificates, two-factor authentication, and role-based access control within the Knowledge Base. The orchestrator is safeguarded through username and password protection. Additionally, the importance of securing individual AI services has been recognized, with plans to enhance protections as the project advances. Future evaluations will assess the effectiveness of these implemented security measures using the WSTG standard.

Testing and Validation: Methods for testing the architecture's integrity and functionality are described, focusing on assessing interactions between components and their integration. This ensures that the system operates seamlessly and maintains high standards of quality and reliability.

The final iteration of this deliverable, D4.3 Final DAI-DSS Prototype, Documentation and Test Report is expected to further refine these components and expand the system's capabilities, guided by ongoing testing and feedback from the implementation phases. This document not only serves as a crucial guide for current stakeholders but also sets the stage for the final prototype documentation and testing report, contributing significantly to the FAIRWork project's goals.

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

