Ref. Ares(2024)2353972 - 29/03/2024

D2.2 – AIDEAS VIEWPOINTS V2

WP2 – Design: AIDEAS Framework Design

Co-funded by the European Union

Document Information

GRANT AGREEMENT NUMBER	101057294	ACRONYM	AIDEAS
FULL TITLE	AI-Driven Industrial Equipment Product Life Cycle Boosting Agility, Sustainability and Resilience		
START DATE	01-10-2022	DURATION	36 months
PROJECT URL	https://aideas-project.eu/		
DELIVERABLE	D2.2 – AIDEAS viewpoints v2		
WORK PACKAGE	WP2 – Design: AIDEAS Framework Design		
DATE OF DELIVERY	CONTRACTUAL 31-Mar-2024	ACTUAL	29-Mar-2024
NATURE	Report	DISSEMINATION LEVEL	Public
LEAD BENEFICIARY	CERTH		
RESPONSIBLE AUTHOR	Miguel A. Mateo-Casalí (UPV), Raul Poler (UPV)		
CONTRIBUTIONS FROM	1-CERTH, 2-UPV, 3-UNINOVA, 4-IKERLAN, 5-TAU, 6-UNIVPM, 7-ITI, 8-CESI, 9-IANUS, 10-XLAB, 11-FBA, 13-PAMA, 14-D2TECH, 15-BBM, 16-MULTISCAN		
TARGET AUDIENCE	1) AIDEAS Project partners; 2) industrial community; 3) other H2020/ Horizon Europe funded projects; 4) scientific community		
DELIVERABLE CONTEXT/ DEPENDENCIES	This document is D2.2 “AIDEAS viewpoints v2”. Its relationship to other documents is as follows: D2.1 Title: “Reference Architecture & Viewpoints v1”.		
EXTERNAL ANNEXES/ SUPPORTING DOCUMENTS	None		
READING NOTES	None		
ABSTRACT	This document is part of the previous AIDEAS deliverable titled "Reference Architecture & Viewpoints v1". This document goes deeper into the design of the solutions, considering the different viewpoints defined in the previous version, as well as the defined architecture. All updates with respect to the previous deliverable are reflected in this document.		

Document History

VERSION	ISSUE DATE	STAGE	DESCRIPTION	CONTRIBUTOR
v0.1	09-Feb-2024	ToC	Table of Contents	UPV
v0.2	27-Feb-2024	Working File	Merged all documentation from the different tasks	UPV and All solution Providers
v0.3	28-Feb-2024	Working File	Development of the introduction, conclusion, references, etc.	UPV
v0.4	29-Feb-2024	Working File	Layout of the deliverable.	UPV
v0.5	04-Mar-2024	Working File	Add improvements to some solutions	UPV
v0.6	05-Mar-2024	Draft	1st Draft available for internal reviews	UPV
v0.7	06-Mar-2024	Review	Conduction of Internal Reviews and plagiarism check	TAU, UNINOVA, UPV
v0.8	23-Mar-2024	Draft	2nd Draft addressing the comments by the reviewers	UPV
v1.0	29-Mar-2024	Fina Doc	Quality check and issue of final document	CERTH

Disclaimer

Copyright message

TABLE OF CONTENTS

EXECUTIVE SUMMARY 19
DOCUMENT STRUCTURE 20
1. INTRODUCTION 21
2. METHODOLOGY 22
2.1. BUSINESS VIEWPOINT APPROACH FOR AIDEAS 22
2.1.1 Business Viewpoint Concepts 22
2.1.2 Business Viewpoint Methodology 22
2.1.3 Survey (Form) Design 23
2.2. USAGE VIEWPOINT APPROACH FOR AIDEAS 24
2.2.1 Usage Viewpoint Model 25
2.3. FUNCTIONAL VIEWPOINT APPROACH FOR AIDEAS 26
2.3.1 RA Domain requirements 27
2.3.2 Data structure: Input and Output Data 28
2.3.3 Data structure of each solution 29
2.3.4 Hardware and Software requirements 30
2.3.5 Solutions Lifecycle 31
2.4. IMPLEMENTATION VIEWPOINT APPROACH FOR AIDEAS 32
2.4.1 AIDEAS Solutions Architecture 33
2.4.2 Implementation Components 34
2.4.3 Technical Description of the Components 34
3. AIDEAS BUSINESS VIEWPOINTS 36
3.1. STAKEHOLDER IDENTIFICATION 36
3.2. STAKEHOLDER'S INTEREST IN AIDEAS SUITES AND TOOLKITS 36
3.3. AIDEAS INDUSTRIAL EQUIPMENT DESIGN SUITE 37
3.3.1 Value Proposition / Pilots' point of view 37
3.3.2 External Stakeholders' Point of view: 37
3.4. AIDEAS INDUSTRIAL EQUIPMENT MANUFACTURING SUITE 38
3.4.1 Value Proposition / Pilots' point of view 38
3.4.2 External Stakeholders' Point of view: 38
3.5. AIDEAS INDUSTRIAL EQUIPMENT USE SUITE 39
3.5.1 Value Proposition / Pilot's point of view 39
3.5.2 External Stakeholders' Point of view: 39
3.6. AIDEAS INDUSTRIAL EQUIPMENT REPAIR-REUSE-RECYCLE SUITE 40
3.6.1 Value Proposition / Pilot's point of view 40
3.6.2 External Stakeholders' Point of view: 41
3.7. AIDEAS MACHINE PASSPORT 41
3.7.1 Value Proposition / Pilot's point of view 41
3.7.2 External Stakeholders' Point of view: 42
3.8. AIDEAS BUSINESS VIEWPOINTS CONCLUSIONS 42
4. AIDEAS USAGE, FUNCTIONAL AND IMPLEMENTATION VIEWPOINT 43
4.1. MACHINE DESIGN OPTIMISER – AI^MDO 43

4.1.1 Usage Viewpoint ... 43
4.1.2 Functional Viewpoint ... 44
4.1.2.1 Data structure of AI^{MDO} ... 44
4.1.2.2 AI^{MDO} Software Requirements ... 45
4.1.2.3 AI^{MDO} Lifecycle ... 45
4.1.2.4 Objects ... 45
4.1.2.5 Description - Login ... 45
4.1.2.6 Description – Optimisation Configuration ... 46
4.1.2.7 Description – Perform Optimisation ... 46
4.1.3 Implementation Viewpoint ... 47
4.1.3.1 AI^{MDO} Implementation Components ... 48
4.1.3.2 Technical Description of AI^{MDO} Components ... 48
4.2. MACHINE SYNTHETIC DATA GENERATOR – AI^{MDG} ... 50
4.2.1 Usage Viewpoint ... 50
4.2.2 Functional Viewpoint ... 52
4.2.2.1 Data structure of AI^{MDG} ... 52
4.2.2.2 AI^{MDG} Software Requirements ... 54
4.2.2.3 AI^{MDG} Lifecycle ... 55
4.2.2.4 Objects ... 55
4.2.2.5 Description – Login: ... 55
4.2.2.6 Description – Design Space Configuration: ... 56
4.2.2.7 Description – Manage Batch: ... 56
4.2.2.8 Description – Manage Model: ... 57
4.2.3 Implementation Viewpoint ... 57
4.2.3.1 AI^{MDG} Implementation Components ... 58
4.2.3.1 Technical Description of the AI^{MDG} Components ... 58
4.3. CAX ADDON – AI^{CAx} ... 60
4.3.1 Usage Viewpoint ... 60
4.3.2 Functional Viewpoint ... 61
4.3.2.1 Data structures of AI^{CAx} ... 62
4.3.2.2 AI^{CAx} Software Requirements ... 64
4.3.2.3 AI^{CAx} Lifecycle ... 65
4.3.2.4 Objects ... 65
4.3.2.5 Description – Login: ... 66
4.3.2.6 Description – Create digital twin: ... 66
4.3.2.7 Description – Start simulation: ... 66
4.3.2.8 Description – Retrieve simulation results: ... 67
4.3.3 Implementation Viewpoint ... 67
4.3.4 AI^{CAx} Implementation Components ... 67
4.3.5 Technical Description of AI^{CAx} Components ... 68
4.4. PROCUREMENT OPTIMISER – AI^{PO} ... 68
4.4.1 Usage Viewpoint ... 68
4.4.2 Functional Viewpoint ... 70
4.4.3 Data structure of AI^{PO} ... 70
4.4.4 AI^{PO} Hardware Requirements ... 71

4.4.5 AI^{PO} Software Requirements ... 72
4.4.6 AI^{PO} Lifecycle ... 72
4.4.6.1 Objects ... 73
4.4.6.2 Description – Login: ... 73
4.4.6.3 Description – Input data: ... 73
4.4.6.4 Description – Obtain optimized procurement plan: ... 73
4.4.7 Implementation Viewpoint ... 74
4.4.8 AI^{PO} Implementation Components ... 75
4.4.9 Technical Description of AI^{PO} Components ... 75
4.5. Fabrication Optimiser – AI^{FO} ... 78
4.5.1 Usage Viewpoint ... 78
4.5.2 Functional Viewpoint ... 80
4.5.3 Data structure of AI^{FO} ... 80
4.5.4 AI^{FO} Software Requirements ... 82
4.5.5 AI^{FO} Lifecycle ... 83
4.5.6 Objects ... 83
4.5.7 Description – Login: ... 83
4.5.8 Description – Database connection: ... 84
4.5.9 Description – Send/RequestData: ... 84
4.5.10 Implementation Viewpoint ... 85
4.5.11 AI^{FO} Implementation Components ... 85
4.5.12 Technical Description of AI^{FO} Components ... 86
4.6. Delivery Optimiser – AI^{DO} ... 89
4.6.1 Usage Viewpoint ... 89
4.6.2 Functional Viewpoint ... 91
4.6.2.1 Data structure of AI^{DO} ... 91
4.6.2.2 AI^{DO} Hardware Requirements ... 97
4.6.2.3 AI^{DO} Software Requirements ... 97
4.6.2.4 AI^{DO} Lifecycle ... 97
4.6.2.5 Objects ... 98
4.6.2.6 Description – Login ... 98
4.6.2.7 Description – Database connection ... 98
4.6.2.8 Description – Send/Request Data ... 99
4.6.3 Implementation Viewpoint ... 100
4.6.3.1 AI^{DO} Implementation Components ... 100
4.6.3.2 Technical Description of the AI^{DO} Components ... 101
4.7. Machine Calibrator – AI^{MC} ... 104
4.7.1 Usage Viewpoint ... 104
4.7.2 Functional Viewpoint ... 106
4.7.2.1 Data structure of AI^{MC} ... 106
4.7.2.2 AI^{MC} Software Requirements ... 107
4.7.2.3 AI^{MC} Lifecycle ... 107
4.7.2.4 Objects ... 108
4.7.2.5 Description – login: ... 108
4.7.2.6 Description – Database Connection Parametrization: ... 108

4.7.2.7 Description – Parametrization: 108
4.7.2.8 Description – Request Calibration 108
4.7.3 Implementation Viewpoint 109
4.7.3.1 AI^{MC} Implementation Components 110
4.7.3.2 Technical Description of AI^{MC} Components 110
4.8. CONDITION EVALUATOR – AI^{CE} 112
4.8.1 Usage Viewpoint 112
4.8.2 Functional Viewpoint 113
4.8.2.1 Data structure of AI^{CE} 113
4.8.2.2 AI^{CE} Software Requirements 115
4.8.2.3 AI^{CE} Lifecycle 115
4.8.2.4 Objects 115
4.8.2.5 Description - Login: 116
4.8.2.6 Description - Database Connection / File Upload: 116
4.8.2.7 Description - Condition Evaluation Parametrization: 117
4.8.2.8 Description – Perform Condition Evaluation: 117
4.8.3 Implementation Viewpoint 118
4.8.3.1 AI^{CE} Implementation Components 119
4.8.3.2 Technical Description of AI^{CE} Components 119
4.9. ANOMALY DETECTOR – AI^{AD} 123
4.9.1 Usage Viewpoint 123
4.9.2 Functional Viewpoint 124
4.9.2.1 Data structure of AI^{AD} 124
4.9.2.2 AI^{AD} Software Requirements 126
4.9.2.3 AI^{AD} Lifecycle 126
4.9.2.4 Objects 126
4.9.2.5 Description - Login: 127
4.9.2.6 Description - Database Connection / File Upload: 127
4.9.2.7 Description - Condition Evaluation Parametrization: 128
4.9.2.8 Description – Perform Anomaly Detection: 128
4.9.3 Implementation Viewpoint 129
4.9.3.1 AI^{AD} Implementation Components 130
4.9.3.2 Technical Description of AI^{AD} Components 130
4.10. ADAPTIVE CONTROLLER – AI^{AC} 134
4.10.1 Usage Viewpoint 134
4.10.2 Functional Viewpoint 135
4.10.2.1 Data structure of AI^{AC} 135
4.10.2.2 AI^{AC} Software Requirements 137
4.10.2.3 AI^{AC} Lifecycle 137
4.10.2.4 Objects 137
4.10.2.5 Description - Login 138
4.10.2.6 Description - Database Connection / File Upload 138
4.10.2.7 Description – Adaptive Controller Parametrization 138
4.10.2.8 Description – Perform Control: 139
4.10.3 Implementation Viewpoint 140

4.10.3.1 AI^{AC} Implementation Components 140
4.10.3.2 Technical Description of AI^{AC} Components 141
4.11. QUALITY ASSURANCE – AI^{QA} 145
4.11.1 Usage Viewpoint 145
4.11.2 Functional Viewpoint 146
4.11.2.1 Data structure of AI^{QA} 146
4.11.2.2 AI^{QA} Software Requirements 147
4.11.2.3 AI^{QA} Lifecycle 147
4.11.2.4 Objects: 148
4.11.2.5 Description: Training and Deployment 148
4.11.3 Implementation Viewpoint 149
4.11.1 AI^{QA} Implementation Components 150
4.11.2 Technical Description of AI^{QA} Components 150
4.12. PRESCRIPTIVE MAINTENANCE – AI^{PM} 152
4.12.1 Usage Viewpoint 152
4.12.2 Functional Viewpoint 153
4.12.2.1 Data structure of AI^{PM} 153
4.12.2.2 AI^{PM} Software Requirements 154
4.12.3 AI^{PM} Lifecycle 155
4.12.3.1 Objects 155
4.12.3.2 Description - Login: 155
4.12.3.3 Description - Database Connection / File Upload: 156
4.12.3.4 Description – Prescriptive Maintenance Parametrization 156
4.12.3.5 Description – Perform Prescriptive Maintenance: 157
4.12.4 Implementation Viewpoint 158
4.12.4.1 AI^{PM} Implementation Components 158
4.12.4.2 Technical Description of AI^{PM} Components 159
4.13. SMART RETROFITTER – AI^{SR} 163
4.13.1 Usage Viewpoint 163
4.13.2 Functional Viewpoint 164
4.13.3 Data structure of AI^{SR} 164
4.13.4 AI^{SR} Hardware Requirements 165
4.13.5 AI^{SR} Software Requirements 165
4.13.6 AI^{SR} Lifecycle 166
4.13.6.1 Objects 166
4.13.6.2 Description – Login 166
4.13.6.3 Description – Database Connection 167
4.13.6.4 Description – Smart Retrofit 167
4.13.7 Implementation Viewpoint 168
4.13.8 AI^{SR} Implementation Components 169
4.13.9 Technical Description of AI^{SR} Components 169
4.14. LCC/LCA/S-LCA – AI^{LC} 170
4.14.1 Usage Viewpoint 170
4.14.2 Functional Viewpoint 172
4.14.2.1 Data structure of AI^{LC} 172

4.14.2.2 AI^{LC} Software Requirements ... 173
4.14.3 AI^{LC} Lifecycle ... 174
4.14.3.1 Objects ... 174
4.14.3.2 Description – Login ... 174
4.14.3.3 Description - Database Connection / File Upload ... 175
4.14.3.4 Description - End-of-Life graph modelling ... 175
4.14.3.5 Description - End-of-Life graph analysis ... 176
4.14.4 Implementation Viewpoint ... 176
4.14.5 AI^{LC} Implementation Components ... 177
4.14.6 Technical Description of AI^{LC} Components ... 178
4.15. DISASSEMBLER – AI^{DIS} ... 180
4.15.1 Usage Viewpoint ... 180
4.15.2 Functional Viewpoint ... 181
4.15.2.1 Data structure of AI^{DIS} ... 182
4.15.2.2 AI^{DIS} Hardware Requirements ... 183
4.15.2.3 AI^{DIS} Software Requirements ... 183
4.15.2.4 AI^{DIS} Lifecycle ... 183
4.15.2.5 Objects ... 184
4.15.2.6 Description – Login ... 184
4.15.3 Description - Database Connection / File Upload: ... 184
4.15.4 Description – Disassembly analysis ... 185
4.15.5 Implementation Viewpoint ... 186
4.15.6 AI^{DIS} Implementation Components ... 186
4.15.7 Technical Description of AI^{DIS} Components ... 186
4.16. MACHINE PASSPORT – AI^{MP} ... 188
4.16.1 Usage Viewpoint ... 188
4.16.2 Functional Viewpoint ... 190
4.16.2.1 Data structure of AI^{MP} ... 191
4.16.3 AI^{MP} Hardware Requirements ... 191
4.16.4 AI^{MP} Software Requirements ... 192
4.16.4.1 AI^{MP} Lifecycle ... 192
4.16.5 Objects ... 193
4.16.6 Description – Database Connection: ... 193
4.16.7 Implementation Viewpoint ... 194
4.16.7.1 AI^{MP} Implementation Components ... 195
4.16.7.2 Technical Description of AI^{MP} Components ... 195

5. CONCLUSIONS ... 198

APPENDIX I ... 199
SURVEY FOR THE PILOTS/VALUE PROPOSITION ... 199
SURVEY FOR EXTERNAL STAKEHOLDERS ... 201

LIST OF FIGURES

Figure 1. AIDEAS Usage Viewpoint Model....26
Figure 2. AIDEAS Reference Architecture....27
Figure 3. Example Data flow in the AIDEAS Reference Architecture....28
Figure 4. Example of sequence diagram....32
Figure 5. AIDEAS Solutions High-level Architecture....33
Figure 6. AIDEAS suites and toolkits....36
Figure 7. AI^{MDO}Usage Viewpoint Activity Diagram....43
Figure 8. Data Flow AI^{MDO}....44
Figure 9. AI^{MDO} Lifecycle....45
Figure 10. AI^{MDO} Implementation Architecture....47
Figure 11. AI^{MDG} Usage Viewpoint Activity Diagram....51
Figure 12. Data Flow AI^{MDG}....52
Figure 13. AI^{MDG} Lifecycle....55
Figure 14. AI^{MDG} Implementation Architecture....57
Figure 15. AI^{CAx} Usage Viewpoint Activity diagram....61
Figure 16. Data Flow AI^{CAx}....62
Figure 17. Data Flow AI^{CAx}....65
Figure 18. AI^{CAx} Implementation Architecture....67
Figure 19. AI^{PO} Activity Diagram (Generate new optimized procurement plan)....69
Figure 20. Data Flow AI^{PO}....70
Figure 21. AI^{PO} Lifecycle....72
Figure 22. AI^{PO} Implementation Architecture....74
Figure 23. AI^{FO} Activity Diagram (Generate corrective parameter to minimize re-machining operations)....78
Figure 24. AI^{FO} Activity Diagram (Generate new production plan and scheduling plan)....79
Figure 25. Data Flow AI^{FO}....80
Figure 26. AI^{FO} Lifecycle....83
Figure 27. AI^{FO} Implementation Architecture....85
Figure 28. Delivery Optimiser Usage Viewpoint Activity Diagram (Optimise delivery)....90
Figure 29. Data Flow AI^{DO}....91
Figure 30. AI^{DO} Lifecycle....97
Figure 31. AI^{DO} Implementation Architecture....100
Figure 32. Machine Calibrator Usage Viewpoint Activity Diagram (Machine Calibration)....104
Figure 33. Machine Calibrator Usage Viewpoint Activity Diagram (Machine Configuration)....105
Figure 34. Data Flow AI^{MC}....106
Figure 35. AI^{MC} Lifecycle....107
Figure 36. AI^{MC} Implementation Architecture....109
Figure 37. Condition Evaluation Usage Viewpoint Activity Diagram (Condition Evaluator)....112
Figure 38. Data Flow AI^{CE}....113
Figure 39. AIDEAS Condition Evaluator Sequence Diagram....115
Figure 40. AI^{CE} Implementation Architecture....118
Figure 41. AI^{AD} Activity Diagram (Anomaly Detector)....123
Figure 42. Data Flow AI^{AD}....124

Figure 43. AIDEAS Anomaly Detector Sequence Diagram....126
Figure 44. AI^{AD} Implementation Architecture....129
Figure 45. Adaptive Controller Usage Viewpoint Activity Diagram....134
Figure 46. Data Flow AI^{AC}....135
Figure 47. AIDEAS Adaptive Controller Sequence Diagram....137
Figure 48. AI^{AC} Implementation Architecture....140
Figure 49. AI^{QA} solution Usage Viewpoint Activity Diagram....145
Figure 50. Data Flow AI^{QA}....146
Figure 51. AI^{QA} Lifecycle....147
Figure 52. AI^{QA} 2D Implementation Architecture....149
Figure 53. AI^{QA} 3D Implementation Architecture....150
Figure 54. Estimation Usage Viewpoint Activity Diagram (Prescriptive Maintenance)....152
Figure 55. Data Flow AI^{PM}....153
Figure 56. AIDEAS Prescriptive Maintenance Sequence Diagram....155
Figure 57. AI^{PM} Implementation Architecture....158
Figure 58. AI^{SR} Usage Viewpoint activity Diagram....163
Figure 59. Data Flow AI^{SR}....164
Figure 60. AI^{SR} Lifecycle....166
Figure 61. AI^{SR} Implementation Architecture....168
Figure 62. AI^{LC} Usage Viewpoint Activity Diagram....171
Figure 63. Data Flow AI^{LC}....172
Figure 64. AI^{LC} Lifecycle....174
Figure 65. AI^{LC} Implementation Architecture....177
Figure 66. AI^{DIS} Usage Viewpoint Activity Diagram....181
Figure 67. Data Flow AI^{DIS}....181
Figure 68. AI^{DIS} Lifecycle....183
Figure 69. AI^{DIS} Implementation Architecture....186
Figure 70. AI^{MP} Activity Diagram (New Machine Configuration)....189
Figure 71. AI^{MP} Activity Diagram (Machine Data Management)....189
Figure 72. Data Flow AI^{MP}....190
Figure 73. AI^{MP} Lifecycle....192
Figure 74. AI^{MP} Implementation Architecture....194

LIST OF TABLES

Table 1. Survey structure (survey for Pilots)....23
Table 2. Survey structure (survey for external stakeholders)....24
Table 3. Input / Output Data Format Template....30
Table 4. Example Hardware required (Camera)....31
Table 5. Example software requirements....31
Table 6. AIDEAS Solutions Implementation Components 34
Table 7. Technical Description of Implementation Components....35
Table 8. Input / Output Data Format AIMDO....44
Table 9. Software requirements AIMDO....45

Table 10. Life-Cycle description Login AIMDO....46
Table 11. Life-Cycle description Anomaly Detector Parametrization AIMDO....46
Table 12. Life-Cycle description Perform Anomaly Detection AIMDO....47
Table 13. AIMDO Implementation Components....48
Table 14. Technical Description of AIMDO “Load Machine Configuration” Implementation Component....48
Table 15. Technical Description of AIMDO “Configure Optimization” Implementation Component....49
Table 16. Technical Description of AIMDO “Perform Optimization” Implementation Component....49
Table 17. Technical Description of AIMDO “Perform Evaluation” Implementation Component....50
Table 18. Input / Output Data Format AIMDG....54
Table 19. Software requirements AIMDG....54
Table 20. Life-Cycle description Login AIMDG....55
Table 21. Life-Cycle description Design Space Configuration AIMDG....56
Table 22. Life-Cycle description Manage Batch AIMDG....56
Table 23. Life-Cycle description Manage Model AIMDG....57
Table 24. AIMDG Implementation Components....58
Table 25. Technical Description of AIMDG “Manage Components” Implementation Component....59
Table 26. Technical Description of AIMDG “Manage Materials” Implementation Component....59
Table 27. Technical Description of AIMDG “Manage Digital Twins” Implementation Component....60
Table 28. Technical Description of AIMDG “Manage Simulations” Implementation Component....60
Table 29. Input / Output Data Format AICAx....64
Table 30. Software Requirements AICAx....64
Table 31. Life-Cycle description Login AICAx....66
Table 32. Life-Cycle description Create digital twin AICAx....66
Table 33. Life-Cycle description Start simulation AICAx....66
Table 34. Life-Cycle description Retrieve simulation results AICAx....67
Table 35. AICAx Implementation Components....67
Table 36. Technical Description of AICAx “Addon for Autodesk Fusion” Implementation Component....68
Table 37. Input / Output Data Format AIPO....71
Table 38. Hardware required AIPO....71
Table 39. Software requirements AIPO....72
Table 40. Life-Cycle login action AIPO....73
Table 41. Life-Cycle Input data action AIPO....73
Table 42. Life-Cycle new procurement plan action AIPO....74
Table 43. AIPO Implementation Components....75
Table 44. Technical Description of AIPO “Import data” Implementation Component....76
Table 45. Technical Description of AIPO “Data validation” Implementation Component....76

Table 46. Technical Description of AIPO “Compose MRP algorithm” Implementation Component77
Table 47. Technical Description of AIPO “Calculate new MRP” Implementation Component....... 77
Table 48. Technical Description of AIPO “Export new MRP” Implementation Component............ 78
Table 49. Input / Output Data Format AIFO 82
Table 50. Software requirements AIFO 82
Table 51. Life-Cycle description Login AIFO 83
Table 52. Life-Cycle description Database connection AIFO 84
Table 53. Life-Cycle description Send/Request Data AIFO 84
Table 54. AIFO Implementation Components 86
Table 55. Technical Description of AIFO “Import Data” Implementation Component 86
Table 56. Technical Description of AIFO “Data Validation and Preprocessing” Implementation Component...87
Table 57. Technical Description of AIFO “Generation of a new scheduling plan” Implementation Component...87
Table 58. Technical Description of AIFO “Generation of a new Production Plan” Implementation Component...88
Table 59. Technical Description of AIFO “Generation of the corrective parameters” Implementation Component 89
Table 60. Technical Description of AIFO “Data Export” Implementation Component 89
Table 61. Input / Output Data Format AIDO.. 96
Table 62. Hardware required AIDO... 97
Table 63. Software requirements AIDO.. 97
Table 64. Life-Cycle description Login AIDO... 98
Table 65. Life-Cycle description Database connection AIDO.. 99
Table 66. Life-Cycle description Send/RequestData AIDO 99
Table 67. AIDO Implementation Components 101
Table 68. Technical Description of AIDO “Import Data” Implementation Component................. 101
Table 69. Technical Description of AIDO “Data Validation and Preprocessing” Implementation Component...102
Table 70. Technical Description of AIDO “Create and Export Model” Implementation Component...102
Table 71. Technical Description of AIDO “Create Predictions and Display Results” Implementation Component...103
Table 72. Input / Output Data Format AIMC.. 107
Table 73. Software requirements AIMC.. 107
Table 74. Life-Cycle description login AIMC 108
Table 75. Life-Cycle description dbConnectionParametrization AIMC 108
Table 76. Life-Cycle description Parametrization AIMC 108
Table 77. Life-Cycle description requestCalibration AIMC 109
Table 78. AIMC Implementation Components 110
Table 79. Technical Description of AIMC “Reading and data preparation” Implementation Component...110
Table 80. Technical Description of AIMC “Data validation” Implementation Component........... 111
Table 81. Technical Description of AIMC “Creating the calibration algorithm” Implementation Component...111

Table 82. Technical Description of AIMC "Generating a calibration configuration" Implementation Component....112
Table 83. Input / Output Data Format AICE.... 114
Table 84. Software requirements AICE.... 115
Table 85. Life-Cycle description Login AICE.... 116
Table 86. Life-Cycle description Database Connection AICE.... 116
Table 87. Life-Cycle description Condition Evaluation Parametrization AICE.... 117
Table 88. Life-Cycle description Perform Condition Evaluation AICE.... 118
Table 89. AICE Implementation Components.... 119
Table 90. Technical Description of AICE "Import Data" Implementation Component.... 120
Table 91. Technical Description of AICE "Data Validation and Preprocessing" Implementation Component....120
Table 92. Technical Description of AICE "Machine Configurator" Implementation Component 121
Table 93. Technical Description of AICE "Create and Export Models" Implementation Component....121
Table 94. Technical Description of AICE "Obtain Predictions and Display Results" Implementation Component....122
Table 95. Technical Description of AICE "Export Data" Implementation Component.... 122
Table 96. Input / Output Data Format AIAD.... 125
Table 97. Software requirements AIAD.... 126
Table 98. Life-Cycle description Login AIAD.... 127
Table 99. Life-Cycle description Database Connection AIAD.... 127
Table 100. Life-Cycle description Anomaly Detector Parametrization AIAD.... 128
Table 101. Life-Cycle description Perform Anomaly Detection AIAD.... 129
Table 102. AIAD Implementation Components.... 130
Table 103. Technical Description of AIAD "Import Data" Implementation Component.... 131
Table 104. Technical Description of AIAD "Data Validation and Preprocessing" Implementation Component....131
Table 105. Technical Description of AIAD "Machine Configurator" Implementation Component132
Table 106. Technical Description of AIAD "Create and Export Models" Implementation Component....132
Table 107. Technical Description of AIAD "Obtain Predictions and Display Results" Implementation Component.... 133
Table 108. Technical Description of AIAD "Export Data" Implementation Component.... 133
Table 109. Input / Output Data Format AIAC.... 136
Table 110. Software requirements AIAC.... 137
Table 111. Life-Cycle description Login AIAC.... 138
Table 112. Life-Cycle description Database Connection AIAC.... 138
Table 113. Life-Cycle description Adaptive Controller Parametrization AIAC.... 139
Table 114. Life-Cycle description Perform Control AIAC.... 139
Table 115. AIAC Implementation Components.... 141
Table 116. Technical Description of AIAC "Import Data" Implementation Component.... 141
Table 117. Technical Description of AIAC "Data Validation and Preprocessing" Implementation Component....142
Table 118. Technical Description of AIAC "Machine Configurator" Implementation Component142

Table 119. Technical Description of AIAC "Create and Export Models" Implementation Component......143
Table 120. Technical Description of AIAC "Create and Export Controllers" Implementation Component......144
Table 121. Technical Description of AIAC "Obtain Controller Evaluation" Implementation Component......144
Table 122. Technical Description of AIAC "Export Data" Implementation Component......145
Table 123. Input / Output Data Format AIQA......146
Table 124. Software requirements AIQA......147
Table 125. Life-Cycle description Training and Deployment AIQA......148
Table 126. AIQA Implementation Components......150
Table 127. 2D Defect Detection Component......151
Table 128. 3D Defect Detection Component......151
Table 129. Input / Output Data Format AIPM......154
Table 130. Software requirements AIPM......154
Table 131. Life-Cycle description Login AIPM......155
Table 132. Life-Cycle description Database Connection AIPM......156
Table 133. Life-Cycle description Prescriptive Maintenance Parametrization AIPM......157
Table 134. Life-Cycle description Perform Prescriptive Maintenance AIPM......157
Table 135. AIPM Implementation Components......159
Table 136. Technical Description of AIPM "Import Data" Implementation Component......159
Table 137. Technical Description of A AIPM "Data Validation and Preprocessing" Implementation Component......160
Table 138. Technical Description of AIPM "Machine Configurator" Implementation Component160
Table 139. Technical Description of AIPM "Create and Export Models" Implementation Component......161
Table 140. Technical Description of AIPM "Obtain Predictions and Display Results" Implementation Component......162
Table 141. Technical Description of AIPM "Export Data" Implementation Component......162
Table 142. Input / Output Data Format AISR......165
Table 143. Hardware required AISR......165
Table 144. Software requirements AISR......165
Table 145. Life-Cycle description Login AISR......166
Table 146. Life-Cycle description Database Connection AISR......167
Table 147. Life-Cycle description Smart Retrofit AISR......168
Table 148. AISR Implementation Components......169
Table 149. Technical Description of AISR "Data acquisition" Implementation Component......169
Table 150. Technical Description of AISR "Upload data" Implementation Component......170
Table 151. Technical Description of AISR "Perform evaluation" Implementation Component..170
Table 152. Input / Output Data Format AILC......173
Table 153. Software requirements AILC......173
Table 154. Life-Cycle description Login AILC......175
Table 155. Life-Cycle description Database Connection AILC......175
Table 156. Life-Cycle description End-of-Life graph modelling AILC......176
Table 157. Life-Cycle description End-of-Life graph analysis AILC......176

Table 158. AILC Implementation Components 177
Table 159. Technical Description of AILC “Import Data” Implementation Component 178
Table 160. Technical Description of AILC “Data Validation” Implementation Component 179
Table 161. Technical Description of AILC “Compose LC algorithm” Implementation Component 179
Table 162. Technical Description of AILC “Run LC algorithm” Implementation Component 180
Table 163. Technical Description of AILC “Export the results” Implementation Component 180
Table 164. Input / Output Data Format AIDIS 182
Table 165. Hardware required AIDIS 183
Table 166. Software requirements AIDIS 183
Table 167. Life-Cycle description Login AIDIS 184
Table 168. Life-Cycle description Database Connection AIDIS 185
Table 169. Life-Cycle description Disassembly Analysis AIDIS 185
Table 170. AIDIS Implementation Components 186
Table 171. Technical Description of AIDIS “Upload Data” Implementation Component 187
Table 172. Technical Description of AIDIS “Image Acquisition” Implementation Component 187
Table 173. Technical Description of AIDIS “Performance Evaluation” Implementation Component 188
Table 174. Input / Output Data Format AIMP 191
Table 175. Hardware required AIMP 191
Table 176. Software requirements AIMP 192
Table 177. Life-Cycle description Database Connection AIMP 194
Table 178. AIMP Implementation Components 195
Table 179. Technical Description of AIMP “Front-End” Implementation Component 196
Table 180. Technical Description of AIMP “Back-End” Implementation Component 196
Table 181. Technical Description of AIMP “Database” Implementation Component 197
Table 182. Technical Description of AIMP “Data Harmonizer” Implementation Component 197

ABBREVIATIONS/ACRONYMS

AAS	Asset Administration Shell
AC	Adaptive Controller
AD	Anomaly Callibrator
AF	Architecture Framework
AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
AR	Augmented Reality
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAM	Computer Aided Manufacturing
CAx	Computer Aided
CE	Condition Evaluator
CNC	Computerised Numerical Control
CSS	Cascading Style Sheets
CSV	Comma Separated Values
DB	Data Bases
DIS	Disassembler
DO	Delivery Optimiser
DSS	Decision Support System
EoL	End of Life
ERP	Enterprise Resource Planning
FO	Fabrication Optimiser
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IIRA	Industrial Internet Reference Architecture
IMG	Image
IoT	Internet of Things

IT	Information Technologies
KPI	Key Performance Indicators
JPEG	Joint Photographic Experts Group
JSON	JavaScript Object Notation
LCA	Life Cycle Assessment
LCC	Life Cycle Costing
M2M	Machine to Machine
MC	Machine Callibrator
MDG	Machine synthetic Data Generator
MDO	Machine Design Optimiser
ML	Machine Learning
MRP	Material Requirements Planning
OS	Operational System
PDF	Portable Document Format
PIL	Python Imaging Library
PLC	Programmable Logic Controller
PM	Prescriptive Maintenance
PNG	Portable Network Graphics
PO	Procurement Optimiser
QA	Quality Assurance
RA	Reference Architecture
REST	Representational State Transfer
RF	Reference Framework
RUL	Remaining Useful Life
S-LCA	Social Life Cycle Assessment
SDG	Sustainable Development Goals
SDK	Software Development Kit
SMEs	Small and Medium-sized Enterprises
SQL	Structured Query Language
SR	Smart Retrofitter
UI	User Interface

UML	Unified Modeling Language
URL	Uniform Resource Locator
WebGL	Web Graphics Library
XML	Extensible Markup Language

Executive summary

The objective of AIDEAS is to develop AI technologies that support the complete lifecycle of industrial equipment, encompassing design, manufacturing, use, and repair/reuse/recycle processes. The primary objective is to enhance European machinery manufacturing companies' sustainability, agility, and resilience. AIDEAS will deploy 4 integrated Suites:

- **Design**: Incorporates AI technologies that seamlessly integrate with CAD/CAM/CAE systems to optimize the design of structural components, mechanisms, and control elements of industrial equipment.
- **Manufacturing**: Focuses on leveraging AI technologies to improve various aspects of industrial equipment production. This includes efficient selection and procurement of purchased components, optimization of manufacturing processes for parts, sequencing of operations, quality control measures, and customization capabilities.
- **Use**: Centres around AI technologies that offer added value to industrial equipment users. These technologies provide enhanced support for installation, initial calibration, production processes, quality assurance, and predictive maintenance, ensuring optimal operating conditions.
- **Repair-Reuse-Recycle:** Utilizes AI technologies to extend the useful lifespan of machines through prescriptive maintenance (repair), enable smart retrofitting for a second life (reuse), and identify the most sustainable end-of-life approaches (recycle).

This document is the second version of the AIDEAS viewpoints. This document is the result of the updated work of the whole consortium on Tasks T2.2 "Business Viewpoint", T2.3 "Usage Viewpoint", T2.4 "Functional Viewpoint" and T2.5 "Implementation Viewpoint".

Document Structure

Section 1 Introduction: In this section, an elaborate introduction to the deliverable is provided, addressing key aspects that will be developed throughout the document.

Section 2 Methodology: In this section, the methodology used in the various viewpoints considered in the project is thoroughly defined. It explores how this methodology guides the approach towards achieving the set objectives.

Section 3 Business Viewpoint: This section delves into the development of the business layer of the project. It analyses the fundamental elements that make up this perspective and their relevance in the context of the work undertaken.

Section 4 AIDEAS Viewpoints v2: This section presents the main viewpoints that will steer the design and implementation of the AIDEAS Reference Framework (AIDEAS RF). These viewpoints are examined in detail, along with their implications on the development of the reference framework.

Section 5 Conclusions: In this section, the conclusions drawn from the development of the AIDEAS Reference Framework are presented. The most significant findings are summarised, and reflections are made on their impact and relevance within the project's context.

1. Introduction

In an increasingly interconnected and technologically advanced world, the integration of solutions is essential to effectively address business and societal challenges. In this context, the AIDEAS project presents in this deliverable the approach used in the development of its solutions.

To this end, the methodology and approaches used in the AIDEAS project for the design, implementation and deployment of solutions are explored in depth. Through a detailed analysis of the different aspects addressed in the project, from the business perspective to the technical implementation, the aim is to provide a complete overview of how these solutions have been developed.

The methodology used in the AIDEAS project is characterised by an approach that integrates business, technical and user aspects at all stages of the development process. From the definition of the vision and business objectives to the technical implementation of the solutions, special attention is paid to the alignment with real market needs and the creation of value for both organisations and society. This deliverable examines in detail four key approaches used in the AIDEAS project: the business viewpoint, the usage viewpoint, the functional viewpoint, and the implementation viewpoint. Each of these approaches offers a unique and complementary perspective that contributes to the overall success of the project.

2. Methodology

This section serves as a guide to the methodology employed in this document. The following pages detail the step-by-step processes designed to gather data, conduct analyses, and derive meaningful insights.

2.1. Business viewpoint approach for AIDEAS

The main objective of this section is to provide a business point of view within the AIDEAS design activities to avoid the risk of "technology-centric" development. This will allow to incorporate, already in the early design phases, requirements and needs that are closer to the real operational needs. To this end, the task will focus on framing the vision, values, and key objectives of the AIDEAS architecture from a business, regulatory and stakeholder perspective. In parallel, it will also identify the capabilities of the consortium's technical team and conduct a business requirements mapping exercise. To improve the effectiveness of the task, specific input will be sought from experienced business partners close to the industry, e.g. through the submission of a questionnaire or a specific interview, like the way in which product development is carried out.

2.1.1 Business Viewpoint Concepts

The task involves articulating the value proposition, vision, and stakeholder interest of the AIDEAS architecture for each suite, considering business, regulatory and stakeholder perspectives. At the same time, the capabilities of the consortium's technical team need to be identified and mapped against the business requirements. Input from experienced business partners close to the industry will be gathered through a dedicated questionnaire or interview process, like standard product development procedures.

2.1.2 Business Viewpoint Methodology

The methodology employed for this task entails several key steps. Firstly, the stakeholders are identified, particularly companies with vested interests in the project. This identification process is pivotal to ensure the business perspective is adequately represented.

Once the stakeholders have been identified, they are asked for their business perspective on each toolkit, from business decision makers to technical managers. The aim is to identify the value proposition, vision and values or new capabilities that are relevant to the organisation. Keep in mind that stakeholders may be interested in certain toolkits and indifferent to others. Therefore, focus on gathering information on only those toolkits that are relevant to stakeholders.

In a structured way, the steps can be outlined as follows:

- Identification of stakeholders.
- Extract information from each stakeholder, each suite, and each toolkit (that they are interested in).

Initially, a questionnaire was administered pertaining to the project's pilots to acquire the value proposition. Subsequently, a second survey targeted external stakeholders, such as SMEs and industries.

2.1.3 Survey (Form) Design

Two surveys have been designed to gather information on requirements that are aligned with real business needs, covering the perspectives of both business decision makers and technical leaders. The questionnaire designed for the pilots in the AIDEAS project consists of three sections in addition to the survey instructions. These sections include:

- Identification of respondents.
- Value proposition (problem identification with pain and gain for each suite of the AIDEAS project).
- Identification of alternative solutions to the identified problem.

Similarly, the questionnaire for external stakeholders is divided into three sections, supplemented by survey instructions. These sections include:

- Identification of respondents.
- Selection of Suites/Toolkit.
- Business perspective.

Table 1 describes the sections and the information required for each, and details the survey sent to the pilots.

Section	Question/Text	Answer type	
Survey Instructions	Welcome, survey duration, survey objective etc.		Text
Stakeholder identification	Organization		Open answer
	Business contact	Name	Open answer
		E-mail	Open answer
Value Proposition (Suites)	AIDEAS Suites		Text
	Jobs to be done		Open answer
	Pains		Open answer
			Open answer
	Gains		Open answer
			Open answer
	Alternative Solution		Open answer

Table 1. Survey structure (survey for Pilots)

Table 2 displays the sections and the information required in each of them for the survey that has been sent to the external stakeholders.

Section	Question/Text		Answer type
Survey instructions	Welcome, survey duration, survey objective…		Text
Identification	Organization		Open answer
	Sector		Open answer
	Business contact	Name	Open answer
		Job position	Open answer
		E-mail	Open answer
		Name	Open answer
		Job position	Open answer
		E-mail	Open answer
Business perspective (Suite and Toolkits selection)	Introduction to AIDEAS Suites and Toolkits		Text
	Interest per Suite		Yes/No/More info
	Description of information to be collected		Text
	Interest per Toolkit	Toolkit name and description	Text
		Interest	Yes/No/More info
	Business Perspective	Problem understanding	Open answer
		Feedback on the way of presenting the solutions	Open answer

Table 2. Survey structure (survey for external stakeholders)

Both Survey models sent to the stakeholders have been included as annexes to this document.

2.2. Usage viewpoint approach for AIDEAS

The main objective of this section is to define the implementation strategy for the capabilities and structure of the AIDEAS framework. The initial focus is on identifying the four key elements of the usage perspective: tasks, roles, activities, and parts within AIDEAS, covering both human and hardware/software systems. Once these elements have been identified, the process moves on to defining functional and implementation maps and specifying the roles responsible for carrying out the tasks.

In the first version of Usage Viewpoints (D2.1), User Diagrams were created to describe the interactions allowed between the different users and the solutions. Mockups were also created for each AIDEAS solution, demonstrating the different phases of the solutions and how the user will interact with them. This is possible to see in the AIDEAS webpage[1].

In this second version, AIDEAS solutions developers were asked to identify whether it was necessary to update the mockups. Section 3.1 describes these updates and why they were changed. To better understand how different stakeholders interact within each solution, each stakeholder was asked to summarise the stakeholders' business views and how they interact with the solution. This way, it is possible to identify the roles involved in the solution and how they interact with it. By type of functions, a set of capabilities and the task it performs are defined, being able to identify the connection between the usage viewpoint and the other viewpoints. How can we check which triggers, effects and constraints make up the task to be performed. This representation is illustrated by "UML activity diagrams" and is described for each AIDEAS solution in section 3.2.

Figure 1 depicts an adaptation of the model of the Industrial Internet Reference Architecture (IIRA) Usage Viewpoint, tailored to the specific developments of the AIDEAS solutions. "The basic unit of work is a task, such as the invocation of an operation, a transfer of data or an action of a party. A task is carried out by a party assuming a role."

2.2.1 Usage Viewpoint Model

In the AIDEAS usage viewpoint model, the party is represented by the "Stakeholders", namely the different partners involved in each system (Solution). The "Roles" will be defined for each solution giving a set of capacities and responsibilities for a stakeholder to execute a task. The "Task" is connected to the "Functional Components" by a functional map that describes the inputs and outputs of each function that it connects to.

The "Activity" represents the coordination of tasks that are required to produce a usage or process of the solution. The "triggers" are the conditions that the tasks, and the effects are the changes that occur in the solution after the completion of the tasks.

[1] https://aideas-project.eu/

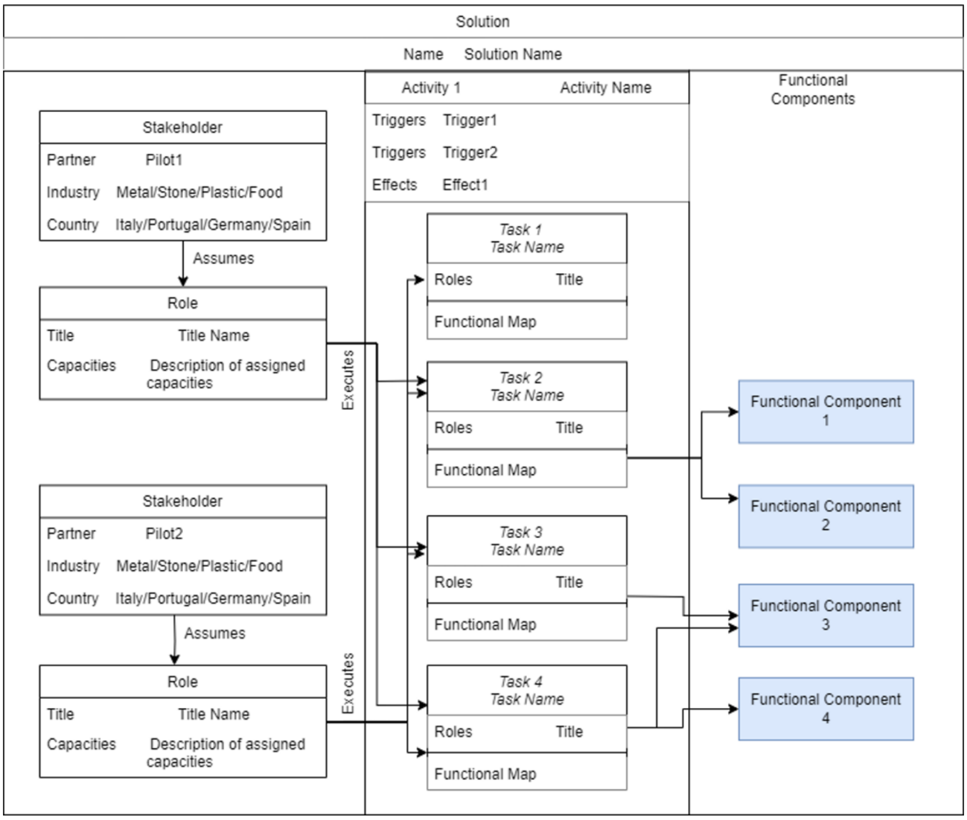


Figure 1. AIDEAS Usage Viewpoint Model

2.3. Functional viewpoint approach for AIDEAS

The functional viewpoint task entails the meticulous breakdown of AIDEAS Solutions into distinct functional components, organised within various functional domains. These domains encompass Control (encompassing functions executed by industrial control and automation systems), Operations (responsible for the management and operation of the control domain), Information (focused on data management and processing), Application (housing functions that implement application logic for specific business functionalities), and Business (comprising functions that support business processes and procedural activities).

Subsequently, the functional viewpoint delves into the intricate web of data, decision, and command/request flows among these functional components. This analysis not only identifies key interfaces between these components but also sheds light on their interactions with external elements in the environment. Following this decomposition, the document will expound upon the controls, coordination, and orchestration mechanisms employed within each domain. To provide a comprehensive conceptual and functional representation, sequence diagrams are employed to depict the most pertinent usages and activities within the overall system.

The primary objectives of this section are as follows:

- **Define Operational Requirements:** Clearly articulate the prerequisites essential for the effective operation of each solution.
- **Data Structure Definition:** Emphasize the critical role of defining input and output data structures, a pivotal aspect in the design of AIDEAS solutions.
- **Hardware and Software Requirements:** Precisely outline the hardware and software prerequisites crucial for the successful implementation of the solutions.

- **Demonstrator Development**: Conclude this section by crafting a comprehensive sequence diagram. This diagram, integral to the FINAL DEMO for exploitation, visually elucidates the sequential processes inherent in the solution's functionality.

2.3.1 RA Domain requirements

This section incorporates considerations related to the Reference Architecture (RA) Domains, with the overall objective of recognizing and mitigating potential risks associated with data connectivity, network design and data processing within the architecture (Figure 2).

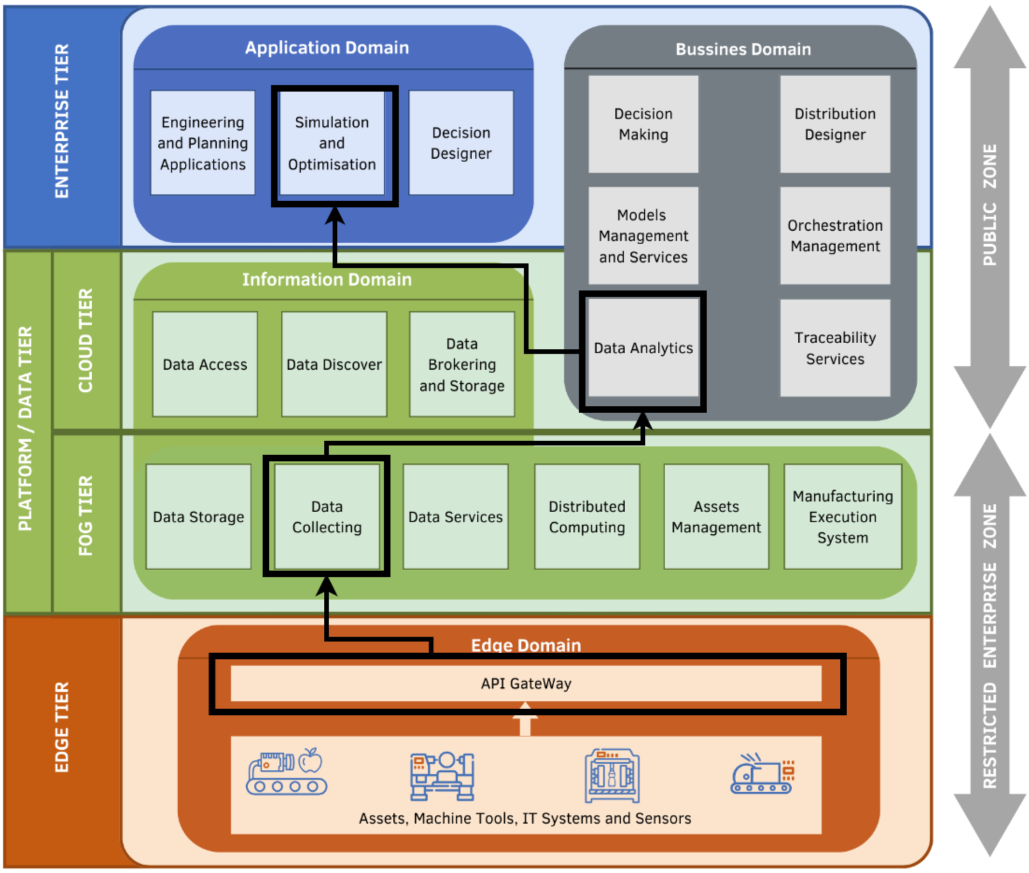


Figure 2. AIDEAS Reference Architecture

The objective is to ensure the robustness and reliability of the solution design, focusing on the following points:

- **Application Domain:** This domain provides specific application services to the system. It plays a key role in providing functionality tailored to the unique requirements of the overall architecture, thus improving the performance and responsiveness of the system.
- **Business Domain:** Responsible for overseeing and managing the system's business processes, the business domain is critical in ensuring that the overall solution is perfectly aligned with the organization's strategic objectives and operational needs.
- **Information Domain:** At the heart of managing the flow of information within the system, the information domain is responsible for managing data consistently and efficiently. This domain ensures that data is properly processed, stored and transmitted, fostering effective communication and decision-making processes.

- **Edge Domain:** Referring to the layer of IT infrastructure located at the edge of the network, closest to the devices and sensors that generate data, the edge domain is critical for real-time processing and immediate response. By residing close to data sources, it minimizes latency and optimizes the efficiency of data-driven operations.

Based on the understanding of the different domains in which the different solutions interact, the development of a data flow map from the AIDEAS Reference architecture is proposed. This analysis will allow addressing aspects of connectivity, data processing, understanding of the solution allowing predicting possible challenges or errors in the design of the solution.

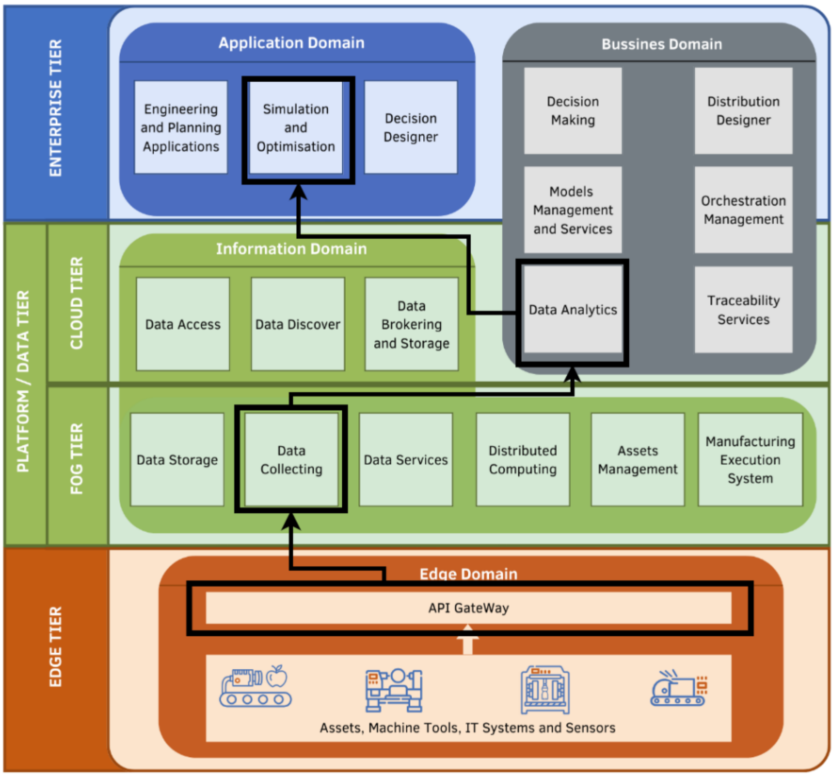


Figure 3. Example Data flow in the AIDEAS Reference Architecture

It involves visually representing the flow of data among different components within each domain, framing and connecting each component with others according to their functionality in the solution.

For instance, a solution might initiate the data flow by gathering information from the Edge through an API. These data are then transmitted to the platform using a Data Collecting module, followed by analysis through Data Analytics tools, and finally utilized within an optimization application in the Application Domain.

2.3.2 Data structure: Input and Output Data

After analysing and understanding the data flow that performs the solutions within the AIDEAS Reference architecture, the types of data with which the solution will work is defined at a functional level. These input and output data can be of the following types:

- **Database:** Defines the structure of tables, data types and values in a database. This includes defining tables, their attributes, relationships, data formats (such as integers, strings, dates) and constraints.

- **Images:** Specifies the type of image (JPEG, PNG, etc.) and if possible, the size.
- **JSON or other formats:** Defines the structure of the data in JSON or other similar formats.
- **Files:** For files, specify the encoding, character sets, tokenization methods and structure (paragraphs, sentences, words).
- **Other:** If there is another undefined data source.

2.3.3 Data structure of each solution

For each solution, the structure of the input and output data is defined. Table 3 represents an example with all kinds of formats, describing whether the data is an input or output and demonstrating an example of this input or output.

Format	Input/Output	Example
IMG	Input	
	Output	
CSV	Input	timestamp, vibration level 2023-11-30T08:00:00Z, 0.5
	Output	timestamp, maintenance_required 2023-11-30T08:30:00Z, true
XML	Input	<machine_data> <timestamp>2023-11-30T08:00:00Z</timestamp> <temperature>65.2</temperature> </machine_data>
	Output	<machine_adjustment> <timestamp>2023-11-30T08:05:00Z</timestamp> <adjustment>cooling</adjustment>

		</machine_adjustment>

JSON	Input	{
		"Name": "ABC123",
		"items": [
		{
		"product_id": "P001",
		"quantity": 10
		}
		],
		"delivery_address": {
		"street": "123 Main Street",
		"city": "Anytown",
		"country": "Country",
		"zipcode": "12345"
		}
		}

		{
		"order_id": "ABC123",
		"estimated_delivery_date": "2023-12-15"
		}

Text Data	Output	• Machine Status: Operational • Temperature: 65.2°C • Runtime: 120 hours • Errors: 0

Table 3. Input / Output Data Format Template

2.3.4 Hardware and Software requirements

This section identifies the hardware elements required for the functionality of the solution. Table 4 details each hardware element, highlighting its importance and exemplifying the respective data associated with it. In case there is no specific hardware requirement for its operation, this table will not be added.

Hardware Element	Importance and Explanation	Data of Element
Image Sensor	Crucial for capturing high-resolution images. Greater sensor size and higher megapixels enhance image clarity and detail.	Sony IMX577, 12.3MP resolution, 1/2.3" sensor size
Lens	Quality lens ensures sharp focus, allowing the capture of clear and detailed images, vital for accurate object detection.	Zeiss Vario-Sonnar T* lens, focal length 24-200mm

Image Processor	Powerful image processing chip enhances captured data, enabling faster analysis and extraction of features for AI detection.	Digic X (Canon) or BIONZ XR (Sony) image processor
Storage Media	Quick and ample storage is necessary for storing high-resolution images and videos required for AI model training and inference.	Example: Lexar Professional 2000x SDXC UHS-II Card, 256GB
Connectivity	Wireless connectivity options (e.g., Wi-Fi, Bluetooth) facilitate easy data transfer to AI model training platforms.	Wi-Fi 6, Bluetooth 5.2
Power Supply	Reliable and long-lasting power source to ensure continuous operation during data acquisition and transmission.	NP-FZ100 Rechargeable Lithium-Ion Battery (Sony)

Table 4. Example Hardware required (Camera)

Software Requirements

This section presents the software requirements that the systems where the solutions are going to be deployed must have for their correct functionality. Table 5 provides an example of software components, its description and required configuration.

Software Component	Description/Role	Required Version/Configuration
Windows OS	Microsoft operating system its need to use specific tool	Windows 11
Microsoft Office Suite	Suite of office with excel applications for document processing.	Office 365
SAP ERP	This Solution need SAP ERP data to use the tool.	V 7.0

Table 5. Example software requirements

2.3.5 Solutions Lifecycle

In this section, a sequence diagram is developed to demonstrate the functionality and interactions of the system components. A sequence diagram represents the sequence of messages exchanged between the various system components during a particular scenario or operation. In the context of the final demonstration for exploitation, this sequence diagram will show the flow of actions and communications between various components, illustrating how data is processed, transferred, and used within the AI solution. It provides a high-level view of system behaviour, illustrating the order of operations, dependencies, and interactions between the various parts of the solution. This diagram not only helps to understand the functionality of the system, but also serves as a valuable tool to present the capabilities, logic, and operational workflow of the solution to stakeholders or end users during the final demonstration phase.

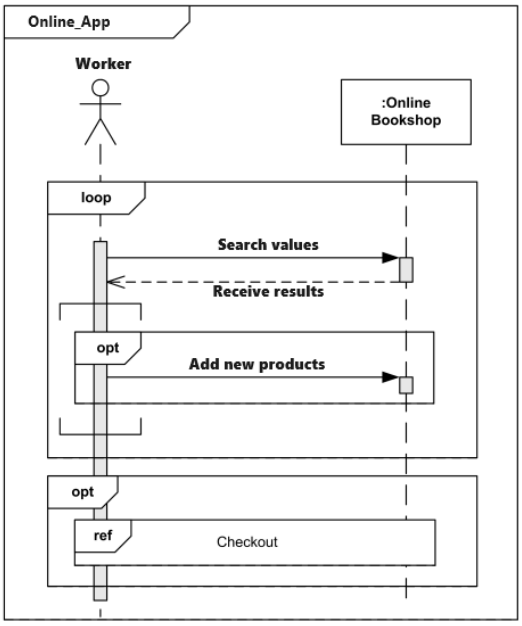


Figure 4. Example of sequence diagram

To develop a sequence diagram, the first step is to identify the actors and functional components involved in the scenario (these functional components were defined in the previous deliverable). This is followed by the definition of the messages or interactions between them, such as method calls or events. Lifelines are created to represent the temporal existence of the actors and objects. The messages in the diagram are then placed in chronological order to show the sequence of interactions over time. If there are conditions or loops, clarifications are added to them in the diagram. A practical example might be the online shopping process (Figure 4), where the user adds products to the cart, initiates the checkout process, the system processes the order, updates the inventory, and the user confirms the purchase. These interactions are represented by arrows between the lifelines of the actors and objects involved.

2.4. Implementation viewpoint approach for AIDEAS

The implementation viewpoint follows the cloud computing patterns. Concerning the Industrial Internet Reference Architecture (IIRA), the implementation viewpoint provides a technical representation of an IIoT system, detailing the system components needed for the provision of the system functionalities as enumerated in the functional viewpoint, and the technologies for implementing the system components. For the AIDEAS solutions, involve the identification of the solution components, the communication between them and the specification of the technologies required to implement the components. The AIDEAS implementation viewpoint describes the following:

- **AIDEAS Solution Architecture:** The architecture of each AIDEAS solution and the interconnections between the different entities making up the solution.
- **Implementation Components:** The implementation components provide the main functionalities of the solution.
- **Technical Description of the Components:** The technical description of the components, detailing the different set of technologies for implementing the components, protocols, interfaces etc.

2.4.1 AIDEAS Solutions Architecture

This section outlines the architecture of AIDEAS Solutions, detailing its primary modules including those for data collection, processing, AI computation, etc., along with their interactions. This aids in formulating a generic approach to representing the architecture of AIDEAS solutions, facilitating the identification of their functional components. The AIDEAS solution is depicted in Figure 5.

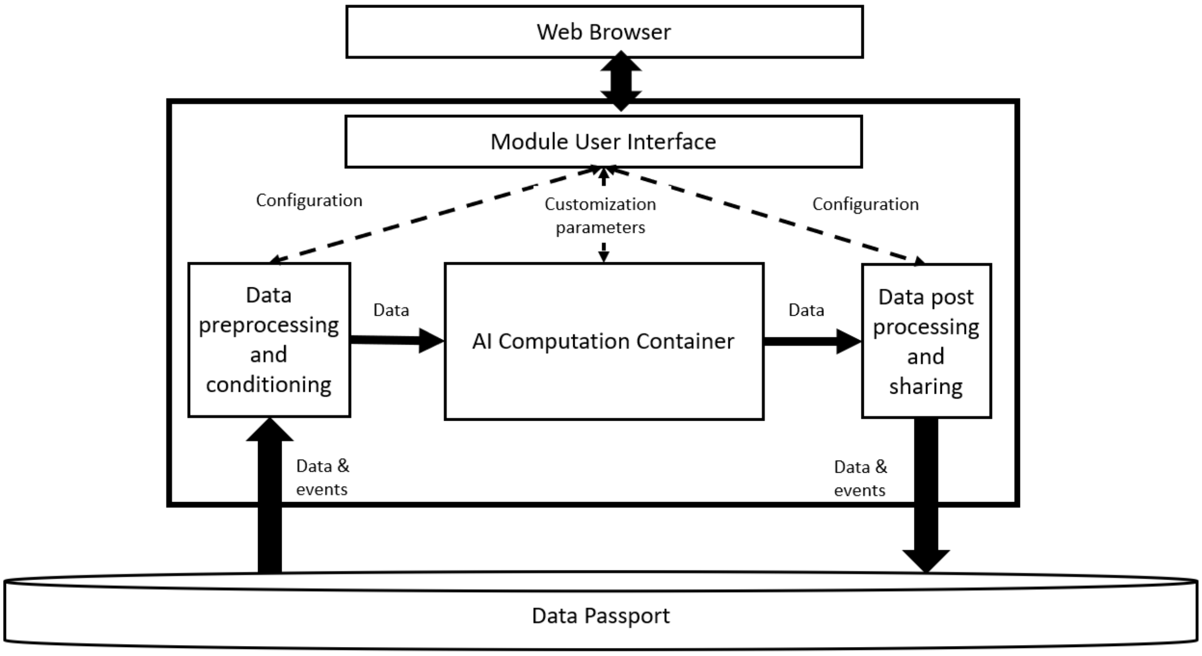


Figure 5. AIDEAS Solutions High-level Architecture

The main modules of the AIDEAS Solutions architecture are described below:

- **Web Browser/Module User Interface:** This is a web-based interface that allows the user to interact with the AIDEAS solutions. The interface also allows the user to configure the AIDEAS solution and observe its status and results of the AI computations.
- **Data Preprocessing and Conditioning**: This is a configurable module for preparing the data for AI training and deployment. This module is configured to listen to predefined events that triggers the AI Computation Container to execute an AI algorithm. This ensures that an AIDEAS solution can be automatically triggered based on the incoming data.
- **Data Post processing and Sharing:** This is a configurable module for preparing the results and output data of the AI computation. These results may also be shared with other AIDEAS solutions. This module can trigger some predefined events to alert other AI solutions about the availability of certain data.
- **AI Computation Container:** This is a set of AI algorithms for performing AI computations (i.e., data analysis and processing) for the purpose of decision-making, leading to the generation of optimal solutions. It is the core aspect of every AIDEAS solutions, which helps the user to optimize different processes.
- **Data Passport:** This is a data and events module that facilitates seamless data exchange and communication between different AIDEAS solutions. It is accessible via APIs and defined topics.

2.4.2 Implementation Components

This section provides a guide to identifying the implementation components of the AIDEAS solutions. The implementation components provide the main functionalities of the solutions. They are the system components required to provide the system functionalities as enumerated in the functional view. Thus, the implementation components can be identified from the functional component diagrams of each AIDEAS solution. They are the components that process the set of input data to generate the desired output data and results of each AIDEAS solution. Table 6 provides a structure for describing the implementation components.

Solution	Implementation Components	Description
AI-XX	Implementation Component 1	Description of Implementation Component 1, including the specification of its functionalities
	Implementation Component 2	Description of Implementation Component 2, including the specification of its functionalities
	Implementation Component 3	Description of Implementation Component 3, including the specification of its functionalities
	Implementation Component 4	Description of Implementation Component 4, including the specification of its functionalities

Table 6. AIDEAS Solutions Implementation Components

2.4.3 Technical Description of the Components

This section serves as a guide to the technical description of AIDEAS solutions. It provides the technical specifications of the components, detailing the different set of technologies used to implement the components, as well as other important aspects such as protocols, interfaces, dependencies, etc. The aim is to achieve a technical representation of the AIDEAS solutions. Table 7 presents a structure for outlining the technical specifications of the AIDEAS solutions. This table is provided for each implementation component. Therefore, the number of tables will depend on the number of implementation components available in each AIDEAS solution.

Implementation component	Implementation Component Name
Description of implementation component	The description of the functionalities associated with the implementation component
Used technologies	The names of the technologies utilised for the implementation of the component
	Dependencies
	Development Language: Libraries: Container:

Technical Description of the Component	Database:
	Interfaces
	User Interface: Synchronous/Asynchronous Interface: Network/Protocols: Data Repository:
	Requires
	e.g., other AIDEAS solutions

Table 7. Technical Description of Implementation Components

3. AIDEAS Business Viewpoints

The fifteen toolkits defined by AIDEAS project are grouped into 5 Suites (Figure 6). The information collected in the surveys is organized according to the different suites.

AIDEAS Industrial Equipment Design Suite	AIDEAS Industrial Equipment Manufacturing Suite	AIDEAS Industrial Equipment Use Suite	AIDEAS Industrial Equipment Repair-Reuse-Recycle Suite
AIDEAS Machine Design Optimiser	AIDEAS Procurement Optimiser	AIDEAS Machine Calibrator	AIDEAS Prescriptive Maintenance
AIDEAS Machine Synthetic Data Generator	AIDEAS Fabrication Optimiser	AIDEAS Condition Evaluator	AIDEAS Smart Retrofitter
AIDEAS CAx Addon	AIDEAS Delivery Optimiser	AIDEAS Anomaly Detector	AIDEAS LCC/LCA/S-LCA
		AIDEAS Adaptive Controller	AIDEAS Disassembler
		AIDEAS Quality Assurance	
AIDEAS Machine Passport			

Figure 6. AIDEAS suites and toolkits

3.1. Stakeholder Identification

The information about the stakeholders (organizations and companies) involved are collected and classified here. The pilots that provided their responses in the first survey about the value proposition are:

- BBM
- MULTISCAN
- D2 TECHNOLOGY
- PAMA S.P.A.

The external stakeholders that are interested in the project and provided their responses in the second survey are:

- Isomat S.A.
- GREEN VEHICLES SRL
- HELLENIC PLAST
- Sisteplant
- Ii40Services
- Poltrona Frau S.P.A.
- BBM

All these organisations are interested in the project and have been asked for information from their business perspective.

3.2. Stakeholder's interest in AIDEAS Suites and Toolkits

As for the value proposition, there are four entities that need to be identified: the jobs to be done, the pains, the gains and the alternative solutions. The jobs to be done refer to the needs or problems that users are trying to solve with each suite. The pains refer to the costs, undesirable situations, risks or other bad experiences that may have occurred before, during or after the jobs to be done. The Gains refer to the desired outcome of using each suite. The Alternative Solution is the way stakeholders are currently solving their problem.

3.3. AIDEAS Industrial Equipment Design Suite

3.3.1 Value Proposition / Pilots' point of view

The Value Proposition for the AIDEAS Industrial Equipment Design Suite has been captured from the pilots as below:

- Jobs to be done:
  1. Optimize the selection of the design solution related to machine components according to the customer real needs.
  2. Try to satisfy efficiently different customer request, improving performance and reducing costs.
  3. Improve design process.
  4. Provide highest quality and reliability, easiest operation, and maintenance.
  5. Create mechanisms to improve the product recording.
- Pains:
  1. Time consuming
  2. Highly customized design solutions that cannot be generalized and managed efficiently by the post-sales assistance.
  3. Having too many design solutions leads to lower reliability.
  4. Adapting a design to custom applications can be frustrating for engineers.
  5. After service costs for mistakes that should had been prevented.
- Gains:
  1. Reduction of redundant activities for engineers.
  2. Reduce design time.
  3. Reducing the waste material to benefit the environment and reduce operating cost.
  4. Improve productivity in all aspects of manufacturing.
  5. Improve reliability and productivity.
  6. Reduce overall costs.
  7. Increase reputation and sales.
- Alternative solutions:

Pilots are using their own market analysis and commercial experience to provide solutions. They mentioned that they are unaware of an alternative solution in the moment because this is cutting edge development.

3.3.2 External Stakeholders' Point of view:

57.1% of external stakeholders indicated that they are interested in the Industrial Equipment Design Suite, while the 28.6% are not sure and need more insight and information and the 14.3% are not interested in this suite.

It is stated that the implementation of the suite will help to automate the process design in coordination with the process control and reduce additional costs for non-value-added activities. It is also stated that some external stakeholders need more information and examples to better understand the use of the suite.

Regarding the **toolkits** of the suite:

- **CAx Addon (AICAx): 75% of external stakeholders are interested in this toolkit, while 25% are not sure and need more insight and information.**
- **Machine Synthetic Data Generator (AI^MDG):** All the external stakeholders are interested in this solution.
- **Machine Design Optimiser (AI^MDO):** All the external stakeholders are interested in this solution.

3.4. AIDEAS Industrial Equipment Manufacturing Suite

3.4.1 Value Proposition / Pilots' point of view

The Value Proposition for the AIDEAS Industrial Equipment Manufacturing Suite has been captured from the pilots as below:

- Jobs to be done:
  1. Optimise the manufacturing process.
  2. Optimise production planning.
  3. Optimise Materials Requirement Planning.
  4. Improve relationship and communication with suppliers and other stakeholders.
  5. Improve safety and security.
- Pains:
  1. The number of steps in the production pipeline is unpredictable and expensive.
  2. A lot of iterations on quality check is needed.
  3. Time consuming.
  4. Difficult data collection
- Gains:
  1. Time optimization.
  2. Reduce lack of inventory (missing components).
  3. Reduction of the machine structure manufacturing time.
  4. Improve manufacturing process with better planning and control.
- Alternative solutions:
  - Software trained employees are using spreadsheet files in the moment to solve problems.

3.4.2 External Stakeholders' Point of view:

85.7% of the external stakeholders stated that they are interested in the Industrial Equipment Manufacturing Suite while the 14.3% are not sure and need more insights and information.

It is indicated that the implementation of the suite will update and modernize the manufacturing processes and improve the allocation of human and material resources, which will lead to increased productivity. They stated that the implementation of the suite would also reduce the productions costs and help companies track their performance better.

Regarding the **toolkits** of the suite:

- **Delivery Optimiser (AI^DO):** 68.7% of the external stakeholders are interested in this toolkit while 16.7% are not sure and need more insights and information and 16.7% are not interested.
- **Fabrication Optimiser (AI^FO):** All the external stakeholders are interested in this solution.

- **Procurement Optimiser (AI^{PO}):** 83.3% of the external stakeholders are interested in this toolkit while 16.7% are not sure and need more insights and information.

3.5. AIDEAS Industrial Equipment Use Suite

3.5.1 Value Proposition / Pilot's point of view

The Value Proposition for the AIDEAS Industrial Equipment Use Suite has been captured from the pilots as below:

- Jobs to be done:
  1. Early prediction of machine failure.
  2. Predictive maintenance.
  3. Reduce the thermal impact on the accuracy of the machine tool for highly precise finishing machining.
  4. Data collection by the acquisition of images at different temperature levels.
  5. Eliminate the cooling system of the machines.
- Pains:
  1. Sudden/unpredictable failures strongly impact the production and may result in high machine downtime.
  2. Sudden/unpredictable failure can result into damage of other components (collateral damage).
  3. Reactive maintenance due to sudden/unpredictable failure is expensive.
  4. Uncontrolled thermal deformation affects the accuracy of the final part.
  5. Inappropriate compensation algorithms and strategies may affect the accuracy of the machined workpiece as an uncontrolled thermal deformation.
  6. Disappointed customers that lead to reduced sales.
- Gains:
  1. Reduce maintenance cost and time.
  2. Increase the accuracy of finishing machining.
  3. Increase the competitiveness and reputation of the company.
  4. Enable new business models (services).
  5. Cost savings.
- Alternative solutions:
  - Reactive maintenance is currently adopted but expensive.

3.5.2 External Stakeholders' Point of view:

85.7% of the external stakeholders stated that they are interested in the Industrial Equipment Use Suite while the 14.3% are not sure and need more insights and information.

It is indicated that the suite is a helpful tool for quality control or defect detection and will be used within the whole quality analysis process. The implementation of the suite will give the opportunity for predictive maintenance, leading to reduction of breakdowns; avoid sudden stoppages of the production and increased productivity. It is mentioned that a hub is needed to manage centrally all operations from production to quality, regulatory and other departments' requirements.

Regarding the **toolkits** of the suite:

- **Quality Assurance (AI^{QA}):** All the external stakeholders are interested in this solution.
- **Adaptive Controller (AI^{AC}):** All the external stakeholders are interested in this solution.
- **Anomaly Detector (AI^{AD}):** All the external stakeholders are interested in this solution.
- **Condition Evaluator (AI^{CE}):** All the external stakeholders are interested in this solution.
- **Machine Calibrator (AI^{MC}):** All the external stakeholders are interested in this solution.

3.6. AIDEAS Industrial Equipment Repair-Reuse-Recycle Suite

3.6.1 Value Proposition / Pilot's point of view

The Value Proposition for the AIDEAS Industrial Equipment Repair-Reuse-Recycle Suite has been captured from the pilots as below:

- Jobs to be done:
  1. Prediction of the Remaining Useful Life of machine components.
  2. Optimize the End-of-Life strategy for machine and components.
  3. Extend the Life of cutting tools/inserts.
  4. Timely assessment of the cutting tool condition (breakage, wear)
  5. Increase the competitiveness and reputation of the company.
  6. Improve relationship and communication with suppliers and other stakeholders.
  7. Improve customer service.
  8. Improve inventory control.
- Pains:
  1. Difficult to manage maintenance schedule.
  2. Wear cutting tools/inserts result in poor surface machining quality and/or uncomplete machining.
  3. Inappropriate End of Life strategy could result in inefficient exploitation of available resources.
  4. Time consuming.
  5. Data gathering.
  6. Not connected systems.
  7. Negative customer reviews leading to decreased sales.
- Gains:
  1. Optimization of maintenance plans.
  2. Reduce cutting tool's waste and connected costs.
  3. Improve machining quality.
  4. Enhance reusability and recyclability of machining components.
  5. Cost savings.
  6. Time and effort saving.
  7. Improve workflow of the company.
  8. Increase sales.
  9. Optimize inventory levels of aftersales products.
- Alternative solutions:
  - Pilots stated that the End-of-Life strategy that is currently adopted is not appropriate. There is an increased effort on preventive maintenance.

3.6.2 External Stakeholders' Point of view:

85.7% of the external stakeholders stated that they are interested in the Industrial Equipment Use Suite while 14.3% are not sure and need more insights and information.

It is indicated that the implementation of the suite will reduce expenses, promote the recycling of equipment, contributing to sustainability purposes and contributing to the Sustainable Development Goals (SDGs). It is mentioned that a direct connection between the Design and Repair-Reuse-Recycle Suite is needed to facilitate the final application of the repair-reuse-recycle services.

Regarding the **toolkits** of the suite:

- **Disassembler (AI^{DIS}):** All the external stakeholders are interested in this solution.
- **LCC/LCA/S-LCA (AI^{LC}):** All the external stakeholders are interested in this solution.
- **Smart Retrofitter (AI^{SR}):** All the external stakeholders are interested in this solution.
- **Prescriptive Maintenance (AI^{PM}):** All the external stakeholders are interested in this solution.

3.7. AIDEAS Machine Passport

3.7.1 Value Proposition / Pilot's point of view

The Value Proposition for the AIDEAS Machine Passport has been captured from the pilots as below:

- Jobs to be done:
    1. Gather relevant information and data in one place.
    2. Unify the relevant information and data between different machines.
    3. Develop a workflow for exchanging the resulting data with the rest of thr machine system.
    4. Test and validate AI applications in order to exploit them and use them effectively.
    5. Improve the system and optimize the information gathered.
    6. Tracking the problems through Machine Passport
- Pains:
    1. Legacy information that is difficult to be interpreted.
    2. Data collection process requires a lot of effort and time
    3. Communication between machine subsystems requires technical support.
    4. Overfitting of algorithms.
    5. Negative customer reviews that lead to decreased sales.
- Gains:
    1. Traceability of the machine life cycle.
    2. Easy retrieval of machine data and information.
    3. Improve efficiency.
    4. Improve workflows.
    5. Time and effort saving.
- Alternative solutions:
    - Pilots stated that legacy methods are used at the moment.

3.7.2 External Stakeholders' Point of view:

85.7% of the external stakeholders stated that they are interested in the Industrial Equipment Use Suite while the 14.3% are not sure and need more insights and information.

It is indicated that the Machine Passport might represent a general trend in engineering that no one should overlook. Machine Passport provides the stakeholders the ability to track the machine's operational data, which will help them in predictive maintenance and in monitoring the production processes. This will help to avoid production downtime, increase productivity, and ensure better quality control and longer equipment lifetime.

3.8. AIDEAS Business viewpoints conclusions

External stakeholders were asked for the easiest way of understanding the value that AIDEAS Suites and Toolkits could bring to them. 42.9% replied that a combination of different technical solutions from different stages of the lifecycle (suites) is what better addresses their problems. 28.6% replied that they prefer individual solutions (toolkits) while 28.6% understand it better when it is presented by use case/suites.

The business perspective has made it possible to identify the suites and toolkits that stakeholders are interested in. Additionally, it has enabled the identification of the specific aspects of improvement that stakeholders expect.

There is an interest in every Suite and Toolkit by the Pilots and the external stakeholders, although in some cases more information, insights and examples are needed so that the solutions would be more understandable. In this case, the Industrial Equipment Use Suite, Industrial Equipment Manufacturing Suite Industrial Equipment Repair-Reuse-Recycle Suite, and Machine Passport gathered equally high interest while the Industrial Equipment Design Suite raised less interest among the external stakeholders.

Among the toolkits the higher interest (equally distributed) is shown on the toolkits below:

- **Machine Synthetic Data Generator (AI^{MDG})**
- **Machine Design Optimiser (AI^{MDO})**
- **Fabrication Optimiser (AI^{FO})**
- **Quality Assurance (AI^{QA})**
- **Adaptive Controller (AI^{AC})**
- **Anomaly Detector (AI^{AD})**
- **Condition Evaluator (AI^{CE})**
- **Machine Calibrator (AI^{MC})**
- **Prescriptive Maintenance (AI^{PM})**

4. AIDEAS Usage, functional and Implementation Viewpoint

4.1. Machine Design Optimiser – AI^{MDO}

4.1.1 Usage Viewpoint

The Usage Activity diagram (Figure 7) within AI^{MDO} demonstrates the interaction between the different tasks and roles. No pilot distinction is made, as these are the same independent of the use case. This diagram illustrates the dynamic flow of interactions, offering a representation of the activities executed by the solution and providing a clear understanding of the activities executed using the AI^{MDO} solution.

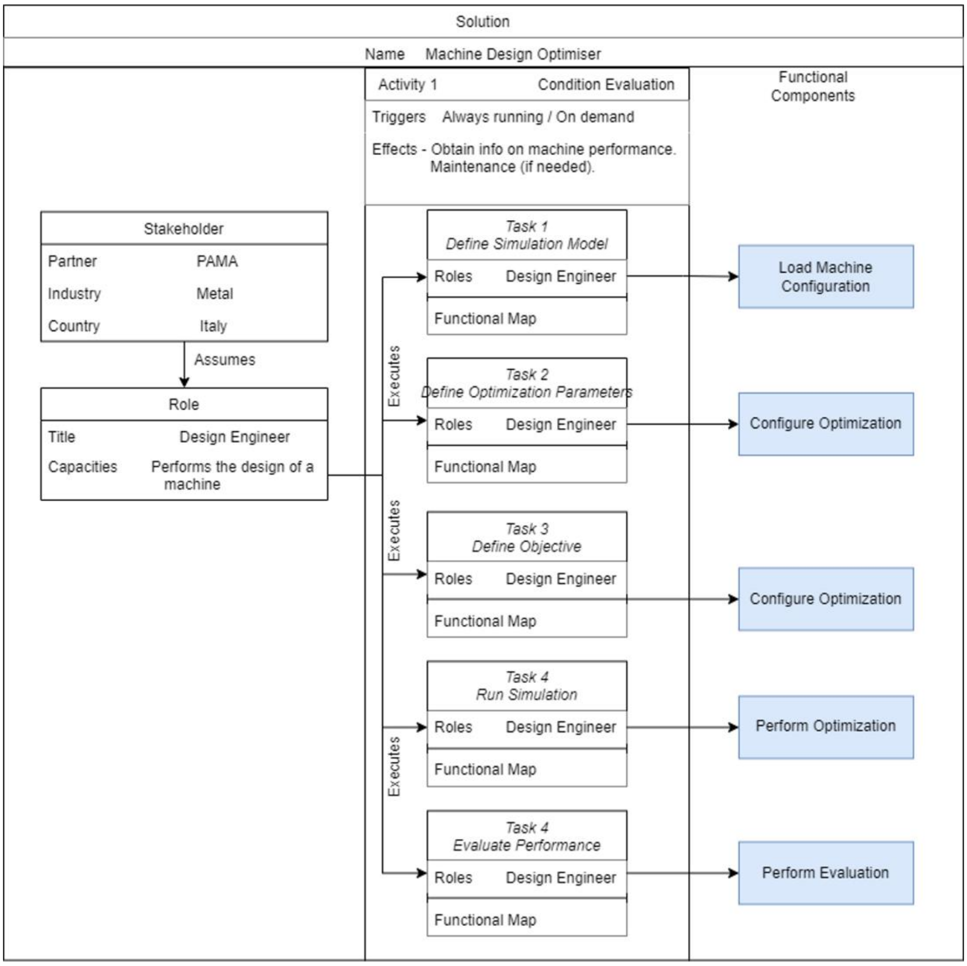


Figure 7. AI^{MDO}Usage Viewpoint Activity Diagram

4.1.2 Functional Viewpoint

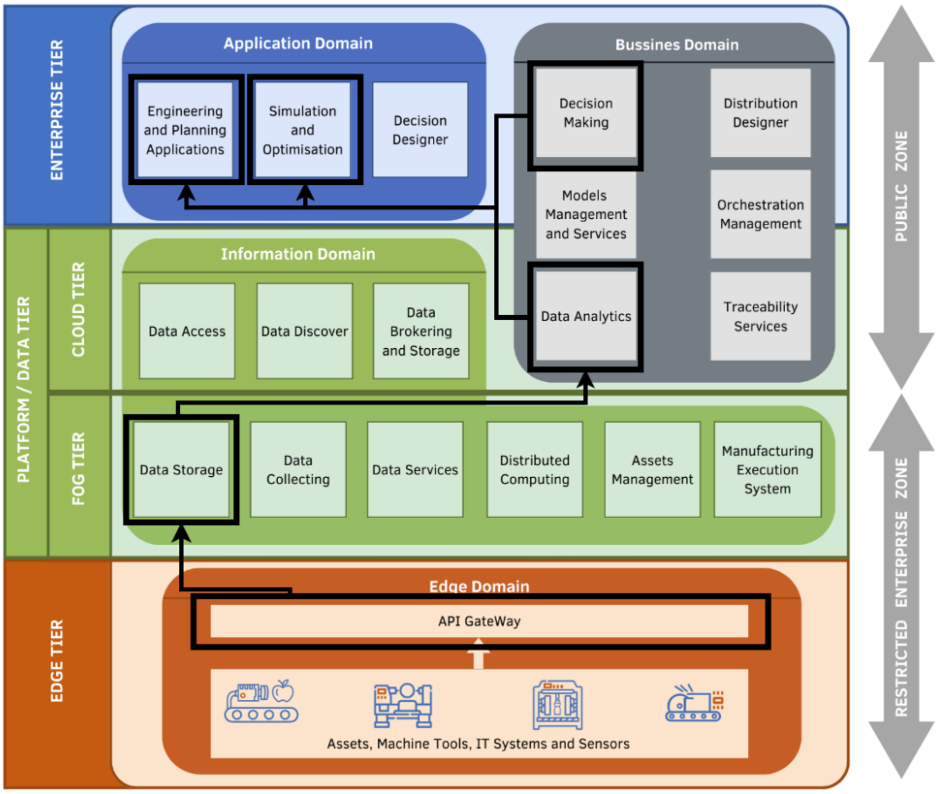


Figure 8. Data Flow AI^{MDO}

Machine Design Optimiser gets machine configuration data, with information on the search space for the optimisation from a .xlsx file, as well as models of the machines in .exe files, from a data storage system. It uses these data to perform an optimisation that helps engineers to make decisions in the design process, determining the most suitable machine configurations according to a specific driver or variable, included in the model.

4.1.2.1 Data structure of AI^{MDO}

Format	Input/Output	Example
XLSX	Input	Parameter, max value, min value, max iteration 1, 15.3, 23.8, 100
EXE	Input	Executable file containing the model that will be used by the AI^{MDO} to simulate different scenarios. This executable file takes as an input the selected values for the parameters within the search space and provides as output the result of the simulation (as a single value).
XLSX	Output	Parameter 1, parameter 2, ..., parameter N, objective 16.7, 2.3, ..., 8.7, 9.4

Table 8. Input / Output Data Format AI^{MDO}

4.1.2.2 AI^{MDO} Software Requirements

Software Component	Description/Role	Required Version/Configuration	Dependencies
Windows OS	Operating system needed to use the tool	Windows 10 Pro	N/A
Docker	Build, share and run containers	latest	N/A

Table 9. Software requirements AI^{MDO}

4.1.2.3 AI^{MDO} Lifecycle

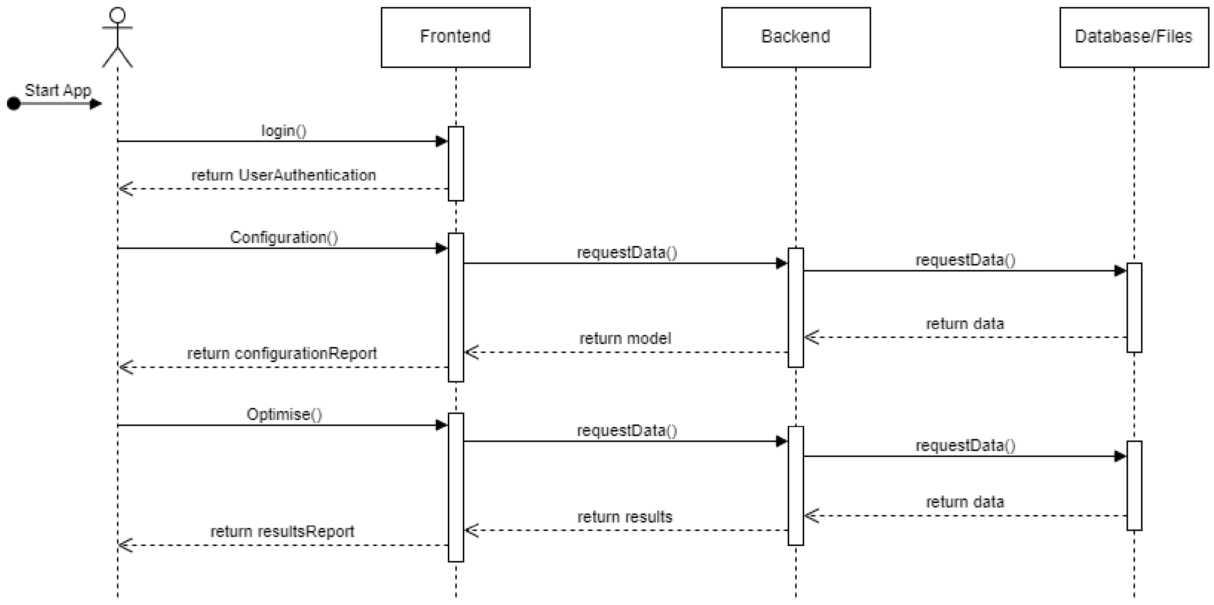


Figure 9. AI^{MDO} Lifecycle

This sequence diagram illustrates the flow of interactions between the User, Frontend, Backend and Data/Files objects during the main processes: login into the AIDEAS platform, configure the optimisation and apply it.

4.1.2.4 Objects

- **User**: Represents the User interacting with the application.
- **Frontend**: Represents the user interface and the presentation layer.
- **Backend**: Represents the application logic and the server layer.
- **Database/Files:** Represents the data or file storage layer.

4.1.2.5 Description - Login

Objects	Description
Start	The User logs into the AIDEAS platform.

User – Frontend Interaction	The User enters user and password.
Frontend - User Interaction	The Frontend shows the User the response from the login action, indicating whether it is logged in or not.
Completion	The application is ready to receive new requests from the User.

Table 10. Life-Cycle description Login AI[MDO]

4.1.2.6 Description – Optimisation Configuration

Objects	Description
Start	The User initiates the action to configure the optimisation
User – Frontend Interaction	The User selects which kind of model wants to use for the optimisation
Frontend - Backend Interaction	The Frontend sends a request to the Backend to process the selected model
Backend - Data/Files Interaction	The Backend requests the storage layer (Data/Files) to authorize the connection or to prepare to receive the data. Data/Files is waiting for the data transfer.
Data/Files - Backend Response	Data/Files sends a response to the Backend
Backend - Frontend Response	The Backend gets the corresponding data and prepares the parameters to be seen by the user
Frontend - User Interaction	The Frontend shows the User the parameters of the model and their corresponding value ranges, having the possibility to edit/save them for its future use
Completion	The application is ready to receive new requests from the User

Table 11. Life-Cycle description Anomaly Detector Parametrization AI[MDO]

4.1.2.7 Description – Perform Optimisation

Objects	Description
Start	The User initiates the action of performing the optimisation
User – Frontend Interaction	The User defines the ranges of the values of the parameters of the model and the number of iterations, and launches the optimisation
Frontend - Backend Interaction	The Frontend sends a request to the Backend to perform the optimisation
Backend - Data/Files Interaction	The Backend requests the storage layer (Data/Files) to authorize the connection or to prepare to receive the data. Data/Files is waiting for the data transfer.

Data/Files - Backend Response	Data/Files sends a response to the Backend.
Backend - Frontend Response	The Backend gets the corresponding data and performs the optimisation
Frontend - User Interaction	The Frontend shows the User the response from the application, obtaining the optimisation report
Completion	The application is ready to receive new requests from the User.

Table 12. Life-Cycle description Perform Anomaly Detection AI[MDO]

4.1.3 Implementation Viewpoint

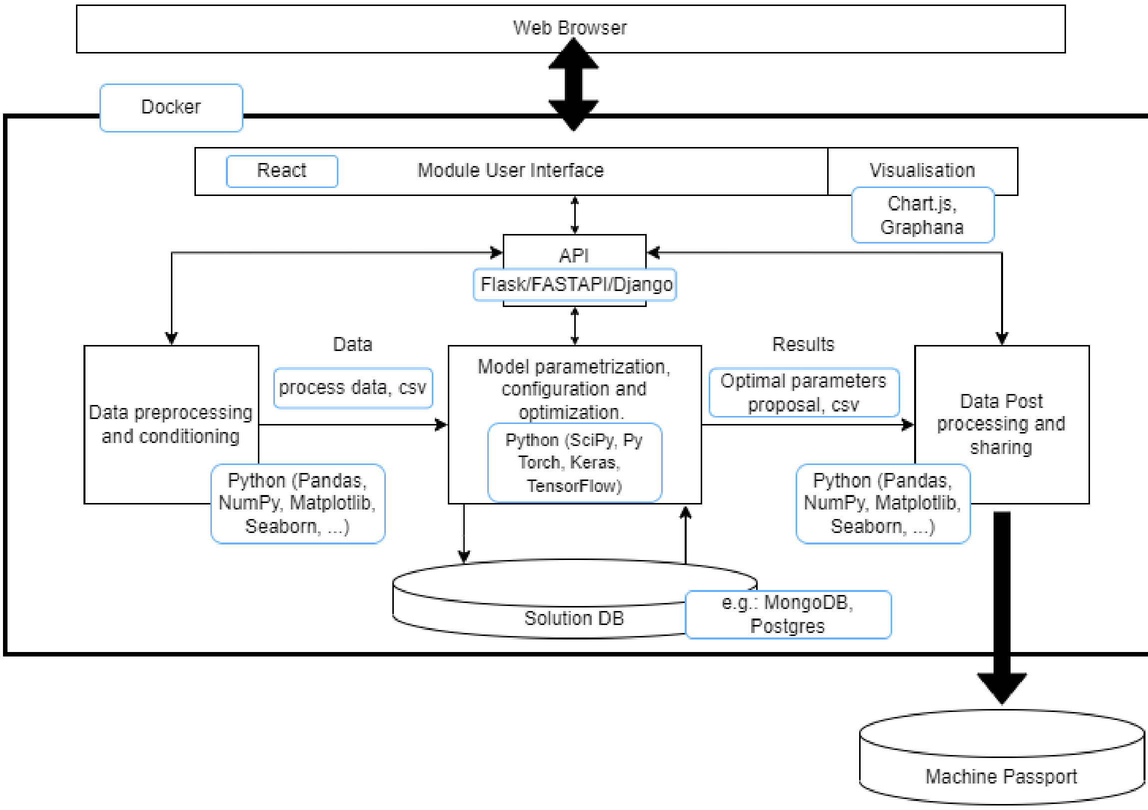


Figure 10. AI[MDO] Implementation Architecture

For AI[MDO], the main development language is Python. Numpy and pandas are used for data management and basic operations, sklearn, keras and tensorflow are used for model training. The solution is containerized using Docker. Input data will be provided by excel and executable files provided by the pilots and final users. And finally, the results and solution's outputs will be sent to the Machine Passport.

4.1.3.1 AI^{MDO} Implementation Components

Implementation Components	Description
Load Machine Configuration	This component facilitates the reading of the inputs required to configure the machine (excel and executable files).
Configure Optimization	This component allows defining the conditions in terms of ranges of acceptable parameter values.
Perform Optimization	This component launches the optimization solver, given the selected model and the configuration of the ranges of values for the parameters.
Perform Evaluation	This component extracts the results of the optimization and provides the best results.

Table 13. AI^{MDO} Implementation Components

4.1.3.2 Technical Description of AI^{MDO} Components

Following the identification and specification of the implementation components in the previous section, the technical description of each component is provided in the following tables. It details the set of technologies required to implement each of the implementation components.

Implementation component	Load Machine Configuration
Description of implementation component	This component facilitates the reading of the inputs required to configure the machine (excel and executable files).
Used technologies	Python
Technical Description of the Component	**Dependencies**
	Development Language: Python Libraries: Pandas, numpy Container: Docker
	Interfaces
	User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS
	Requires
	N/A

Table 14. Technical Description of AI^{MDO} "Load Machine Configuration" Implementation Component

Implementation component	Configure Optimization
Description of implementation component	This component allows defining the conditions in terms of ranges of acceptable parameter values.
Used technologies	Python
Technical Description of the Component	**Dependencies**
	Development Language: Python Libraries: Pandas, numpy Container: Docker
	Interfaces
	User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS
	Requires
	Load Machine Configuration

Table 15. Technical Description of AI^{MDO} "Configure Optimization" Implementation Component

Implementation component	Perform Optimization
Description of implementation component	This component launches the optimization solver, given the selected model and the configuration of the ranges of values for the parameters.
Used technologies	Python
Technical Description of the Component	**Dependencies**
	Development Language: Python Libraries: Pandas, numpy, scipy, keras, tensorflow Container: Docker
	Interfaces
	User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS
	Requires
	Configure Optimization

Table 16. Technical Description of AI^{MDO} "Perform Optimization" Implementation Component

Implementation component	Perform Evaluation	
Description of implementation component	This component extracts the results of the optimization and provides the best results.	
Used technologies	Python	
Technical Description of the Component	Dependencies	
	Development Language: Python Libraries: Pandas, numpy Container: Docker Database: N/A	
	Interfaces	
	User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: N/A	
	Requires	
	Perform Optimization	

Table 17. Technical Description of AI^{MDO} "Perform Evaluation" Implementation Component

4.2. Machine Synthetic Data Generator – AI^{MDG}

4.2.1 Usage Viewpoint

Figure 11 depicts the Usage Activity Diagram for the Machine Synthetic Data Generator. Note that Tasks labelled with "Manage" indicate a set of functions that are clustered here for better overview.

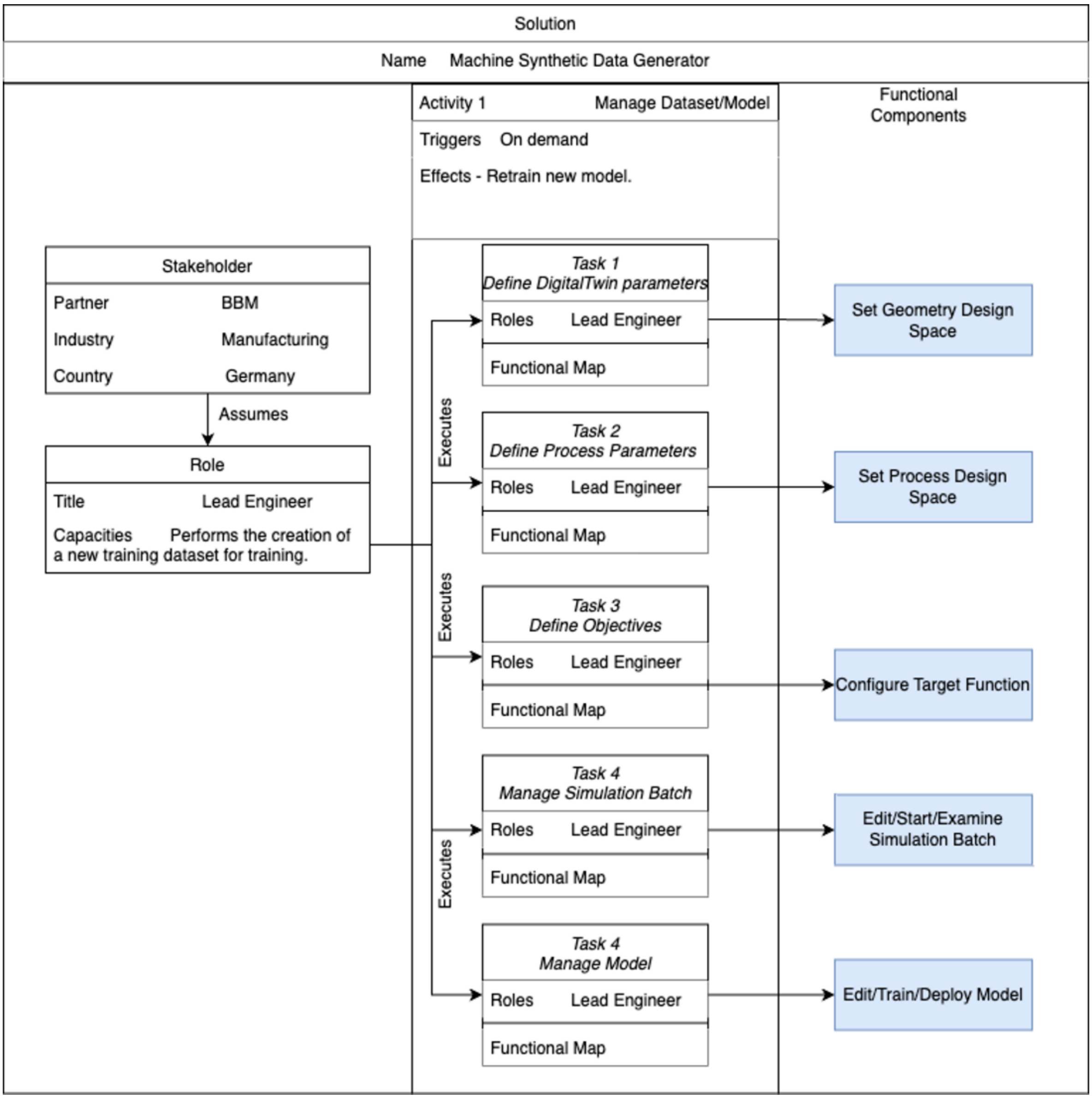


Figure 11. AI^{MDG} Usage Viewpoint Activity Diagram

4.2.2 Functional Viewpoint

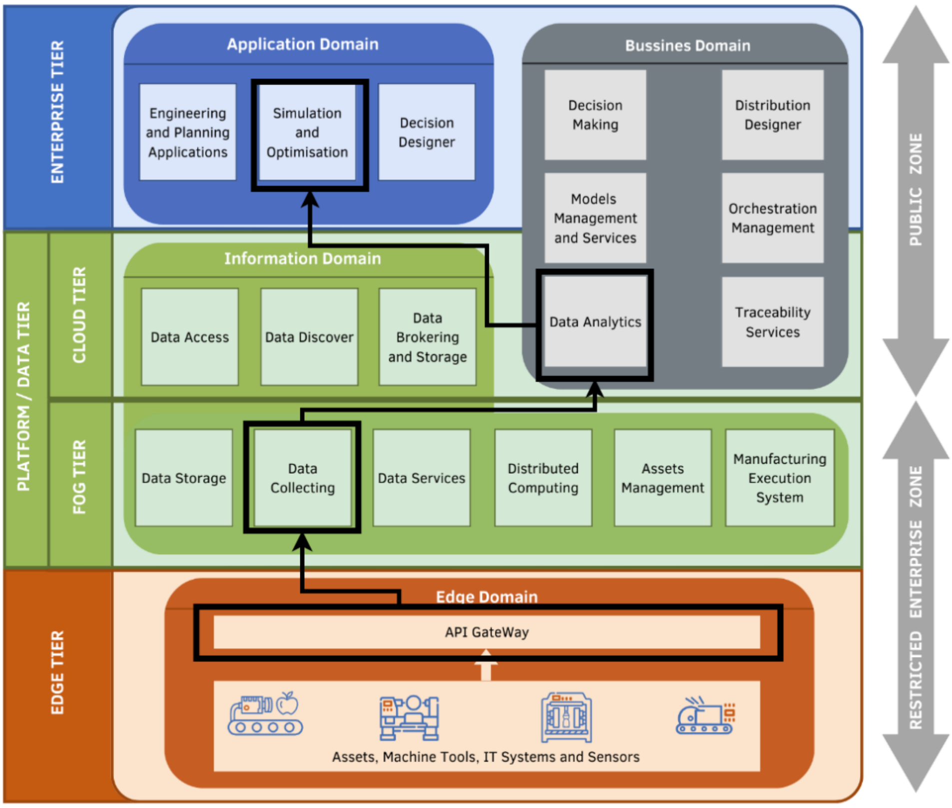


Figure 12. Data Flow AI^{MDG}

The Machine Synthetics Data Generator receives machine configuration data with transfer model domain information for simulation from the trained design models from a data storage system. Using this data, it performs simulations and optimisations to help engineers make decisions in the design process by determining an appropriate prediction of the most suitable design configurations and generating a prototype depending on variable geometry data sets.

4.2.2.1 Data structure of AI^{MDG}

In the examples below, single instances of the set of dictionaries stored in the JSON files dataset are depicted, up to the point where BASE64 data is attached. The underlying grammar of the dictionaries may vary from one AI^{MDG} use case to another and depends on the selected dimensional parameters for the requested dataset. Finally, the examples have been simplified to be able to fit in here.

Format	Input/Output	Example
JSON	Input	{ "simulation_id": 12866, "name": "340.508_13mk", "company": "ANONYMIZED", "company_id": 177, "simulation_form": "Automatisch|Profilwerkzeuge", "submit_date": "2024-03-01 14:50:52", "status": "complete",

		"input_parameter": {
		"temperature": {
		"type": "number",
		"name": {
		"de": "Eingangstemperatur [\u00b0C]",
		"en": "Inflowtemperature [\u00b0C]"
		},
		"placeholder": "-",
		"var": "temperature",
		"optional": false,
		"value": 150
		}, "placeholder": "",
		"template": [
		{
		"type": "materials",
		"name": {
		"de": "Materialauswahl",
		"en": "Material selection"
		},
		"placeholder": "-",
		"var": "material",
		"optional": false
		},
		{
		"type": "number",
		"name": {
		"de": "Durchsatz [kg\/h]",
		"en": "Throughput [kg\/h]"
		},
		"placeholder": "-",
		"var": "massthroughput",
		"optional": false
		}
		],
		"buttontext": {
		"de": "Weiterer Inflow",
		"en": "Add inflow"
		},
		"var": "inflow",
		"multi": null,
		"value": [
		{
		"material": {
		"type": "materials",
		"name": {

		"de": "Materialauswahl",
		"en": "Material selection"
		},
		"placeholder": "-",
		"var": "material",
		"optional": false,
		"value": "658"
		},
		"massthroughput": {
		"type": "number",
		"name": {
		"de": "Durchsatz [kg\/h]",
		"en": "Throughput [kg\/h]"
		},
		"placeholder": "-",
		"var": "massthroughput",
		"optional": false,
		"value": 25
		}}]}},
		"output_parameter": [],
		"simulation_form_var": "gendie|general",
		"application": "Gendie",
		"results": [
		{
		"file": "data:application\/x-zip-compressed;base64,UEsDBD8AAg...}]

JSON	output	{
		"pressure": {
		"inflows": [{
		"pressure_diff": 15.209617538382691
		}]
		},
		"velocity": {
		"outflow_flowrate": 0.1222477668638478,
		"outflow_mean": 0.0920015349984169}}

Table 18. Input / Output Data Format AI[MDG]

4.2.2.2 AI[MDG] Software Requirements

Software Component	Description/Role	Required Version/Configuration	Dependencies
FeatFlower	Simulation Worker Code.	Up-to-date checkout.	ParMetis

Table 19. Software requirements AI[MDG]

4.2.2.3 AI^MDG Lifecycle

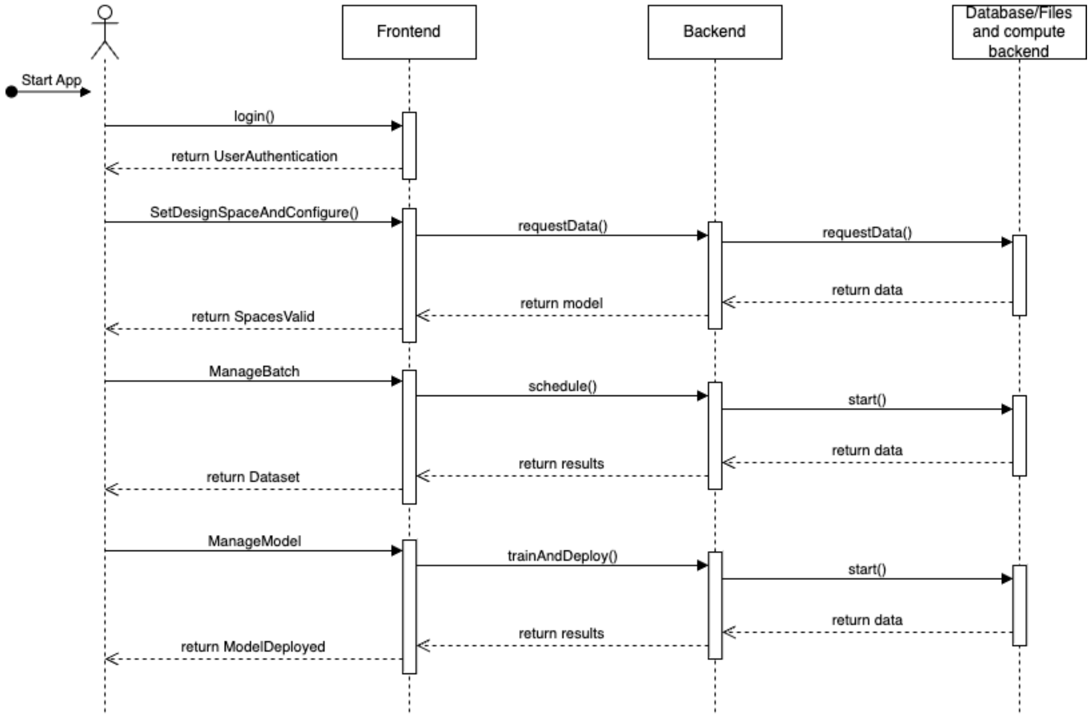


Figure 13. AI^MDG Lifecycle

The AI^MDG lifecycle is characterised by the lifecycle of configuration, synthetic dataset, and readily trained model.

4.2.2.4 Objects

- **User**: Represents the User interacting with the application.
- **Frontend**: Represents the user interface and the presentation layer.
- **Backend**: Represents the application logic and the server layer.
- **Database/Files:** Represents the data or file storage layer.

4.2.2.5 Description – Login:

Objects	Description
Start	The user opens the "manage dataset" UI element
User - Frontend Interaction	The user selects parameters for input and output layers
Frontend - User Interaction	The Frontend shows the User the response from the login action, indicating whether it is logged in or not.
Completion	The application is ready to receive new requests from the User

Table 20. Life-Cycle description Login AI^MDG

4.2.2.6 Description – Design Space Configuration:

Objects	Description
Start	The User initiates the action to configure the material parameters
User - Frontend Interaction	The User defines the values of the material and process parameters and launches the simulation
Frontend - Backend Interaction	The Frontend sends a request to the Backend to perform the simulation
Backend - Data/Files Interaction	The Backend requests the storage layer (Data/Files) to authorize the connection or to prepare to receive the data. Data/Files is waiting for the data transfer.
Data/Files - Backend Response	Data/Files sends a response to the Backend.
Backend - Frontend Response	The Backend gets the corresponding data and performs the simulations
Frontend - User Interaction	The Frontend shows the User the response from the application, obtaining the simulations report
Completion	The application is ready to receive new requests from the User

Table 21. Life-Cycle description Design Space Configuration AI[MDG]

4.2.2.7 Description – Manage Batch:

Objects	Description
Start	The user initiates the action to configure the application.
User - Frontend Interaction	The user explicitly orders to act on the previously configured and validated batch of simulations.
Frontend - Backend Interaction	The simulations are scheduled in the backend for later execution in the computing backend
Backend - Data/Files Interaction	The backend starts the simulations in the computing backend
Data/Files - Backend Response	The computing backend returns simulated data to the Backend
Backend - Frontend Response	The backend post-processes the simulation results and returns them to frontend
Frontend - User Interaction	The front end shows the user the response of the application and the simulation report
Completion	The application is ready to receive new requests from the User

Table 22. Life-Cycle description Manage Batch AI[MDG]

4.2.2.8 Description – Manage Model:

Objects	Description
Start	The user starts examining the finished dataset
User - Frontend Interaction	The user selects test/train split and explicitly orders training
Frontend - Backend Interaction	The Frontend orders training and deploying the datasets in the backend.
Backend - Data/Files Interaction	The backend starts the training iterations of the data sets in the computing backend database
Data/Files - Backend Response	Training is burned in the compute backend
Backend - Frontend Response	The backend processes the available simulation results and adds them to the training data set
Frontend - User Interaction	Frontend shows the trained data model to examine by user and explicitly deploy for usage in other services
Completion	STEP file of the trained data model can be downloaded and opened in the CAD software

Table 23. Life-Cycle description Manage Model AI^{MDG}

4.2.3 Implementation Viewpoint

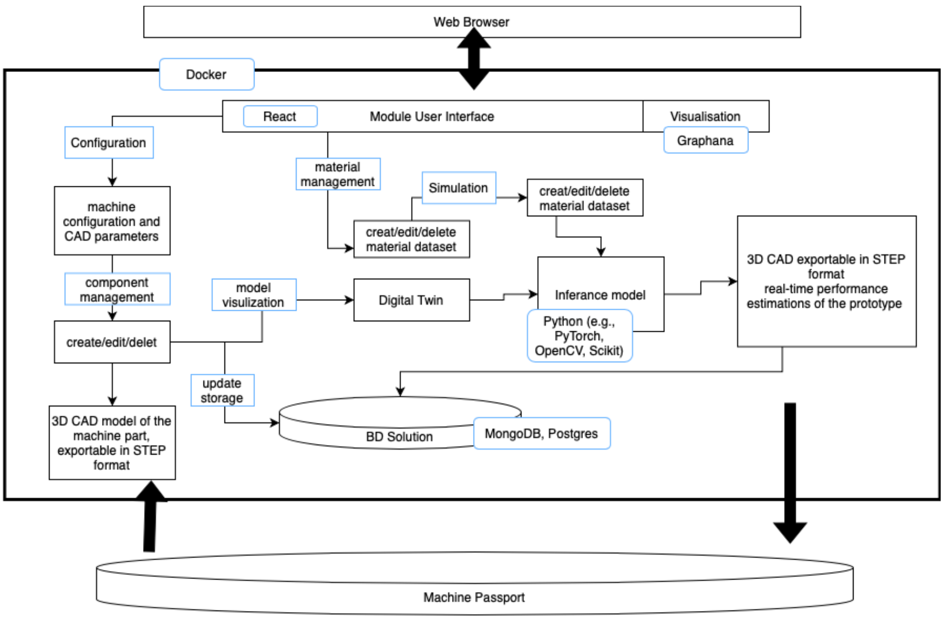


Figure 14. AI^{MDG} Implementation Architecture

At the core of the application lies a configuration module that allows users to specify parameters and settings for machines and their components. This module is directly responsible for the creation, editing, and deletion of 3D CAD models of machine parts, which are made available in STEP format. These models serve as a digital representation of the physical components, facilitating accurate simulations and visualizations.

The web application integrates a material management system that enables users to create, modify, and delete datasets pertaining to different materials. This data is crucial for simulating the real-world behaviour of machine parts under various conditions and loads. The simulation component utilizes these datasets to model and predict the performance of the materials within the configured machine parameters.

For visual interpretation, the system incorporates Graphana, a visualization platform that displays the simulation results in an intuitive and interactive format. This aids in the rapid assessment of the machine's performance characteristics and potential areas for improvement. The digital twin aspect of the architecture reflects a virtual representation of the machines that synchronizes with the real-world counterpart. This dynamic model allows for real-time monitoring and analysis, ensuring that the digital and physical entities are aligned in terms of performance and operational data.

The inference model, powered by Python libraries such as PyTorch, OpenCV, and Scikit, processes the input data to make predictions about the machine's behaviour. This model is a crucial aspect of the system's ability to provide actionable insights and enhance decision-making processes. Underpinning the entire application is a robust database solution that employs both MongoDB and Postgres to handle the diverse data storage requirements. These databases ensure the integrity and accessibility of the configuration details, material properties, and simulation results.

The web application is containerized using Docker, which encapsulates the dependencies and environment needed to run the software reliably across different computing environments. The use of REACT in the development of the user interface suggests a responsive and dynamic experience for the users, enabling them to interact with the system efficiently through a web browser.

4.2.3.1 AI^{MDG} Implementation Components

Implementation Components	Description
Manage Components	Create, edit, delete components and/or parameters that are supposed to be added into a dataset.
Manage Materials	Create, edit, delete process parameters for dataset.
Manage Digital Twins	Create, edit delete digital twins and/or their parameters to be added into the dataset.
Manage Simulations	Start, examine, and control the simulations in a batch belonging to the synthetic dataset, start training and deploy the model.

Table 24. AI^{MDG} Implementation Components

4.2.3.1 Technical Description of the AI^{MDG} Components

Following the identification and specification of the implementation components in the previous section, the technical description of each component is provided in the following tables. It details the set of technologies required to implement each of the implementation components.

Implementation component	Manage Components
Description of implementation component	Create, edit, delete components and/or parameters that are supposed to be added into a dataset.
Used technologies	FreeCAD, python, full web app
Technical Description of the Component	Dependencies
	Development Language: python, typescript Libraries: open3d Container: docker Database: MariaDB, MongoDB
	Interfaces
	User Interface: Web UI, WebGL Synchronous/Asynchronous Interface: Synchronous Network/Protocols: https Data Repository: static assets, backend bulk storage

Table 25. Technical Description of AI[MDG] "Manage Components" Implementation Component

Implementation component	Manage Materials
Description of implementation component	Create, edit, delete process parameters for dataset.
Used technologies	python, full web app
Technical Description of the Component	Dependencies
	Development Language: python, typescript, php Libraries: laravel Container: docker Database: MariaDB
	Interfaces
	User Interface: Web UI, Synchronous/Asynchronous Interface: Synchronous Network/Protocols: https

Table 26. Technical Description of AI[MDG] "Manage Materials" Implementation Component

Implementation component	Manage Digital Twins
Description of implementation component	Create, edit delete digital twins and/or their parameters to be added into the dataset.
Used technologies	python, full web app, WebGL, FreeCAD

Technical Description of the Component	Dependencies	
	Development Language: python, typescript, php Libraries: laravel, nuxt Container: docker Database: MariaDB	
	Interfaces	
	User Interface: Web UI, WebGL Synchronous/Asynchronous Interface: Synchronous Network/Protocols: https Data Repository: --	

Table 27. Technical Description of AI^{MDG} "Manage Digital Twins" Implementation Component

Implementation component	Manage Simulations
Description of implementation component	Start, examine, and control the simulations in a batch belonging to the synthetic dataset, start training and deploy the model.
Used technologies	python, full web app, WebGL, FreeCAD, matplotlib, pyvista, FeatFlower, pytorch
Technical Description of the Component	Dependencies
	Development Language: python, typescript, php Libraries: laravel, nuxt, pytorch, pyvista Container: docker Database: MariaDB
	Interfaces
	User Interface: Web UI, WebGL Synchronous/Asynchronous Interface: Synchronous Network/Protocols: https Data Repository: --

Table 28. Technical Description of AI^{MDG} "Manage Simulations" Implementation Component

4.3. CAx Addon – AI^{CAx}

4.3.1 Usage Viewpoint

The Usage Activity diagram (Figure 15) within AI^{CAx} depicts the interaction between the different tasks and roles within the selected pilot. This diagram illustrates the dynamic flow of interactions, offering a representation of the activities executed by the solution providing a clear understanding of the activities executed using the AI^{CAx} solution.

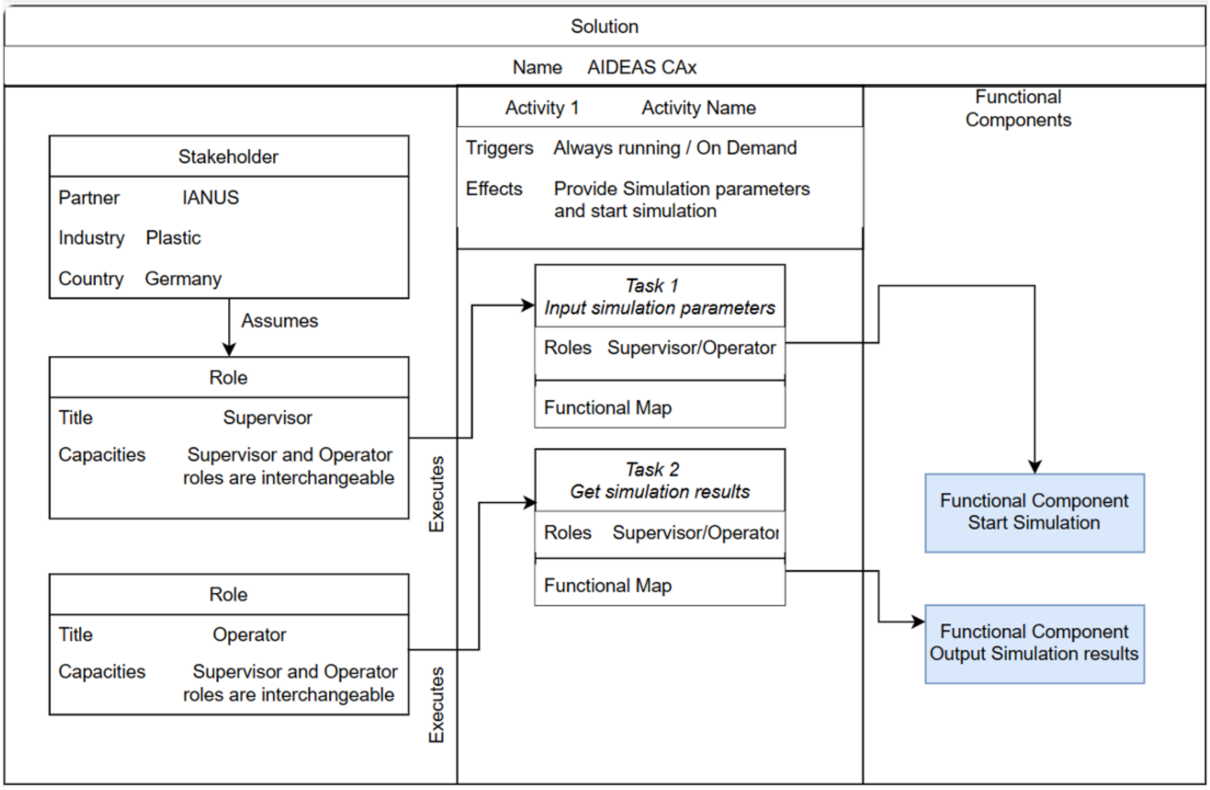


Figure 15. AI^{CAx} Usage Viewpoint Activity diagram

4.3.2 Functional Viewpoint

The AI^{CAx} component implements integration and Machine-2-Machine (M2M) communication between the AI^{MDG} component and the external tools (e.g., external simulation tools). The implementation of the AI^{CAx} adds high-added value to the iterative and manual model optimization process. This addon depends on (i) chosen AI^{CAx} software (e.g., SolidWorks, AD Fusion etc.) and the requirements imposed by AI^{CAx} add-on development capabilities and limitations of the chosen AI^{CAx} (ii) AI^{MDG} solution limitations to be consumed by the usage of the AI^{CAx} module. Currently, AD Fusion is supported.

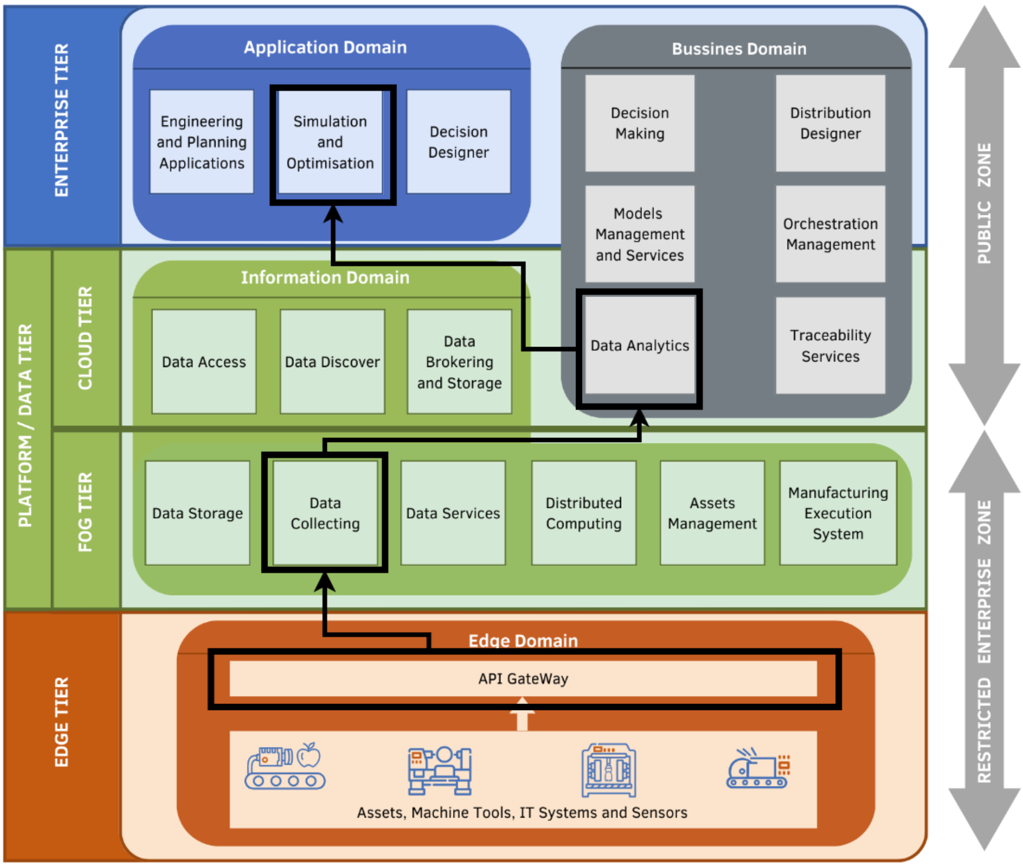


Figure 16. Data Flow AI^{CAx}

4.3.2.1 Data structures of AI^{CAx}

Format	Input/Output	Example

JSON	Input	List of digital twins:
		{
		"success":[
		{
		"digital_twin_id":9392,
		"name":"ExampleAIDEAS",
		"created_at":"2023 Oct 4 - 12:02:03 CEST"
		}
		]
		}

JSON	Output	Create digital twin request:
		{
		"name": "Golf Ball",
		"file": "STEP file content”
		}

JSON	Input	List of usable materials: { "success":[

		{
		"material_id":1627,
		"name":"PE"
		}
		]
		}

JSON	Input	List of usable forms:
		
		{
		"success":[
		{
		"form_id":55,
		"name":"Automatisch|Profilwerkzeuge",
		"parameters":{
		"temperature":"number",
		"inflow":"multiblock"
		}
		}
		]
		}

JSON	Input	List of simulations for selected digital twin:
		{
		"success":[
		{
		"simulation_id":11452,
		"status":"complete",
		"materials":[
		{
		"name":"PE",
		"value":1627
		}
		],
		"parameters":{
		"temperature":200,
		"inflow":[
		{
		"material":"1627",
		"massthroughput":100
		}
		]
		},
		"created_at":"2023 Nov 8 - 14:39:54 CET"
		}
		]}

JSON	Output	Start simulation request:

		{
		"digital_twin_id":"9392",
		"parameters":{
		"temperature":200,
		"inflow":[
		{
		"material":"1627",
		"massthroughput":100
		}
		]
		}
		}

JSON	Input	Simulation status:
		{
		"success": {
		"status": "complete"
		}
		}

JSON	Input	Simulation result download url:
		{
		"success":
		"https://backend.simod.de/api/automated_user/result/ce932699-7dd3-408f-ae58-80a15d44d396"
		}

PDF	Input	Simulation result: PDF document

Table 29. Input / Output Data Format AI^{CAx}

4.3.2.2 AI^{CAx} Software Requirements

Software Component	Description/Role	Required Version/Configuration	Dependencies
Windows OS	Microsoft operating system its need to use specific tool	Windows 10	Autodesk Fusion

Table 30. Software Requirements AI^{CAx}

4.3.2.3 AI CAx Lifecycle

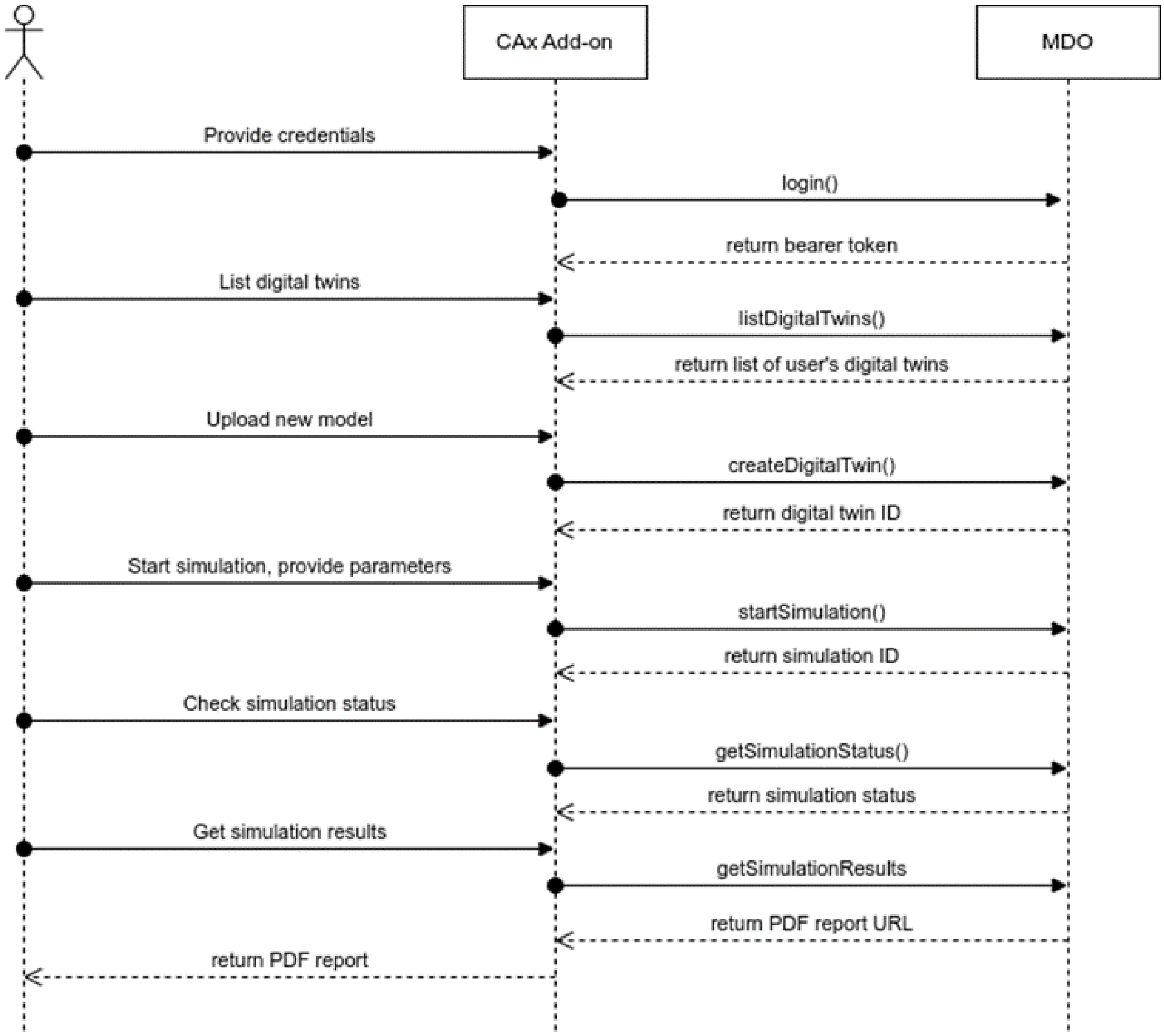


Figure 17. Data Flow AI CAx

The AI CAx component provides Machine-2-Machine (M2M) communication between the AI MDG component and the external tools (e.g., external simulation tools). The current implementation is a AutoDesk Fusion plugin that integrates with pilots AI MDG backend and is aimed to automate the process of product design optimization. The deployment is done through the AutoDesk Fusion itself.

4.3.2.4 Objects

- **User**: represents the user interacting with the AI CAx add-on.
- **CAx add-on** represents the add-on for CAD/CAE/CAM software.
- **MDO**: represents Machine Design Optimizer solution.

4.3.2.5 Description – Login:

Objects	Description
Start	The User opens Login view in the CAx
User - CAx interaction	The User enters username and password into login form
CAx – MDO interaction	The CAx sends login request to the MDO
MDO – CAx response	MDO validates the user credentials and returns a bearer token if credentials are correct. Otherwise, it returns an error code.
CAx – User response	CAx displays login success message
Completion	CAx stores the bearer token and uses it in further communication with MDO

Table 31. Life-Cycle description Login AI^{CAx}

4.3.2.6 Description – Create digital twin:

Objects	Description
Start	The User opens 'Create digital twin' view in the CAx
User - CAx interaction	The User enters digital twin name
CAx – MDO interaction	The CAx sends create-digital-twin request to the MDO containing the model in STEP format
MDO – CAx response	MDO creates digital twin for the specified model and returns digital twin ID
CAx – User response	CAx informs the User that digital twin has been created
Completion	New digital twin is displayed in the Digital twins table

Table 32. Life-Cycle description Create digital twin AI^{CAx}

4.3.2.7 Description – Start simulation:

Objects	Description
Start	The User opens 'Start simulation' view in the CAx
User - CAx interaction	The User enters simulation parameters
CAx – MDO interaction	The CAx sends start-simulation request to the MDO containing the digital twin ID and simulation parameters
MDO – CAx response	MDO validates the simulation parameters, starts simulation and returns simulation ID
CAx – User response	CAx informs the User that simulation has been started
Completion	New simulation is displayed in the Simulations table

Table 33. Life-Cycle description Start simulation AI^{CAx}

4.3.2.8 Description – Retrieve simulation results:

Objects	Description
Start	The User opens 'List simulations' view in the CAx
User - CAx interaction	The User finds simulation in the table and checks simulation status
CAx – MDO interaction	The CAx sends get-simulation-results request to the MDO containing the simulation ID
MDO – CAx response	MDO returns URL to the PDF document with simulation results
CAx – User response	CAx opens PDF document in the Web Browser
Completion	PDF report is downloaded and opened in the Web Browser

Table 34. Life-Cycle description Retrieve simulation results AI^CAx

4.3.3 Implementation Viewpoint

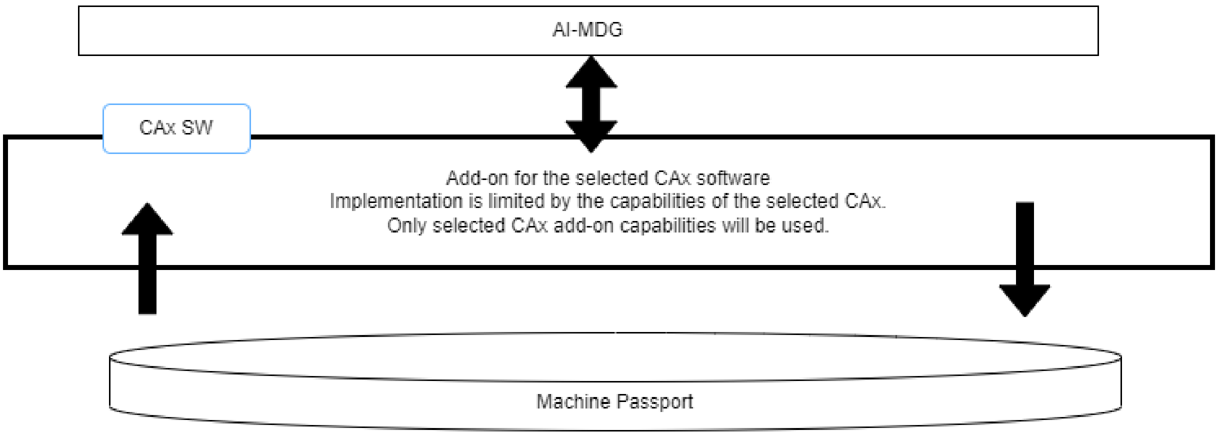


Figure 18. AI^CAx Implementation Architecture

The core of the add-on is developed in Python programming language and uses Fusion Python API for interaction with Fusion. The user interface of the add-on is implemented using web technologies (HTML, CSS, Javascript, jQuery) which provide much richer user interface and user experience than standard Fusion UI components. The add-on communicates with IANUS Strömungsraum simulation and optimization software through HTTP API. The communication is secured using https protocol and uses bearer token authentication scheme for API calls authentication and authorization.

4.3.4 AI^CAx Implementation Components

Implementation Components	Description
Addon for Autodesk Fusion	The add-on provides integration with IANUS Strömungsraum simulation and optimization software.

Table 35. AI^CAx Implementation Components

4.3.5 Technical Description of AI^{CAx} Components

Following the identification and specification of the implementation components in the previous section, the technical description of each component is provided in the following tables. It details the set of technologies required to implement each of the implementation components.

Implementation component	Addon for Autodesk Fusion
Description of implementation component	The add-on provides integration with IANUS Strömungsraum simulation and optimization software.
Used technologies	Python, HTML, CSS, JavaScript
Technical Description of the Component	**Dependencies**
	Development Languages: Python, JavaScript Libraries: Python: fusion360utils, adsk JavaScript: jQuery, DataTables Container: N/A Database: N/A
	Interfaces
	User Interface: HTML, CSS, JavaScript Synchronous/Asynchronous Interface: N/A Network/Protocols: HTTP/HTTPS Data Repository: N/A
	Requires
	IANUS Strömungsraum API

Table 36. Technical Description of AI^{CAx} "Addon for Autodesk Fusion" Implementation Component

4.4. Procurement Optimiser – AI^{PO}

4.4.1 Usage Viewpoint

The following usage activity diagram (Figure 19) depicts the pilot actors as roles and how they interact with different tasks and functions for the main activity in the AI^{PO} solution. This is intended to facilitate understanding the different functionalities and the flow of interactions to generate a new optimized purchases plan. The two pilots in AI^{PO} are very similar, the main difference is the possibility of parsing emails as an input, which is only available as optional in the D2TECH pilot.

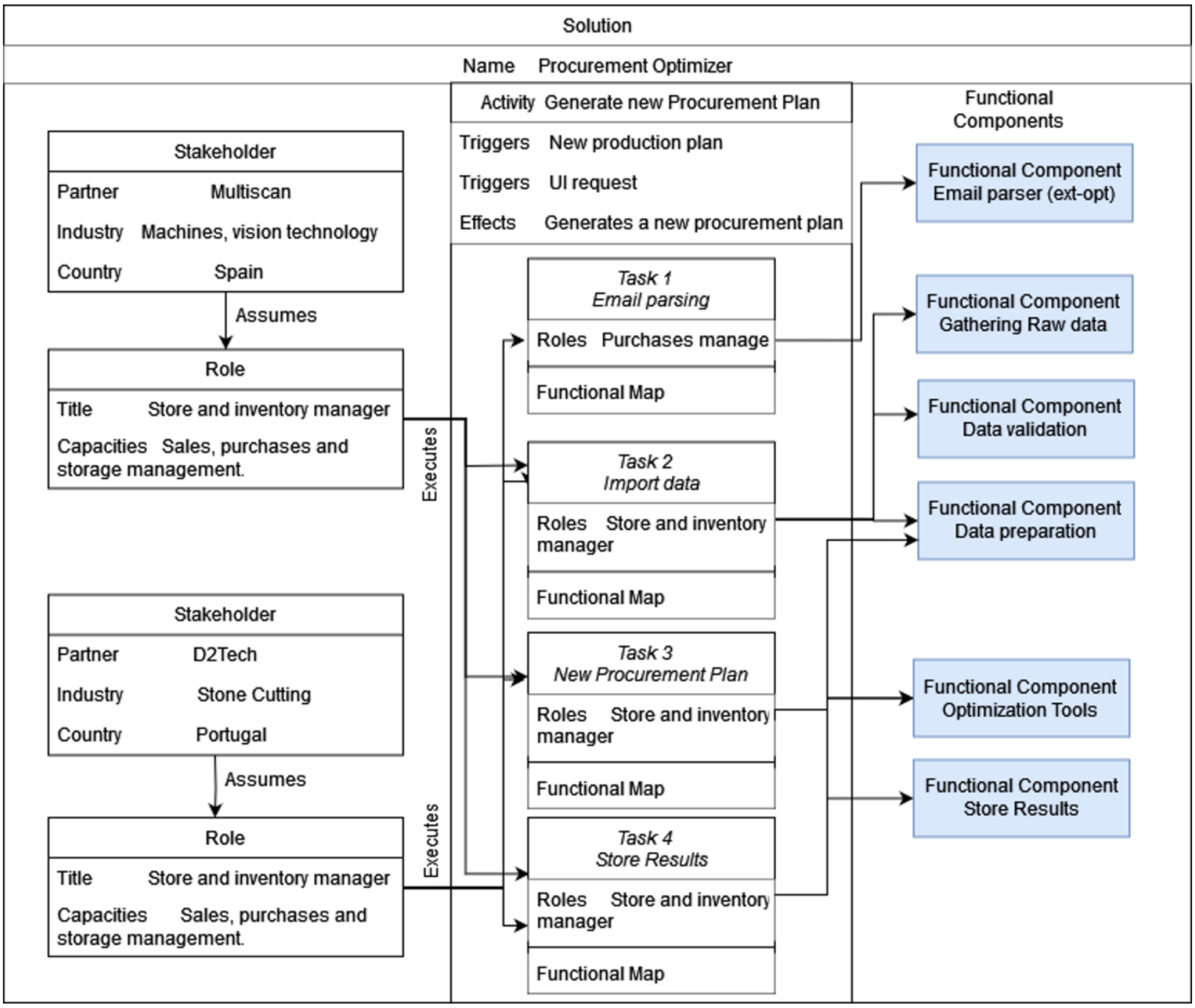


Figure 19. AI^{PO} Activity Diagram (Generate new optimized procurement plan)

4.4.2 Functional Viewpoint

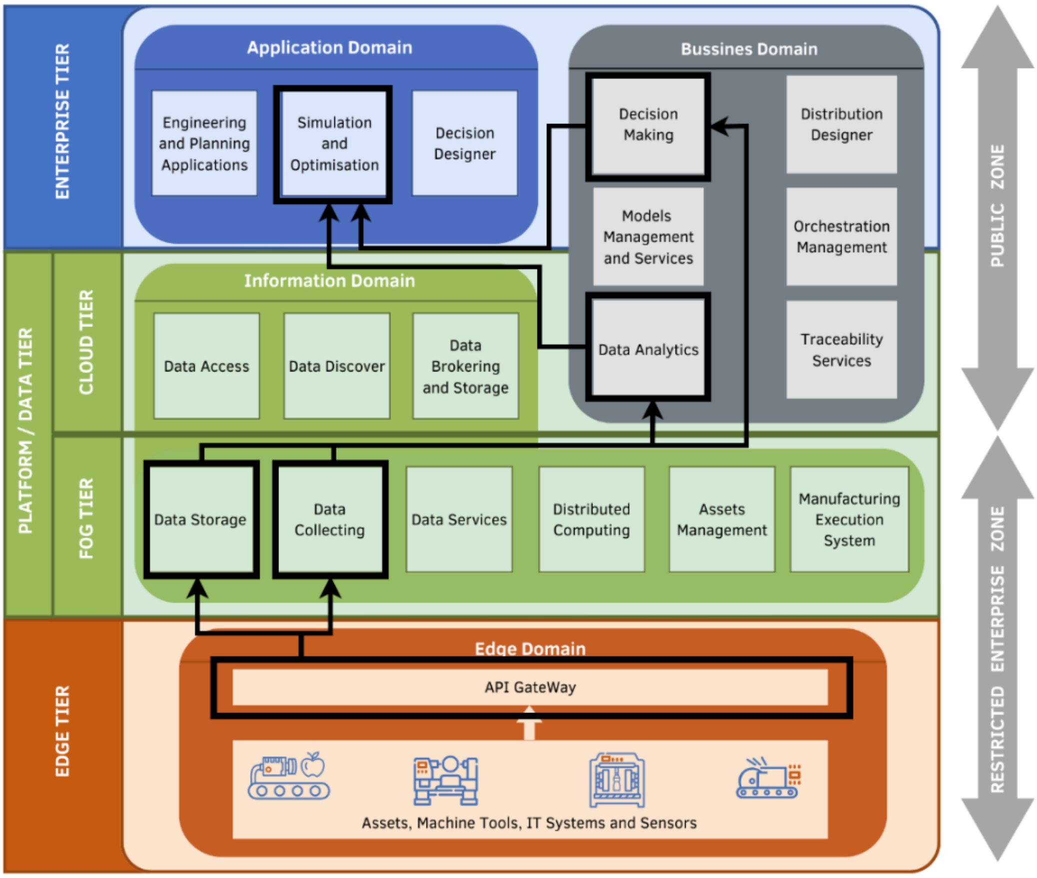


Figure 20. Data Flow AI[PO]

Procurement Optimizer generates a list of dated necessities of raw materials based on an initial production plan and the minimum stocks for each material. Given these necessities and the current state of the material stocks, the application uses the suppliers's information and the possible offers to obtain a possible solution, ID set, and a list of dated purchases of materials. Procurement Optimizer will try to obtain optimised solutions for the problem based on the KPIs or the pilot priorities. Once concluded, the procurement plan returns with some alerts of materials that cannot be received on time, warning that the production plan may need some alterations.

4.4.3 Data structure of AI[PO]

Format	Input/Output	Example
Database	Input	Database based in standard specifications from: https://doi.org/10.1016/j.compind.2021.103398
JSON	Input	Production Plan: [// basic information of the plan like date… “PlanID”: “03122024_PP24”, “StartingDate”: "2024-12-28T00:00:00", …, “ProductionPlanLines”: [{

		"PartID": "part-001", "PeriodID": 202402, "RequerimentAmount": 150, "ReceptionAvailabilityAmount": 80, "AmountIsForecasted": false }, ...] }]

JSON	Output	Procurement plan data, list of purchases, list of unable demand.
		{
		"ODProcurementPlanID": "Plan-001", "Date": "2024-02-09T00:00:00", "Message": "Procurement plan for Q1 2024",
		"ProposedPurchase": [
		{
		"ODProcurementPlanID": "Plan-001", "OfferID": 101, "PartID": "part-002",
		"PartDescription": "Electric Motor 2erqw3", "SupplierID": "supplier-003",
		"SupplierDescription": "lorem Motors Inc.", "ODRequirementAmount": 50,
		"ODPurchasePrice": 120.5, "ReceptionPeriodID": "202403",
		"ReceptionPeriodDate": "2024-03-15"
		}, ...
		],
		"UnavailableItems": [
		{
		"ODProcurementPlanID": "Plan-001", "PartID": "part-005", "ODProcurementPlanPartID": "Prod-009", "DelayAmount": 30, "PeriodID": "202404"
		}, ...
		]
		}

Table 37. Input / Output Data Format AI^{PO}

4.4.4 AI^{PO} Hardware Requirements

Hardware Element	Importance and Explanation	Data of Element
Database Server	The pilots (Multiscan/D2TECH) will have a database in a server with internet connection.	N/A
AI-PO Server	All the software related to the optimizer will be on a server with an Internet connection.	N/A

Table 38. Hardware required AI^{PO}

4.4.5 AI[PO] Software Requirements

Software Component	Description/Role	Required Version/Configuration	Dependencies
Database	Microsoft SQL Server	Unknown	No
Windows	Windows	11	

Table 39. Software requirements AI[PO]

4.4.6 AI[PO] Lifecycle

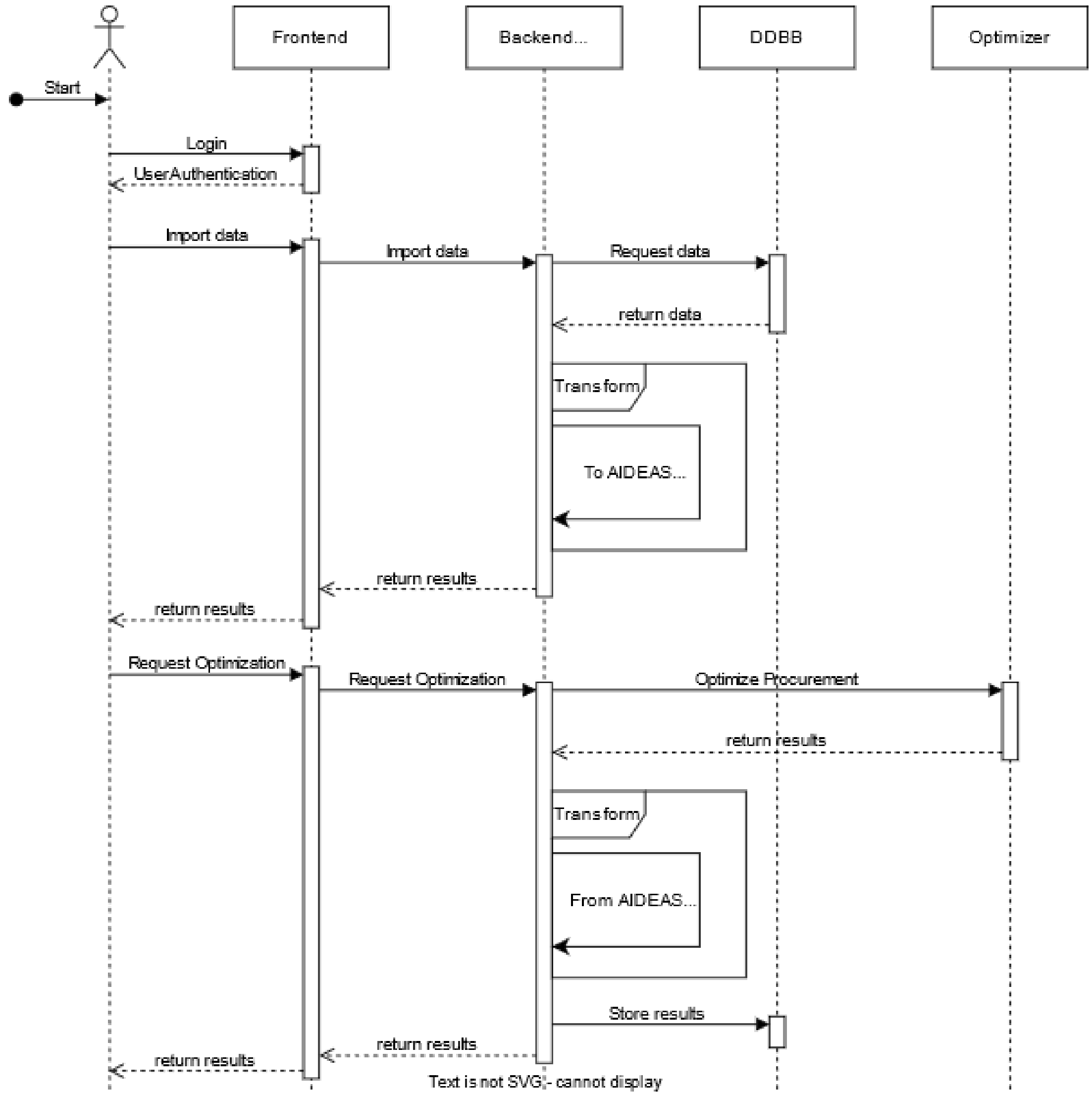


Figure 21. AI[PO] Lifecycle

This sequence diagram illustrates the interactions between User, Frontend, Backend, Database and Optimizer modules. The main actions illustrated include the login in the AIDEAS platform, the input of new data into the system, the connection with the database, the transformations of data required for optimization, the user request of a new plan, the optimization process and the storage and display of a new plan.

4.4.6.1 Objects

- **User**: Represents the user interacting with the application.
- **Frontend**: Represents the user interface and the presentation layer for the AIDEAS solution.
- **Backend bridge**: Represents the application logic and the server layer. This module controls the flow of data, transforms the data, and contains all the application logic except the optimization process.
- **Database (DDBB)**: Represents the data layer.
- **Optimizer**: This application module generates the optimized procurement plans.

4.4.6.2 Description – Login:

Objects	Description
Start	Start the login.
User – Frontend Interaction	The User logs into the AIDEAS platform.
Frontend - User Interaction	The AIDEAS platform authenticates the user.
Completion	The user can access the application.

Table 40. Life-Cycle login action AI[PO]

4.4.6.3 Description – Input data:

Objects	Description
User – Frontend Interaction	The user requests to import data selecting the data source.
Frontend - Backend	The Frontend propagates the demand to backend bridge.
Backend - Database	The backend requests the updated data from the Database, the Database returns this data.
Backend-Backend	Backend transforms this data into AIDEAS data structures generating a problem to optimize.
Backend - Frontend	Backend returns the results of the data gathering to the frontend.
Frontend – User	Frontend shows the results of the data gathering process

Table 41. Life-Cycle Input data action AI[PO]

4.4.6.4 Description – Obtain optimized procurement plan:

Objects	Description
User – Frontend Interaction	The User requests the system to generate a new procurement plan.
Frontend - Backend	The Frontend propagates the demand to backend bridge.
Backend - Optimize	The backend sends the procurement optimization problem to Optimize (i.e., FACOP framework) and waits for a response.

Optimize - Optimize	The AI^PO generates a solution for the problem with a procurement plan and a list of incompatible products.
Optimize - Backend	The procurement plan returns to the backend bridge.
Backend - Backend	Backend transforms procurement plan data from AIDEAS data structures to database format.
Backend - Database	Backend sends the results to the database to be stored, and the database responds with an acknowledgement.
Backend - Frontend	Backend returns procurement plan data to the frontend.
Frontend – User	Frontend shows the results of the procurement plant to the user.

Table 42. Life-Cycle new procurement plan action AI^PO

4.4.7 Implementation Viewpoint

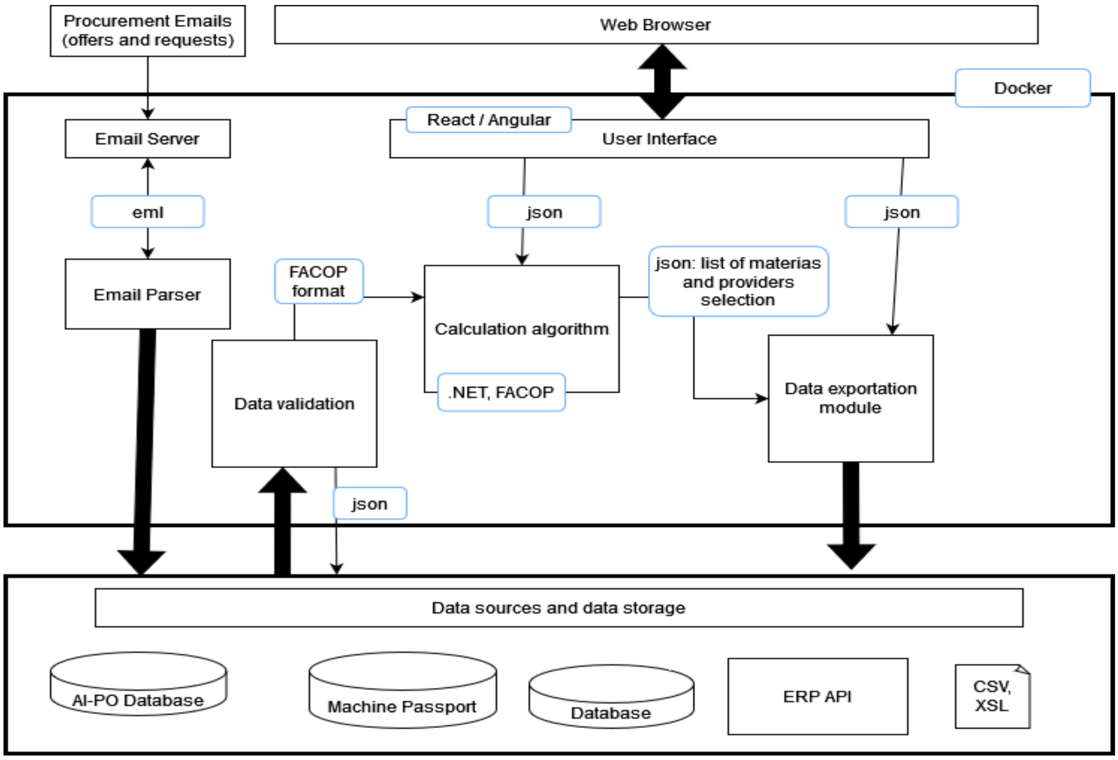


Figure 22. AI^PO Implementation Architecture

AI^PO development language will be mostly C#, using .Net libraries and the core library for optimization will be FACOP. The deployment and containerization will be done with Docker. The databases to store pilot information will depend on clients' licenses; for the MULTISCAN pilot, it will be Microsoft SQL Server, and for D2TECH, PostgreSQL has been proposed. The database contains information related to AI^PO problem and solution. The Database is fed with information gathered from client's ERP, the email parser and AI^PO solution.

There have been some updates to these implementation architecture diagram and the general design of the application. These changes affect mostly how components communicate, as a database has been included, which can be read or modified by few components. This database serves multiple purposes: it acts as a buffer for client information, as an output destination for the email parser, a consistent input source for the MRP calculation module, and a reliable output source for the MRP solutions.

4.4.8 AI^{PO} Implementation Components

Implementation Components	Description
Import data	This component facilitates the reading of data from different data sources like databases, emails, files etc.
Data validation	This component validates the training data and ensures that the input data is in the correct format before feeding it into the model.
Compose MRP algorithm	This component generates (based in a previous configuration) an algorithm with different smaller subparts that combined forms a full functional algorithm tasked with solving the procurement optimization problem. This component also associates the input data with this algorithm.
Calculate new MRP	The task of this component is to solve the procurement optimization problem and to generate a feasible and valid procurement plan. It will also allow the detection of products that cannot be produced at the time indicated in the production plan due to lack of time needed to purchase the materials. This will bring about an adjustment to the current production plan.
Export new MRP	The solution and output data for the new MRP now needs to be properly formatted and exported to AIDEAS system to be stored.

Table 43. AI^{PO} Implementation Components

4.4.9 Technical Description of AI^{PO} Components

Following the identification and specification of the implementation components in the previous section, the technical description of each component is provided in the following tables. It details the set of technologies required to implement each of the implementation components.

Implementation component	Import data
Description of implementation component	This component validates the training data and ensures that the input data is in the correct format before feeding it into the model.
Used technologies	.Net
	Dependencies
	Development Language: C# Libraries: .Net

Technical Description of the Component	Container: Docker Database: To be defined
	Interfaces
	User Interface: Yes Synchronous/Asynchronous Interface: Synchronous Network/Protocols: HTTPS

Table 44. Technical Description of AI^PO "Import data" Implementation Component

Implementation component	Data validation
Description of implementation component	This component validates the training data and ensures that the input data is in the correct format before feeding it into the model.
Used technologies	.Net
Technical Description of the Component	**Dependencies**
	Development Language: C# Libraries: .Net Container: Docker Database: No
	Interfaces
	User Interface: Yes Synchronous/Asynchronous Interface: Synchronous Network/Protocols: HTTP

Table 45. Technical Description of AI^PO "Data validation" Implementation Component

Implementation component	Compose MRP algorithm
Description of implementation component	This component generates (based in a previous configuration) an algorithm with different smaller subparts that combines and forms a full functional algorithm tasked with solving the procurement optimization problem. This component also associates the input data with this algorithm.
Used technologies	FACOP and .Net
	Dependencies
	Development Language: C# Libraries: .Net, FACOP Container: Docker

Technical Description of the Component	Database: No
	Interfaces
	User Interface: No
	Synchronous/Asynchronous Interface: Asynchronous

Table 46. Technical Description of AI^{PO} “Compose MRP algorithm” Implementation Component

Implementation component	Calculate new MRP
Description of implementation component	The task of this component is to solve the procurement optimization problem and to generate a feasible and valid procurement plan. It will also allow the detection of products that cannot be produced at the time indicated in the production plan due to lack of time needed to purchase the materials. This will bring about an adjustment to the current production plan.
Used technologies	FACOP and .Net
Technical Description of the Component	**Dependencies**
	Development Language: C#
	Libraries: .Net, FACOP
	Container: Docker
	Database: No
	Interfaces
	User Interface: No
	Synchronous/Asynchronous Interface: Asynchronous

Table 47. Technical Description of AI^{PO} “Calculate new MRP” Implementation Component

Implementation component	Export new MRP
Description of implementation component	The solution and output data for the new MRP now needs to be properly formatted and exported to AIDEAS system to be stored.
Used technologies	.Net
Technical Description of the Component	**Dependencies**
	Development Language: C#
	Libraries: .Net
	Container: Docker
	Database: To be defined
	Interfaces
	User Interface: Yes

	Synchronous/Asynchronous Interface: Synchronous Network/Protocols: HTTPS Data Repository:

Table 48. Technical Description of AI^{PO} “Export new MRP” Implementation Component

4.5. Fabrication Optimiser – AI^{FO}

4.5.1 Usage Viewpoint

The usage activity diagrams below represent the roles of the actors in the pilot cases and their interaction with the functionalities of the two AI^{FO} solutions. In the first diagram, the solution aims to clarify the functionality for obtaining correction parameters to be passed as input to the machine for machining a key component. This way, rework steps are avoided. In the second diagram, pilots have the possibility to generate new scheduling plans based on production orders. The difference between the two pilot cases is that for the MULTISCAN pilot, there is also the possibility of generating the master production plan, which will provide the input data for the scheduling plan.

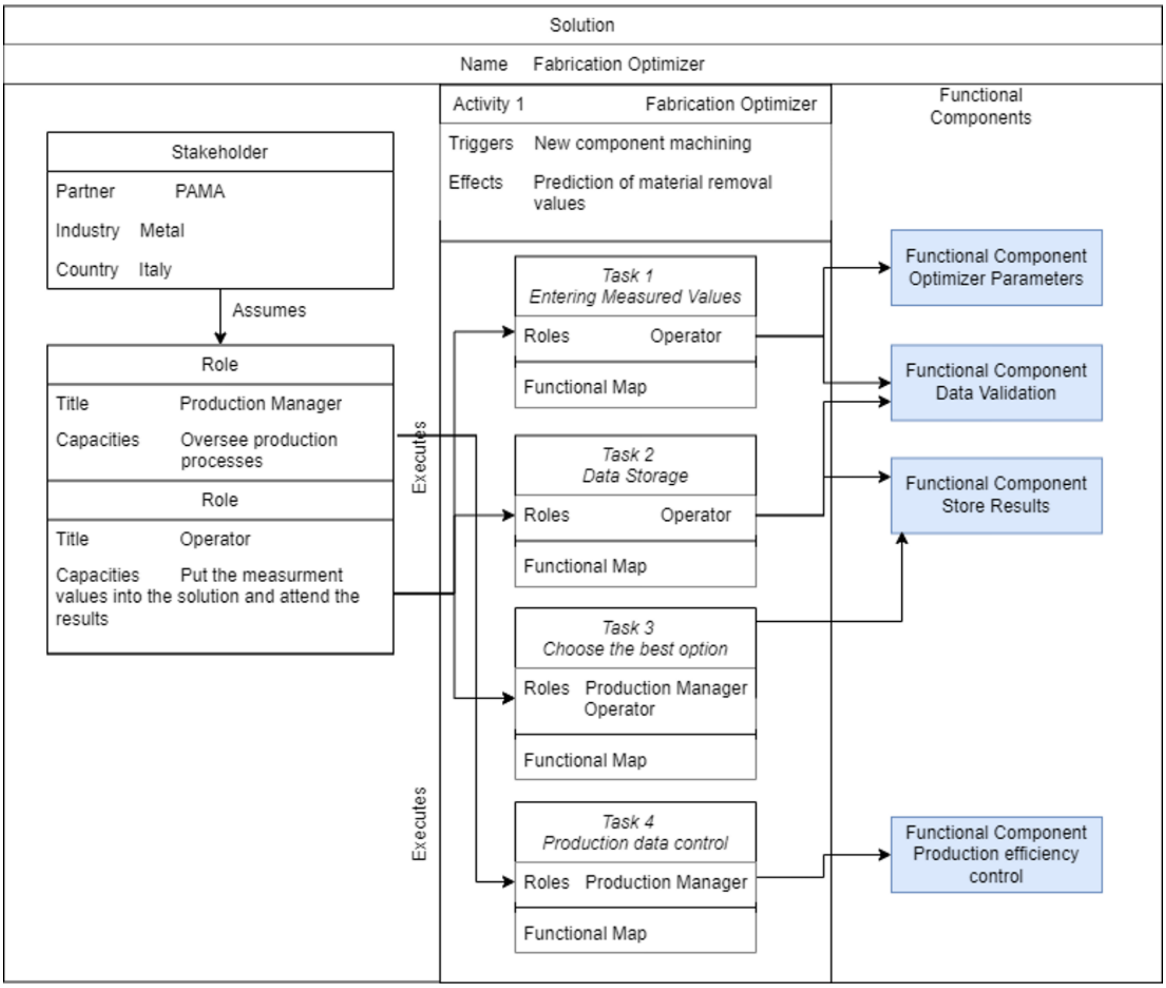


Figure 23. AI^{FO} Activity Diagram (Generate corrective parameter to minimize re-machining operations)

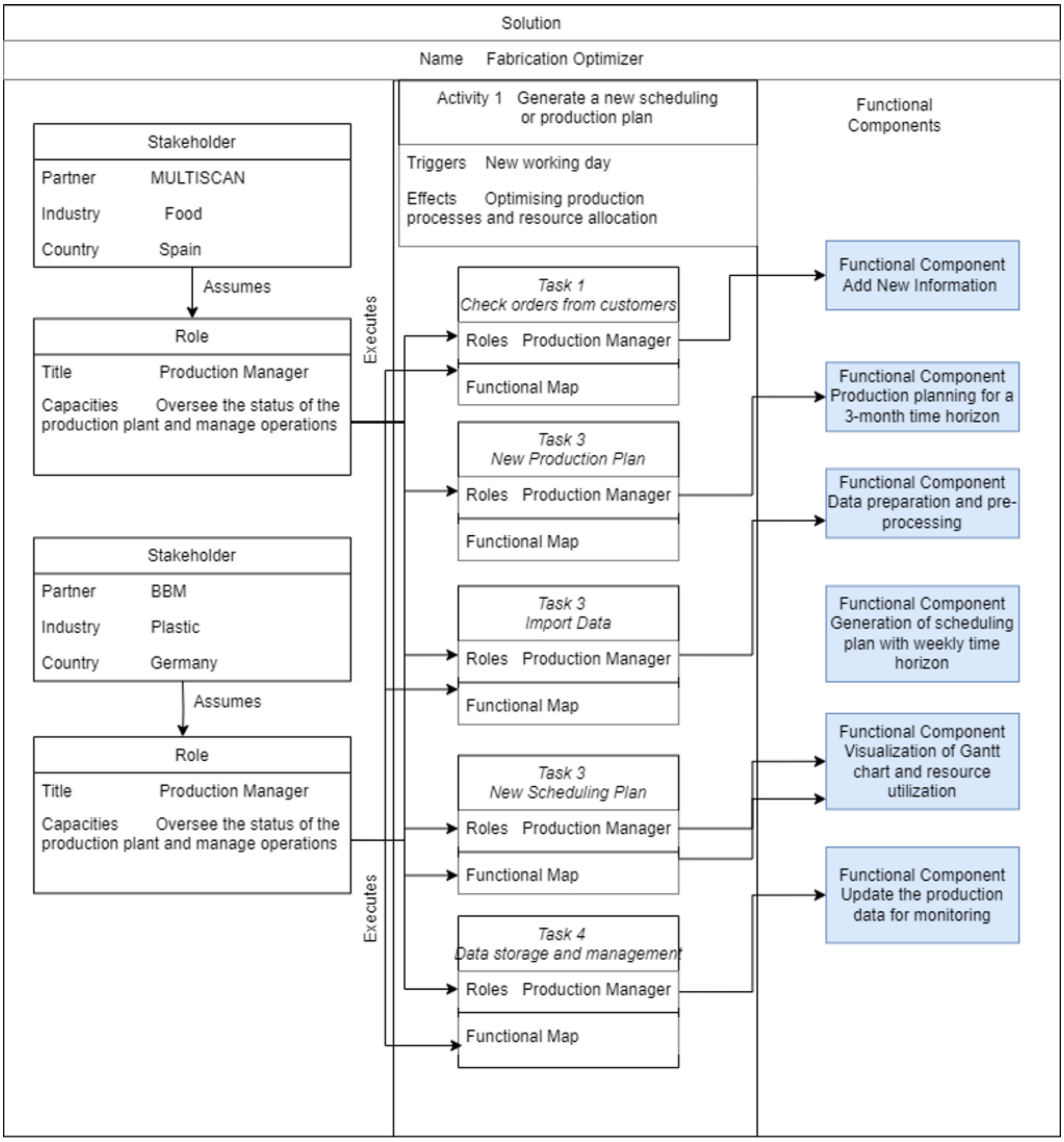


Figure 24. AI[FO] Activity Diagram (Generate new production plan and scheduling plan)

4.5.2 Functional Viewpoint

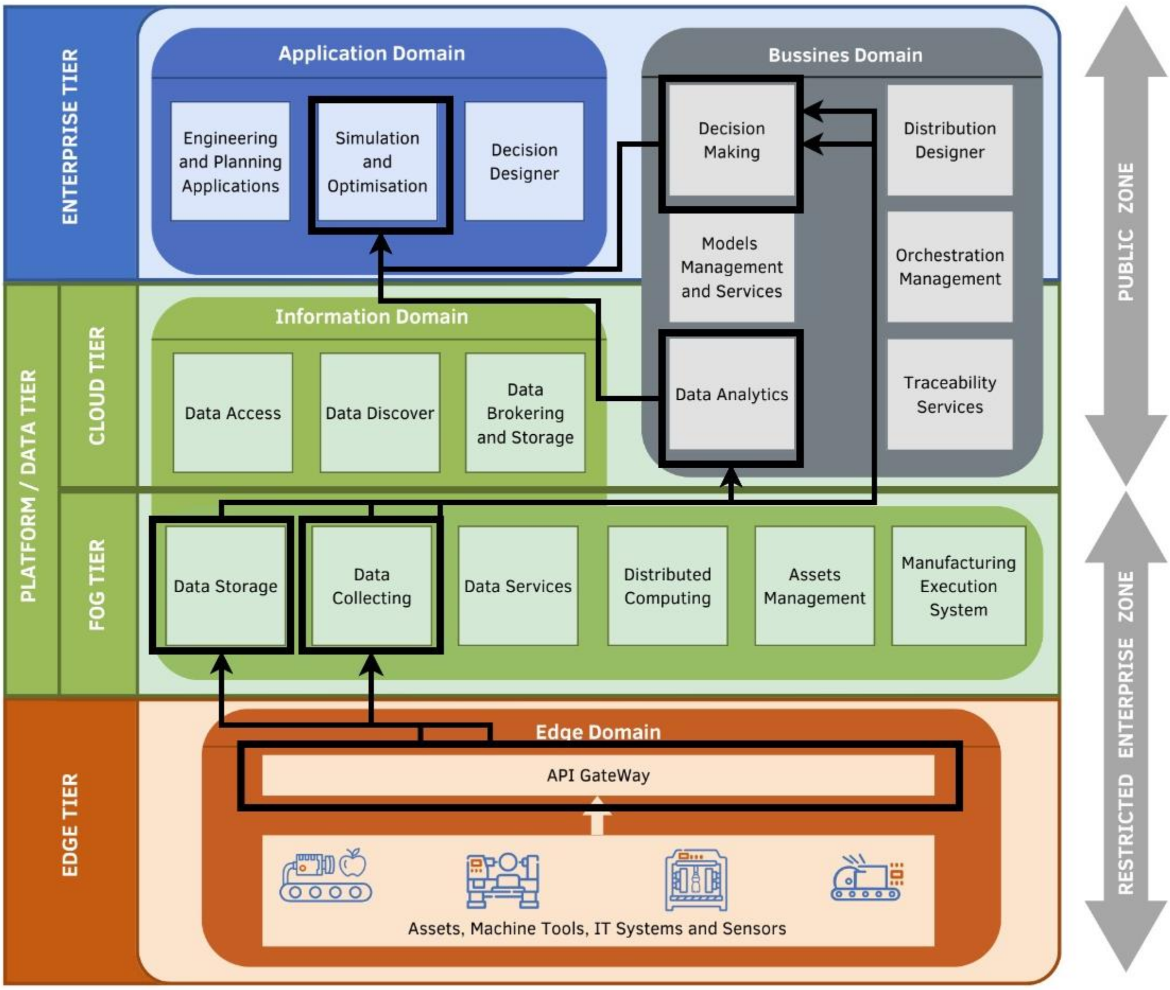


Figure 25. Data Flow AI^FO

Fabrication Optimiser for the PAMA solution receives production data manually from the operator (user). The input data is sent to the backend where it is automatically pre-processed and passed as input to two ANNs that predict the correction factor to be passed as input to a 5-axis CNC machine. Meanwhile, for the production plan and scheduling tool which must be implemented in MULTISCAN and BBM, the data comes from a database, after which a collection module captures and pre-processes it. When the data is ready, an AI model will manipulate it to find the best production sequences.

4.5.3 Data structure of AI^FO

Format	Input/Output	Example

JSON	Input	For PAMA Solution
		data = {
		“tests”: [
		{
		"Job_name": "PS4B52",
		"order": "133711",
		"test_number": 1,

		"Operator_name": "CAZZANELLI MARCO",
		"data": "14/07/2022",
		"hour": "04:30",
		"temperature": 23.0,
		"lenght": 7500,
		"input_measurments":
		{
		"meas_guide_ant_list": […],
		"meas_guide_post_list": […],
		"meas_side_ant_list": […],
		"meas_side_post_list": […]
		}
		]
		}

JSON	Output	Output for PAMA solution
		Output_data = {
		"corr_guide_ant_list": […],
		"corr_guide_post_list": […],
		"corr_side_ant_list": […],
		"corr_side_post_list": […],
		"image_guide" : " .png",
		"image_side" : " .png",
		"Excel_results": " .xlsx"
		}

DATABASE	Input	For MULTISCAN & BBM Solution MySQL -> Standard C2NET -> https://doi.org/10.1016/j.compind.2021.103398

JSON	Input	ProductionPlanData: {
		"PartID": ( ),
		"Demand": ,
		"Inventory": ,
		"Capacity": ,
		"ProcurementPlan": ,
		"LabourID": ,
		"OrderID": ,
		"Part_Operation_material": { },
		"Calendar_availability_Resources: { }
		}

JSON	Output	ProductionPla_Output: {
		"LabourID": ,
		"OrderID": ,
		"Start_Date": ,
		"Finish_Date":,
		}

JSON	Input	SchedulingData: {

		“TaskID”: ( ), “ProcessingTime”: , “ProductionConstraint”: , “Progress”: , “Due_Date”: “OperatorGroup”: , “LabourID”: , “OrderID”: , “Start_Date”: , “Finish_Date”:, “Part_Operation_material”: { }, “Calendar_availability_Resources: { } }

JSON	Output	Schedulign_output: {
		“TaskeID”: ( ),
		“Start_Date”: ,
		“Finish_Date”: ,
		“LabourID”: ,
		“Gantt_Labour”: “ .png”,
		“Gantt_Part”: " .png“
		}

Table 49. Input / Output Data Format AI^{FO}

4.5.4 AI^{FO} Software Requirements

Software Component	Description/Role	Required Version/Configuration
Windows OS	Microsoft operating system its need to use specific tool	Windows 11
Microsoft Office Suite	Suite of office with excel applications for document processing.	Office 365

Table 50. Software requirements AI^{FO}

4.5.5 AI^FO Lifecycle

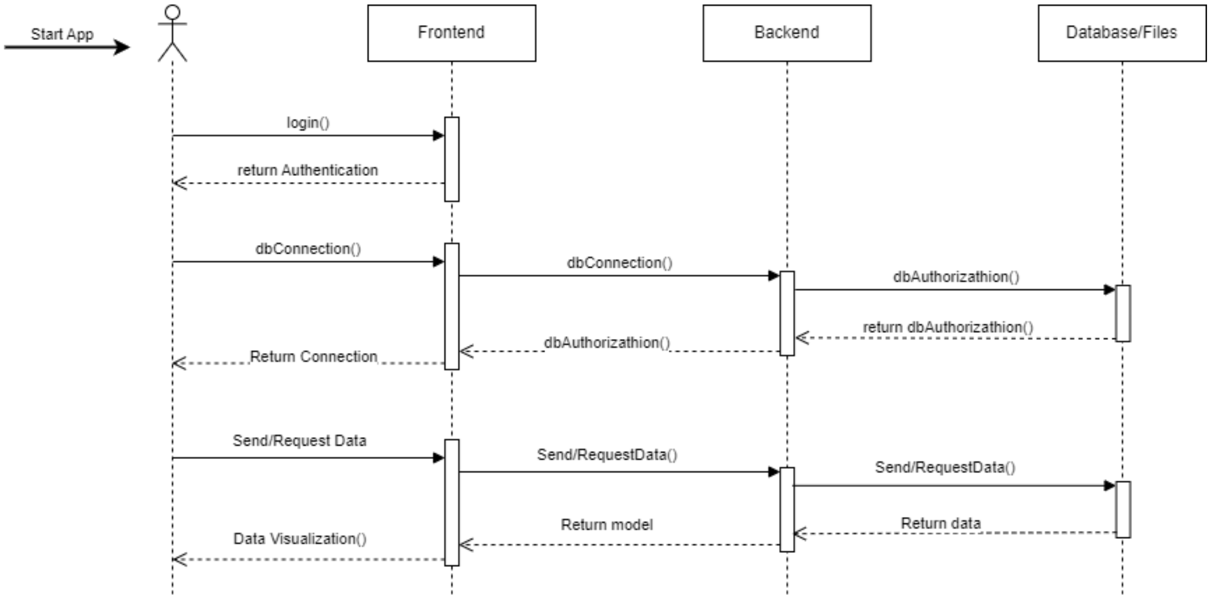


Figure 26. AI^FO Lifecycle

This sequence diagram illustrates the flow of interactions between the User, Frontend, Backend and Data/Files objects during the main processes. Login into the AIDEAS platform. In the case of PAMA solution, the user will send the data to the backend, which will receive it as input and generate the outputs values and relatives' images and files. Meanwhile, for the scheduling tool, the user will take data from the database after an authentication phase.

4.5.6 Objects

- **User**: Represents the User interacting with the application.
- **Frontend**: Represents the user interface and the presentation layer.
- **Backend**: Represents the application logic and the server layer.
- **Database/Files:** Represents the data or file storage layer

4.5.7 Description – Login:

Objects	Description
Start	The user logs into the AIDEAS platform.
User – Frontend Interaction	The user enters username and password.
Frontend - User Interaction	The Frontend shows the user the response from the login action, indicating whether it is logged in or not.
Completion	The application is ready to receive new requests from the User.

Table 51. Life-Cycle description Login AI^FO

4.5.8 Description – Database connection:

Objects	Description
User – Frontend Interaction	The user requests access to the database or to send files.
Frontend - Backend Interaction	The Frontend sends a request to the Backend to process a connection to the database or to process the file upload.
Backend - Data/Files Interaction	The Backend requests the storage layer (Data/Files) to authorize the connection or to prepare to receive the file. Data/Files is waiting for the authentication or for the file transfer
Data/Files - Backend Response	Data/Files confirms that the connection has been completed or the file has been stored correctly and sends a response to the Backend.
Backend - Frontend Response	The Backend informs the Frontend about the success or failure of the connection or file upload operation.
Frontend - User Interaction	The Frontend shows the user the response from the application, indicating whether the connection or file upload was successful.
Completion	The application is ready to receive new requests from the User.

Table 52. Life-Cycle description Database connection AI[FO]

4.5.9 Description – Send/RequestData:

Objects	Description
User – Frontend Interaction	The user interacts with the Frontend. In the case of PAMA solution, the user inserts the data in the manufacturing suite. In the scheduling tool, the user takes the necessary information from the database.
Frontend - Backend Interaction	The Frontend sends a request to the Backend to process the input data in both solutions.
Backend - Data/Files Interaction	The Backend requests the storage layer (Data/Files) to authorize the connection or to prepare to receive the data.
Data/Files - Backend Response	Data/Files sends a response to the Backend.
Backend - Frontend Response	The Backend gets the corresponding data and evaluates the model.
Frontend - User Interaction	The Frontend shows the User the response from the application, obtaining the condition evaluation report for the selected components
Completion	The application is ready to receive new requests from the user.

Table 53. Life-Cycle description Send/Request Data AI[FO]

4.5.10 Implementation Viewpoint

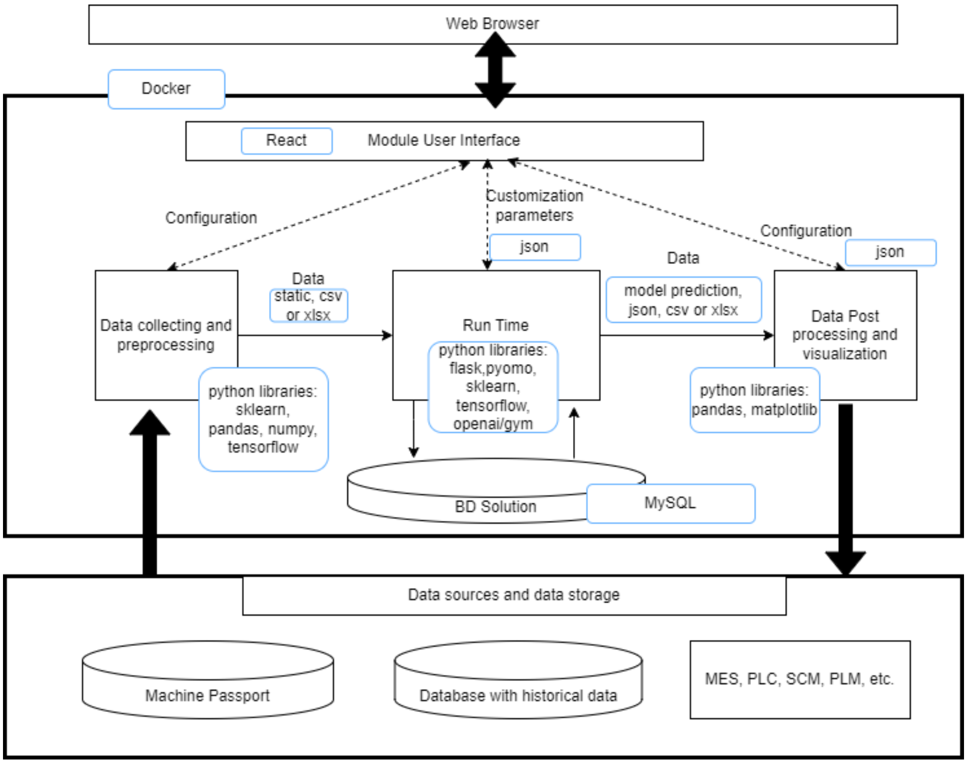


Figure 27. AI^{FO} Implementation Architecture

Python was used as the main development language for the realisation of the two AI^{FO} solutions. The main libraries used are Pandas, NumPy and Sklearn for data manipulation and pre-processing. Sklearn was used for the PAMA pilot (solution 1) to train and validate the artificial neural network model, while TensorFlow and Gym from OpenAI were used for the scheduling algorithm (solution 2). Doker was adopted for containerisation. An important update to the architecture of the previous solution concerns the introduction of Flask as a library to create the backend endpoint. For solution 1, the data is entered manually by the user. For solution 2, the data comes from the company's MySQL database.

4.5.11 AI^{FO} Implementation Components

Implementation Components	Description
Import Data	This component allows the data required for the two solutions to be entered from different sources (Database, Frontend, etc.)
Data Validation and Preprocessing	This component allows the input data to be validated to ensure the correct format. It also pre-processes the data to make it ready for AI model input.
Generation of a new scheduling plan	This component allows the generation of a master production plan based on production orders, production forecasts and resource availability. (Only for solution 2)

Generation of a new production plan	This component allows a production scheduling plan to be generated based on the results of the main production plan and the results of the AI-PO solution. It constitutes the production/assembly plan for the machines, allocates the necessary resources for implementation and sets the theoretical end date of production (Only for solution 2)
Generation of the corrective parameters	This component allows calculation of the quantities of material to be removed to avoid subsequent reworking of the component. (Only for solution 1)
Data Export	This component allows results to be exported and saved in Machine Passport, in the company database or in the cloud.

Table 54. AI^{FO} Implementation Components

4.5.12 Technical Description of AI^{FO} Components

Following the identification and specification of the implementation components in the previous section, the technical description of each component is provided in the following tables. It details the set of technologies required to implement each of the implementation components.

Implementation component	Import Data
Description of implementation component	This component allows the data required for the two solutions to be entered from different sources (Database, Frontend)
Used technologies	Python, REACT
Technical Description of the Component	Dependencies
	Development Language: Python Libraries: Pandas, Flask Container: Docker Database: To be defined
	Interfaces
	User Interface: Yes Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: https

Table 55. Technical Description of AI^{FO} "Import Data" Implementation Component

Implementation component	Data Validation and Preprocessing
Description of implementation component	This component allows the input data to be validated to ensure the correct format. It also pre-processes the data to make it ready for AI model input.
Used technologies	Python

Technical Description of the Component	Dependencies	
	Development Language: Python Libraries: Pandas, Numpy, Sklearn Container: Docker Database: No	
	Interfaces	
	User Interface: No Synchronous/Asynchronous Interface: No Network/Protocols: No Data Repository: No	
	Requires	
	Import Data component	

Table 56. Technical Description of AI^{FO} "Data Validation and Preprocessing" Implementation Component

Implementation component	Generation of a new scheduling plan
Description of implementation component	This component allows the generation of a master production plan based on production orders, production forecasts and resource availability. (Only for solution 2)
Used technologies	Python
Technical Description of the Component	Dependencies
	Development Language: Python Libraries: TensorFlow, Gym, Pandas, Matplotlib, Numpy Container: Docker Database: To be defined
	Interfaces
	User Interface: NO Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: https
	Requires
	Data Validation and Preprocessing component

Table 57. Technical Description of AI^{FO} "Generation of a new scheduling plan" Implementation Component

Implementation component	Generation of a new Production Plan
Description of implementation component	This component allows a production scheduling plan to be generated based on the results of the main production plan and the results of the AI^{PO} solution. It constitutes the production/assembly plan for the machines, allocates the necessary resources for implementation and sets the theoretical end date of production (Only for solution 2).
Used technologies	Python
Technical Description of the Component	Dependencies
	Development Language: Python Libraries: Pyomo, Pandas Container: Docker Database: To be defined
	Interfaces
	User Interface: Yes Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: https
	Requires
	Data Validation and Preprocessing

Table 58. Technical Description of AI^{FO} "Generation of a new Production Plan" Implementation Component

Implementation component	Generation of the corrective parameters
Description of implementation component	This component allows calculation of the quantities of material to be removed to avoid subsequent reworking of the component. (Only for solution 1)
Used technologies	Python
Technical Description of the Component	Dependencies
	Development Language: Python Libraries: Sklearn, Pandas, Openpyxl Container: Docker Database: No
	Interfaces
	User Interface: NO Synchronous/Asynchronous Interface: RESTful APIs

	Network/Protocols: https. Data Repository: PAMA Drive
	Requires
	Data Validation and Preprocessing

Table 59. Technical Description of AI^{FO} "Generation of the corrective parameters" Implementation Component

Implementation component	Data Export
Description of implementation component	This component allows results to be exported and saved in the Machine Passport, in the company database or in the cloud.
Technical Description of the Component	Dependencies
	Development Language: Python Libraries: Sklearn, Numpy, Pandas, Matplotlib Container: Docker Database: To be defined
	Interfaces
	User Interface: Yes, React Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: https. Data Repository: PAMA Drive (Solution 1)
	Requires
	Generation of the corrective parameters' component or, Generation of a new production plan and Generation of a new scheduling plan component

Table 60. Technical Description of AI^{FO} "Data Export" Implementation Component

4.6. Delivery Optimiser – AI^{DO}

4.6.1 Usage Viewpoint

The usage activity diagram (Figure 28) within AI^{DO} illustrates the connections between the tasks and roles in the MULTISCAN use-case. This usage activity diagram shows the flow of activities performed by the AI^{DO} solution. The roles, tasks, processes, and functional components help to provide a clear understanding of the activities executed using the AI^{DO} solution.

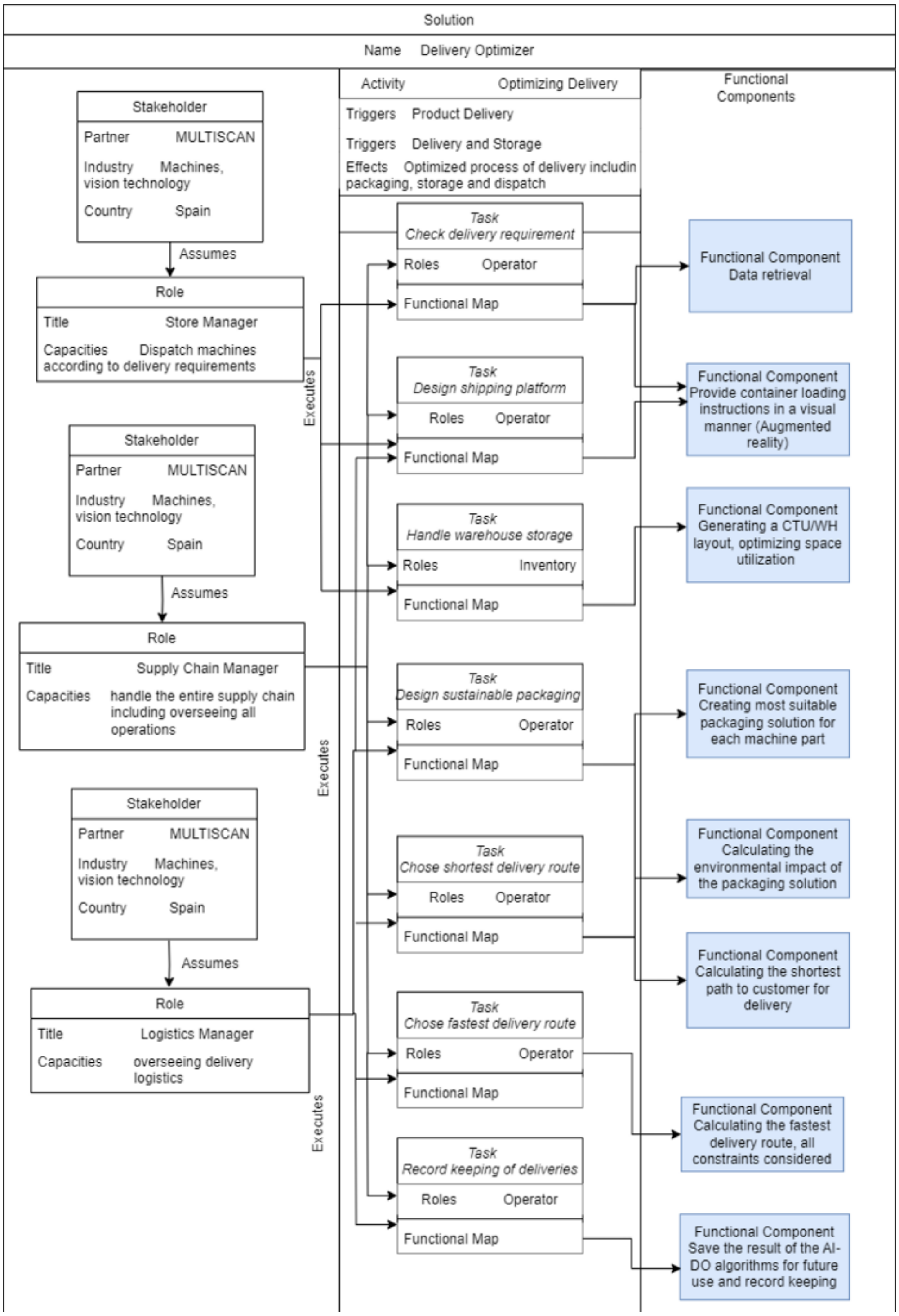


Figure 28. Delivery Optimiser Usage Viewpoint Activity Diagram (Optimise delivery).

4.6.2 Functional Viewpoint

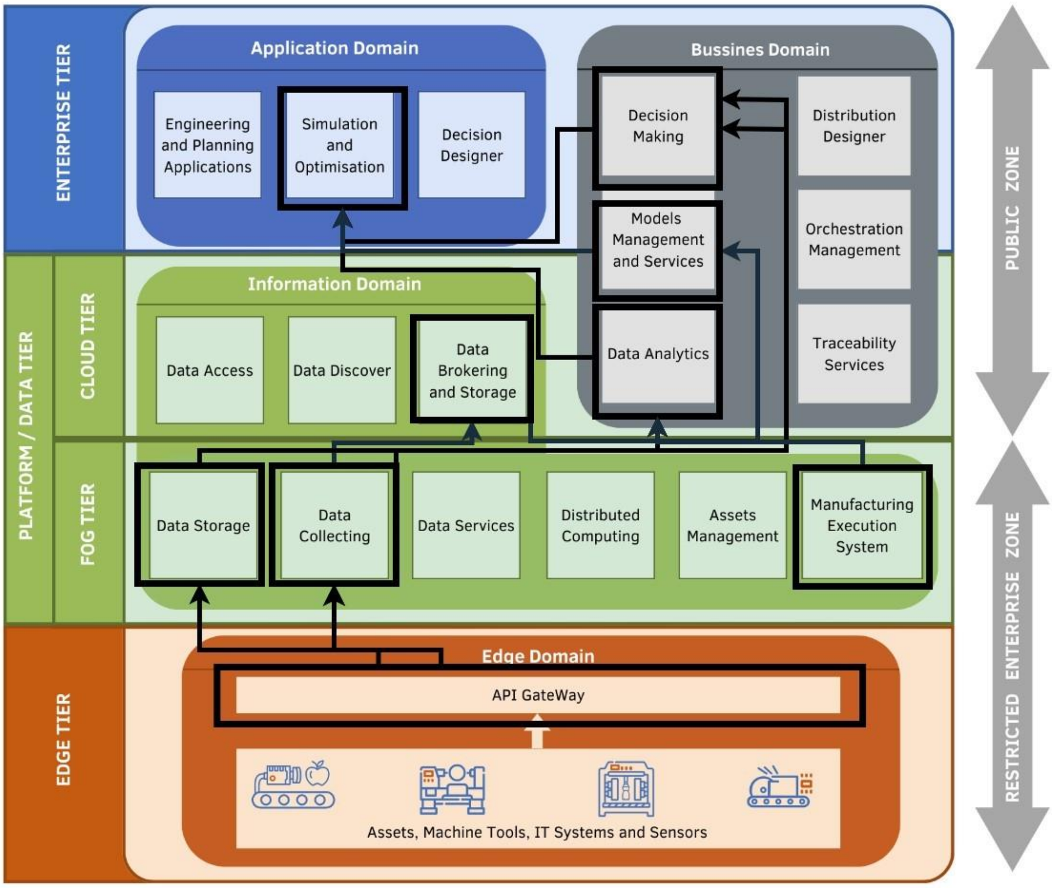


***Figure 29**. Data Flow AI^{DO}*

The Delivery Optimiser receives input data from different data sources such as databases, files, and ERPs. The input data is pre-processed and validated before it is fed into the AI model, which makes predictions for optimal packaging, storage, and product delivery. The AI^{DO} solution is to be validated in the MULTISCAN use-case.

4.6.2.1 Data structure of AI^{DO}

Format	Input/Output	Example

JSON	Input	Packaging input data
		{
		"data": ["part length(cm)",
		"Part width(cm)",
		"Part height(cm)",
		"Part weight(kg)",
		"Part fragility(binary)",
		"Atmospheric seal(binary)",
		"Storage time(days)"]
		}
		Example
		{"data": [56, 43, 29, 49, 1, 0, 100] }

JSON	Output	Packaging output data
		Output data = {
		"Packaging material": "choice of material",
		"Extra protection": "choice of material/not required",
		Environmental impact of package: "scale of 1 - 10"
		}
		
		Example
		{
		"Packaging material": "polystyrene",
		"Extra protection": "foam beans",
		"impact": "6"
		}

JSON	Input	Storage input data consists of the following fields.
		{
		orderID,
		"Truck dimension": list of integers providing length, width and height of truck/container,
		"customerN": list of lists, providing the items/boxes belonging to customer. Each inner list consists of 6 elements, the first three elements provide length, width, and height. The last three elements provide volume, value associated to a box and an integer that provides the belonging of box to customer.
		}
		
		Example
		data = {
		"0": {
		"truck dimension": [//container dimensions
		589,
		235,
		239
		],
		"customer1": [// items belonging to customer
		[
		92,
		81,
		55,
		409860,
		236,
		1
		]....,
		"customer2": [// items belonging to customer
		[
		108,

		76, 30, 246240, 199, 2], "customer3": [// items belonging to customer [92, 81, 55, 409860, 236, 3]....., }

JSON	Output	Storage output data consists of different placement layouts of boxes in the container. Each layout has following fields:
		unique **ID.**
		"fitness" which is a list provide utilization of container volume, average value and percentage of boxes packed.
		"num" provide number of boxes are packed.
		"res" contains solution for boxes placement in the container. The solution is sorted in the lowest to highest customer priority.
		"customer_1" is the list of lists. Each list contains the x,y,z,l,w,h data of a box. The first 3 elements provide coordinates where a box will be placed in the container and the last three elements provide length, width and height of the box.
		"num_Cn": N, a number which tells how many boxes belonging to a customer are placed in the container.
		Example:
		Output_data = {
		"00": {
		"fitness": [
		62.07%,
		96.0%,
		96.43%
		],
		"num": 70,
		"res": {"customer_1": [
		[
		0,
		0,
		0,
		81,

		55, 92, 1]..., "num_C1": 25, "customer_2": [[173, 0, 147, 55, 81, 92, 2], "num_C1": 24, "customer_3": [[0, 147, 0, 55, 81, 92, 3], "num_C1": 23.....,

JSON	Input	Delivery input data
		data = {
		"agents": {
		"0": {
		"agent_ID": 0,
		"X": 0.0,
		"Y": 0.0
		},
		"X": {
		"agent_ID": ,
		"X": 0.0,
		"Y": 0.0
		}
		},
		"locations": {
		"0": {
		"location_ID": 0,
		"X": 0.0,
		"Y": 0.0
		},
		"1": {

		"location_ID": 1,
		"X": 0.0,
		"Y": 0.0
		},
		"X": {
		"location_ID": X,
		"X": 36.0,
		"Y": 4.0
		}
		},
		"distances": {
		"0_0": 0.0,
		"0_1": 0.0,
		"X_X": 40.0,
		},
		"parameters": {
		"pop_size": 100,
		"prob_mutation": 1,
		"time_calc": 10,
		"max_loops": 10000
		}
		}

JSON	Output	Delivery output data
		Output_data = {
		"agents": {
		"0": {
		"agent_ID": 0,
		"X": 85.0,
		"Y": 38.0,
		"route": {
		"0": {
		"location_ID": 9,
		"X": 85.0,
		"Y": 38.0
		},
		"1": {
		"location_ID": 5,
		"X": 58.0,
		"Y": 44.0
		},
		"X": {
		"location_ID": X,
		"X": 47.0,
		"Y": 74.0
		}

		},
		"1": {
		"agent_ID": 1,
		"X": 67.0,
		"Y": 10.0,
		"route": {
		"0": {
		"location_ID": 6,
		"X": 67.0,
		"Y": 10.0
		},
		"1": {
		"location_ID": 2,
		"X": 36.0,
		"Y": 4.0
		},
		"X": {
		"location_ID": X,
		"X": 0.0,
		"Y": 0.0
		}
		}
		},
		"solution": {
		"Objective_Information": {
		"Objective": 428.0,
		"Sense": "minimize",
		"Max route": 6.0,
		"Status": "ok",
		"Termination condition": "optimal",
		"Lower bound": 428.0,
		"Upper bound": 428.0,
		"GAP": 0.0,
		"Time": 2.1446940898895264
		}
		}
		}

DATABASE	Input	For MULTISCAN Solution MySQL or Postgres

Table 61. Input / Output Data Format AI^{DO}

4.6.2.2 AI^{DO} Hardware Requirements

Hardware Element	Importance and Explanation
AR Hardware and Android device	For the visualization of the AI^{DO} results and output. This is needed to support the human worker to implement the optimal solution provided by the tool e.g., a guide on how to arrange the machine parts in the boxes and in the shipping container. The AR hardware should be capable of running an HTTPS server. The Android device should have Chrome 81 or newer installed

Table 62. Hardware required AI^{DO}

4.6.2.3 AI^{DO} Software Requirements

Software Component	Description/Role	Required Version/Configuration
Windows OS	Microsoft operating system its need to use specific tool	Windows 11
Microsoft Office Suite	Suite of office with excel applications for document processing.	Office 365

Table 63. Software requirements AI^{DO}

4.6.2.4 AI^{DO} Lifecycle

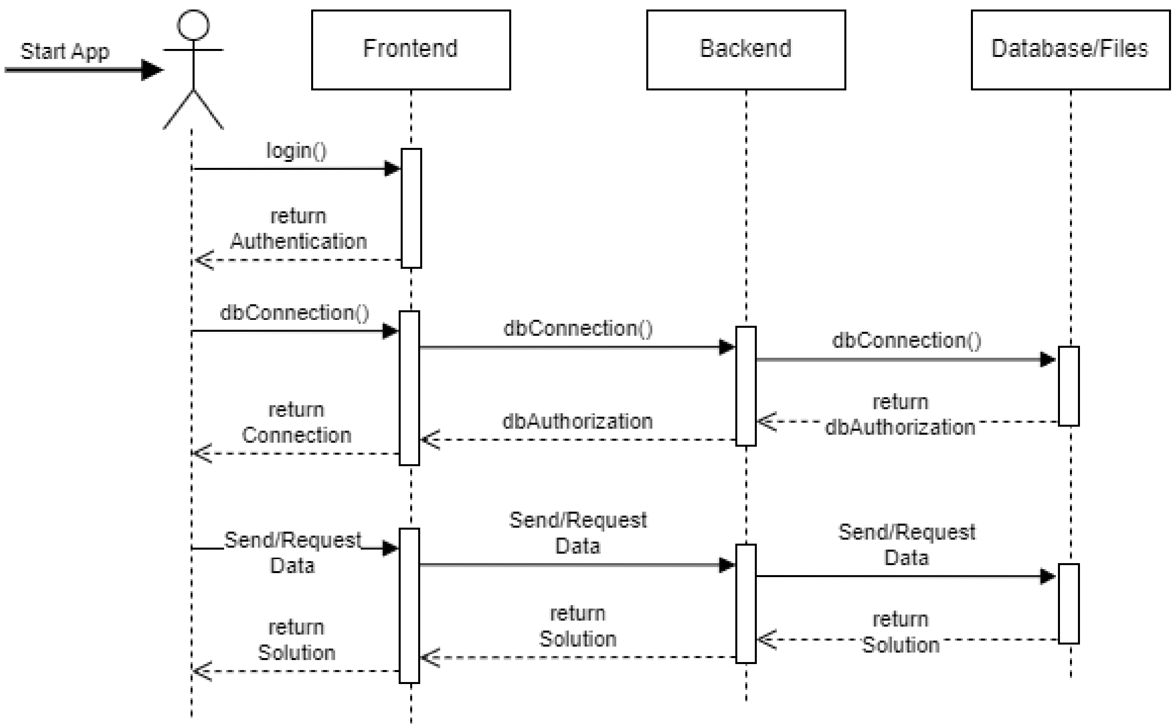


Figure 30. AI^{DO} Lifecycle

This sequence diagram illustrates interactions between the Frontend, Backend and Database. The user will have to login via the AIDEAS platform. The output of the AI model (optimised solution for packaging, storage, and delivery) is visualized on the Frontend.

4.6.2.5 Objects

- **User**: Represents the user interacting with the application.
- **Frontend**: Represents the user interface and the presentation layer.
- **Backend**: Represents the application logic and the server layer.
- **Database/Files:** Represents the data or file storage layer.

4.6.2.6 Description – Login

Objects	Description
Start	The user logs into the AIDEAS platform.
User – Frontend Interaction	The user enters username and password.
Frontend - User Interaction	The Frontend shows the User the response from the login action, indicating whether it is logged in or not.
Completion	The user has access to the AIDEAS suite and components related to the user's level of access, displayed on the dashboard.

Table 64. Life-Cycle description Login AI^{DO}

4.6.2.7 Description – Database connection

Objects	Description
User – Frontend Interaction	The user requests access to the database or to send files.
Frontend - Backend Interaction	The Frontend sends a request to the Backend to process a connection to the database or to process the file upload with multiple entries or a single entry, user is also able to access old entries into the database and save results from the AI^{DO} algorithm
Backend - Data/Files Interaction	The Backend requests the storage layer (data/files) to authorize the connection or to prepare to receive the file/entry. Data/Files is waiting for the authentication or for the file transfer
Data/Files - Backend Response	Data/Files confirms that the connection has been completed or the file/entry has been stored correctly and sends a response to the Backend.
Backend - Frontend Response	The Backend informs the Frontend about the success or failure of the connection or file upload/add entry operation.

Frontend - User Interaction	The Frontend shows the User the response from the application, indicating whether the connection or file upload/new entry was successful.
Completion	The application is ready to receive new requests from the User.

Table 65. Life-Cycle description Database connection AI[DO]

4.6.2.8 Description – Send/Request Data

Objects	Description
User – Frontend Interaction	The User will interact with the Frontend. For example, the user can enter the machine/part data along with the customer data and receive the most suitable packaging solution/storage solution/delivery route. This can either be done as a single entry or as a csv upload.
Frontend - Backend Interaction	The Frontend sends a request to the Backend to process the input data.
Backend - Data/Files Interaction	The Backend requests the storage layer to save the user entered data, while at the same time processes the data according to the deployed AI models and sends the output to the storage layer as well.
Data/Files - Backend Response	Data/Files sends a response to the Backend.
Backend - Frontend Response	The Backend, after computing the results and getting a response from the database, send either an error response or a success response to the frontend
Frontend - User Interaction	The Frontend shows the User the response from the application, i.e. the packaging/storage/delivery results as per the user ordered
Completion	The application is ready to receive new requests from the User.

Table 66. Life-Cycle description Send/RequestData AI[DO]

4.6.3 Implementation Viewpoint

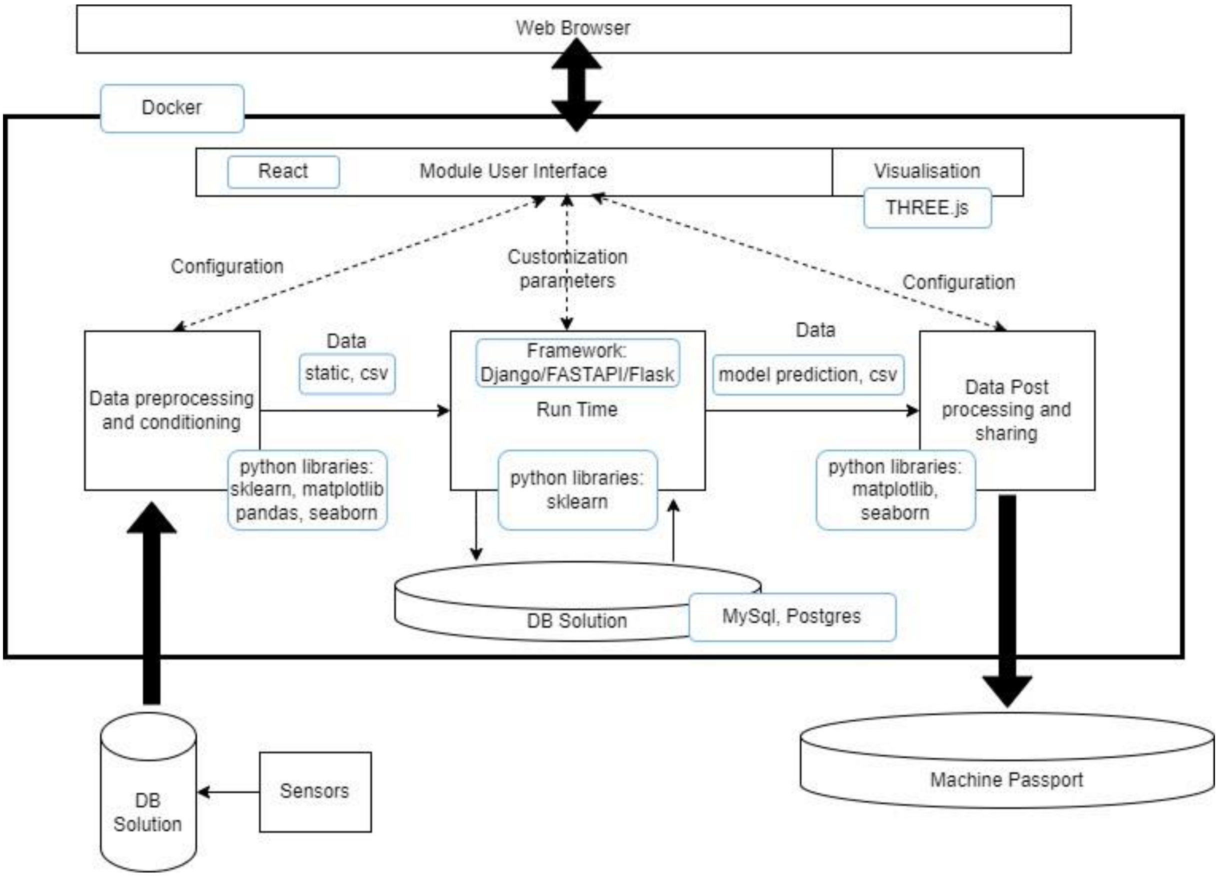


Figure 31. AI^{DO} Implementation Architecture

The main development language for AI^{DO} is Python, due to its data processing and machine learning functionalities. The main libraries utilised are sklearn for model training and Pymoo for the optimisation algorithm. Docker is also used for containerisation. A major update following the AI^{DO} solution architecture that was provided in M9, is the use of the "Three.js" library in place of Grafana for data visualisation and the use of MySQL database in place of MongoDB. In addition, input data will be accessed and collected from databases and ERP systems provided by the pilots, while the results and output data of the AI^{DO} solution will be stored in the Machine Passport.

4.6.3.1 AI^{DO} Implementation Components

Implementation Components	Description
Import Data	This component facilitates the reading of data from different data sources like databases, ERPs, files etc.
Data Validation and Preprocessing	This component validates the training data and ensures that the input data is in the correct format before feeding it into the model.
Create and Export Respective ML Models	This component provides different ML models, which train on the available pre-processed data to make predictions for optimising product packaging, storage, and delivery.

Create Predictions and Display Results	This component predicts optimal storage, delivery and packaging with new data using the trained models.

Table 67. AI^DO Implementation Components

4.6.3.2 Technical Description of the AI^DO Components

Following the identification and specification of the implementation components in the previous section, the technical description of each component is provided in the following tables. It details the set of technologies required to implement each of the implementation components.

Implementation component	Import Data
Description of implementation component	This component facilitates the reading of data from different data sources like databases, ERPs, files etc.
Used technologies	Python
Technical Description of the Component	Dependencies
	Development Language: Python, JavaScript Libraries: Pandas Container: Docker Database: Different clients (Postgres, MySQL, minIO)
	Interfaces
	User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: Postgres, MySQL, minIO

Table 68. Technical Description of AI^DO "Import Data" Implementation Component

Implementation component	Data Validation and Preprocessing
Description of implementation component	This component validates the training data and ensures that the input data is in the correct format before feeding it into the model.
Used technologies	Python
	Dependencies
	Development Language: Python Libraries: Pandas, sklearn Container: Docker
	Interfaces
	User Interface: No

Technical Description of the Component	Synchronous/Asynchronous Interface: No Network/Protocols: No Data Repository: No
	Requires
	"Import Data" component

Table 69. Technical Description of AI[DO] "Data Validation and Preprocessing" Implementation Component

Implementation component	Create and export respective ML models
Description of implementation component	This component provides different ML models, which train on the available pre-processed data to make predictions for optimising product packaging, storage and delivery.
Used technologies	Python
Technical Description of the Component	Dependencies
	Development Language: Python Libraries: sklearn, numPy, pickle Container: Docker Database: No
	Interfaces
	User Interface: No Synchronous/Asynchronous Interface: No Network/Protocols: No Data Repository: No
	Requires
	"Data validation and preprocessing" component

Table 70. Technical Description of AI[DO] "Create and Export Model" Implementation Component

Implementation component	Create Predictions and display results
Description of implementation component	This component predicts optimal storage, delivery and packaging with new data using the trained models.
Used technologies	Python, Django, REACT
	Dependencies
	Development Language: Python, JavaScript Libraries: Three.js, sklearn, numPy, matplotlib Container: Docker Database: Postgres, MySQL, minIO

Technical Description of the Component	Interfaces
	User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: Postgres, MySQL, minIO
	Requires
	"Create and export respective ML models" component

Table 71. Technical Description of AI[DO] "Create Predictions and Display Results" Implementation Component

4.7. Machine Calibrator – AI^{MC}

4.7.1 Usage Viewpoint

The usage activity diagram (Figure 32) within AI^{MC} demonstrates the interaction between the tasks and roles of D2TECH and BBM. This visual representation illustrates the dynamic flow of interactions, offering a representation of the activities executed by the solution. Through this diagram, the operational processes and collaborative engagements between different tasks and roles are presented, providing a clear understanding of the activities carried out by AI^{MC} in its functioning.

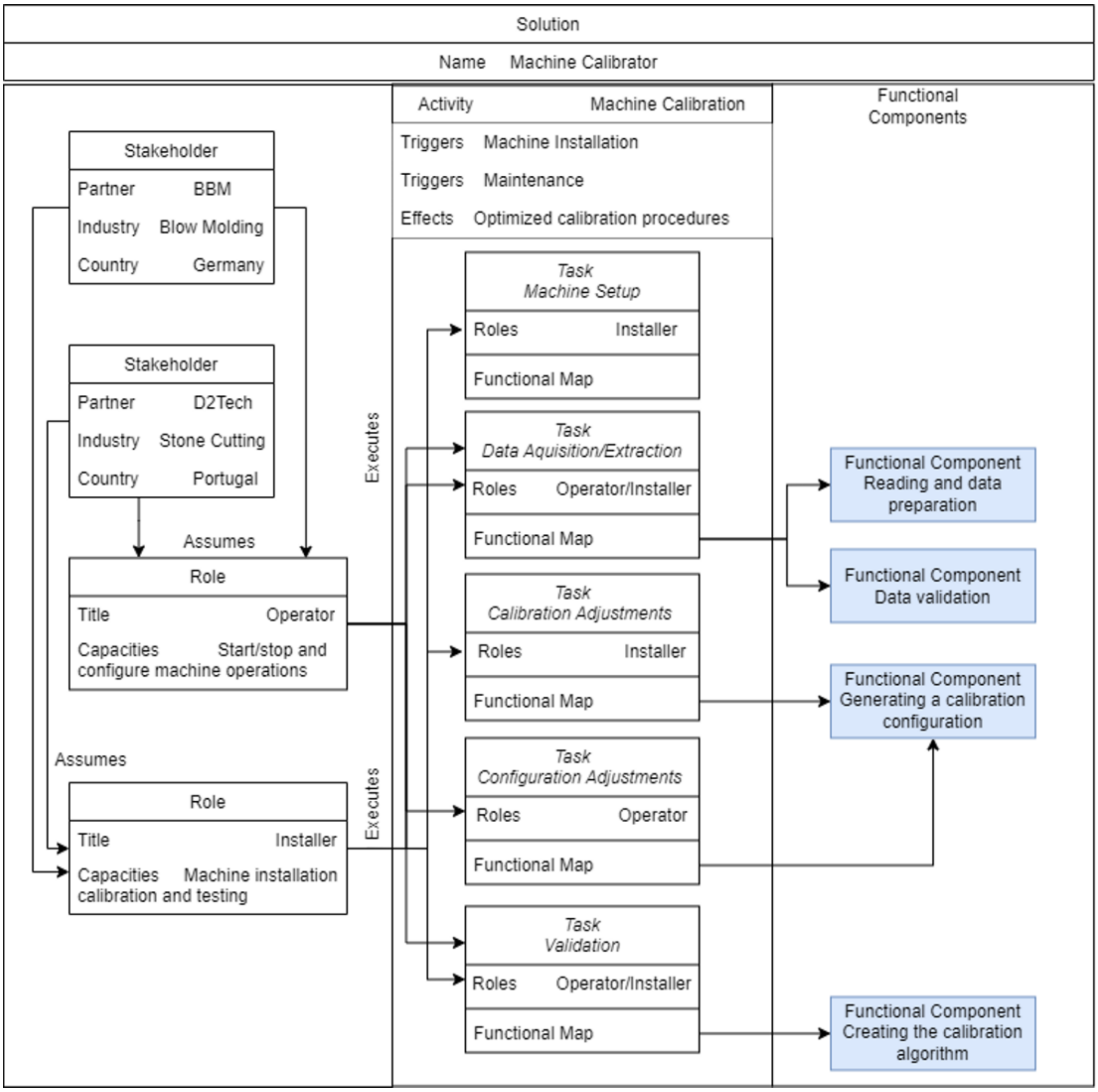


Figure 32. Machine Calibrator Usage Viewpoint Activity Diagram (Machine Calibration).

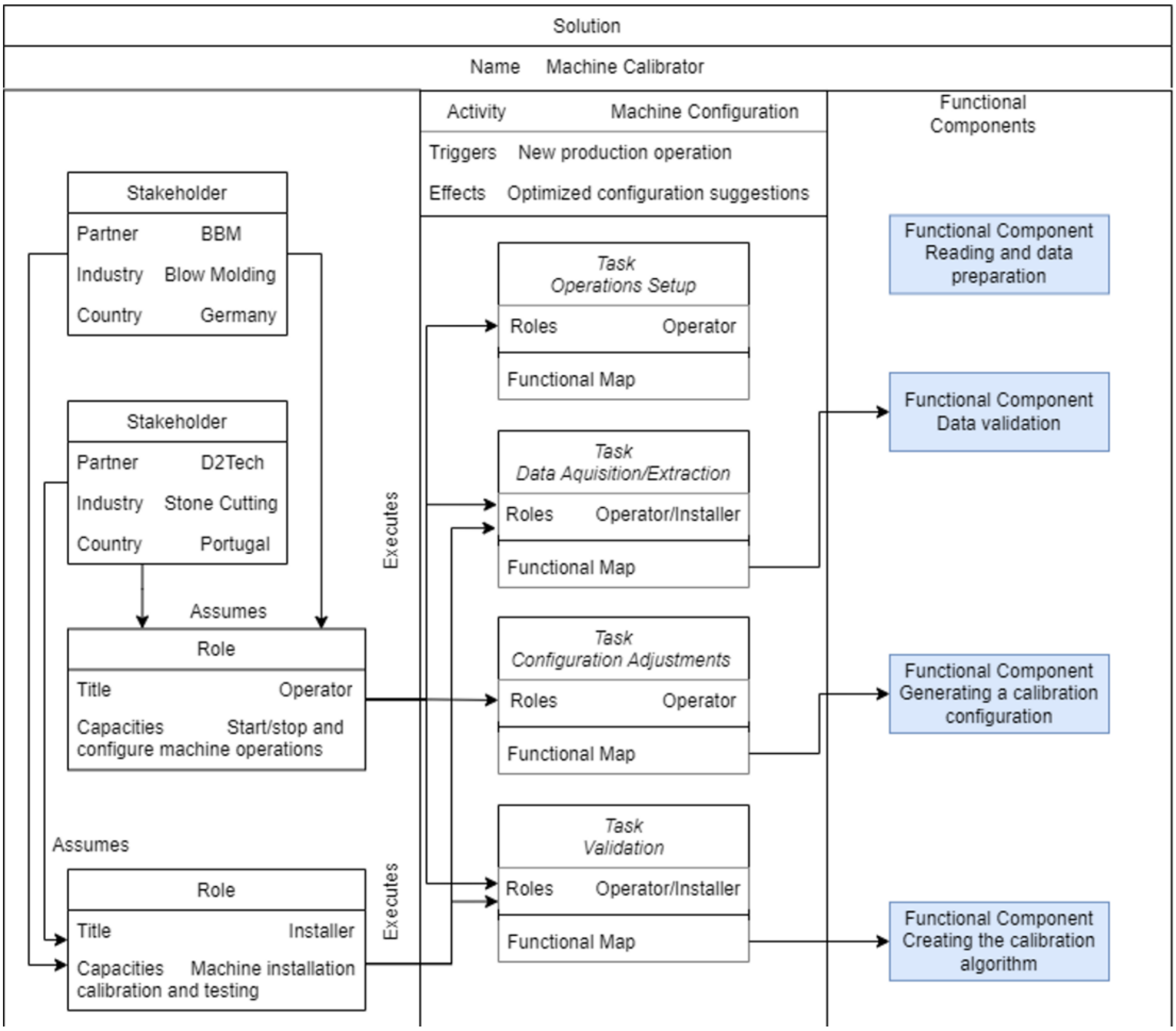


Figure 33. Machine Calibrator Usage Viewpoint Activity Diagram (Machine Configuration).

4.7.2 Functional Viewpoint

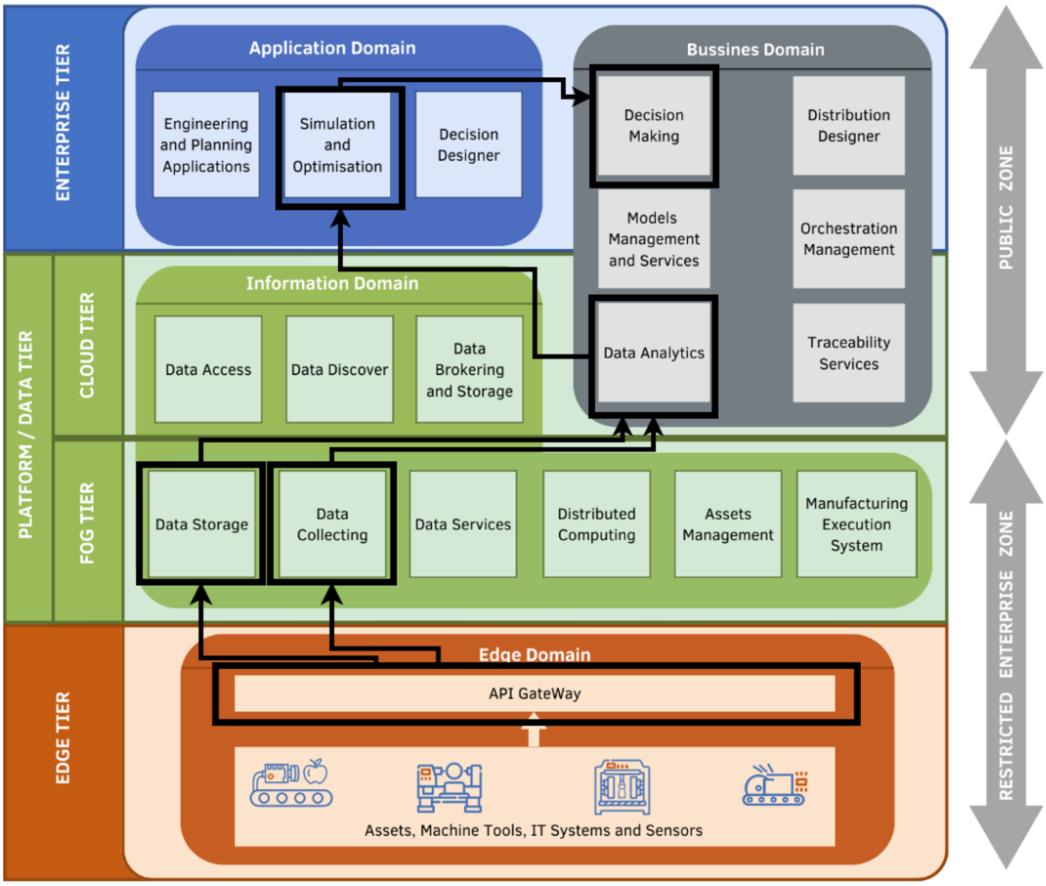


Figure 34. Data Flow AI^MC^

Machine Calibrator solution will gather machine data and other data from acquisition systems, during the production, setup, testing, configuration, installation, and operation phases. The raw data collected will be pooled, cleaned, and stored in a database. Afterwards, a data analysis will evaluate the calibration parameters, to advise the operator of corrective adjustments.

4.7.2.1 Data structure of AI^MC^

Format	Input/Output	Example
Database	Input	PostGRES Timestamp, Machine parameters.
CSV	Input	Timestamp, Machine parameters.
Video	Input	Video acquisition of machine operation.
JSON	Output	Corrective adjustment suggestions { "MachineId": { "Operation_Order":1, "Parameter":"Parameter_Name",

		{
		"CorrectiveValue":"Corrective_Value"
		}
		}
		}

Table 72. Input / Output Data Format AI^{MC}

4.7.2.2 AI^{MC} Software Requirements

Software Component	Description/Role	Required Version/Configuration	Dependencies
OS	Base OS	Linux (Ubuntu 22.04)/Windows 10 Pro	N/A
Docker	Build, share and run containers	latest	N/A

Table 73. Software requirements AI^{MC}

4.7.2.3 AI^{MC} Lifecycle

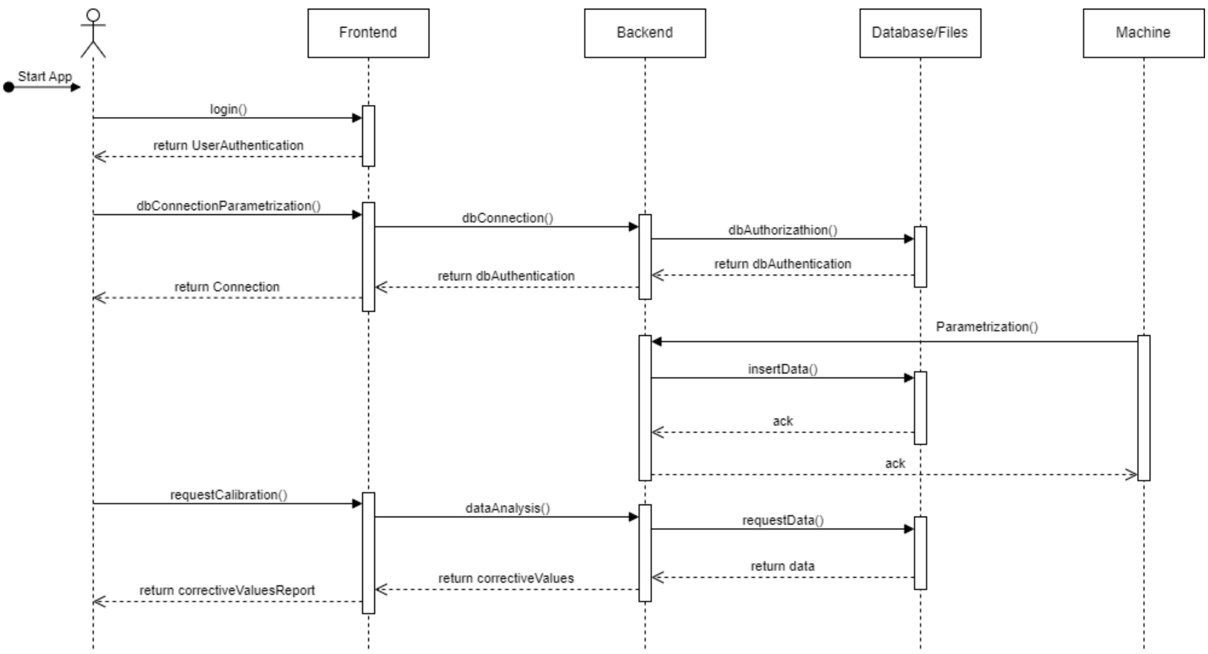


Figure 35. AI^{MC} Lifecycle

This sequence diagram illustrates the flow of interactions between the User, Frontend, Backend and Data/Files objects during the machine calibration processes. The parametrizations are reported by the machine sensors/controllers to the backend for storage and analysis, when the operator is performing the installation and calibration of the machines, presenting the corrective values in the frontend.

4.7.2.4 Objects

- **User**: Represents the user interacting with the application
- **Frontend**: Represents the user interface and the presentation layer.
- **Backend**: Represents the application logic and the server layer.
- **Database/Files**: Represents the data or file storage layer.
- **Machine**: Represents the machine outputs.

4.7.2.5 Description – login:

Objects	Description
Start	The user logs into the AIDEAS platform.
User – Frontend Interaction	The user enters username and password.
Frontend - User Interaction	The Frontend shows the User the response from the login action, indicating whether it is logged in or not.
Completion	The application is ready to receive new requests from the User.

Table 74. Life-Cycle description login AI[MC]

4.7.2.6 Description – Database Connection Parametrization:

Objects	Description
User – Requests Parametrization Update	The user issues an update for visualizing the parametrization data.
Frontend-Backend-DB data fetching	The request is authenticated and authorized by the backend and parametrization data is fetched from the db.
Refresh frontend data visualizer	The frontend displays the parametrization data.

Table 75. Life-Cycle description dbConnectionParametrization AI[MC]

4.7.2.7 Description – Parametrization:

Objects	Description
Parametrization	Machine updates the current parametrization data into the backend
insertData	The backend inserts the machine data in the db.

Table 76. Life-Cycle description Parametrization AI[MC]

4.7.2.8 Description – Request Calibration

Objects	Description
User- Requests new calibration	The User requests a new calibration in the frontend interface

dataAnalysis	The backend runs the data analytics on the machine data.
correctiveValues	The corrective values for the machine parameters are calculated.
correctiveValuesReport	The corrective values are displayed in the frontend for the user to apply on the machine.

Table 77. Life-Cycle description requestCalibration AI[MC]

4.7.3 Implementation Viewpoint

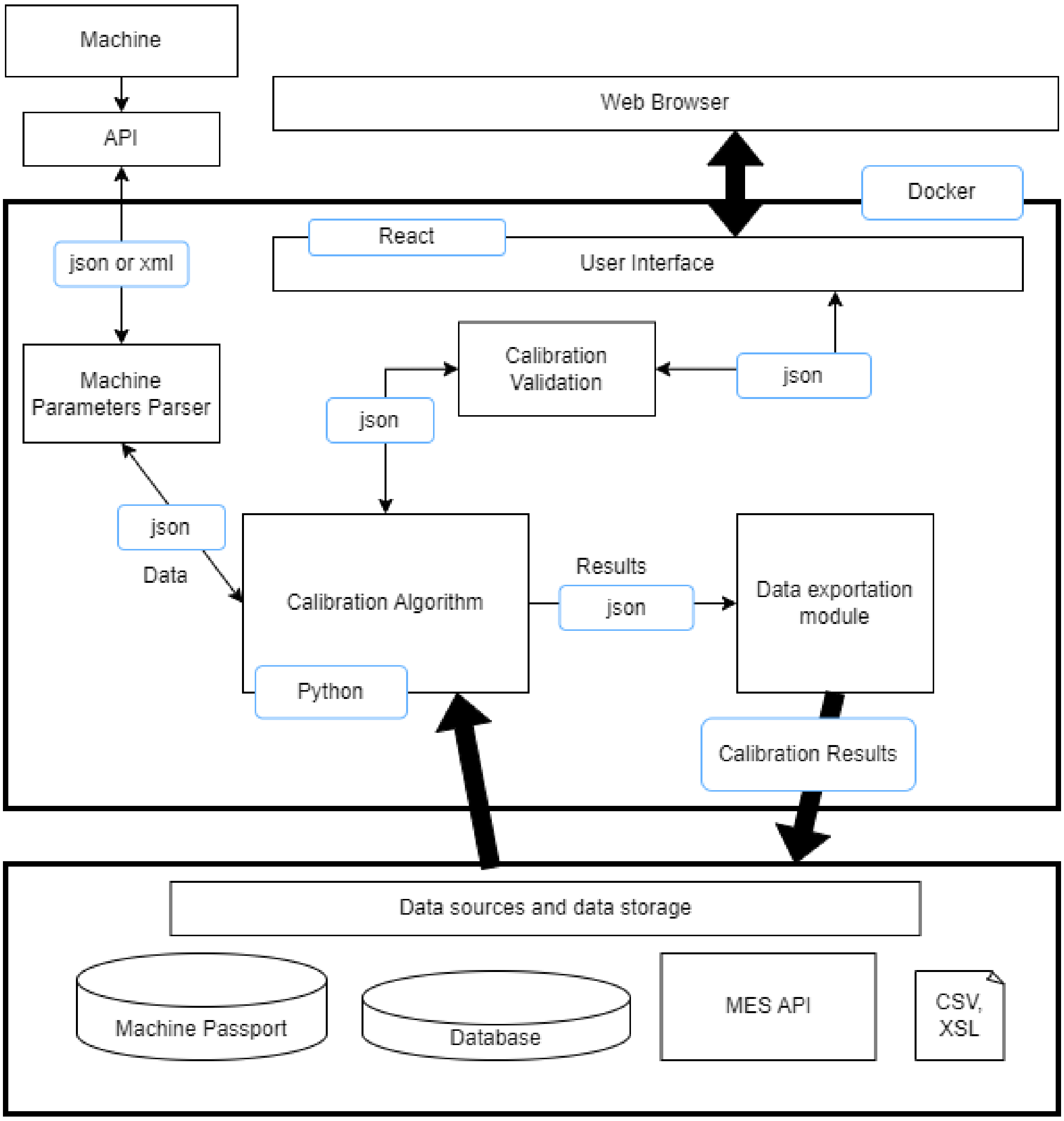


Figure 36. AI[MC] Implementation Architecture

The MC will leverage the data acquired from the machine operators on the initial equipment configuration for future parameter suggestions. The solution will be developed in Python, supported by extensive machine learning packages, using a SQL database in a Docker container environment. Data inputs/outputs will be consumed using a RESTful API and the calibration suggestions (as well as historical data) will be made available to the machine operators in a User Interface based on the AIDEAS project REACT template.

4.7.3.1 AI^{MC} Implementation Components

Implementation Components	Description
Reading and data preparation	In this step, the solution receives the data and proceed to data processing and conditioning
Data validation	Checking the conformity/fidelity of the data, to see if it is within the acceptable limits
Creating the calibration algorithm	The validated data is used to generate the machine calibration assistance algorithm
Generating a calibration configuration	The algorithm is executed by analysing in real time whether the calibration data entered by the operator conforms to the target parameters for a successful configuration.

Table 78. AI^{MC} Implementation Components

4.7.3.2 Technical Description of AI^{MC} Components

Implementation component	Reading and data preparation
Description of implementation component	In this step, the solution receives the data and proceeds to data processing and conditioning
Technical Description of the Component	**Dependencies**
	Development Language: Python Libraries: Python ML libraries Container: Docker Database: PostGRESql
	Interfaces
	User Interface: React Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: PostGRESql/GitLab

Table 79. Technical Description of AI^{MC} "Reading and data preparation" Implementation Component

Implementation component	Data validation
Description of implementation component	Checking the conformity/fidelity of the data, to see if it is within the acceptable limits.

Technical Description of the Component	Dependencies Development Language: Python Libraries: Python ML libraries Container: Docker Database: PostGRESql Interfaces User Interface: React Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: PostGRESql/GitLab

Table 80. Technical Description of AI[MC] "Data validation" Implementation Component

Implementation component	Creating the calibration algorithm
Description of implementation component	The validated data is used to generate the machine calibration assistance algorithm
Technical Description of the Component	Dependencies Development Language: Python Libraries: Python ML libraries Container: Docker Database: PostGRESql Interfaces User Interface: React Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: PostGRESql/GitLab

Table 81. Technical Description of AI[MC] "Creating the calibration algorithm" Implementation Component

Implementation component	Generating a calibration configuration
Description of implementation component	The algorithm is executed by analysing in real time whether the calibration data entered by the operator conforms to the target parameters for a successful configuration.
	Dependencies Development Language: Python Libraries: Python ML libraries Container: Docker

Technical Description of the Component	Database: PostGRESql
	Interfaces
	User Interface: React Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: PostGRESql/GitLab

Table 82. Technical Description of AI^{MC} "Generating a calibration configuration" Implementation Component

4.8. Condition Evaluator – AI^{CE}

4.8.1 Usage Viewpoint

The usage activity diagram (Figure 37) within AI^{CE} demonstrates the interaction between the different tasks and roles. No pilot distinction is made, as these are the same independent on the use case. This diagram illustrates the dynamic flow of interactions, offering a representation of the activities executed by the solution providing a clear understanding of the activities executed using the AI^{CE} solution.

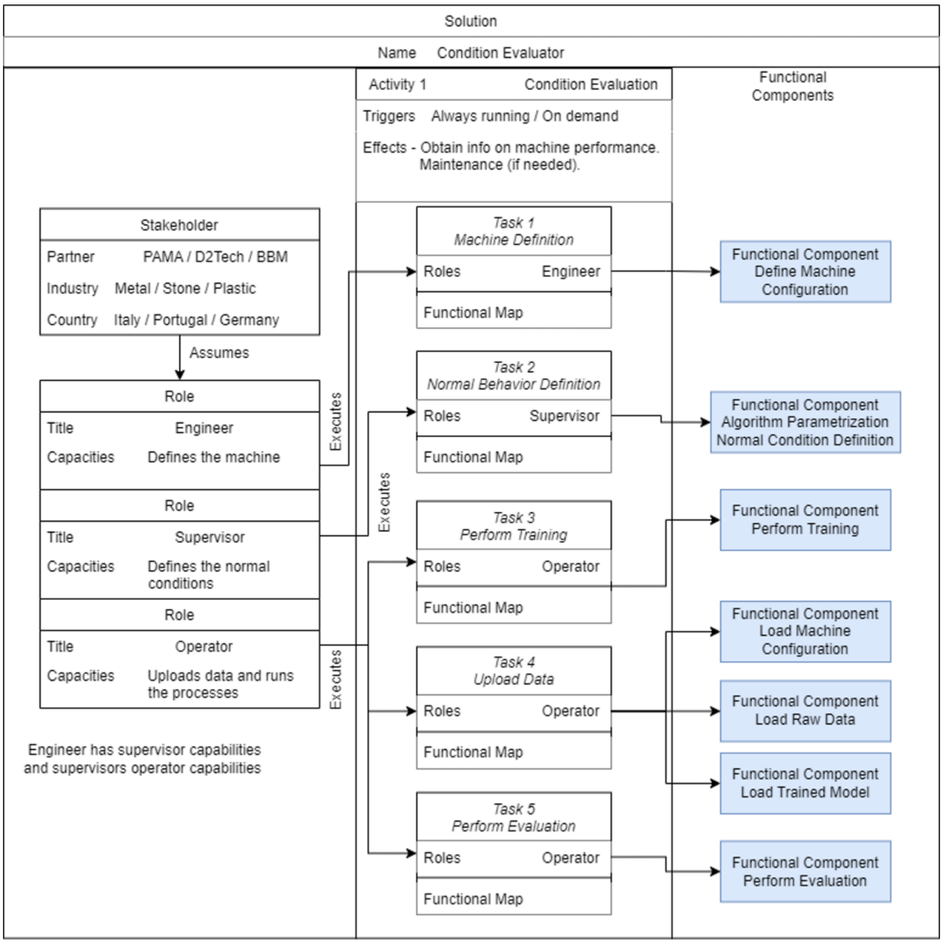


Figure 37. Condition Evaluation Usage Viewpoint Activity Diagram (Condition Evaluator).

4.8.2 Functional Viewpoint

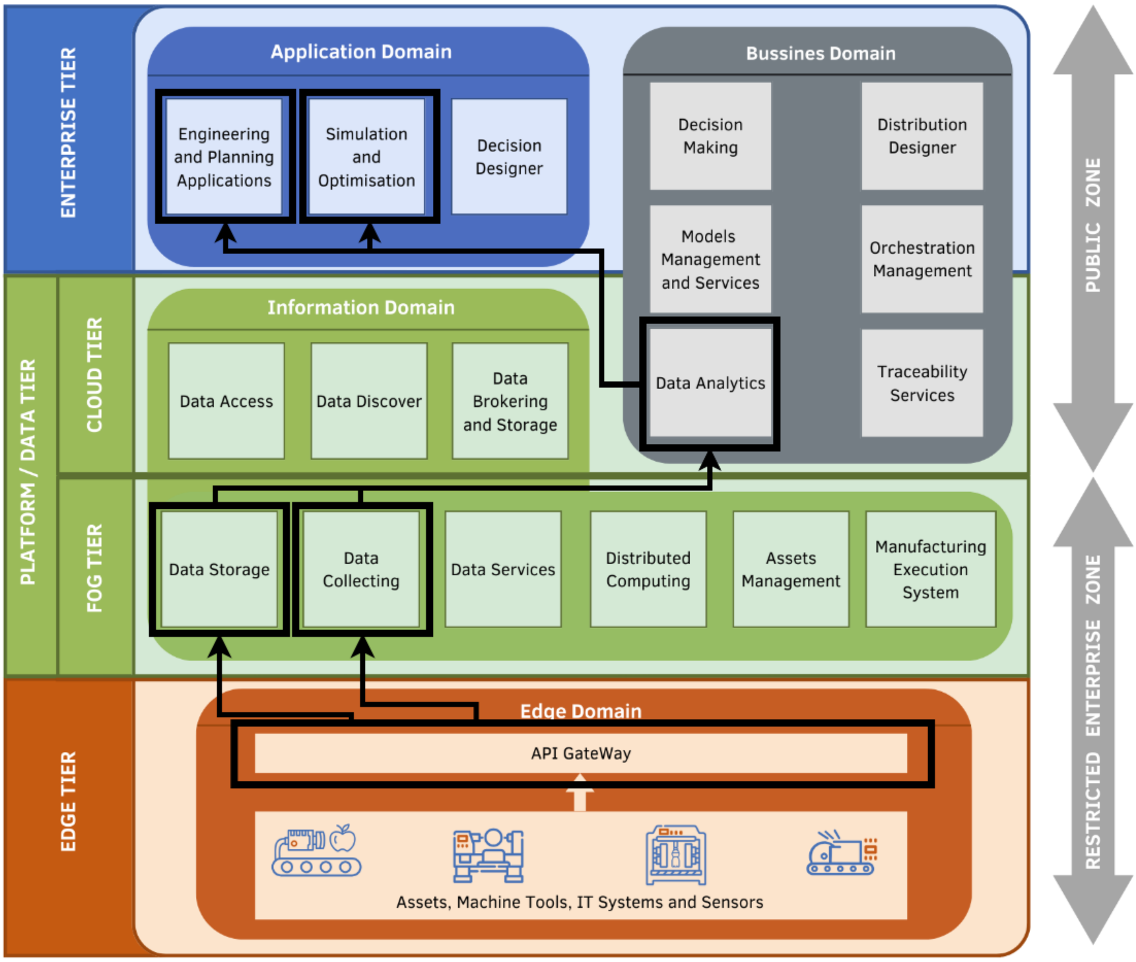


Figure 38. Data Flow AICE

Condition Evaluator gets machine operation data, gathered from PLCs and data acquisition systems in the edge, databases or .csv files through a data collection module to perform several analyses to determine if the machine, or any of its component, is working as expected in the factory where it is being used or otherwise it is deviating from what is known as normal behaviour, helping operators to know the current status of the machines.

4.8.2.1 Data structure of AICE

Format	Input/Output	Example
Database	Input	MONGODB timestamp, vibration level. 2023-11-30T08:00:00Z,0.5
CSV	Input	timestamp, vibration level. 2023-11-30T08:00:00Z,0.5
JSON	Input	Can also be created from solution's frontend.

		[{ "Name": "X_Axis_Pos", "Description": "X Axis Position", "MinVal": 0, "MaxVal": 1000, "MeasureUnit": "mm", "ParentGroup": "X Axis" }, { "Name": "X_Axis_Vel", "Description": "X Axis Speed", "MinVal": 0, "MaxVal": 50, "MeasureUnit": "mm/s", "ParentGroup": "X Axis" } , …]

JSON	Output	Based on time interval
		
		{
		“StartTime”: “2023-11-30T08:00:00Z”,
		“EndTime”: “2023-11-30T10:00:00Z”,
		“Machine%”:
		{
		“Green%”: 0.8,
		“Orange%”: 0.15,
		“Red%”: 0.05
		} ,
		“Components%”:
		[
		{
		“Component”: “Component1”,
		“Green%”: 0.6,
		“Orange%”: 0.3,
		“Red%”: 0.1
		},
		…
		]
		}

Table 83. Input / Output Data Format AI^{CE}

4.8.2.2 AI^{CE} Software Requirements

Software Component	Description/Role	Required Version/Configuration	Dependencies
Linux OS or Windows OS	Operating system needed to use the tool	Ubuntu 22.04 / Windows 10 Pro	N/A
Docker	Build, share and run containers	latest	N/A

Table 84. Software requirements AI^{CE}

4.8.2.3 AI^{CE} Lifecycle

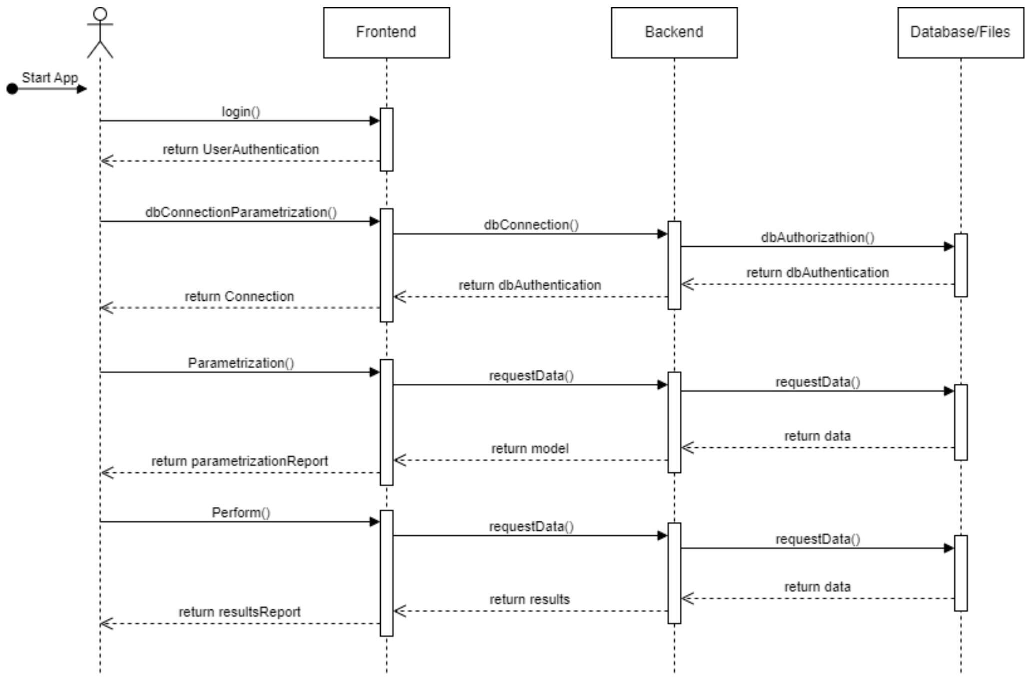


Figure 39. AIDEAS Condition Evaluator Sequence Diagram

This sequence diagram illustrates the flow of interactions between the User, Frontend, Backend and Data/Files objects during the main processes. Login into the AIDEAS platform, parametrize a connection to an external database, parametrize the condition evaluation and perform it.

4.8.2.4 Objects

- **User**: Represents the user interacting with the application.
- **Frontend**: Represents the user interface and the presentation layer.
- **Backend**: Represents the application logic and the server layer.
- **Database/Files:** Represents the data or file storage layer.

4.8.2.5 Description - Login:

Objects	Description
Start	The user logs into the AIDEAS platform.
User – Frontend Interaction	The user enters username and password.
Frontend - User Interaction	The Frontend shows the User the response from the login action, indicating whether it is logged in or not.
Completion	The application is ready to receive new requests from the User.

Table 85. Life-Cycle description Login AI[CE]

4.8.2.6 Description - Database Connection / File Upload:

Objects	Description
Start	The user initiates the action of parametrizing a database connection or uploads data files.
User – Frontend Interaction	The user parametrizes the database connection or uploads a data file.
Frontend - Backend Interaction	The Frontend sends a request to the Backend to process a connection to the database or to process the file upload.
Backend - Data/Files Interaction	The Backend requests the storage layer (Data/Files) to authorize the connection or to prepare to receive the file. Data/Files is waiting for the authentication or for the file transfer
Data/Files - Backend Response	Data/Files confirms that the connection has been completed or the file has been stored correctly and sends a response to the Backend.
Backend - Frontend Response	The Backend informs the Frontend about the success or failure of the connection or file upload operation.
Frontend - User Interaction	The Frontend shows the user the response from the application, indicating whether the connection or file upload was successful.
Completion	The application is ready to receive new requests from the User.

Table 86. Life-Cycle description Database Connection AI[CE]

4.8.2.7 Description - Condition Evaluation Parametrization:

Objects	Description
Start	The User initiates the action of parametrizing the condition evaluation.
User – Frontend Interaction	The User selects which kind of model wants to build, with which data and defines what is to be considered as normal behaviour
Frontend - Backend Interaction	The Frontend sends a request to the Backend to process the parameters sent
Backend - Data/Files Interaction	The Backend requests the storage layer (Data/Files) to authorize the connection or to prepare to receive the data. Data/Files is waiting for the data transfer.
Data/Files - Backend Response	Data/Files sends a response to the Backend
Backend - Frontend Response	The Backend gets the corresponding data and builds the model with the inputs defined previously by the user
Frontend - User Interaction	The Frontend shows the User the response from the application, obtaining a report for the model training and having the possibility to save it for its future use
Completion	The application is ready to receive new requests from the User

Table 87. Life-Cycle description Condition Evaluation Parametrization AI[CE]

4.8.2.8 Description – Perform Condition Evaluation:

Objects	Description
Start	The User initiates the action of performing the condition evaluation
User – Frontend Interaction	The User parametrizes in which components wants to perform the condition evaluation, indicates the desired model to evaluate and defines where the new data comes from
Frontend - Backend Interaction	The Frontend sends a request to the Backend to process the parameters sent.
Backend - Data/Files Interaction	The Backend requests the storage layer (Data/Files) to authorize the connection or to prepare to receive the data. Data/Files is waiting for the data transfer.
Data/Files - Backend Response	Data/Files sends a response to the Backend.

Backend - Frontend Response	The Backend gets the corresponding data and evaluates the model.
Frontend - User Interaction	The Frontend shows the User the response from the application, obtaining the condition evaluation report for the selected components
Completion	The application is ready to receive new requests from the User.

Table 88. Life-Cycle description Perform Condition Evaluation AI^CE

4.8.3 Implementation Viewpoint

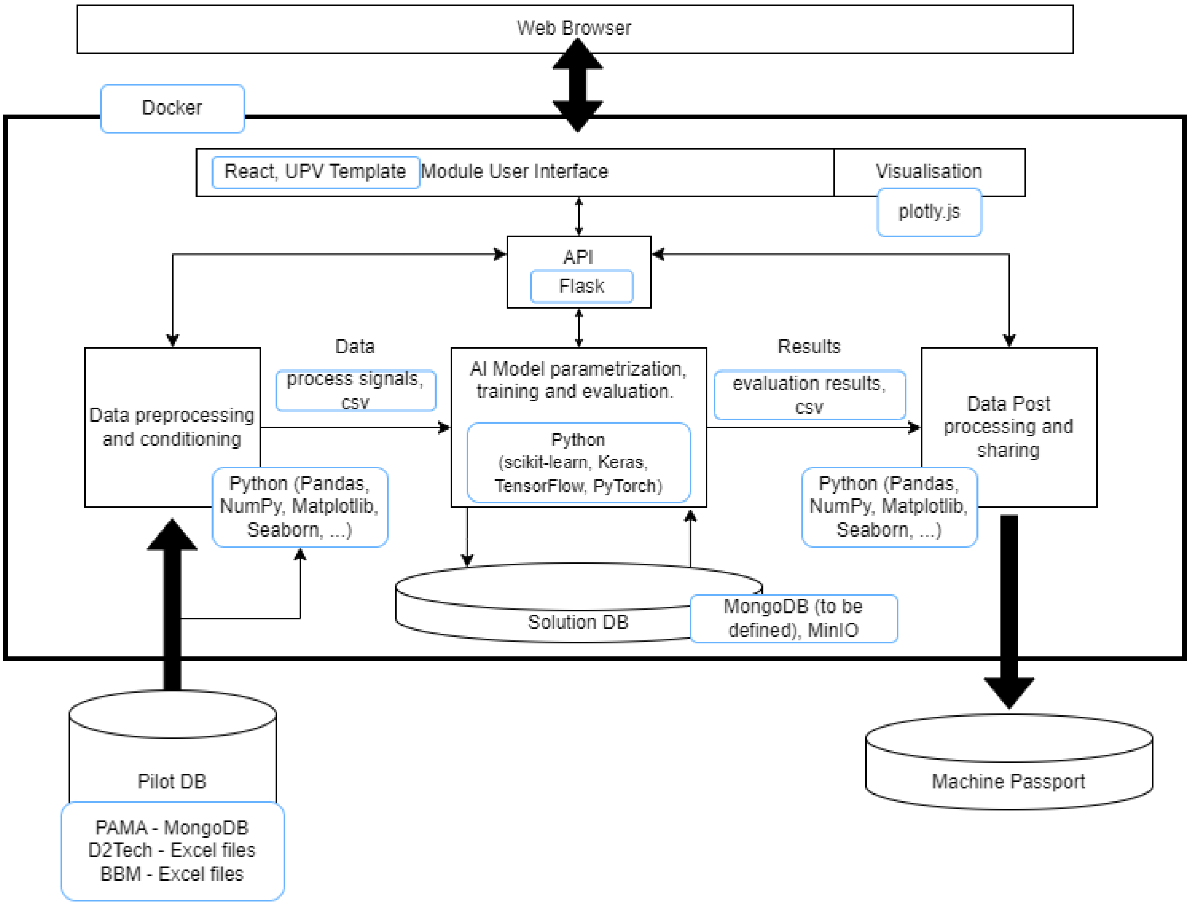


Figure 40. AI^CE Implementation Architecture

For AI^CE, the main development language is Python, NumPy and pandas are used for data management and basic operations, sklearn, keras and TensorFlow are used for model training. The solution is containerized using Docker. A minor update following the solution architecture provided in M9 is the use of plotly.js library in place of the proposed options for data visualization purposes and the use of Flask as the framework for the API server. The use of an internal DB for the solution is still pending to be defined. At, the moment, no internal database is used. In addition, input data will be accessed and collected from database (MongoDB) or files (excel files) provided by the pilots. Specific data preprocessing must be done on the pilots' data to get the expected input data. Finally, the results and solution's outputs will be sent to the AIDEAS Machine Passport.

4.8.3.1 AI^CE Implementation Components

Implementation Components	Description
Import Data	This component facilitates the reading of data from different data sources like databases, files etc.
Data Validation and Preprocessing	This component validates the training data and ensures that the input data is in the correct format before feeding it into the model.
Machine Configurator	This component allows the definition of the current machine configuration to identify its component and associated variables. It also allows the definition of what is to be considered as normal behaviour.
Create and Export Models	This component provides different algorithms, which train on the available pre-processed data to determine if the system is working properly, or if the system is not behaving as expected.
Obtain Predictions and Display Results	This component determines if the system is behaving as expected or if it is deviating from normal behaviour, giving a score, with new data using the trained models.
Export Data	This component sends the data to the Machine Passport and enables the user to export it directly if needed.

Table 89. AI^CE Implementation Components

4.8.3.2 Technical Description of AI^CE Components

Following the identification and specification of the implementation components in the previous section, the technical description of each component is provided in the following tables. It details the set of technologies required to implement each of the implementation components.

Implementation component	Import Data
Description of implementation component	This component facilitates the reading of data from different data sources like databases, files etc.
Used technologies	Python
Technical Description of the Component	**Dependencies**
	Development Language: Python Libraries: Pandas Container: Docker Database: to be defined
	Interfaces
	User Interface: Yes, REACT

	Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: MongoDB, minIO

Table 90. Technical Description of AI[CE] "Import Data" Implementation Component

Implementation component	Data Validation and Preprocessing
Description of implementation component	This component validates the training data and ensures that the input data is in the correct format before feeding it into the model.
Used technologies	Python
Technical Description of the Component	**Dependencies** Development Language: Python Libraries: Pandas, numpy Container: Docker Database: to be defined **Interfaces** User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: MongoDB, minIO **Requires** Import Data

Table 91. Technical Description of AI[CE] "Data Validation and Preprocessing" Implementation Component

Implementation component	Machine Configurator
Description of implementation component	This component facilitates the definition of the machine that will be the target of the condition evaluation, defining its components and its relevant process variables. It also allows defining different thresholds to assess how good the machine is performing.
Used technologies	Python
	Dependencies Development Language: Python Libraries: Pandas, numpy Container: Docker Database: to be defined

Technical Description of the Component	**Interfaces**	
	User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: MongoDB, minIO	
	Requires	
	Import Data	

Table 92. Technical Description of AI[CE] "Machine Configurator" Implementation Component

Implementation component	Create and Export Models
Description of implementation component	This component facilitates the parametrization and creation of new machine learning models, which train on the previously pre-processed data given a specific machine configuration.
Used technologies	Python
Technical Description of the Component	**Dependencies**
	Development Language: Python Libraries: sklearn, keras, tensorflow, pickle Container: Docker Database: to be defined
	Interfaces
	User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: MongoDB, minIO
	Requires
	Import Data, Data Validation and Preprocessing, Machine Configurator

Table 93. Technical Description of AI[CE] "Create and Export Models" Implementation Component

Implementation component	Obtain Predictions and Display Results
Description of implementation component	This component predicts how good the machine is performing given new data using previously trained models. Results are shown in different formats.
Used technologies	Python
	Dependencies

Technical Description of the Component	Development Language: Python, Javascript Libraries: sklearn, keras, tensorflow, pickle, plotly.js Container: Docker Database: to be defined
	Interfaces
	User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: MongoDB, minIO
	Requires
	Import Data, Data Validation and Preprocessing, Machine Configurator, Create and Export Models

Table 94. Technical Description of AI^{CE} "Obtain Predictions and Display Results" Implementation Component

Implementation component	Export Data
Description of implementation component	This component facilitates exporting the obtained results to a local storage or send them to the Machine Passport.
Used technologies	Python
Technical Description of the Component	**Dependencies**
	Development Language: Python Libraries: Pandas, numpy Container: Docker Database: MongoDB
	Interfaces
	User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: MongoDB, minIO
	Requires
	Machine Configurator, Obtain Predictions and Display Results

Table 95. Technical Description of AI^{CE} "Export Data" Implementation Component

4.9. Anomaly Detector – AI[AD]

4.9.1 Usage Viewpoint

The usage activity diagram (Figure 41) within AI[AD] demonstrates the interaction between the different tasks and roles. No pilot distinction is made, as these are the same independent of the use case. This diagram illustrates the dynamic flow of interactions, offering a representation of the activities executed by the solution providing a clear understanding of the activities executed using the AI[AD] solution.

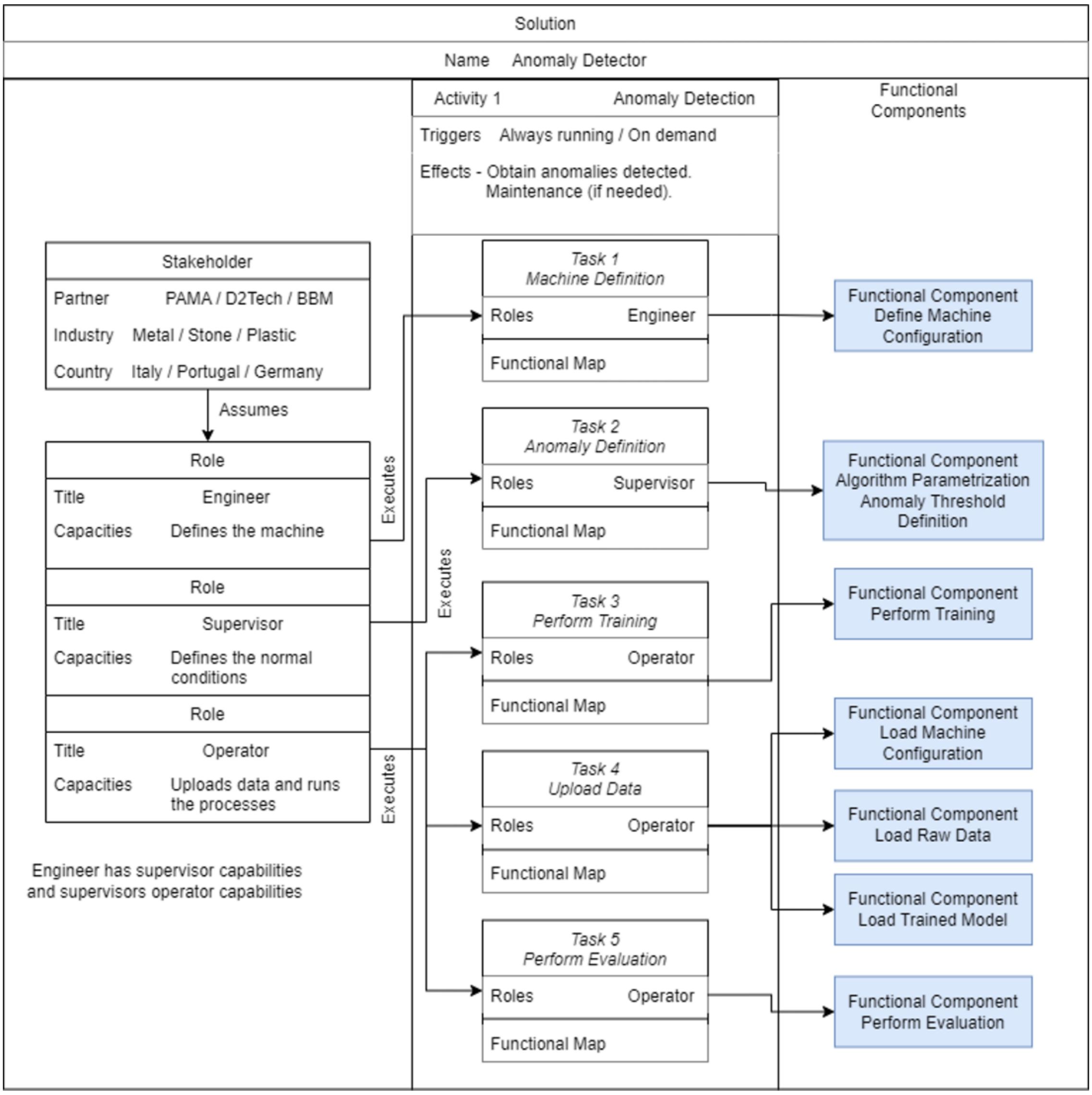


Figure 41. AI[AD] Activity Diagram (Anomaly Detector)

4.9.2 Functional Viewpoint

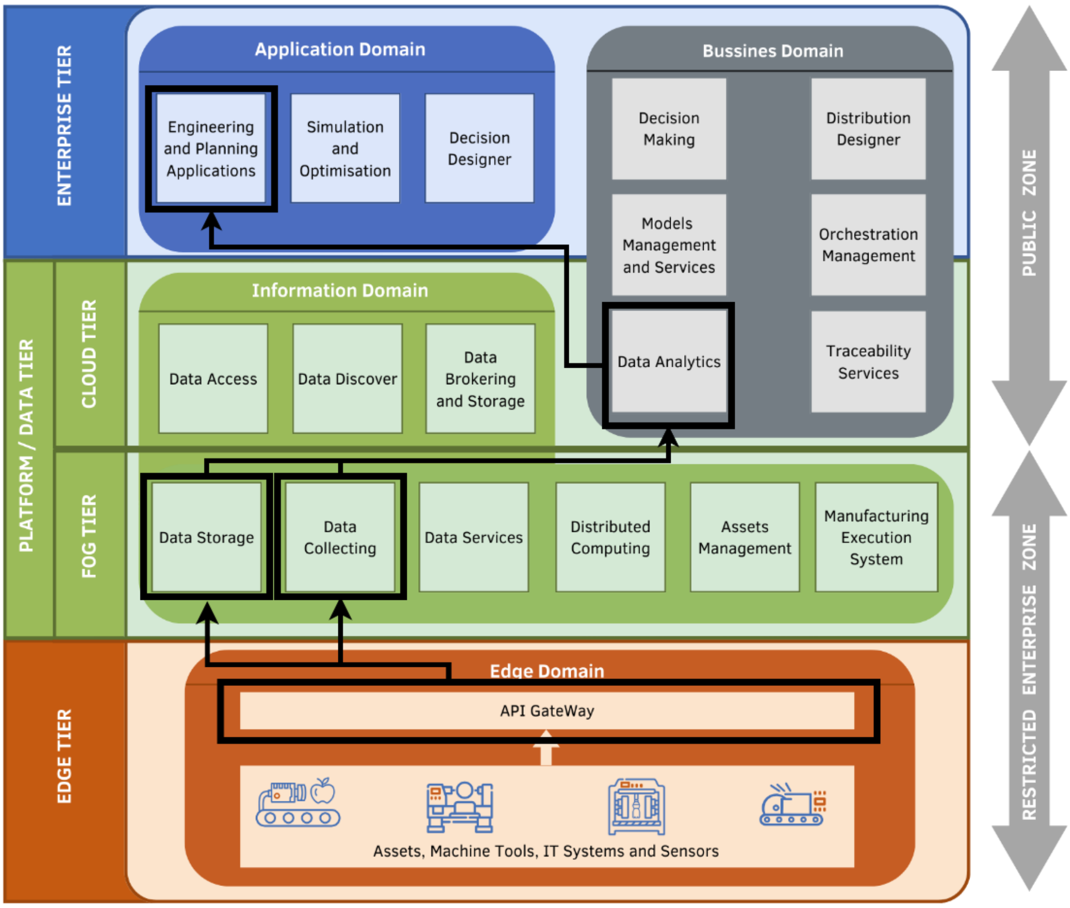


Figure 42. Data Flow AI[AD]

Anomaly Detector gets machine operation data, gathered from PLCs and data acquisition systems in the edge, databases or .csv files through a data collection module to perform several analyses to determine if the machine, or any of its component, has encountered any anomaly while it is working as expected in the factory where it is being used, helping operators to know if there is any deviation from the expected behaviour.

4.9.2.1 Data structure of AI[AD]

Format	Input/Output	Example
Database: MONGODB	Input	timestamp, vibration level. 2023-11-30T08:00:00Z,0.5
CSV	Input	timestamp, vibration level. 2023-11-30T08:00:00Z,0.5
JSON	Input	Can also be created from solution's frontend. [{ "Name": "X_Axis_Pos",

		"Description": "X Axis Position", "MinVal": 0, "MaxVal": 1000, "MeasureUnit": "mm", "ParentGroup": "X Axis" }, { "Name": "X_Axis_Vel", "Description": "X Axis Speed", "MinVal": 0, "MaxVal": 50, "MeasureUnit": "mm/s", "ParentGroup": "X Axis" } , ...]

JSON	Output	Based on event
		
		{
		"Timestamp": "2023-11-30T08:00:00Z",
		"AnomalyDescription": "An anomaly has been detected on X axis motor, current is above 4.5A"
		}

JSON	Output	Based on time interval
		
		{
		"StartTime": "2023-11-30T08:00:00Z",
		"EndTime": "2023-11-30T10:00:00Z",
		"Anomaly Lyst":
		[
		{
		"Timestamp": "2023-11-30T08:00:00Z",
		"AnomalyDescription": "An anomaly has been detected on X axis motor, current is above 4.5A"
		}
		]
		}

Table 96. Input / Output Data Format AI[AD]

4.9.2.2 AI^AD Software Requirements

Software Component	Description/Role	Required Version/Configuration	Dependencies
Linux OS or Windows OS	Operating system needed to use the tool	Ubuntu 22.04 / Windows 10 Pro	N/A
Docker	Build, share and run containers	latest	N/A

Table 97. Software requirements AI^AD

4.9.2.3 AI^AD Lifecycle

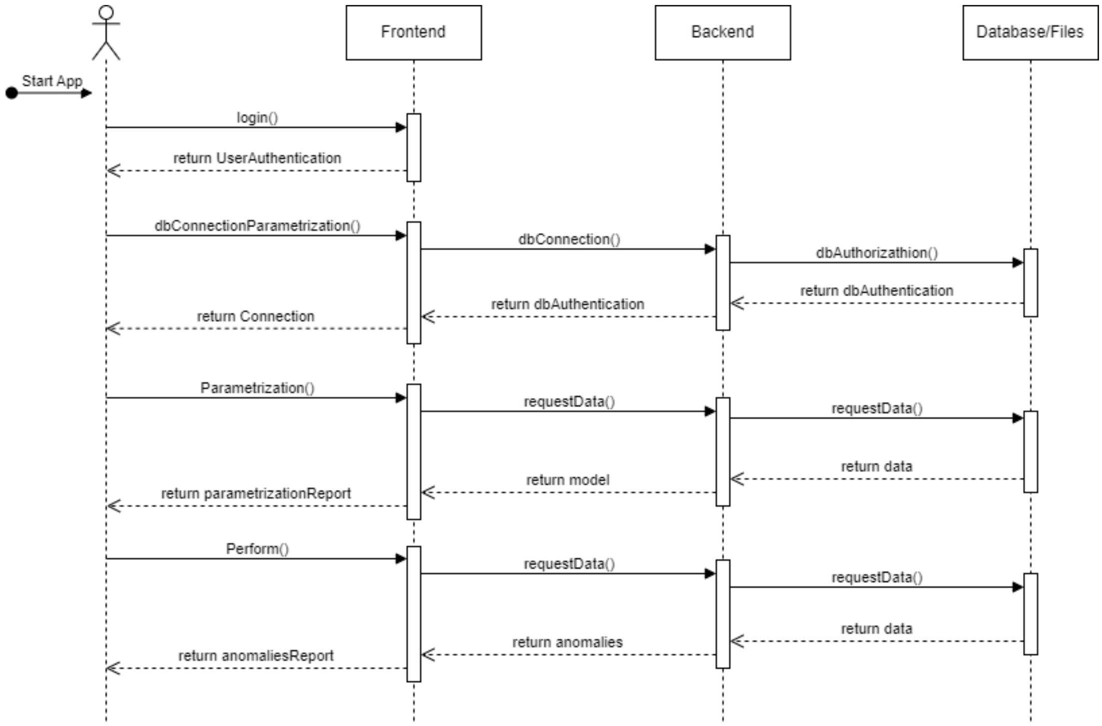


Figure 43. AIDEAS Anomaly Detector Sequence Diagram

This sequence diagram illustrates the flow of interactions between the User, Frontend, Backend and Data/Files objects during the main processes. Login into the AIDEAS platform, parametrize a connection to an external database, parametrize the anomaly detection and perform it.

4.9.2.4 Objects

- **User**: Represents the user interacting with the application.
- **Frontend**: Represents the user interface and the presentation layer.
- **Backend**: Represents the application logic and the server layer.
- **Database/Files: Represents the data or file storage layer.**

4.9.2.5 Description - Login:

Objects	Description
Start	The User logs into the **AIDEAS** platform.
User – Frontend Interaction	The User enters user and password.
Frontend - User Interaction	The Frontend shows the User the response from the login action, indicating whether it is logged in or not.
Completion	The application is ready to receive new requests from the User.

Table 98. Life-Cycle description Login AI[AD]

4.9.2.6 Description - Database Connection / File Upload:

Objects	Description
Start	The User initiates the action of parametrizing a database connection or uploads data files.
User – Frontend Interaction	The User parametrizes the database connection or uploads a data file.
Frontend - Backend Interaction	The Frontend sends a request to the Backend to process a connection to the database or to process the file upload.
Backend - Data/Files Interaction	The Backend requests the storage layer (Data/Files) to authorize the connection or to prepare to receive the file. Data/Files is waiting for the authentication or for the file transfer
Data/Files - Backend Response	Data/Files confirms that the connection has been completed or the file has been stored correctly and sends a response to the Backend.
Backend - Frontend Response	The Backend informs the Frontend about the success or failure of the connection or file upload operation.
Frontend - User Interaction	The Frontend shows the User the response from the application, indicating whether the connection or file upload was successful.
Completion	The application is ready to receive new requests from the User.

Table 99. Life-Cycle description Database Connection AI[AD]

4.9.2.7 Description - Condition Evaluation Parametrization:

Objects	Description
Start	The User initiates the action of parametrizing the AD.
User – Frontend Interaction	The User selects which kind of model wants to build and with which data
Frontend - Backend Interaction	The Frontend sends a request to the Backend to process the parameters sent
Backend - Data/Files Interaction	The Backend requests the storage layer (Data/Files) to authorize the connection or to prepare to receive the data. Data/Files is waiting for the data transfer.
Data/Files - Backend Response	Data/Files sends a response to the Backend
Backend - Frontend Response	The Backend gets the corresponding data and builds the model with the inputs defined previously by the user
Frontend - User Interaction	The Frontend shows the User the response from the application, obtaining a report for the model training and having the possibility to save it for its future use
Completion	The application is ready to receive new requests from the User

Table 100. Life-Cycle description Anomaly Detector Parametrization AI[AD]

4.9.2.8 Description – Perform Anomaly Detection:

Objects	Description
Start	The User initiates the action of performing the anomaly detection.
User – Frontend Interaction	The User parametrizes in which components wants to perform the anomaly detection, indicates the desired model to evaluate and defines where the new data comes from.
Frontend - Backend Interaction	The Frontend sends a request to the Backend to process the parameters sent.
Backend - Data/Files Interaction	The Backend requests the storage layer (Data/Files) to authorize the connection or to prepare to receive the data. Data/Files is waiting for the data transfer.
Data/Files - Backend Response	Data/Files sends a response to the Backend.
Backend - Frontend Response	The Backend gets the corresponding data and evaluates the model.

Frontend - User Interaction	The Frontend shows the User the response from the application, obtaining the anomaly detection report for the selected components
Completion	The application is ready to receive new requests from the User.

Table 101. Life-Cycle description Perform Anomaly Detection AI[AD]

4.9.3 Implementation Viewpoint

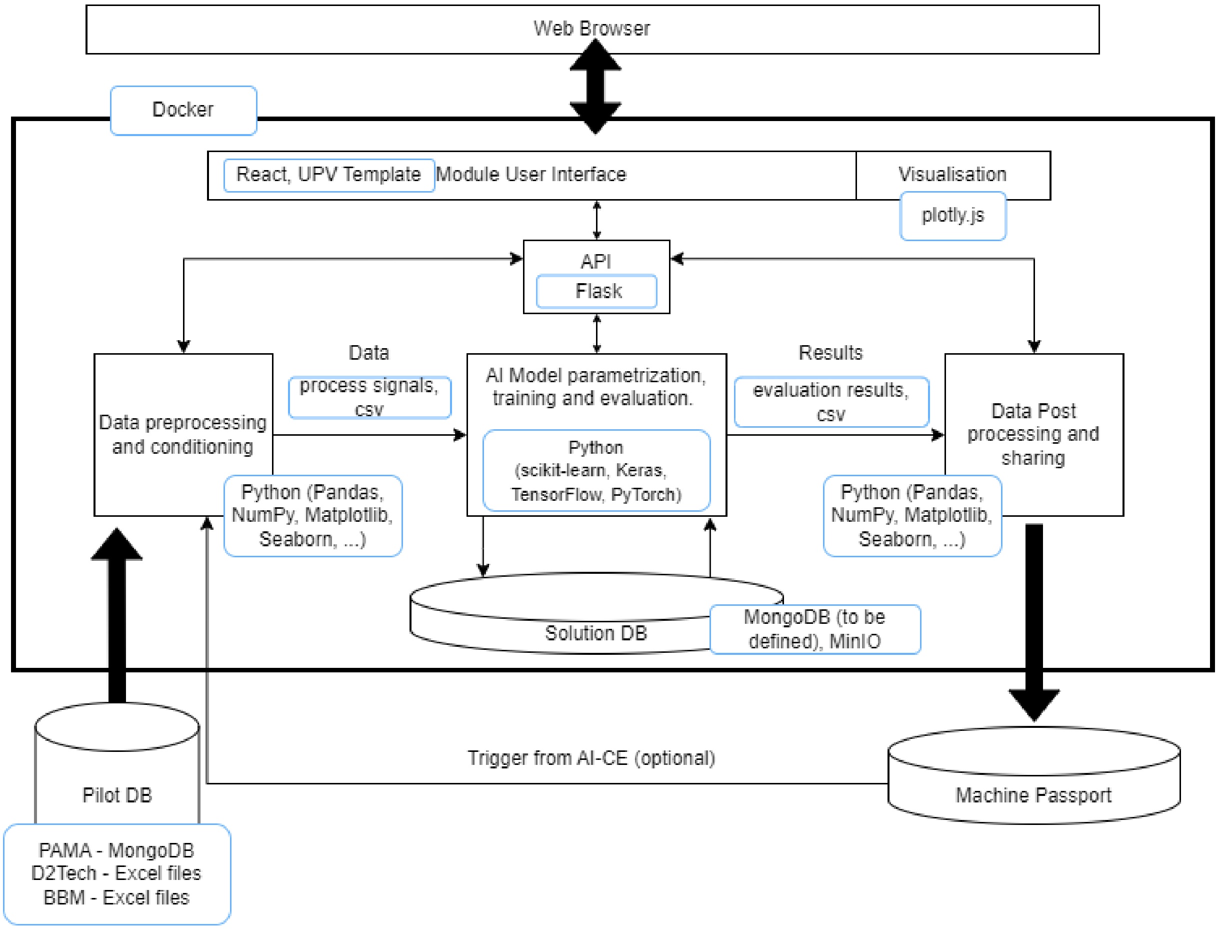


Figure 44. AI[AD] Implementation Architecture

For AI[AD], the main development language is Python, numpy and pandas are used for data management and basic operations, sklearn, keras and tensorflow are used for model training. The solution is containerized using Docker. A minor update following the solution architecture provided in M9 is the use of plotly.js library in place of the proposed options for data visualization purposes and the use of Flask as the framework for the API server. The use of an internal DB for the solution is still pending to be defined. At the moment, no internal database is used. In addition, input data will be accessed and collected from database (MongoDB) or files (excel files) provided by the pilots. Specific data preprocessing must be done on the pilots' data to get the expected input data. Finally, the results and solution's outputs will be sent to the Machine Passport.

4.9.3.1 AI^{AD} Implementation Components

Implementation Components	Description
Import Data	This component facilitates the reading of data from different data sources like databases, files etc.
Data Validation and Preprocessing	This component validates the training data and ensures that the input data is in the correct format before feeding it into the model.
Machine Configurator	This component allows the definition of the current machine configuration to identify its components and associated variables.
Create and Export Models	This component provides different algorithms, which train on the available pre-processed data to determine if there is an anomaly, or if the system is behaving as expected.
Obtain Predictions and Display Results	This component predicts if there is an anomaly or if the system is behaving as expected with new data using the trained models.
Export Data	This component sends the data to the Machine Passport and enables the user to export it directly if needed.

Table 102. AI^{AD} Implementation Components

4.9.3.2 Technical Description of AI^{AD} Components

Following the identification and specification of the implementation components in the previous section, the technical description of each component is provided in the following tables. It details the set of technologies required to implement each of the implementation components.

Implementation component	Import Data
Description of implementation component	This component facilitates the reading of data from different data sources like databases, files etc.
Used technologies	Python
Technical Description of the Component	**Dependencies** Development Language: Python Libraries: Pandas Container: Docker Database: to be defined **Interfaces** User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs

	Network/Protocols: HTTP/HTTPS Data Repository: MongoDB, minIO

Table 103. Technical Description of AI^{AD} "Import Data" Implementation Component

Implementation component	Data Validation and Preprocessing
Description of implementation component	This component validates the training data and ensures that the input data is in the correct format before feeding it into the model
Used technologies	Python
Technical Description of the Component	**Dependencies** Development Language: Python Libraries: Pandas, numpy Container: Docker Database: to be defined **Interfaces** User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: MongoDB, minIO **Requires** Import Data

Table 104. Technical Description of AI^{AD} "Data Validation and Preprocessing" Implementation Component

Implementation component	Machine Configurator
Description of implementation component	This component facilitates the definition of the machine that will be the target of the anomaly detection, defining its components and its relevant process variables.
Used technologies	Python
Technical Description of the Component	**Dependencies** Development Language: Python Libraries: Pandas, numpy Container: Docker Database: to be defined **Interfaces** User Interface: Yes, REACT

	Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: MongoDB, minIO
	Requires
	Import Data

Table 105. Technical Description of AI^{AD} "Machine Configurator" Implementation Component

Implementation component	Create and Export Models
Description of implementation component	This component facilitates the parametrization and creation of new machine learning models, which train in the previously pre-processed data given a specific machine configuration.
Used technologies	Python
Technical Description of the Component	Dependencies
	Development Language: Python Libraries: sklearn, keras, TensorFlow, pickle Container: Docker Database: to be defined
	Interfaces
	User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: MongoDB, minIO.
	Requires
	Import Data, Data Validation and Preprocessing, Machine Configurator

Table 106. Technical Description of AI^{AD} "Create and Export Models" Implementation Component

Implementation component	Obtain Predictions and Display Results
Description of implementation component	This component predicts if new data is considered an anomaly or not using previously trained models. Results are shown in different formats.
Used technologies	Python
	Dependencies
	Development Language: Python, Javascript Libraries: sklearn, keras, TensorFlow, pickle, plotly.js

Technical Description of the Component	Container: Docker Database: to be defined	
	Interfaces	
	User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: MongoDB, minIO	
	Requires	
	Import Data, Data Validation and Preprocessing, Machine Configurator, Create and Export Models	

Table 107. Technical Description of AI[AD] "Obtain Predictions and Display Results" Implementation Component

Implementation component	Export Data
Description of implementation component	This component facilitates exporting the obtained results to local storage or to the Machine Passport.
Used technologies	Python
Technical Description of the Component	Dependencies
	Development Language: Python Libraries: Pandas, numpy Container: Docker Database: MongoDB
	Interfaces
	User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: MongoDB, minIO
	Requires
	Machine Configurator, Obtain Predictions and Display Results

Table 108. Technical Description of AI[AD] "Export Data" Implementation Component

4.10. Adaptive Controller – AI^AC

4.10.1 Usage Viewpoint

The usage activity diagram (Figure 45) within AI^AC demonstrates the interaction between the different tasks and roles. No pilot distinction is made as these are the same independent of the use case. This diagram illustrates the dynamic flow of interactions, offering a representation of the activities executed by the solution and providing a clear understanding of the activities executed using the AI^AC solution.

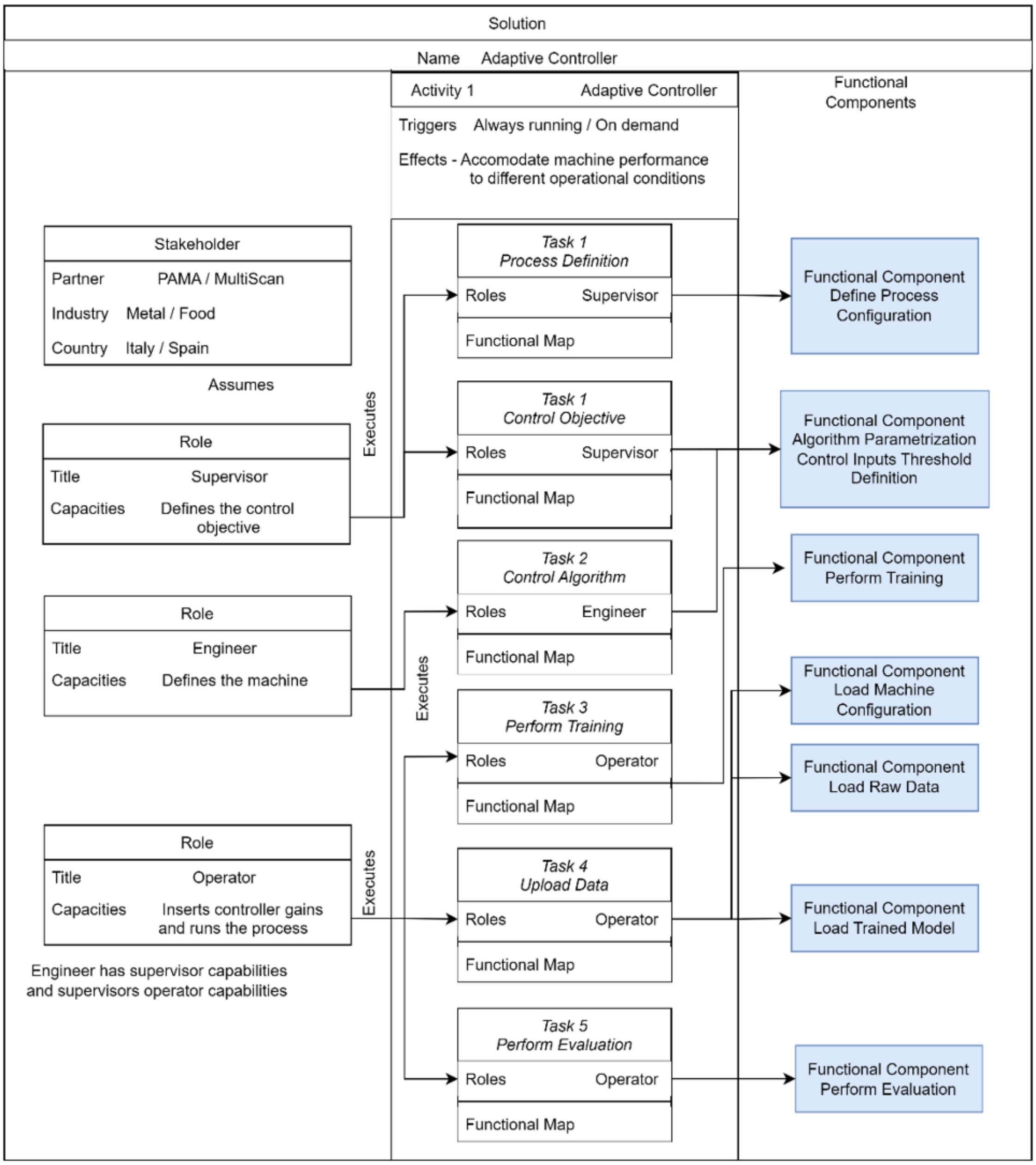


Figure 45. Adaptive Controller Usage Viewpoint Activity Diagram.

4.10.2 Functional Viewpoint

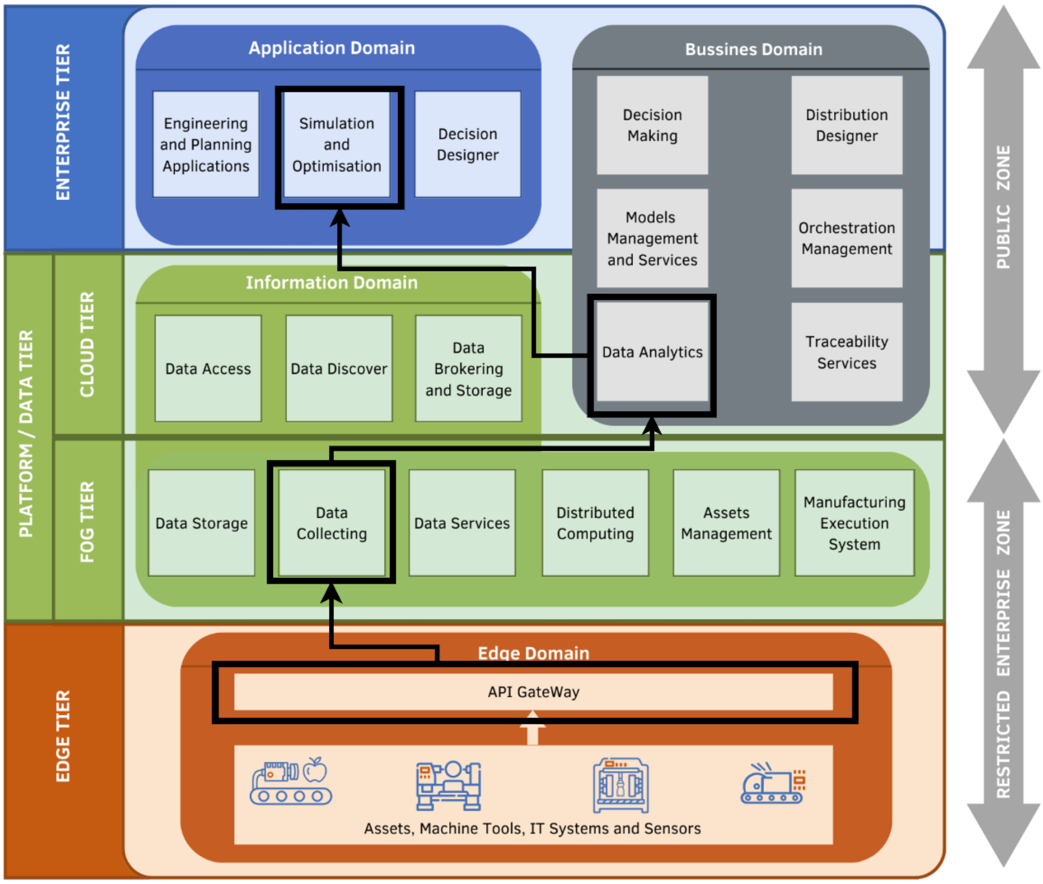


Figure 46. Data Flow AI^{AC}

Adaptive Controller gets machine operation data, gathered from PLCs and data acquisition systems in the edge, databases or .csv files through a data collection module to calculate the control inputs that should be sent to the machines. This will help operators to improve the machine efficiency by applying the calculated control input to optimize the process.

4.10.2.1 Data structure of AI^{AC}

Format	Input/Output	Example
Database	Input	MONGODB timestamp, vibration level. 2023-11-30T08:00:00Z,0.5
CSV	Input	timestamp, reference tracking error. 2023-11-30T08:00:00Z,0.5
JSON	Input	Can also be created from solution's frontend. [{

		"Name": "Temp.spin.front.bear", "Description": "Spindle Temperature", "MinVal": 0, "MaxVal": 100, "MeasureUnit": "ºC", "ParentGroup": "Spindle Temperature" }, { "Name": "X_Axis_Pos", "Description": "X Axis Position", "MinVal": 0, "MaxVal": 1000, "MeasureUnit": "mm", "ParentGroup": "X Axis" } …]

JSON	Output	Based on event
		
		{
		"Timestamp": "2023-11-30T08:00:00Z",
		"ControlDescription": "The tracking error has reached maximum allowed value of 10mm."
		}

JSON	Output	Based on time interval
		
		{
		"StartTime": "2023-11-30T08:00:00Z",
		"EndTime": "2023-11-30T10:00:00Z",
		"Control Inputs":
		[
		{
		"Timestamp": "2023-11-30T08:00:00Z",
		"ControlDescription": "The motor velocity or torque should be modified."
		} , …
		]
		}

Table 109. Input / Output Data Format AI^{AC}

4.10.2.2 AI^AC Software Requirements

Software Component	Description/Role	Required Version/Configuration	Dependencies
Linux OS or Windows OS	Operating system needed to use the tool	Ubuntu 22.04 / Windows 10 Pro	N/A
Docker	Build, share and run containers	latest	N/A

Table 110. Software requirements AI^AC

4.10.2.3 AI^AC Lifecycle

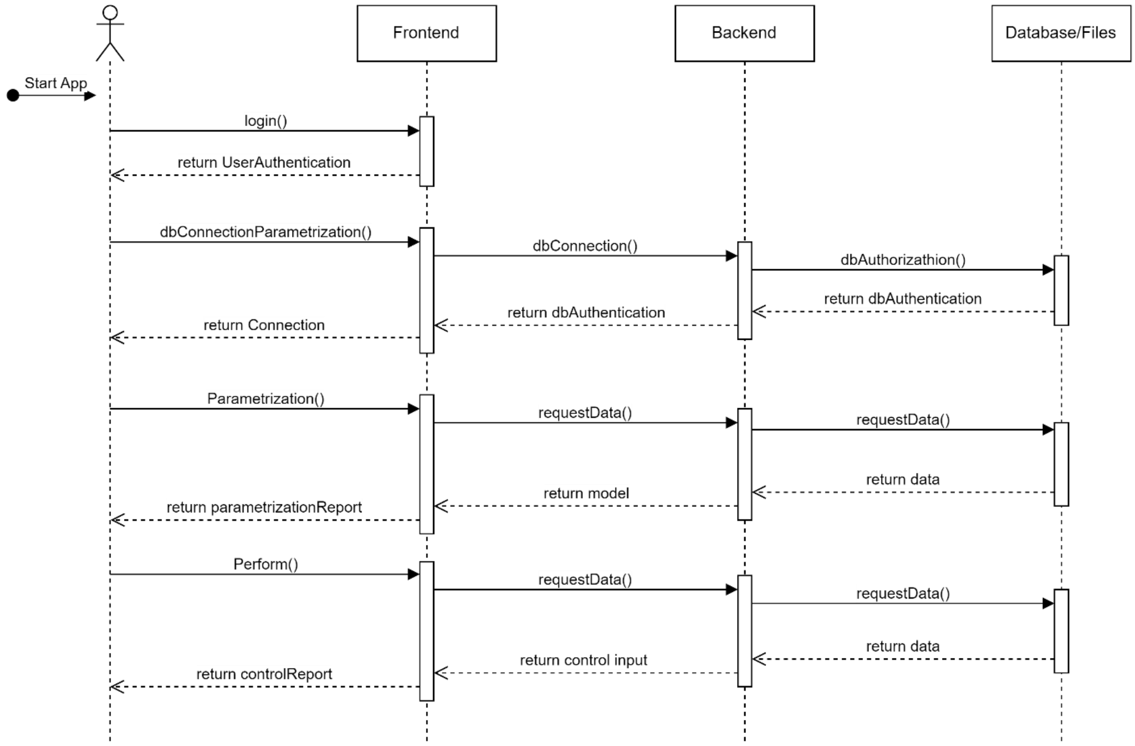


Figure 47. AIDEAS Adaptive Controller Sequence Diagram

This sequence diagram illustrates the flow of interactions between the User, Frontend, Backend and Data/Files objects during the main processes. Login into the AIDEAS platform, parametrize a connection to an external database, parametrize the anomaly detection and perform it.

4.10.2.4 Objects

- **User**: Represents the user interacting with the application.
- **Frontend**: Represents the user interface and the presentation layer.
- **Backend**: Represents the application logic and the server layer.
- **Database/Files:** Represents the data or file storage layer.

4.10.2.5 Description - Login

Objects	Description
Start	The User logs into the **AIDEAS** platform.
User – Frontend Interaction	The User enters username and password.
Frontend - User Interaction	The Frontend shows the User the response from the login action, indicating whether it is logged in or not.
Completion	The application is ready to receive new requests from the User.

Table 111. Life-Cycle description Login AI[AC]

4.10.2.6 Description - Database Connection / File Upload

Objects	Description
Start	The User initiates the action of parametrizing a database connection or uploads data files.
User – Frontend Interaction	The User parametrizes the database connection or uploads a data file.
Frontend - Backend Interaction	The Frontend sends a request to the Backend to process a connection to the database or to process the file upload.
Backend - Data/Files Interaction	The Backend requests the storage layer (Data/Files) to authorize the connection or to prepare to receive the file. Data/Files is waiting for the authentication or for the file transfer
Data/Files - Backend Response	Data/Files confirms that the connection has been completed or the file has been stored correctly and sends a response to the Backend.
Backend - Frontend Response	The Backend informs the Frontend about the success or failure of the connection or file upload operation.
Frontend - User Interaction	The Frontend shows the User the response from the application, indicating whether the connection or file upload was successful.
Completion	The application is ready to receive new requests from the User.

Table 112. Life-Cycle description Database Connection AI[AC]

4.10.2.7 Description – Adaptive Controller Parametrization

Objects	Description
Start	The User initiates the action of parametrizing de model and the controller structure.

User – Frontend Interaction	The User selects which kind of model wants to build and with which data, and the controller structure.
Frontend - Backend Interaction	The Frontend sends a request to the Backend to process the parameters sent.
Backend - Data/Files Interaction	The Backend requests the storage layer (Data/Files) to authorize the connection or to prepare to receive the data. Data/Files is waiting for the data transfer.
Data/Files - Backend Response	Data/Files sends a response to the Backend.
Backend - Frontend Response	The Backend gets the corresponding data and builds the model and the controller structure with the inputs defined previously by the user.
Frontend - User Interaction	The Frontend shows the User the response from the application, obtaining a report for the model training with the controller selected and having the possibility to save it for its future use
Completion	The application is ready to receive new requests from the User

Table 113. Life-Cycle description Adaptive Controller Parametrization AI[AC]

4.10.2.8 Description – Perform Control:

Objects	Description
Start	The User initiates the machine control.
User – Frontend Interaction	The User starts the tuning of the controller and indicates which variables need to be controller, indicates the desired model to evaluate the controller, and the performance variable.
Frontend - Backend Interaction	The Frontend sends a request to the Backend to process the parameters sent.
Backend - Data/Files Interaction	The Backend requests the storage layer (Data/Files) to authorize the connection or to prepare to receive the data. Data/Files is waiting for the data transfer.
Data/Files - Backend Response	Data/Files sends a response to the Backend.
Backend - Frontend Response	The Backend gets the corresponding data and evaluates the controller process.
Frontend - User Interaction	The Frontend shows the User the response from the application, obtaining the performance indicator for the selected variables.
Completion	The application is ready to receive new requests from the User.

Table 114. Life-Cycle description Perform Control AI[AC]

4.10.3 Implementation Viewpoint

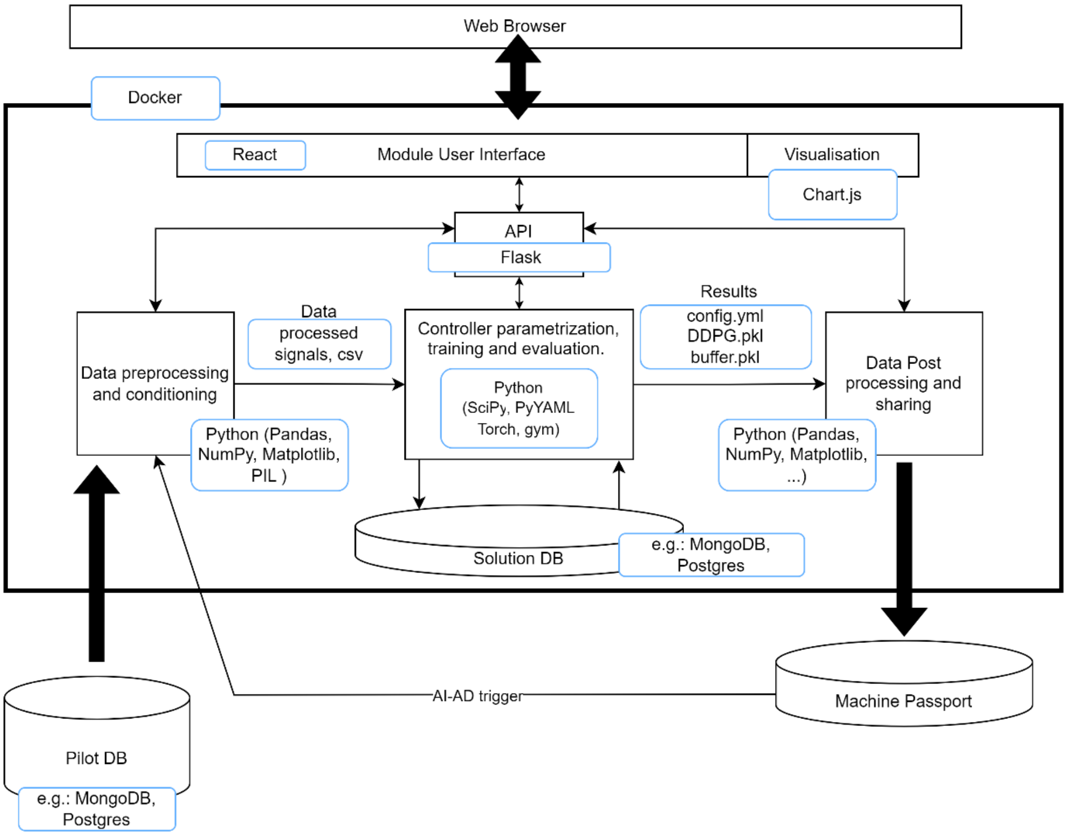


Figure 48. AI^AC Implementation Architecture

For AI^AC, the main development language is Python, NumPy and pandas are used for data management and basic operations, sklearn, keras and TensorFlow are used for model training and the controller design. The solution is containerized using Docker. A minor update following the solution architecture provided in M9 is the use of PIL library to be able to read images provided by MULTISCAN pilot, to carry out image recognition. The use of an internal DB for the solution is still pending to be defined. Now, no internal database is used. In addition, input data will be accessed and collected from database (MongoDB) or files (excel files) provided by the pilots. Specific data preprocessing must be done on the pilots' data to get the expected input data. Finally, the results and solution's outputs will be sent to the Machine Passport.

4.10.3.1 AI^AC Implementation Components

Implementation Components	Description
Import Data	This component facilitates the reading of data from different data sources like databases, files etc.
Data Validation and Preprocessing	This component validates the training data and ensures that the input data is in the correct format before feeding it into the model.
Machine Configurator	This component allows the definition of the current machine configuration to identify its components and its associated variables.

Create and Export Models	This component provides different algorithms, which train on the available pre-processed data to obtain an accurate relationship of the actual input/outputs.
Create and Export Controllers	This component provides an adaptive controller algorithm, which trains on user-defined models, controls the system and optimizes it.
Obtain Controller Evaluation	This component analyses and evaluates the controller performance.
Export Data	This component sends the data to the Machine Passport and enables the user to export it directly if needed.

Table 115. AI^{AC} Implementation Components

4.10.3.2 Technical Description of AI^{AC} Components

Following the identification and specification of the implementation components in the previous section, the technical description of each component is provided in the following tables. It details the set of technologies required to implement each of the implementation components.

Implementation component	Import Data
Description of implementation component	This component facilitates the reading of data from different data sources like databases, files etc.
Used technologies	Python
Technical Description of the Component	**Dependencies** Development Language: Python Libraries: Pandas, PIL Container: Docker Database: to be defined **Interfaces** User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: MongoDB, minIO

Table 116. Technical Description of AI^{AC} "Import Data" Implementation Component

Implementation component	Data Validation and Preprocessing
Description of implementation component	This component validates the training data and ensures that the input data is in the correct format before feeding it into the model.
Used technologies	Python

Technical Description of the Component	**Dependencies**
	Development Language: Python Libraries: Pandas, numpy Container: Docker Database: to be defined
	Interfaces
	User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: MongoDB, minIO
	Requires
	Import Data

Table 117. Technical Description of AI^{AC} “Data Validation and Preprocessing” Implementation Component

Implementation component	Machine Configurator
Description of implementation component	This component facilitates the definition of the machine and the control objective, which will be the target of the control design, defining its components and its relevant process variables.
Used technologies	Python
Technical Description of the Component	**Dependencies**
	Development Language: Python Libraries: Pandas, numpy Container: Docker Database: to be defined
	Interfaces
	User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: MongoDB, minIO
	Requires
	Import Data

Table 118. Technical Description of AI^{AC} “Machine Configurator” Implementation Component

Implementation component	Create and Export Models
Description of implementation component	This component provides different algorithms, which train on the available pre-processed data to obtain an accurate relationship of the actual input/outputs.
Used technologies	Python
Technical Description of the Component	**Dependencies** Development Language: Python Libraries: sklearn, keras, tensorflow, pickle, scipy Container: Docker Database: to be defined **Interfaces** User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: MongoDB, minIO **Requires** Import Data, Data Validation and Preprocessing, Machine Configurator

Table 119. Technical Description of AI[AC] “Create and Export Models” Implementation Component

Implementation component	Create and Export Controllers
Description of implementation component	This component provides an adaptive controller algorithm, which trains on user-defined models, controls the system and optimizes it.
Used technologies	Python
Technical Description of the Component	**Dependencies** Development Language: Python Libraries: sklearn, keras, tensorflow, pickle, scipy Container: Docker Database: to be defined **Interfaces** User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS

	Data Repository: MongoDB, minIO	
	Requires	
	Import Data, Data Validation and Preprocessing, Machine Configurator	

Table 120. Technical Description of AI[AC] "Create and Export Controllers" Implementation Component

Implementation component	Obtain Controller Evaluation
Description of implementation component	This component analyses and evaluates the controller performance.
Used technologies	Python
Technical Description of the Component	Dependencies
	Development Language: Python, Javascript Libraries: sklearn, keras, tensorflow, pickle, plotly.js Container: Docker Database: to be defined
	Interfaces
	User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: MongoDB, minIO
	Requires
	Import Data, Data Validation and Preprocessing, Machine Configurator, Create and Export Models

Table 121. Technical Description of AI[AC] "Obtain Controller Evaluation" Implementation Component

Implementation component	Export Data
Description of implementation component	This component facilitates exporting the obtained results to a local storage or to the Machine Passport.
Used technologies	Python
	Dependencies
	Development Language: Python Libraries: Pandas, NumPy Container: Docker Database: MongoDB
	Interfaces

Technical Description of the Component	User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: MongoDB, minIO
	Requires
	Machine Configurator, Obtain Predictions and Display Results

Table 122. Technical Description of AI^{AC} "Export Data" Implementation Component

4.11. Quality Assurance – AI^{QA}

4.11.1 Usage Viewpoint

The usage activity diagram (Figure 49) within AI^{QA} depicts the interaction between the tasks and roles, currently targeting one pilot. Nevertheless, similar approaches will be reused, as these are the same independent of the use case. This diagram illustrates the dynamic flow of interactions, offering a representation of the activities executed by the solution providing a clear understanding of the activities executed using the AI^{QA} solution.

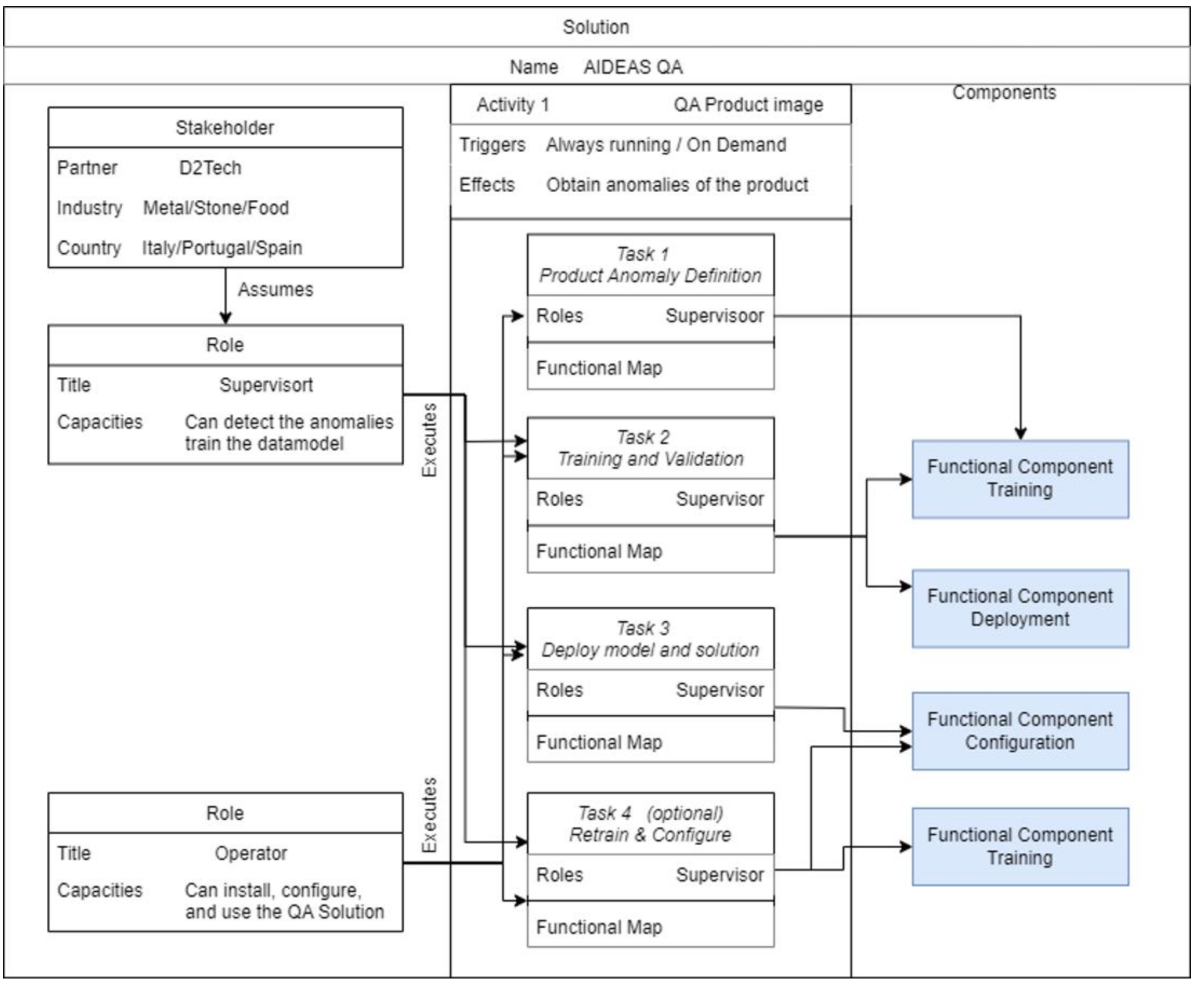


Figure 49. AI^{QA} solution Usage Viewpoint Activity Diagram

4.11.2 Functional Viewpoint

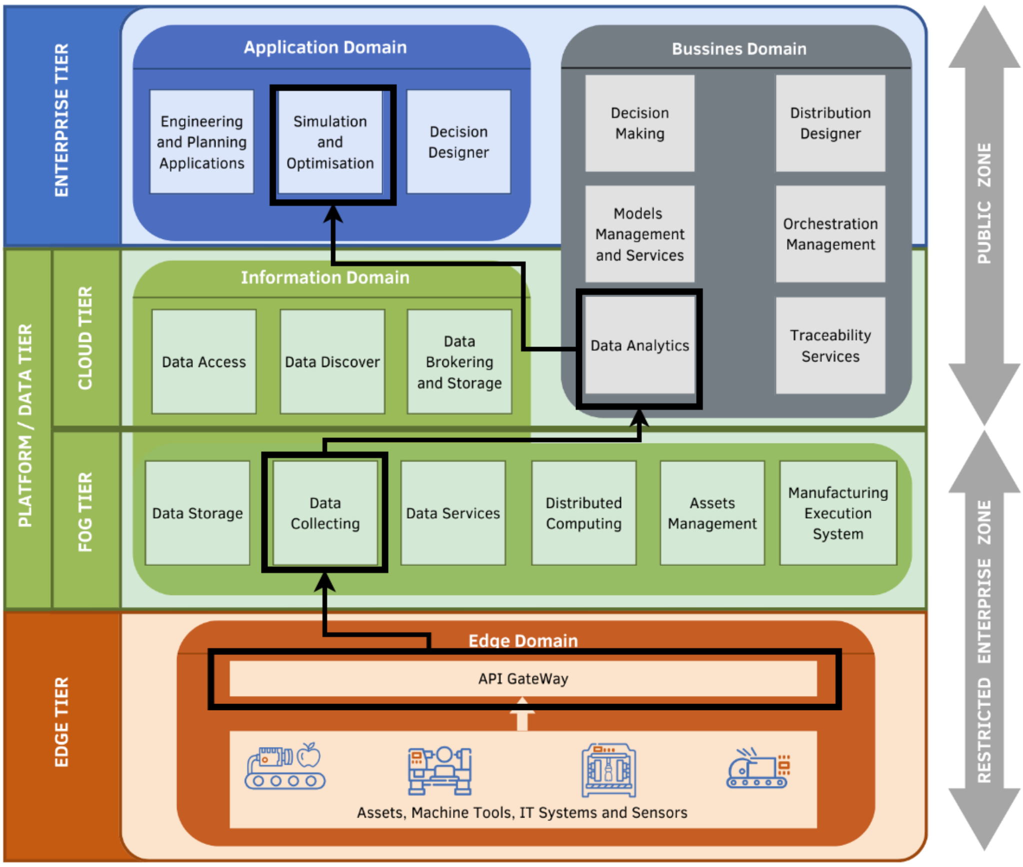


Figure 50. Data Flow AI^QA

The solution uses an anomaly detection framework to learn how to classify a "defect-free" image of the product from a defective product to generate a model. This allows it to identify typical characteristics of defect-free product images. The model is then used to identify defects on a random product image by creating an output heatmap of the image produced and a defect evaluation rank number. Higher the number the more likely defects are present in the image.

4.11.2.1 Data structure of AI^QA

Format	Input/Output	Example
PNG image	input	An image of the product (cut stone image)
PNG image	output	An image of the product with a heatmap overlay (transparent->red) depicting the probability of the defect detected in the image.
dset_number	output	A number ranking – the higher the result, the more likely there are defects detected in the image

Table 123. Input / Output Data Format AI^QA

4.11.2.2 AI^{QA} Software Requirements

Software Component	Description/Role	Required Version/Configuration
Linux OS or Windows OS	Operating system needed to use the tool	Ubuntu 22.04 / Windows 10 Pro
Docker	Build, share and run containers	latest

Table 124. Software requirements AI^{QA}

4.11.2.3 AI^{QA} Lifecycle

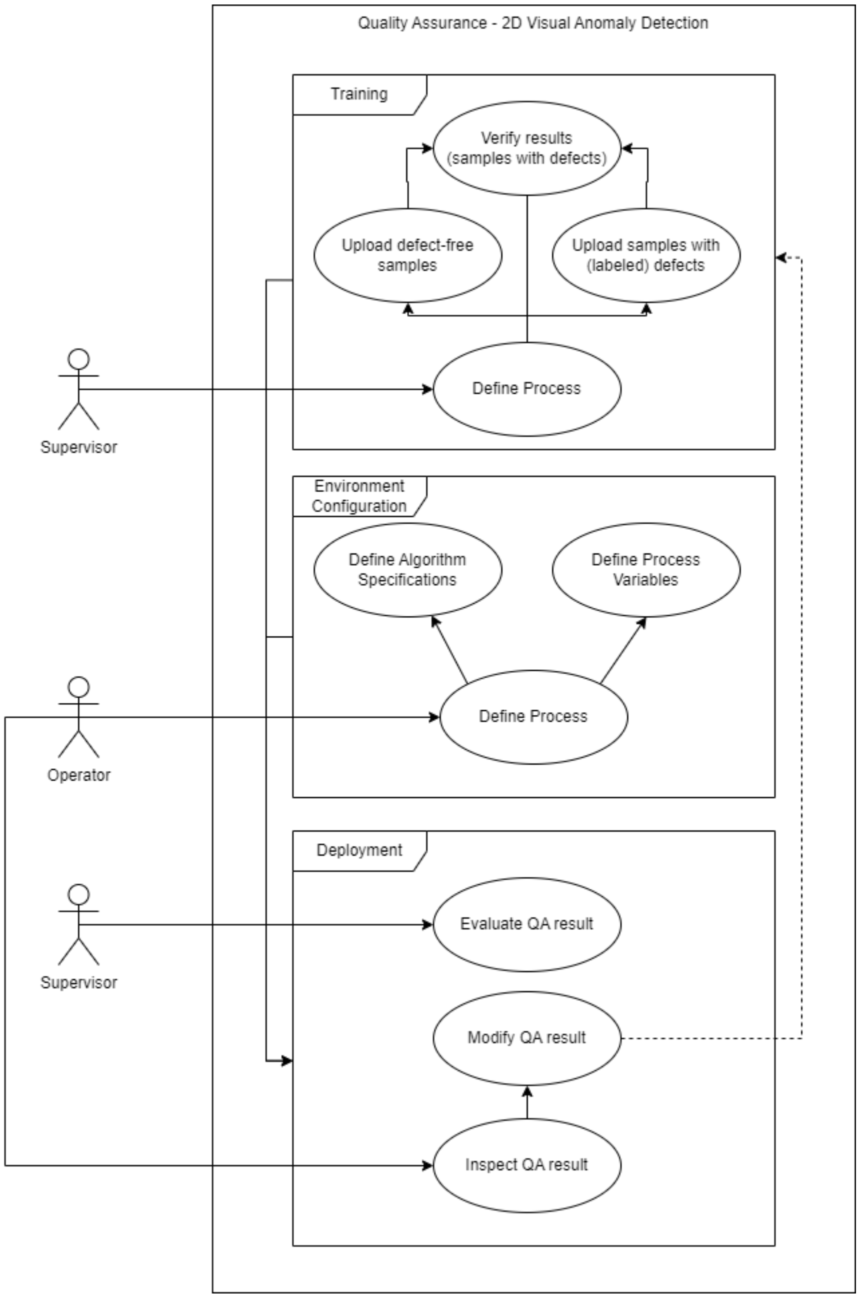


Figure 51. AI^{QA} Lifecycle

- **Training**: The user defines the process (machine and material characteristics) and uploads a set of images of objects labelled as defect-free or/and with defects detected and annotated. The AI[QA] -2D method is trained on these samples. The training is usually performed and verified by the supervisor.
- **Environment configuration:** The operator configures the application with machine characteristics.
- **Deployment:** AI[QA] -2D solution is deployed to (usually) on-premises edge/machine environment and applied to 2D images to perform a visual inspection (QA). The operator might inspect the results (correct classification of incoming objects), the operator might also modify the results and provide feedback to the AI[QA] -2D module to facilitate the feedback loop which is optional.

4.11.2.4 Objects:

- **Supervisor**: Represents the User interacting with the AI[QA] solution while generating/updating the data model.
- **Operator**: Represents the user interfacing the edge machine in production.
- **Training**: Represents the process of training from collected defect annotated and defect-free images of products for building the decision model.
- **Configuration**: Represents the setup and configuration of AI[QA] solution on the edge device (customer/pilot machine).
- **Deployment**: Represents the user interaction with the AI[QA] solution deployed

4.11.2.5 Description: Training and Deployment

Objects	Description
Training – Define process	The Supervisor initiates the collection and classification of images (defect-free and defect annotated) for initiating training.
Training – Verify results	The Supervisor verifies the results of data-model evaluation and if the model tests are approved the solution is deployed on the edge machine for usage
Configuration – setup	The Operator sets up configures the deployed QA solution and starts using the QA solution with production images on site
Deployment – QA verification	The Supervisor verifies the results and overall execution of the deployed QA solution – if needed starts a new Training classification based on changed parameters/use cases.

Table 125. Life-Cycle description Training and Deployment AI[QA]

4.11.3 Implementation Viewpoint

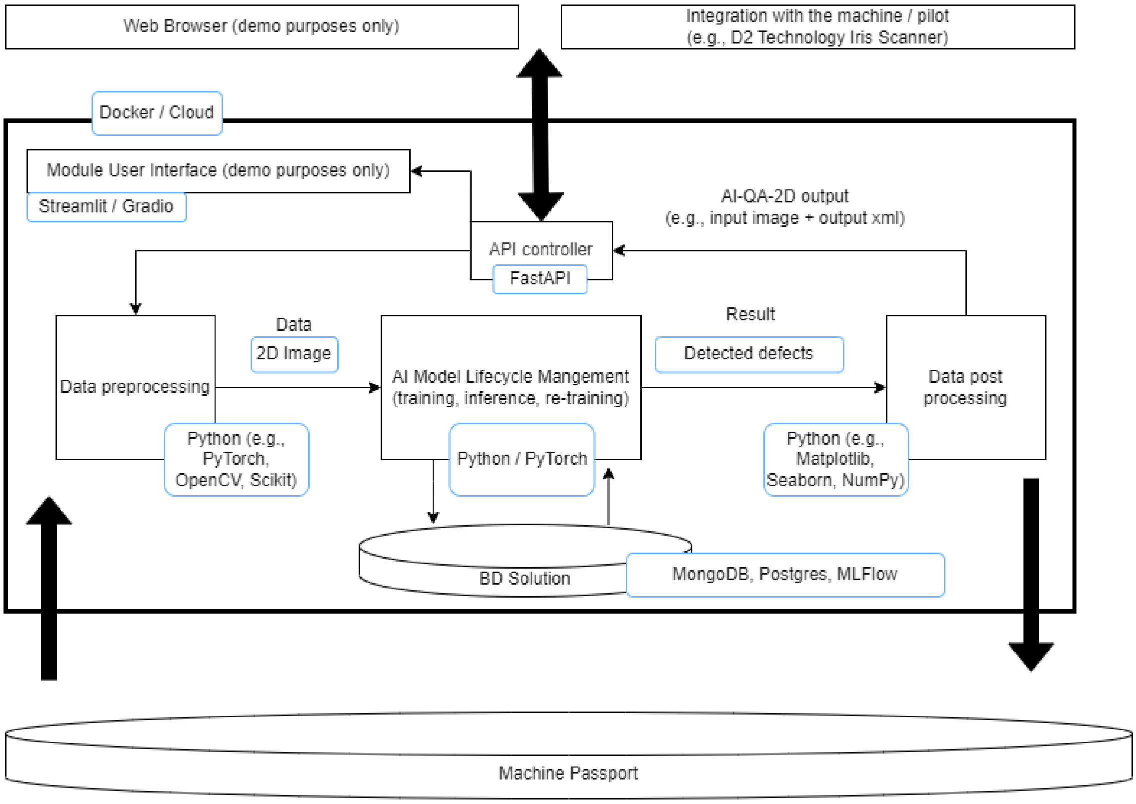


Figure 52. AI^{QA} 2D Implementation Architecture

AI^{QA} components are developed in Python, using pandas for data management and basic operations while anomalib (PyTorch) is used for model training. The component is containerized using Docker. The matplotlib library is used for creating a heat map of the detected defects in 2D product images and the use of FastAPI as the framework for the API server. The use of an internal DB for the solution for storing metadata is planned, for the moment no internal database is used, but probably MongoDB. Images are currently stored in a mounted volume, but minIO API could also be used to access the images in the cloud since specific data preprocessing must be done for adequate pilots' images to get the expected dataset and train the model. Finally, the results and component's outputs will be sent to the Machine Passport.

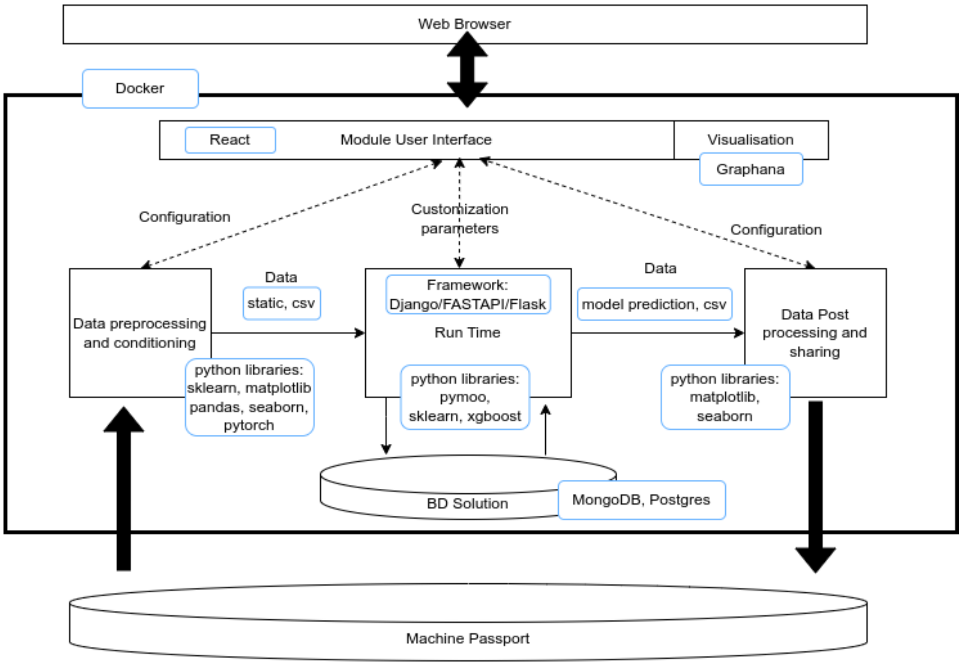


Figure 53. AI^{QA} 3D Implementation Architecture

4.11.1 AI^{QA} Implementation Components

Solution	Implementation Components	Description
AI^{QA}	2DDefectDetection	The tool can detect and rank the possible anomaly based on a 2D image of the product.
AI^{QA}	3DDefectDetection	The tool can detect, localize, and classify 3D captures of products.

Table 126. AI^{QA} Implementation Components

4.11.2 Technical Description of AI^{QA} Components

Following the identification and specification of the implementation components in the previous section, the technical description of each component is provided in the following tables. It details the set of technologies required to implement each of the implementation components.

Implementation component	2DDefectDetection
Description of implementation component	The tool can detect and rank the possible anomaly based on a 2D image of the product.
Used technologies	The tool can detect and rank the possible anomaly based on a 2D
	Dependencies
	Development Language: Python Libraries: anomalib, torch, open-clip-torch, pandas, imgaug, opencv-python, pandas, matplotlib

Technical Description of the Component	Container: Docker Database: MongoDB	
	Interfaces	
	User Interface: REST API Synchronous/Asynchronous Interface: REST API Async Network/Protocols: HTTP/HTTPS Data Repository: MongoDB, datalakes	
	Requires	
	Machine passport	

Table 127. 2D Defect Detection Component

Implementation component	3DDefectDetection
Description of implementation component	The tool can detect and rank the possible anomaly based on a 3D image set of the product.
Used technologies	Python
Technical Description of the Component	Dependencies
	Development Language: Python Libraries: sklearn, matplotlib, pandas, seaborn, pytorch Container: Docker Database: SQLite
	Interfaces
	User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: MongoDB, minIO
	Requires
	Machine passport

Table 128. 3D Defect Detection Component

4.12. Prescriptive Maintenance – AI^PM

4.12.1 Usage Viewpoint

The usage activity diagram (Figure 54) within AI^PM demonstrates the interaction between the different tasks and roles. No pilot distinction is made as these are the same independent of the use case. This diagram illustrates the dynamic flow of interactions, offering a representation of the activities executed by the solution providing a clear understanding of the activities executed using the AI^PM solution.

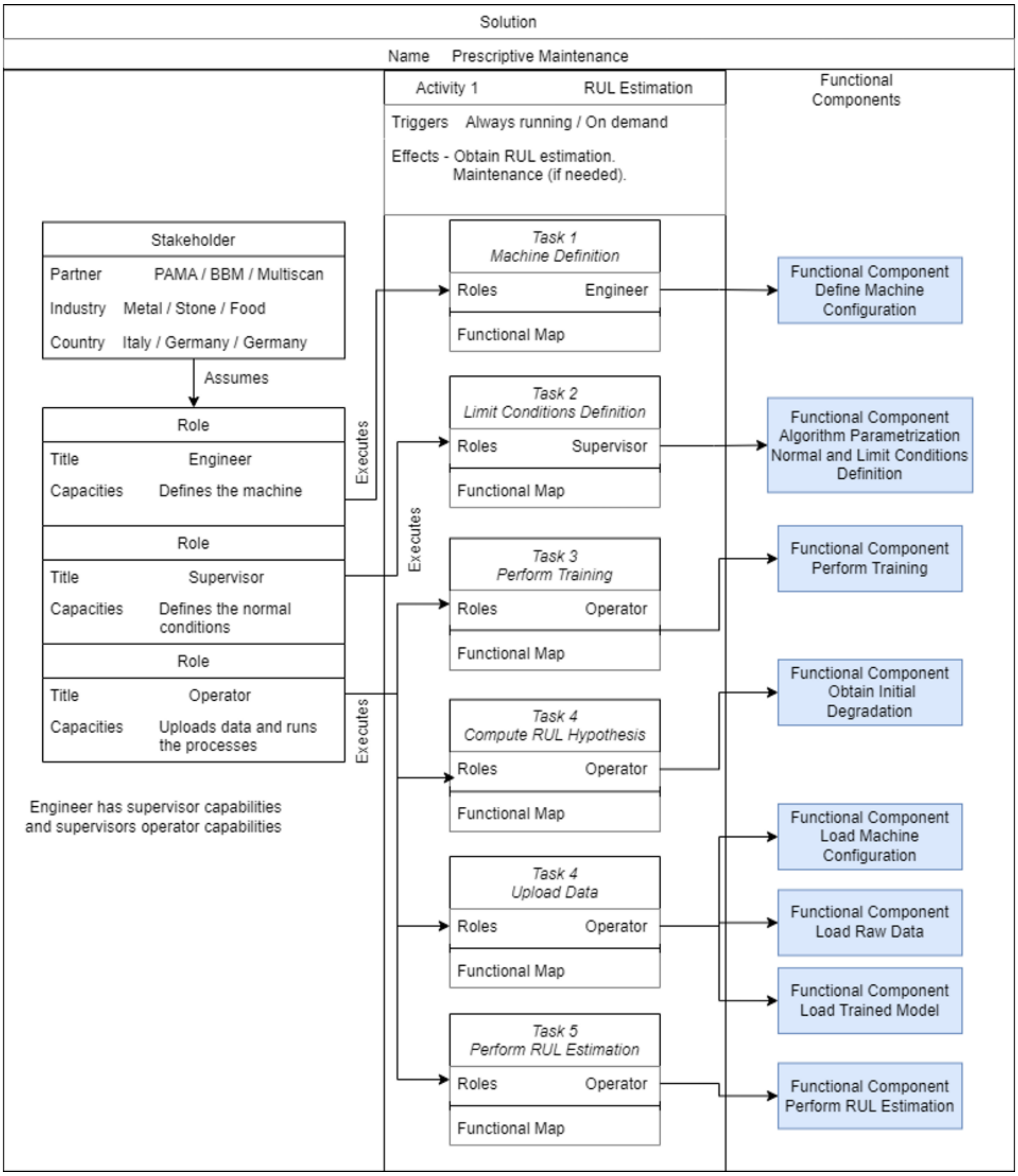


Figure 54. Estimation Usage Viewpoint Activity Diagram (Prescriptive Maintenance)

4.12.2 Functional Viewpoint

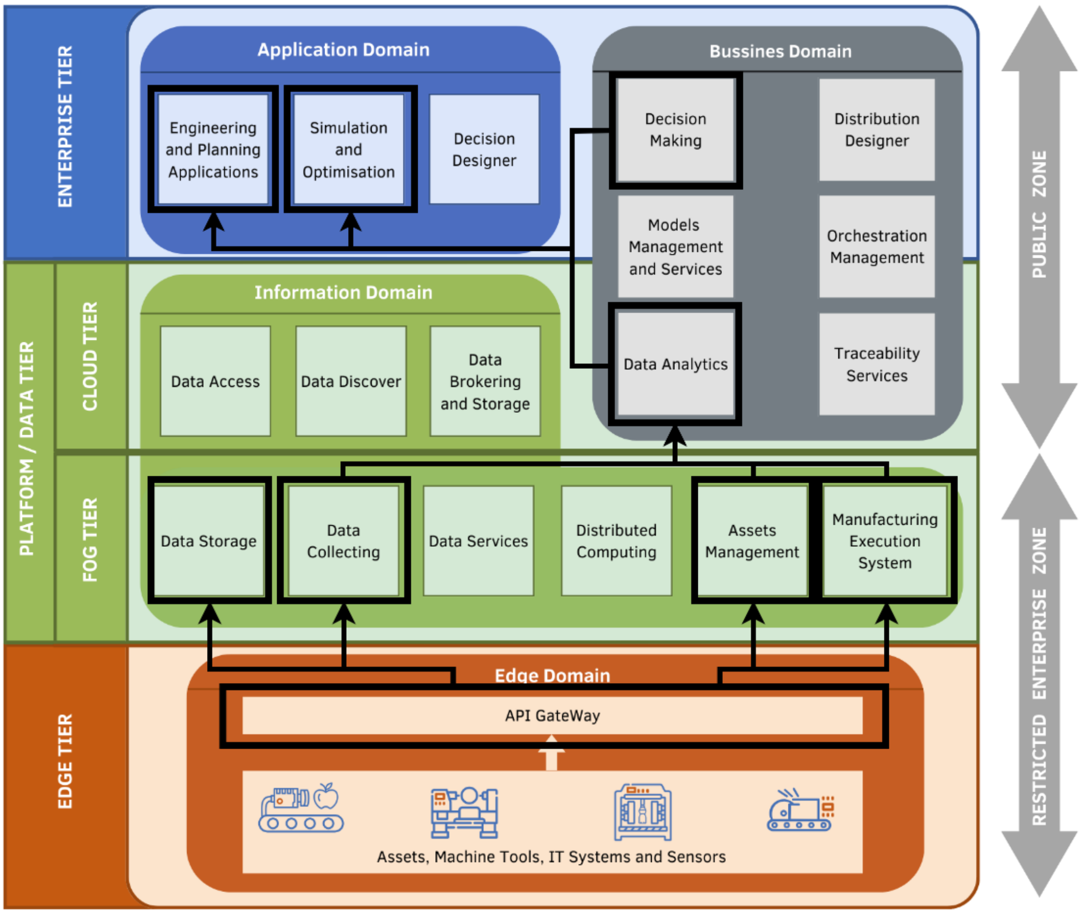


Figure 55. Data Flow AI^{PM}

Prescriptive Maintenance gets machine operation data, gather from PLCs and data acquisition systems in the edge, databases or .csv files through a data collection module and also maintenance records and operational context information to assess the failure risk with an uncertainty threshold or to assess the damage estimating the remaining useful life (RUL) of the machine, or any of its components, while its working in the factory where it is being used and recommending operators and maintenance managers to perform corresponding actions.

4.12.2.1 Data structure of AI^{PM}

Format	Input/Output	Example
Database: MONGODB	Input	timestamp, vibration level. 2023-11-30T08:00:00Z,0.5
CSV	Input	timestamp, vibration level. 2023-11-30T08:00:00Z,0.5
JSON	Input	Can also be created from solution's frontend. [

		{ "Name": "X_Axis_Pos", "Description": "X Axis Position", "MinVal": 0, "MaxVal": 1000, "MeasureUnit": "mm", "ParentGroup": "X Axis" }, { "Name": "X_Axis_Vel", "Description": "X Axis Speed", "MinVal": 0, "MaxVal": 50, "MeasureUnit": "mm/s", "ParentGroup": "X Axis" }]

JSON	Output	Based on time interval, once a day or defined in solution's frontend
		
		{
		"StartTime": "2023-11-30T08:00:00Z",
		"EndTime": "2023-11-30T10:00:00Z",
		"MachineRUL": 450
		"ComponentsRUL":
		[
		{
		"Component": "Component1",
		"RUL": 330
		}
		]
		}

Table 129. Input / Output Data Format AIPM

4.12.2.2 AIPM Software Requirements

Software Component	Description/Role	Required Version/Configuration	Dependencies
Linux OS or Windows OS	Operating system needed to use the tool	Ubuntu 22.04 / Windows 10 Pro	N/A
Docker	Build, share and run containers	latest	N/A

Table 130. Software requirements AIPM

4.12.3 AI^{PM} Lifecycle

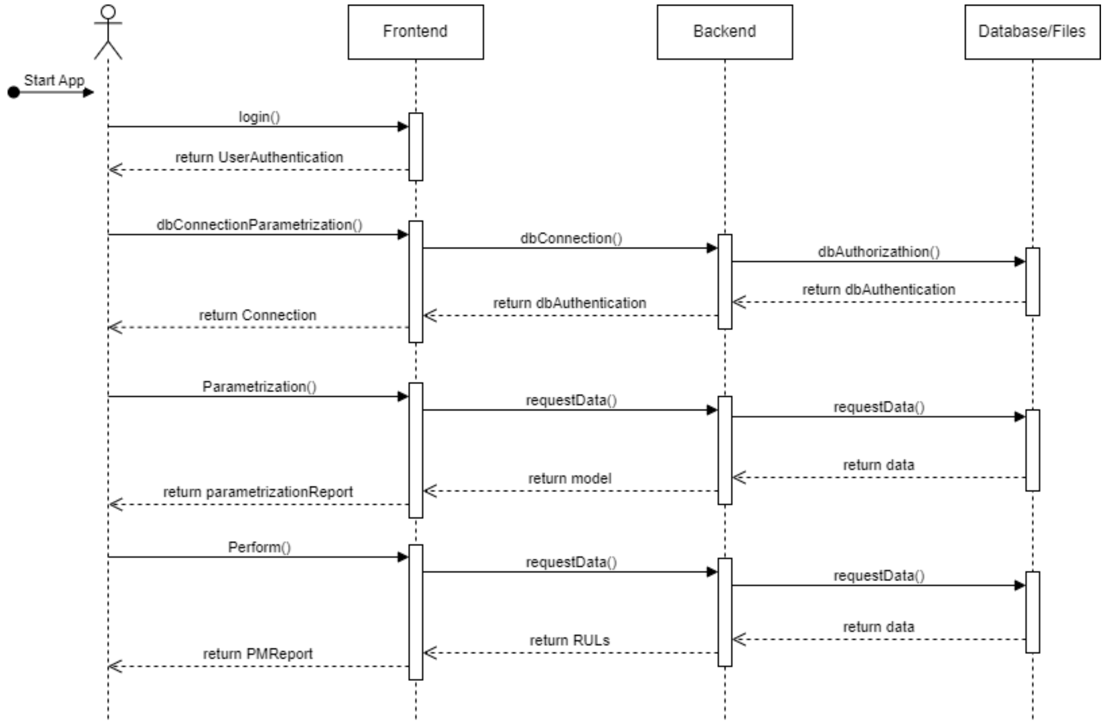


Figure 56. AIDEAS Prescriptive Maintenance Sequence Diagram

This sequence diagram illustrates the flow of interactions between the User, Frontend, Backend and Data/Files objects during the main processes. Login into the AIDEAS platform, parametrize a connection to an external database, parametrize the anomaly detection and perform it.

4.12.3.1 Objects

- **User**: Represents the User interacting with the application.
- **Frontend**: Represents the user interface and the presentation layer.
- **Backend**: Represents the application logic and the server layer.
- **Database/Files:** Represents the data or file storage layer.

4.12.3.2 Description - Login:

Objects	Description
Start	The User logs into the AIDEAS platform.
User – Frontend Interaction	The User enters user and password.
Frontend - User Interaction	The Frontend shows the User the response from the login action, indicating whether it is logged in or not.
Completion	The application is ready to receive new requests from the User.

Table 131. Life-Cycle description Login AI^{PM}

4.12.3.3 Description - Database Connection / File Upload:

Objects	Description
Start	The User initiates the action of parametrizing a database connection or uploads data files.
User – Frontend Interaction	The User parametrizes the database connection or uploads a data file.
Frontend - Backend Interaction	The Frontend sends a request to the Backend to process a connection to the database or to process the file upload.
Backend - Data/Files Interaction	The Backend requests the storage layer (Data/Files) to authorize the connection or to prepare to receive the file. Data/Files is waiting for the authentication or for the file transfer
Data/Files - Backend Response	Data/Files confirms that the connection has been completed or the file has been stored correctly and sends a response to the Backend.
Backend - Frontend Response	The Backend informs the Frontend about the success or failure of the connection or file upload operation.
Frontend - User Interaction	The Frontend shows the User the response from the application, indicating whether the connection or file upload was successful.
Completion	The application is ready to receive new requests from the User.

Table 132. Life-Cycle description Database Connection AI[PM]

4.12.3.4 Description – Prescriptive Maintenance Parametrization

Objects	Description
Start	The user initiates the action of parametrizing the prescriptive maintenance.
User – Frontend Interaction	The user selects the kind of model to be built as well as the data to be utilized.
Frontend - Backend Interaction	The Frontend sends a request to the Backend to process the parameters sent
Backend - Data/Files Interaction	The Backend requests the storage layer (Data/Files) to authorize the connection or to prepare to receive the data. Data/Files is waiting for the data transfer.
Data/Files - Backend Response	Data/Files sends a response to the Backend

Backend - Frontend Response	The Backend gets the corresponding data and builds the model with the inputs defined previously by the user.
Frontend - User Interaction	The Frontend shows the user the response from the application, obtaining a report for the model training and having the possibility to save it for its future use.
Completion	The application is ready to receive new requests from the User.

Table 133. Life-Cycle description Prescriptive Maintenance Parametrization AI[PM]

4.12.3.5 Description – Perform Prescriptive Maintenance:

Objects	Description
Start	The User initiates the action of performing the prescriptive maintenance
User – Frontend Interaction	The User parametrizes in the components to perform the prescriptive maintenance, indicates the desired model to evaluate and defines where the new data comes from.
Frontend - Backend Interaction	The Frontend sends a request to the Backend to process the parameters sent.
Backend - Data/Files Interaction	The Backend requests the storage layer (Data/Files) to authorize the connection or to prepare to receive the data. Data/Files is waiting for the data transfer.
Data/Files - Backend Response	Data/Files sends a response to the Backend.
Backend - Frontend Response	The Backend gets the corresponding data and evaluates the model.
Frontend - User Interaction	The Frontend shows the user the response from the application, obtaining the prescriptive maintenance report for the selected components
Completion	The application is ready to receive new requests from the User.

Table 134. Life-Cycle description Perform Prescriptive Maintenance AI[PM]

4.12.4 Implementation Viewpoint

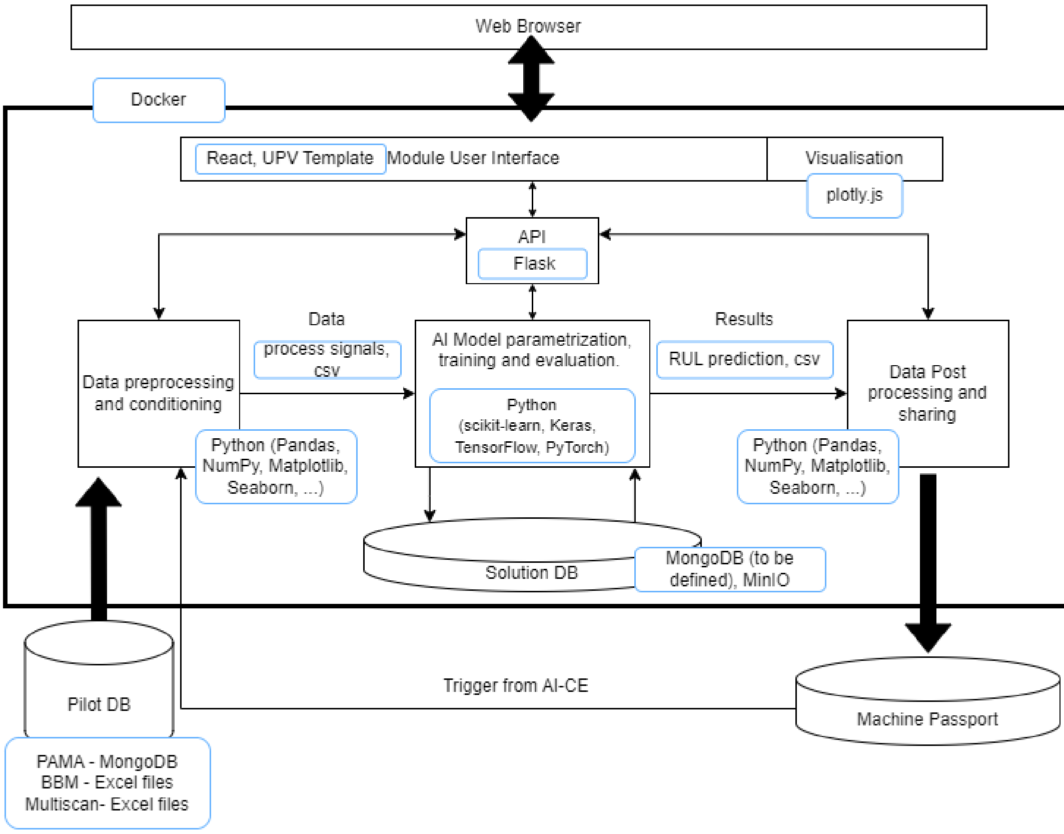


Figure 57. AI^{PM} Implementation Architecture

For AI^{PM}, the main development language is Python, NumPy and pandas are used for data management and basic operations, sklearn, keras and TensorFlow are used for model training. The solution is containerized using Docker. A minor update following the solution architecture provided in M9 is the use of plotly.js library in place of the proposed options for data visualization purposes and the use of Flask as the framework for the API server. The use of an internal DB for the solution is still pending to be defined. Now, no internal database is used. In addition, input data will be accessed and collected from database (MongoDB) or files (excel files) provided by the pilots. Specific data preprocessing must be done on the pilots' data to get the expected input data. Finally, the results and solution's outputs will be sent to the AIDEAS Machine Passport.

4.12.4.1 AI^{PM} Implementation Components

Implementation Components	Description
Import Data	This component facilitates the reading of data from different data sources like databases, files etc.
Data Validation and Preprocessing	This component validates the training data and ensures that the input data is in the correct format before feeding it into the model.

Machine Configurator	This component allows the definition of the current machine configuration to identify its components and associated variables.
Create and Export Models	This component provides different algorithms, which train on the available pre-processed data to determine the remaining useful life or failure risk of a component or system. Two approaches/models: a health-predictive model and a survival model to assess the RUL and the failure risk.
Obtain RUL and Failure Risk Estimation and Display Results	This component allows to predict a RUL estimation in both the machine and its components. It assesses the failure risk at component level with new data using the trained models.
Export Data	This component sends the data to the Machine Passport and enables the user to export it directly if needed.

Table 135. AI^{PM} Implementation Components

4.12.4.2 Technical Description of AI^{PM} Components

Following the identification and specification of the implementation components in the previous section, the technical description of each component is provided in the following tables. It details the set of technologies required to implement each of the implementation components.

Implementation component	Import Data
Description of implementation component	This component facilitates the reading of data from different data sources like databases, files etc.
Used technologies	Python
Technical Description of the Component	**Dependencies**
	Development Language: Python Libraries: Pandas Container: Docker
	Interfaces
	User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: MongoDB, minIO

Table 136. Technical Description of AI^{PM} "Import Data" Implementation Component

Implementation component	Data Validation and Preprocessing
Description of implementation component	This component facilitates the reading of data from different data sources like databases, files etc.

Used technologies	Python	
Technical Description of the Component	Dependencies	
	Development Language: Python Libraries: Pandas, numpy Container: Docker Database: to be defined	
	Interfaces	
	User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: MongoDB, minIO	
	Requires	
	Import Data	

Table 137. Technical Description of A AI[PM] "Data Validation and Preprocessing" Implementation Component

Implementation component	Machine Configurator
Description of implementation component	This component facilitates the definition of the machine that will be the target of the anomaly detection, defining its components and its relevant process variables.
Used technologies	Python
Technical Description of the Component	Dependencies
	Development Language: Python Libraries: Pandas, numpy Container: Docker Database: to be defined
	Interfaces
	User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: MongoDB, minIO
	Requires
	Import Data

Table 138. Technical Description of AI[PM] "Machine Configurator" Implementation Component

Implementation component	Create and Export Models
Description of implementation component	This component facilitates the parametrization and creation of new machine learning models, which train on the previously pre-processed data given a specific machine configuration.
Used technologies	Python
Technical Description of the Component	**Dependencies**
	Development Language: Python Libraries: sklearn, keras, tensorflow, pickle Container: Docker Database: to be defined
	Interfaces
	User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: MongoDB, minIO
	Requires
	Import Data, Data Validation and Preprocessing, Machine Configurator

Table 139. Technical Description of AI^{PM} "Create and Export Models" Implementation Component

Implementation component	Obtain RUL Estimation and Display Results
Description of implementation component	This component predicts if new data is considered an anomaly or not using previously trained models. Results are shown in different formats.
Used technologies	Python
Technical Description of the Component	**Dependencies**
	Development Language: Python, Javascript Libraries: sklearn, keras, tensorflow, pickle, plotly.js Container: Docker Database: to be defined
	Interfaces
	User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS

	Data Repository: MongoDB, minIO
	Requires
	Import Data, Data Validation and Preprocessing, Machine Configurator, Create and Export Models

Table 140. Technical Description of AI[PM] "Obtain Predictions and Display Results" Implementation Component

Implementation component	Export Data
Description of implementation component	This component facilitates exporting the obtained results to a local storage or the Machine Passport.
Used technologies	Python
Technical Description of the Component	Dependencies
	Development Language: Python Libraries: Pandas, numpy Container: Docker Database: MongoDB
	Interfaces
	User Interface: Yes, REACT Synchronous/Asynchronous Interface: RESTful APIs Network/Protocols: HTTP/HTTPS Data Repository: MongoDB, minIO
	Requires
	Machine Configurator, Obtain Predictions and Display Results

Table 141. Technical Description of AI[PM] "Export Data" Implementation Component

4.13. Smart Retrofitter – AI^SR

4.13.1 Usage Viewpoint

The usage activity diagram (Figure 58) within AI^SR shows the interaction between different tasks and roles. This diagram illustrates the dynamic flow of interactions, offering a representation of the activities performed by the solution and providing a clear understanding of the activities performed with the AI^SR solution.

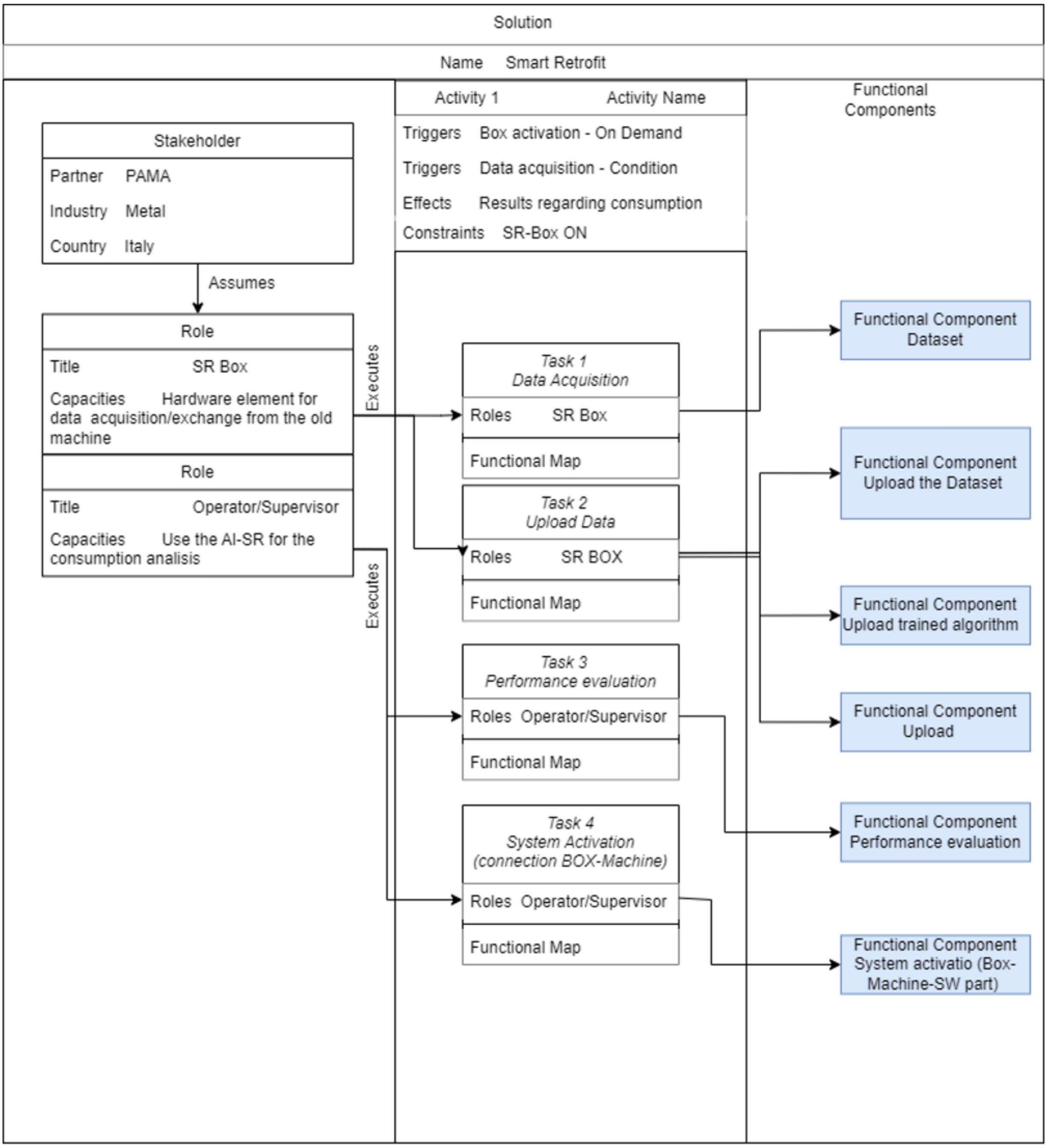


Figure 58. AI^SR Usage Viewpoint activity Diagram

4.13.2 Functional Viewpoint

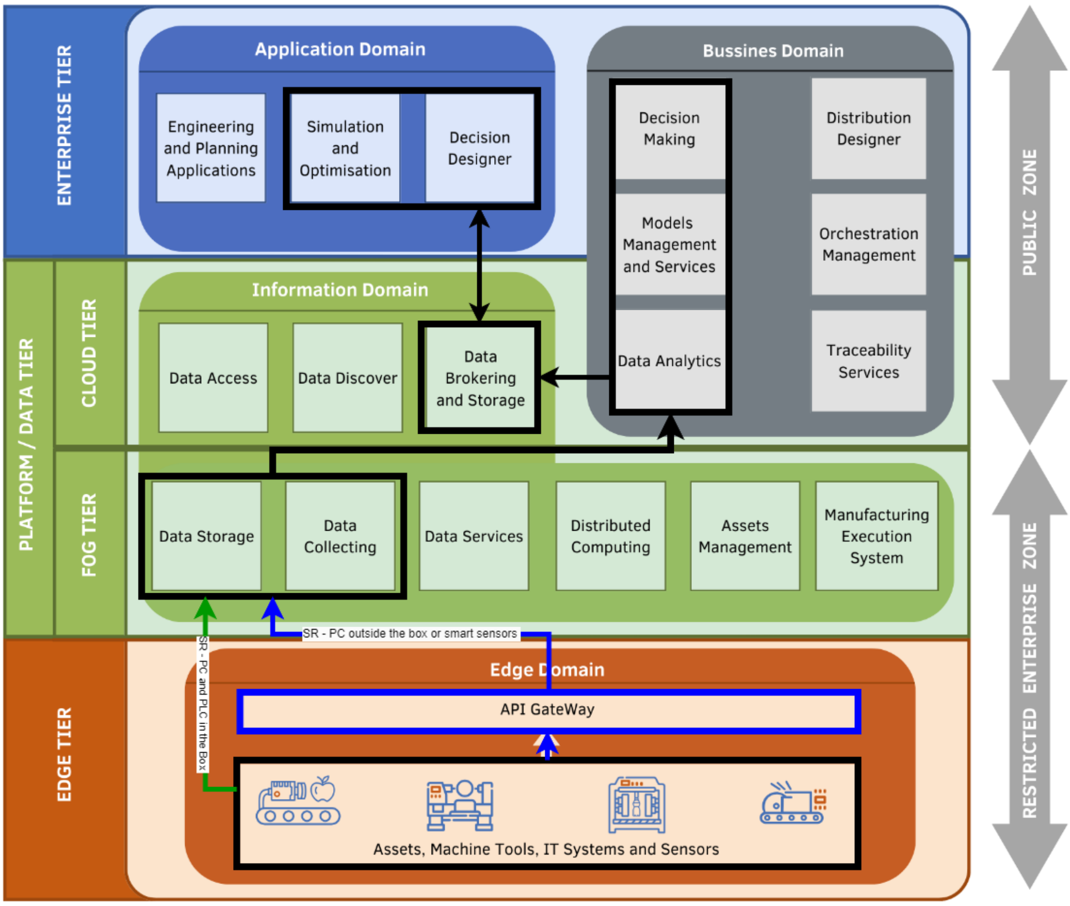


Figure 59. Data Flow AI^{SR}

The Smart Retrofit obtains the machine operating data, collected by the hardware part of the solution (box installed in the edge), and processes it through an artificial intelligence algorithm (software part of the solution) to understand the operating conditions of the machine and operate/decide in relation to them. The solution stores data and processes it locally and then sends it together with the results to a data collection cloud.

4.13.3 Data structure of AI^{SR}

Format	Input/Output	Example
Database	Input	MONGODB Timestamp, Value Id, Value 2024-01-08T05:05:34.115+00:00 , 20001, 0
JSON	Input	[{ 'Time': 'ValueId': 'Value' : }, ...]

JSON	Input	[{ "_id": , "Name": “” , "LastUpdate": { "$date": " " }, "Description": , "DescriptionEn": , "LastValue": , "MinVal": , "MaxVal": , "MeasureUnit": , "IntervalMs": , }, ...]

Table 142. Input / Output Data Format AI^{SR}

4.13.4 AI^{SR} Hardware Requirements

Smart Retrofits' solution involves the integration of a hardware part called the Smart Retrofit Box on the machine, which provides for the integration of all the devices required to bring the machine closer to the industry 4.0 paradigm. The SR-Box is designed to be generic and modular, thus adaptable to any situation. In fact, following a preliminary analysis on the machine, it is possible to identify if/which sensors are needed/present on the machine. Based on the initial conditions of the machine, the SR-Box is then integrated with new sensors or connected to existing ones. In this way, it is possible acquire all the parameters/data and carry out the desired analysis.

Hardware Element	Importance and Explanation
WiFi connection	WiFi internet connection is necessary to realise the connection between the Smart Retrofit solution and the Cloud system. Alternatively, an intranet connection is enough to for SR operation

Table 143. Hardware required AI^{SR}

4.13.5 AI^{SR} Software Requirements

Software Component	Description/Role	Required Version/Configuration	Dependencies
Linux OS or Windows OS	Operating system needed to use the tool	Ubuntu Debian / Windows 10 Pro	N/A
Docker	Build, share and run containers	latest	N/A

Table 144. Software requirements AI^{SR}

4.13.6 AI^{SR} Lifecycle

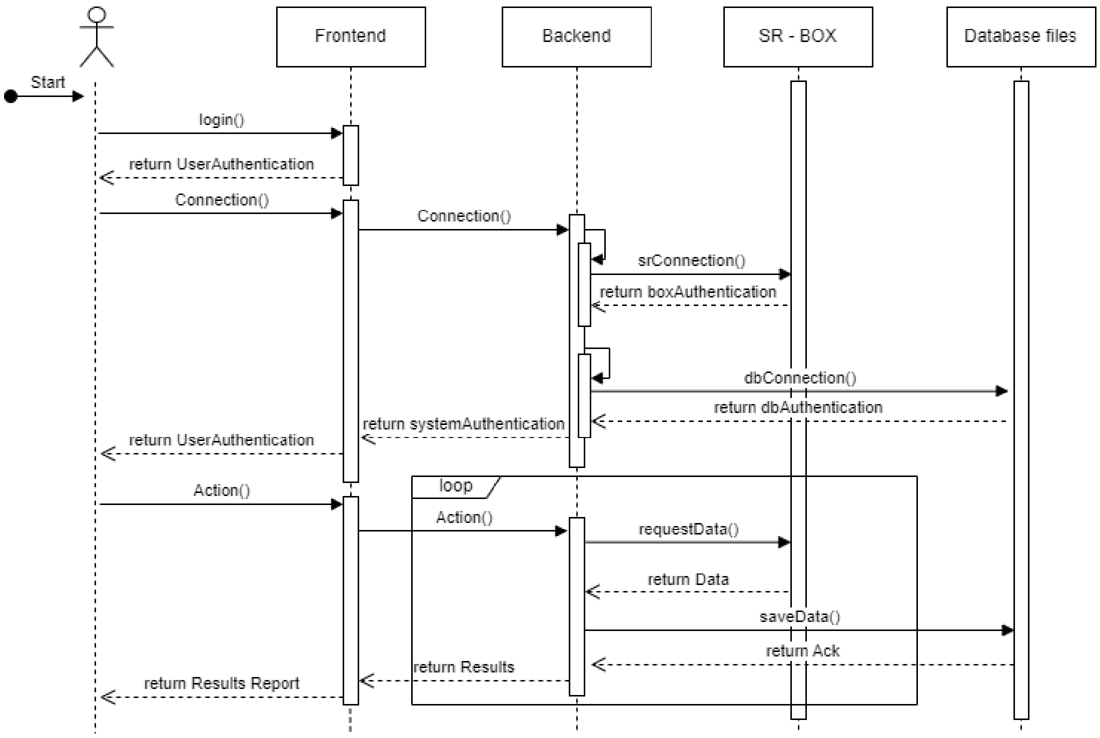


Figure 60. AI^{SR} Lifecycle

This sequence diagram illustrates the flow of interactions between the User, the Frontend, the Backend, the SR-Box objects, and the Database during the main processes. The user logs in to the AIDEAS platform, establish connections with the Box and the Database, initiate the main actions that request and process data to obtain useful results.

4.13.6.1 Objects

- **User**: Represents the user interacting with the solution.
- **Frontend**: Represents the user interface and the presentation layer.
- **Backend**: Represents the application logic and the server layer.
- **SR-BOX**: Represents the hardware/acquisition layer.
- **Database**: Represents the data or file storage layer.

4.13.6.2 Description – Login

Objects	Description
Start	The user logs into the platform.
User – Frontend Interaction	The user enters username and password.
Frontend - User Interaction	The Frontend shows the User the response from the login action, indicating whether it is logged in or not.
Completion	The application is ready to receive new requests from the User.

Table 145. Life-Cycle description Login AI^{SR}

4.13.6.3 Description – Database Connection

Objects	Description
Start	The user requires the Connection.
User – Frontend Interaction	The user starts the Connection.
Frontend – Backend Interaction	The Frontend sends a request to the Backend to process a connection to the database and the Box or to process the file upload.
Backend – SR-Box	The Backend requests the SR-Box to authorize the connection or to prepare to send the file. SR-Box is waiting for the authentication or for the file transfer.
Backend – Database	The Backend requests the Database to authorize the connection or to prepare to receive the file. Database is waiting for the authentication or for the file transfer.
Database - Backend Response	Database confirms that the connection has been completed or the file has been stored correctly and sends a response to the Backend.
SR-Box - Backend Response	SR-Box confirms that the connection has been completed or the file can be sent correctly. It sends a response to the Backend.
Backend - Frontend Response	The Backend informs the Frontend about the success or failure of the connection or file upload operation.
Frontend - User Interaction	The Frontend shows the User the response from the application, indicating whether the connection or file upload was successful.
Completion	The application is ready to be used by the User.

Table 146. Life-Cycle description Database Connection AI^{SR}

4.13.6.4 Description – Smart Retrofit

Objects	Description
Start	The User initiates the action of performing the Smart Retrofit solution
User – Frontend Interaction	The User start the action to monitor and control the machine behaviour
Frontend - Backend Interaction	The Frontend sends a request to the Backend to realise the action.

Backend – SR-Box Interaction	The Backend requests the SR-BOX layer to send data in real time (loop).
SR-Box - Backend Response	SR-Box return data and start the processing phase.
Backend - Database Interaction	The Backend requests the storage layer to save the data and results.
Database - Backend Response	Database sends a response to the Backend.
Backend - Frontend Response	The Backend gets the corresponding data that are displayed on the Frontend.
Frontend - User Interaction	The Frontend shows the User the response from the application regarding the behaviour of the machine.
Completion	The application is ready to receive new requests from the User.

Table 147. Life-Cycle description Smart Retrofit AI^{SR}

4.13.7 Implementation Viewpoint

The AI^{SR} is a solution with a hardware part and a software part that works by leaning on the hardware base. It is also possible to deploy the software part on a Docker as shown below (Figure 61). The main development language of AI^{SR} is Python because of its data processing and machine learning capabilities. The main libraries used are sklearn for model training and matplotlib, pandas, numpy for data processing. For its use in the "Web version", containerization takes place by using Docker. Input data will be collected from sensors in the hardware part of the solution or sent MongoDB from databases, while results and output data from the AI^{SR} solution will be stored locally (possibly also in the cloud) and in the Machine Passport.

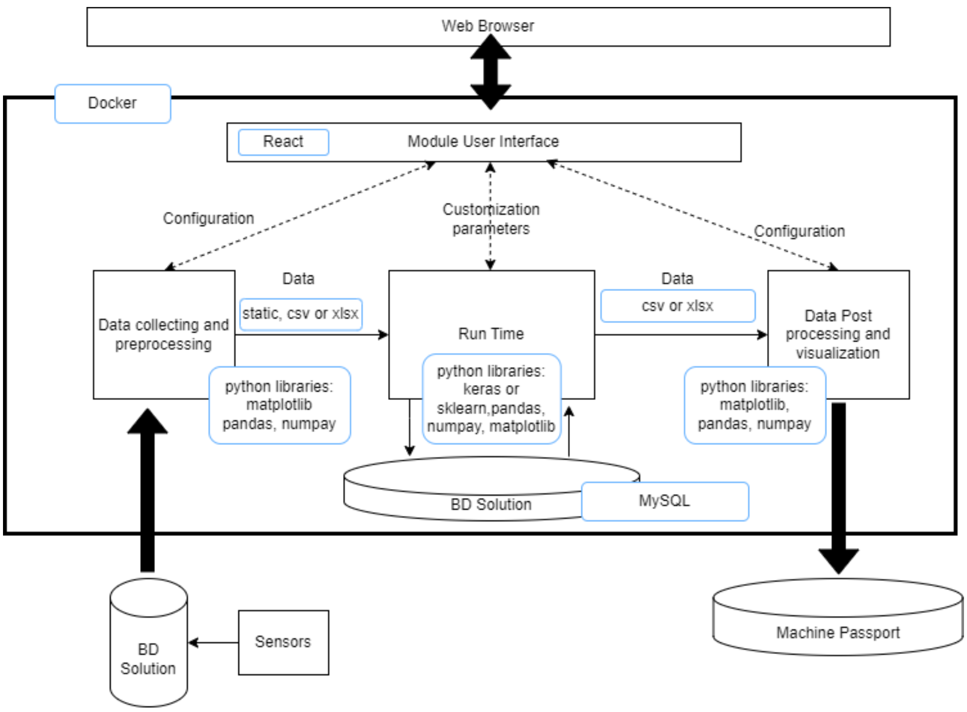


Figure 61. AI^{SR} Implementation Architecture

4.13.8 AI^{SR} Implementation Components

Implementation Components	Description
Data acquisition	The Smart Retrofit Box allows the acquisition of the data also from the old machine that are not equipped with sensors
Upload Data	Upload Valid Data as input and the AI algorithm
Perform evaluation	Evaluation of the performance and the results obtained from it

Table 148. AI^{SR} Implementation Components

4.13.9 Technical Description of AI^{SR} Components

Following the identification and specification of the implementation components in the previous section, the technical description of each component is provided in the following tables. It details the set of technologies required to implement each of the implementation components.

Implementation component	Data acquisition
Description of implementation component	The Smart Retrofit Box allows the acquisition of the data also from the old machine that are not equipped with sensors.
Used technologies	Python
Technical Description of the Component	**Dependencies** Development Language: Python Libraries: pandas, matplotlib, sklearn, numpy Container: Docker Database: MongoDB **Interfaces** User Interface: no Synchronous/Asynchronous Interface: no Network/Protocols: no Data Repository: no

Table 149. Technical Description of AI^{SR} "Data acquisition" Implementation Component

Implementation component	Upload data
Description of implementation component	This component provides the trained SR models and the acquired and pre-processed data.
Used technologies	Python
	Dependencies

Technical Description of the Component	Development Language: Python
	Libraries: pandas, matplotlib, sklearn, numpy
	Container: Docker
	Database: MongoDB
	Interfaces
	User Interface: no
	Synchronous/Asynchronous Interface: no
	Network/Protocols: no
	Data Repository: no

Table 150. Technical Description of AI^{SR} "Upload data" Implementation Component

Implementation component	Perform evaluation
Description of implementation component	Evaluate the performance and the results from the analysis by the AI-SR algorithm. Graph and data will be displayed on the UI.
Used technologies	Python
Technical Description of the Component	Dependencies
	Development Language: Python
	Libraries: pandas, matplotlib, sklearn, numpy
	Container: Docker
	Database: Local\MongoDB
	Interfaces
	User Interface: yes
	Synchronous/Asynchronous Interface: Synchronous (real time/near real time)
	Network/Protocols: no
	Data Repository: no

Table 151. Technical Description of AI^{SR} "Perform evaluation" Implementation Component

4.14. LCC/LCA/S-LCA – AI^{LC}

4.14.1 Usage Viewpoint

The usage activity diagram (Figure 62) within AI^{LC} shows the interaction between different tasks and roles. This diagram illustrates the dynamic flow of interactions, offering a representation of the activities performed by the solution and providing a clear understanding of the activities performed with the AI^{LC} solution.

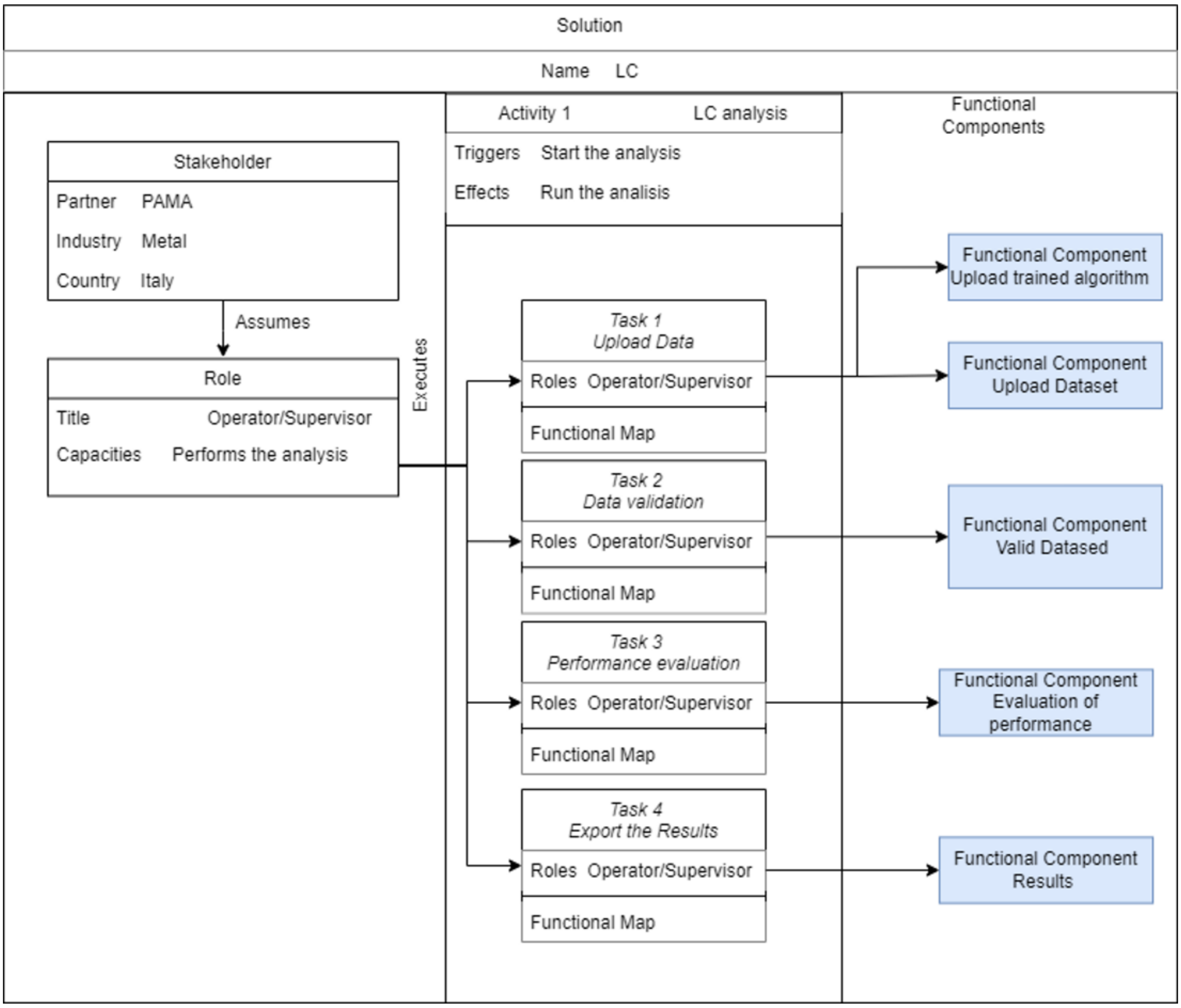


Figure 62. AI^{LC} Usage Viewpoint Activity Diagram

4.14.2 Functional Viewpoint

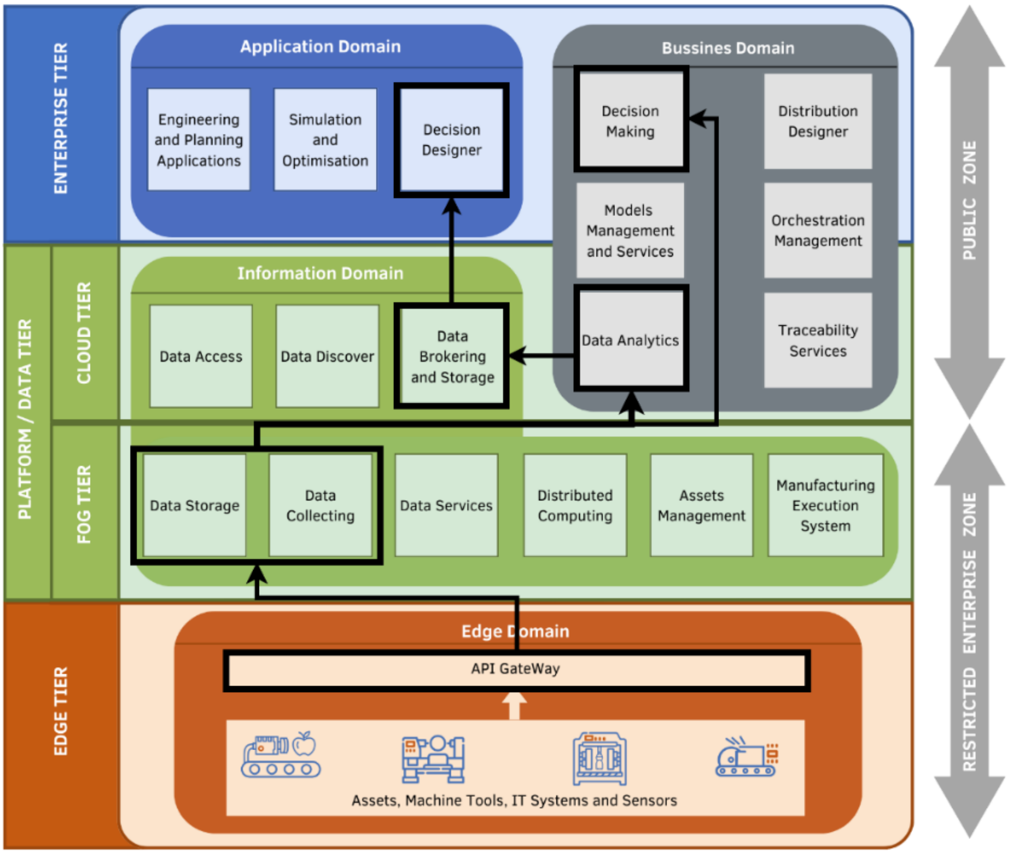


Figure 63. Data Flow AI^{LC}

The AI^{LC} allows users to model the End-of-Life graph of industrial equipment and connect to it all the information related to Life Cycle Assessment, Life Cycle Costing and Social-Life Cycle assessment. It then processes it through an artificial intelligence algorithm to understand the best end-of-life approach for the selected equipment. The solution stores data and processes it locally and sends it together with the results to a data collection cloud.

4.14.2.1 Data structure of AI^{LC}

Format	Input/Output	Example
Database: MySQL	Input	ProcessId, LCCcost, LCAcost,SLCAcost P1, 1500,2000,3000
JSON	Input	{ "vc": ["C1","C20","C28",...], "vp": ["P9","P10","P11",...], "codec": ["C0","C1","C1.1","C1.1.1", ...], "codep": ["P1","P1.1","P1.1.1",...], "mcp": [[1,0, 0...],

		[1,0, 0...], [1,0, 0...],], "mpc": [[1,0, 0...], [1,0, 0...], [1,0, 0...],] }

JSON	Output	{
		"Paths": [
		["P1", "P2" ...],
		["P1", "P2" ...],
		],
		"Paths Costs": [1, 21, 10, ...],
		"Best Path": [["P1", "P2" ...]],
		"Best Cost": 100,
		}

Table 152. Input / Output Data Format AI^{LC}

4.14.2.2 AI^{LC} Software Requirements

Software Component	Description/Role	Required Version/Configuration	Dependencies
Linux OS or Windows OS	Operating system needed to use the tool	Ubuntu 22.04 / Windows 10 Pro	N/A
Docker	Build, share and run containers	latest	N/A

Table 153. Software requirements AI^{LC}

4.14.3 AI^LC^ Lifecycle

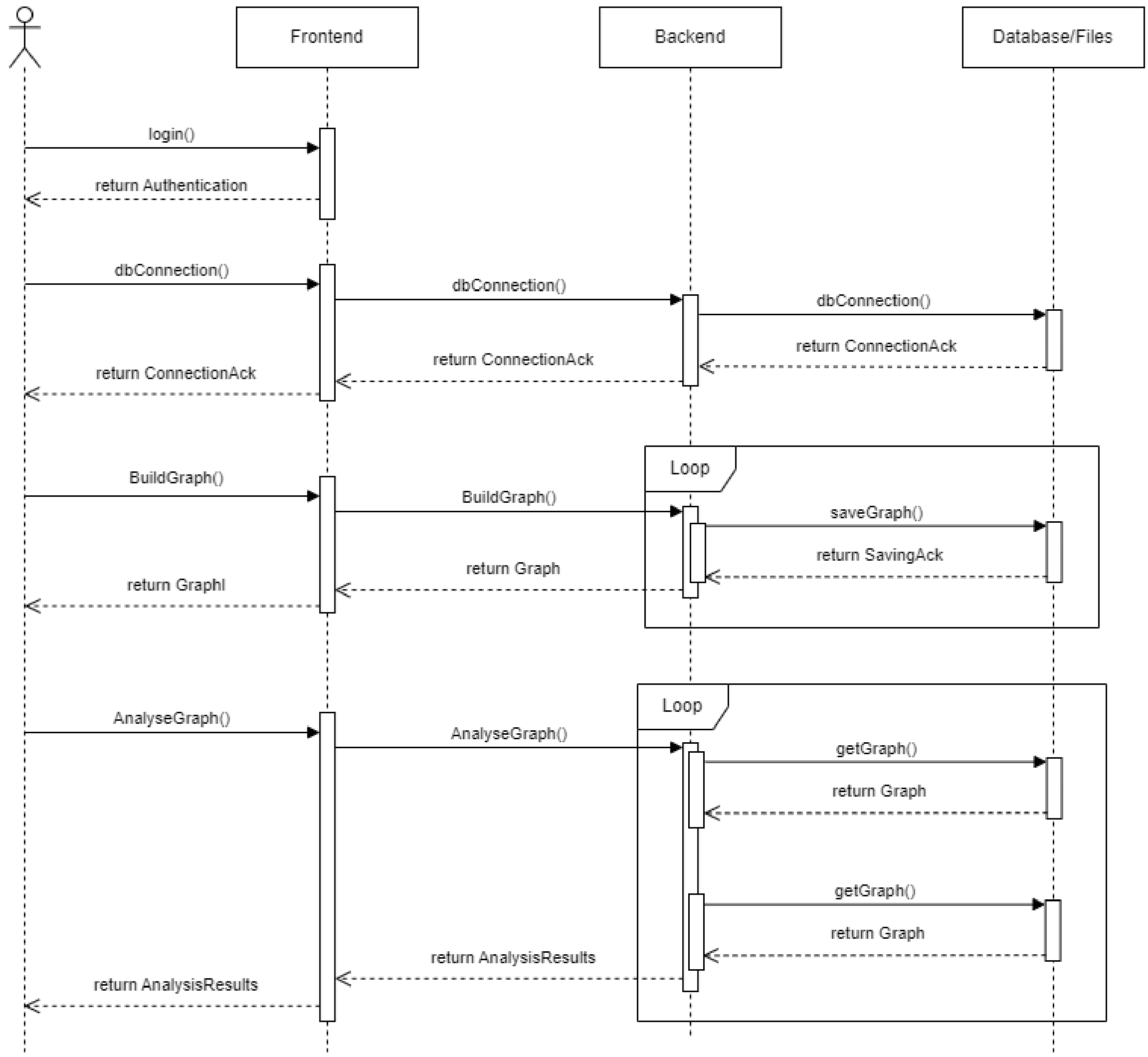


Figure 64. AI^LC^ Lifecycle

This sequence diagram illustrates the flow of interactions between the User, Frontend, Backend, and Data/Files objects during the main processes. Login into the AIDEAS platform, establish a connection to an internal/external database, model the End-of-Life graph, and analyse it.

4.14.3.1 Objects

- **User**: Represents the User interacting with the application.
- **Frontend**: Represents the user interface and the presentation layer.
- **Backend**: Represents the application logic and the server layer.
- **Database/Files:** Represents the data or file storage layer.

4.14.3.2 Description – Login

Objects	Description
Start	The User logs into the AIDEAS platform.

User – Frontend Interaction	The User enters username and password.
Frontend - User Interaction	The Frontend shows the User the response from the login action, indicating whether it is logged in or not.
Completion	The application is ready to receive new requests from the User.

Table 154. Life-Cycle description Login AI[LC]

4.14.3.3 Description - Database Connection / File Upload

Objects	Description
Start	The User initiates the action of establishing a database connection or uploads data files.
User – Frontend Interaction	The User parametrizes the database connection or uploads a data file.
Frontend - Backend Interaction	The Frontend sends a request to the Backend to process a connection to the database or to process the file upload.
Backend - Data/Files Interaction	The Backend requests the storage layer (Data/Files) to authorize the connection or to prepare to receive the file. Data/Files is waiting for the authentication or for the file transfer
Data/Files - Backend Response	Data/Files confirms that the connection has been completed or the file has been stored correctly and sends a response to the Backend.
Backend - Frontend Response	The Backend informs the Frontend about the success or failure of the connection or file upload operation.
Frontend - User Interaction	The Frontend shows the User the response from the application, indicating whether the connection or file upload was successful.
Completion	The application is ready to receive new requests from the User.

Table 155. Life-Cycle description Database Connection AI[LC]

4.14.3.4 Description - End-of-Life graph modelling

Objects	Description
Start	The User initiates the action of End-of-Life modelling
User – Frontend Interaction	The User model the graph using the user interface buttons or uploading saved model on DB
Frontend - Backend Interaction	The Frontend sends a request to the Backend to adapt the modelled graph for future analyses

Backend - Data/Files Interaction	The Backend requests the storage layer (Data/Files) to authorize the connection or to prepare to receive the data. Data/Files is waiting for the data transfer.
Data/Files - Backend Response	Data/Files sends a response to the Backend
Backend - Frontend Response	The Backend gets the corresponding data and builds the model with the inputs defined previously by the user. During the process, data are saved on db periodically and autonomously.
Frontend - User Interaction	The Frontend shows the User the response from the application, obtaining the graphical representation of the EoL graph and all the related information, and having the possibility to save it for its future use
Completion	The application is ready to receive new requests from the User

Table 156. Life-Cycle description End-of-Life graph modelling AI[LC]

4.14.3.5 Description - End-of-Life graph analysis

Objects	Description
Start	The User initiates the action of performing the EoL graph analysis
User – Frontend Interaction	The User can model a new EoL graph or upload an existing one.
Frontend - Backend Interaction	The Frontend sends a request to the Backend to model or upload a saved graph and perform the analysis.
Backend - Data/Files Interaction	The Backend requests the upload a saved model or adapt the modelled for the analysis and analyse it. During the process, data are saved on db periodically and autonomously. Data/Files is waiting for the data transfer.
Data/Files - Backend Response	Data/Files sends a response to the Backend.
Backend - Frontend Response	The Backend gets the corresponding data and evaluates the model.
Frontend - User Interaction	The Frontend shows the User the response from the application, obtaining the can exhaustive report for the selected EoL graph.
Completion	The application is ready to receive new requests from the User.

Table 157. Life-Cycle description End-of-Life graph analysis AI[LC]

4.14.4 Implementation Viewpoint

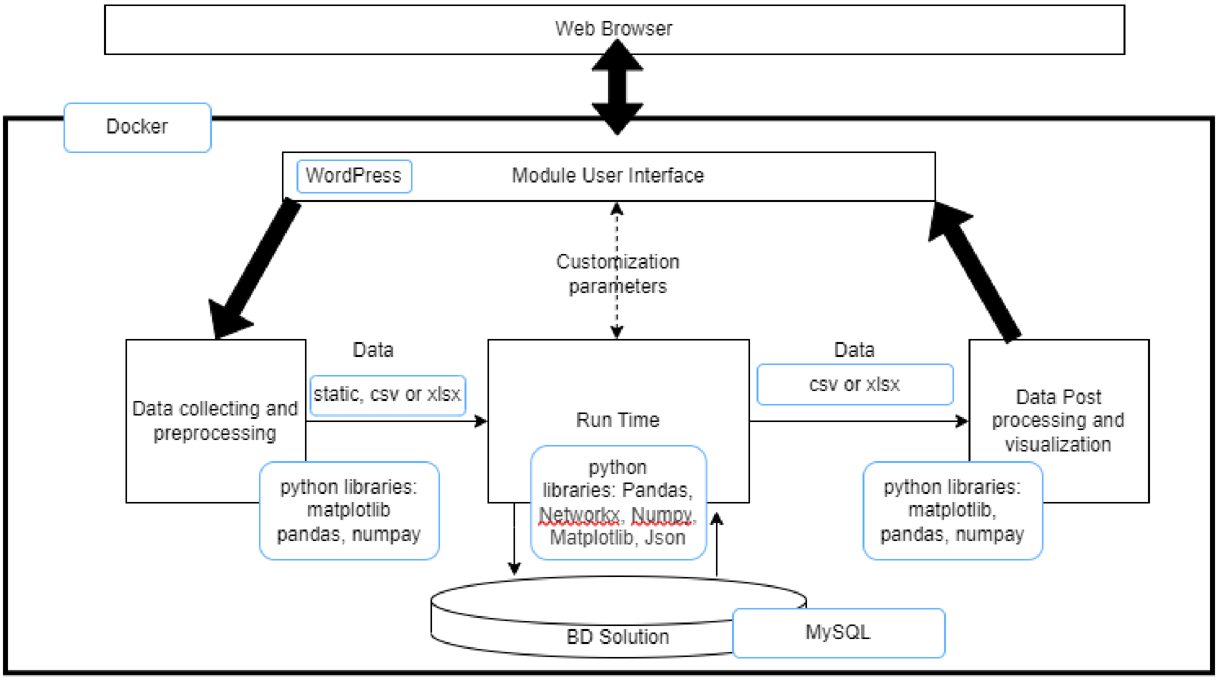


Figure 65. AI^{LC} Implementation Architecture

4.14.5 AI^{LC} Implementation Components

Implementation Components	Description
Import Data	A function designed to import data from databases to execute commands and fetch results, while user data could be collected via interactive prompts or GUI elements, allowing for a flexible and user-driven data input process.
Data Validation	The function is designed to perform comprehensive checks on incoming data, ensuring its integrity by validating formats, types, and values against predefined criteria, possibly incorporating both automated rules for database-derived data and interactive feedback mechanisms for user-provided inputs. This process is crucial in maintaining data quality and reliability within systems that handle diverse data sources.
Compose LC algorithm	The function is designed to allow the users to model the EoL graph and insert all the information related to the evaluation of the best end of life.
Run LC algorithm	The function is devoted to performing the EoL graph analysis and providing results.
Export the results	The function is designed to efficiently transfer processed data to external files or systems, likely supporting various formats such as CSV, JSON, or direct database insertion. It ensures that the analysis outcomes are accessible for further use or presentation, catering to different stakeholder needs and integration requirements.

Table 158. AI^{LC} Implementation Components

4.14.6 Technical Description of AI^{LC} Components

Following the identification and specification of the implementation components in the previous section, the technical description of each component is provided in the following tables. It details the set of technologies required to implement each of the implementation components.

Implementation component	Import Data
Description of implementation component	A function designed to import data from databases and users.
Used technologies	Python
Technical Description of the Component	**Dependencies** Development Language: Python Libraries: pandas, matplotlib, sklearn, numpy, json, csv Container: Docker Database: MySQL **Interfaces** User Interface: Yes, bootsrap and codeigniter Synchronous/Asynchronous Interface: Restfull APIs Network/Protocols: HTTP/HTTPS Data Repository: MySQL

Table 159. Technical Description of AI^{LC} "Import Data" Implementation Component

Implementation component	Data Validation
Description of implementation component	The function is likely designed to perform comprehensive checks on incoming data.
Used technologies	Python
Technical Description of the Component	**Dependencies** Development Language: Python, PHP Libraries: pandas, matplotlib, sklearn, numpy, json, csv Container: Docker Database: MySQL **Interfaces** User Interface: no Synchronous/Asynchronous Interface: no Network/Protocols: no

	Data Repository: no

Table 160. Technical Description of AI^{LC} “Data Validation” Implementation Component

Implementation component	Compose LC algorithm
Description of implementation component	The function is designed to allow the users to model the EoL graph and insert all the information related to the evaluation of the best end of life.
Used technologies	Python
Technical Description of the Component	Dependencies
	Development Language: PHP, JavaScript, Python Libraries: pandas, matplotlib, sklearn, numpy, json, csv, codigniter (php) Container: Docker Database: MySQL
	Interfaces
	User Interface: Yes, Bootstrap and Codeigniter Synchronous/Asynchronous Interface: Restful APIs Network/Protocols: HTTP/HTTPS Data Repository: MySQL

Table 161. Technical Description of AI^{LC} “Compose LC algorithm” Implementation Component

Implementation component	Run LC algorithm
Description of implementation component	The function is devoted to performing the EoL graph analysis and providing results.
Used technologies	Python
Technical Description of the Component	Dependencies
	Development Language: Python Libraries: pandas, matplotlib, sklearn, numpy, Networkx, JSON, CSV Container: Docker Database: MySQL
	Interfaces
	User Interface: Yes, Bootstrap and Codeigniter Synchronous/Asynchronous Interface: Restful APIs Network/Protocols: HTTP/HTTPS

	Data Repository: MySQL

Table 162. Technical Description of AI[LC] “Run LC algorithm” Implementation Component

Implementation component	Export the results
Description of implementation component	The function is designed to efficiently transfer processed data to external files or systems.
Used technologies	Python
Technical Description of the Component	Dependencies
	Development Language: Python Libraries: pandas, matplotlib, sklearn, numpy Container: Docker Database: MySQL
	Interfaces
	User Interface: Yes, Bootstrap and Codeigniter Synchronous/Asynchronous Interface: Restful APIs Network/Protocols: HTTP/HTTPS Data Repository: MySQL

Table 163. Technical Description of AI[LC] “Export the results” Implementation Component

4.15. Disassembler – AI[DIS]

4.15.1 Usage Viewpoint

The usage activity diagram (Figure 66) within AI[DIS] shows the interaction between different tasks and roles. This diagram illustrates the dynamic flow of interactions, offering a representation of the activities performed by the solution and providing a clear understanding of the activities performed with the AI[DIS] solution.

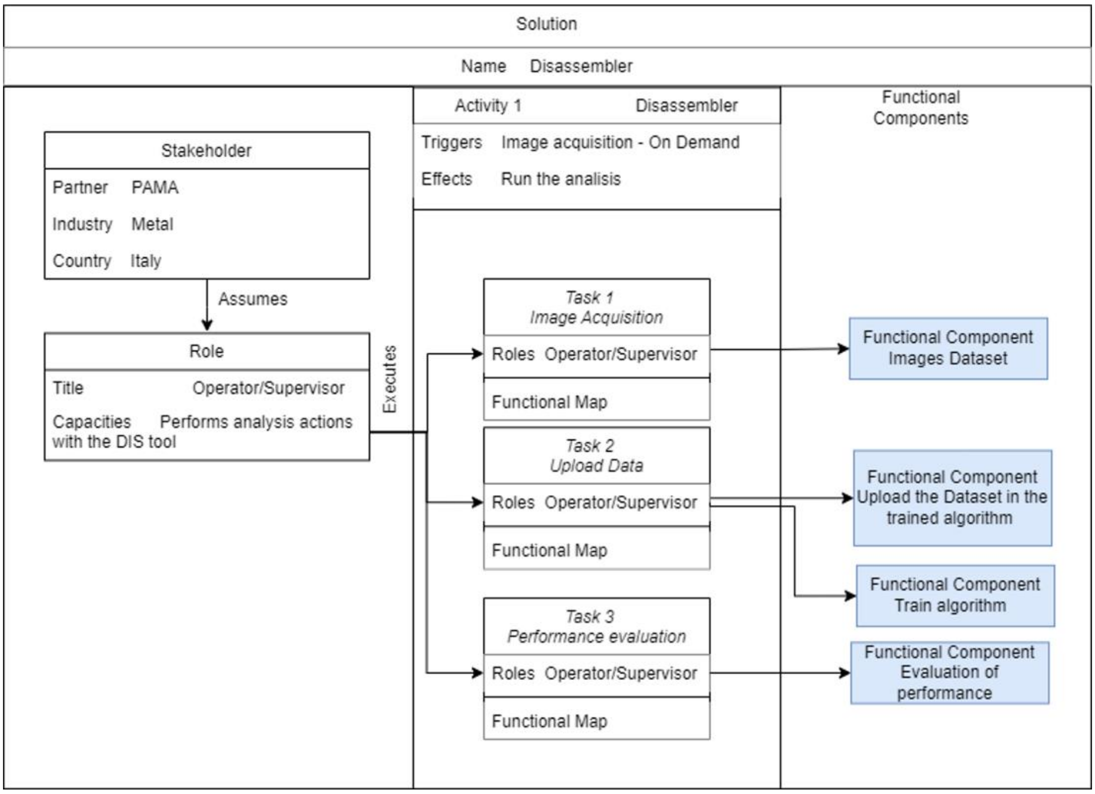


Figure 66. AI[DIS] Usage Viewpoint Activity Diagram

4.15.2 Functional Viewpoint

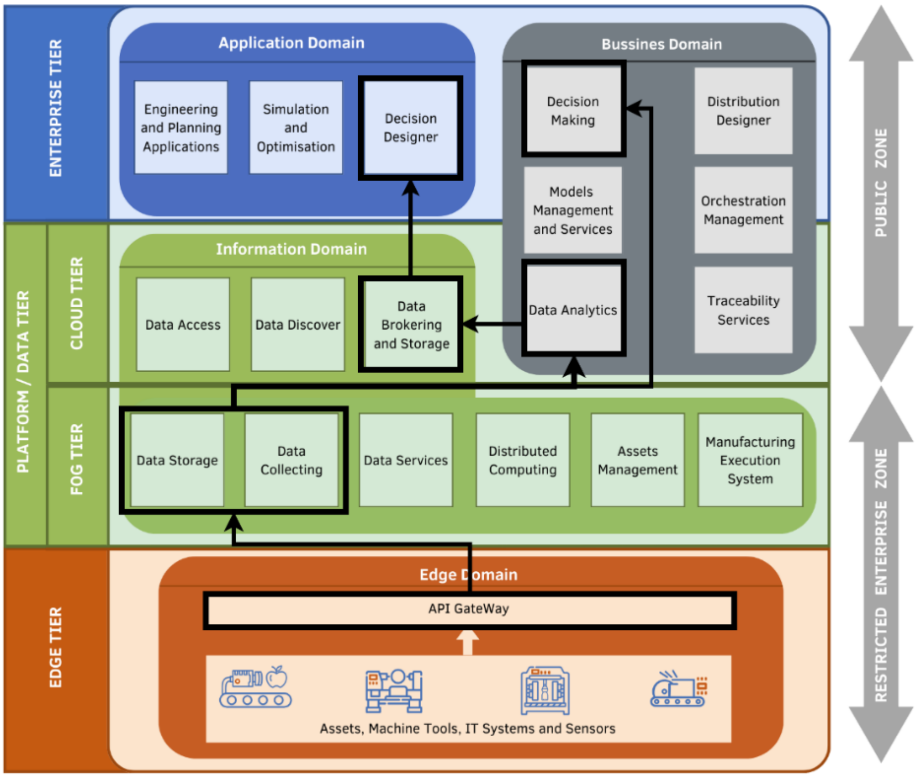


Figure 67. Data Flow AI[DIS]

The AI^{DIS} allows users to analyse the status of an equipment disassembly with all the related information through image acquisition. Then, processes the image through an artificial intelligence algorithm to understand the component status. The solution stores data and processes it locally and then sends it together with the results to a data collection cloud.

4.15.2.1 Data structure of AI^{DIS}

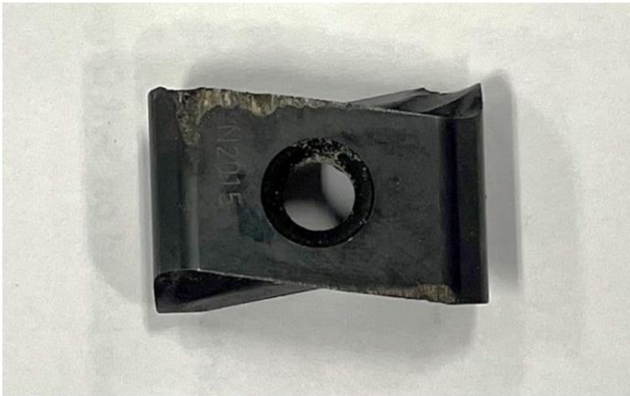
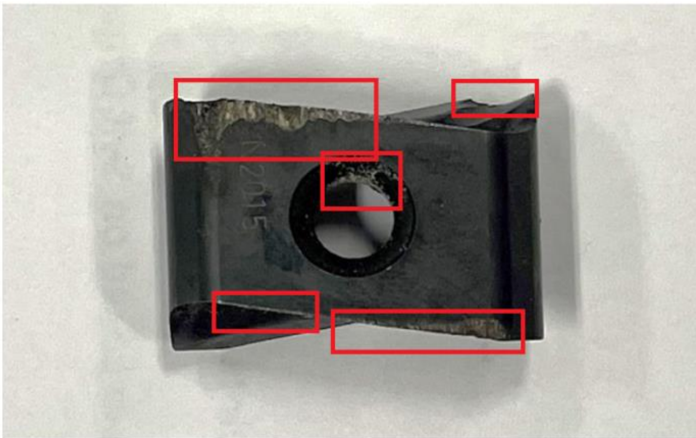
Format	Input/Output	Example
IMG	Input	
IMG	Output	
JSON	Output	{ "Time"= 2024-01-08T05:05:34.115+00:00, "IdMachine"= "AS232", "IdImage"="DF151", "QualityEvaluation" = 70, "Note" = "description…", }

Table 164. Input / Output Data Format AI^{DIS}

4.15.2.2 AI^DIS Hardware Requirements

Hardware Element	Importance and Explanation	Data of Element
Industrial camera	A high-resolution camera for image acquisition in industrial field	Example: Triton2 20 MP Model (IMX183), Fujinon, 50 mm, 1.1", 23MP

Table 165. Hardware required AI^DIS

4.15.2.3 AI^DIS Software Requirements

Software Component	Description/Role	Required Version/Configuration	Dependencies
Linux OS or Windows OS	Operating system needed to use the tool	Ubuntu 22.04 / Windows 10 Pro	N/A
Docker	Build, share and run containers	latest	N/A

Table 166. Software requirements AI^DIS

4.15.2.4 AI^DIS Lifecycle

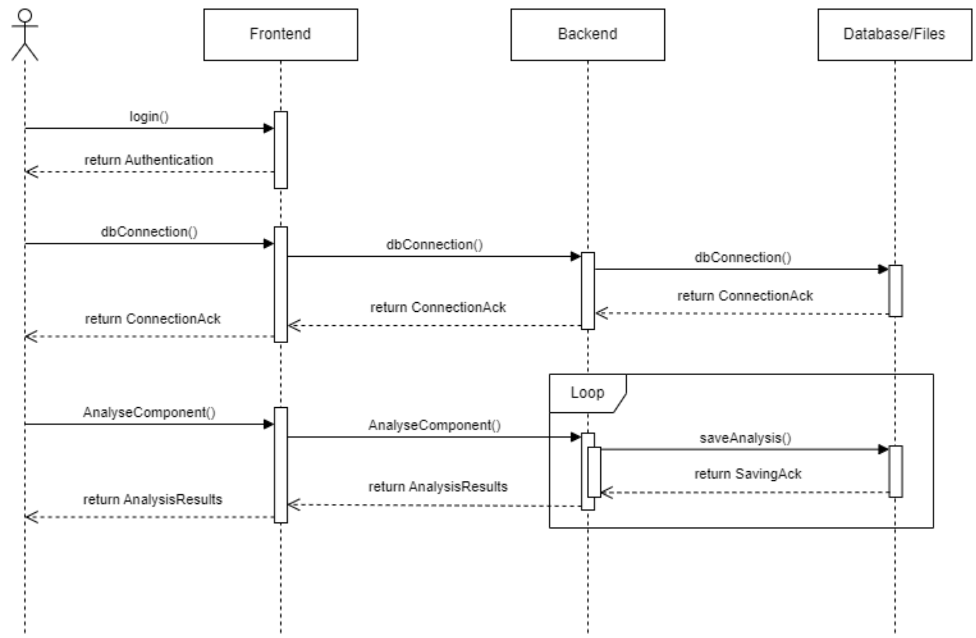


Figure 68. AI^DIS Lifecycle

This sequence diagram illustrates the flow of interactions between the User, Frontend, Backend, and Data/Files objects during the main processes. Login into the AIDEAS platform, establish a connection to an internal/external database, acquiring image from equipment, and analyse it.

4.15.2.5 Objects

- **User**: Represents the User interacting with the application.
- **Frontend**: Represents the user interface and the presentation layer.
- **Backend**: Represents the application logic and the server layer.
- **Database/Files: Represents the data or file storage layer.**

4.15.2.6 Description – Login

Objects	Description
Start	The User logs into the AIDEAS platform.
User – Frontend Interaction	The User enters username and password.
Frontend - User Interaction	The Frontend shows the User the response from the login action, indicating whether it is logged in or not.
Completion	The application is ready to receive new requests from the User.

Table 167. Life-Cycle description Login AI[DIS]

4.15.3 Description - Database Connection / File Upload:

Objects	Description
Start	The User initiates the action of establishing a database connection or uploads data files.
User – Frontend Interaction	The User parametrizes the database connection or uploads a data file.
Frontend - Backend Interaction	The Frontend sends a request to the Backend to process a connection to the database or to process the file upload.
Backend - Data/Files Interaction	The Backend requests the storage layer (Data/Files) to authorize the connection or to prepare to receive the file. Data/Files is waiting for the authentication or for the file transfer
Data/Files - Backend Response	Data/Files confirms that the connection has been completed or the file has been stored correctly and sends a response to the Backend.
Backend - Frontend Response	The Backend informs the Frontend about the success or failure of the connection or file upload operation.
Frontend - User Interaction	The Frontend shows the User the response from the application, indicating whether the connection or file upload was successful.

Completion	The application is ready to receive new requests from the User.

Table 168. Life-Cycle description Database Connection AI[DIS]

4.15.4 Description – Disassembly analysis

Objects	Description
Start	The User initiates the action of performing the smart disassembly analysis
User – Frontend Interaction	The User can activate the equipment images acquisition.
Frontend - Backend Interaction	The Frontend sends a request to the Backend to perform the analysis.
Backend - Data/Files Interaction	The Backend provide the analyses. During the process, data are saved on db periodically and autonomously. Data/Files is waiting for the data transfer.
Data/Files - Backend Response	Data/Files sends a response to the Backend.
Backend - Frontend Response	The Backend gets the corresponding data and evaluates the model.
Frontend - User Interaction	The Frontend shows the User the response from the application, obtaining an exhaustive report for the disassembly evaluation.
Completion	The application is ready to receive new requests from the User.

Table 169. Life-Cycle description Disassembly Analysis AI[DIS]

4.15.5 Implementation Viewpoint

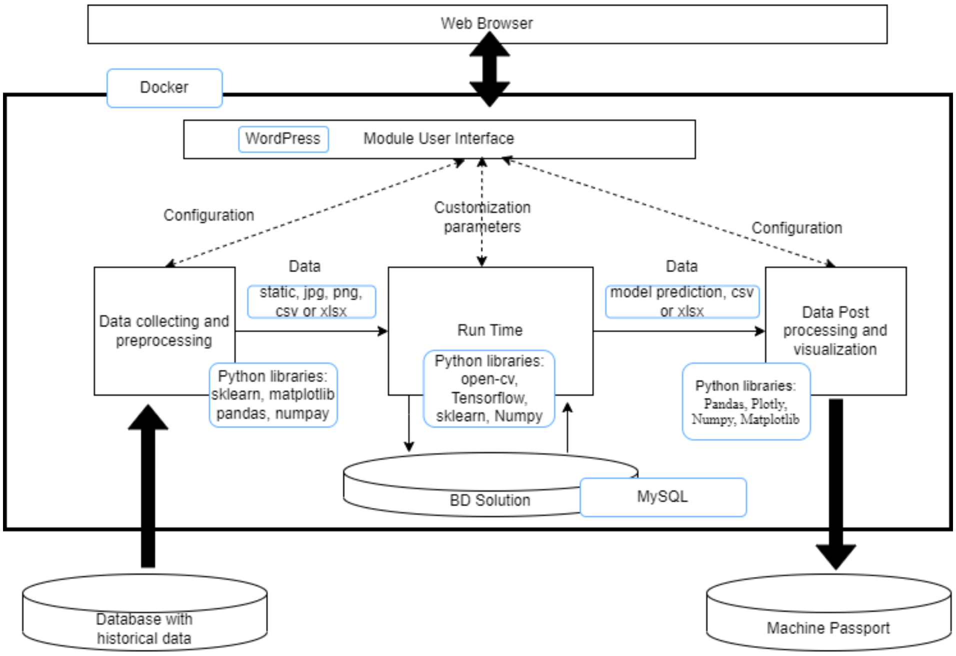


Figure 69. AI^{DIS} Implementation Architecture

The main development language for AI^{DIS} is Python, due to its data processing and machine learning functionalities. The main libraries utilised are sklearn for the AI algorithm. Docker is also used for containerisation. Input data will be collected from databases or directly with and acquisition system, while the results and output data of the AI^{DIS} solution will be stored in cloud and in the Machine Passport.

4.15.6 AI^{DIS} Implementation Components

Implementation Components	Description
Upload Data	Upload images and the trained AI algorithm to understand if a cutting tool must be disassembled and replaced or not.
Image Acquisition	Acquire an image of a cutting tool to be analysed.
Performance evaluation	Run the analysis and evaluate the results

Table 170. AI^{DIS} Implementation Components

4.15.7 Technical Description of AI^{DIS} Components

Following the identification and specification of the implementation components in the previous section, the technical description of each component is provided in the following tables. It details the set of technologies required to implement each of the implementation components.

Implementation component	Upload Data

Description of implementation component	Upload images and the trained AI algorithm for understand if a cutting tool must be disassembled and replaced or not.
Used technologies	Python
Technical Description of the Component	Dependencies
	Development Language: Python Libraries: Pandas, NumPy, matplotlib, sklearn Container: Docker Database: MongoDB
	Interfaces
	User Interface: no Synchronous/Asynchronous Interface: no Network/Protocols: HTTP/HTTPS Data Repository: MySQL

Table 171. Technical Description of AI[DIS] "Upload Data" Implementation Component

Implementation component	Image Acquisition
Description of implementation component	Acquire an image of a cutting tool to be analysed.
Used technologies	Python
Technical Description of the Component	Dependencies
	Development Language: Python Libraries: Pandas , numpy, matplotlib,sklearn Container: Docker Database: MongoDB
	Interfaces
	User Interface: yes Synchronous/Asynchronous Interface: Asynchronous Network/Protocols: HTTP/HTTPS Data Repository: MySQL

Table 172. Technical Description of AI[DIS] "Image Acquisition" Implementation Component

Implementation component	Performance Evaluation
Description of implementation component	Run the analysis and evaluate the results displayed on the UI.
Used technologies	Python

Technical Description of the Component	Dependencies	
	Development Language: Python Libraries: Pandas, NumPy, matplotlib, sklearn Container: Docker Database: MongoDB	
	Interfaces	
	User Interface: yes Synchronous/Asynchronous Interface: Asynchronous Network/Protocols: HTTP/HTTPS Data Repository: MySQL	

Table 173. Technical Description of AI^{DIS} “Performance Evaluation” Implementation Component

4.16. Machine Passport – AI^{MP}

4.16.1 Usage Viewpoint

The usage activity diagrams (Figures 70 & 71) within AI^{MP} demonstrate the interaction between the different tasks and roles. No distinction between pilots is made, as the solution's functionality remains consistent across various use cases and stages of the product life cycle. These diagrams depict the dynamic flow of interactions, showcasing the activities performed by the AI^{MP} solution. They provide a clear understanding of the operational processes and collaborative engagements among different tasks and roles.

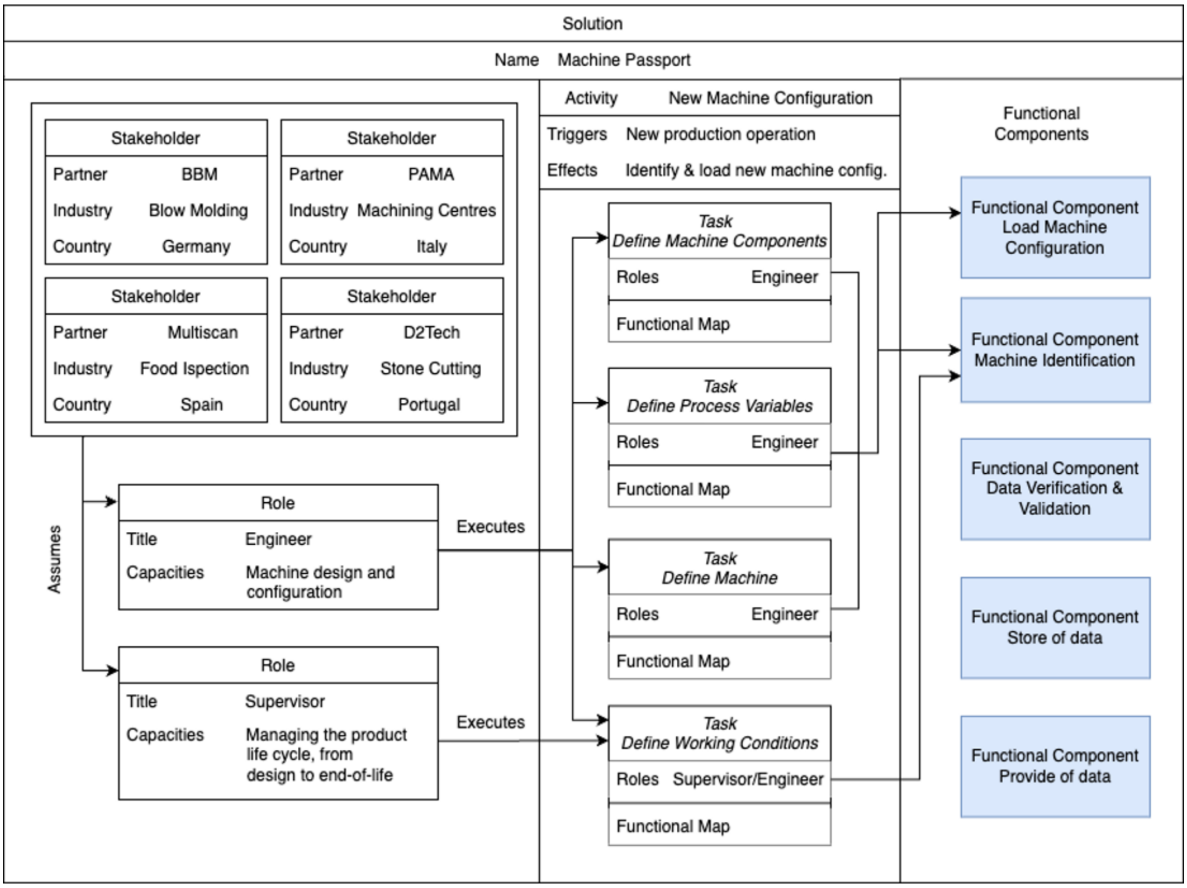


Figure 70. AI^{MP} Activity Diagram (New Machine Configuration)

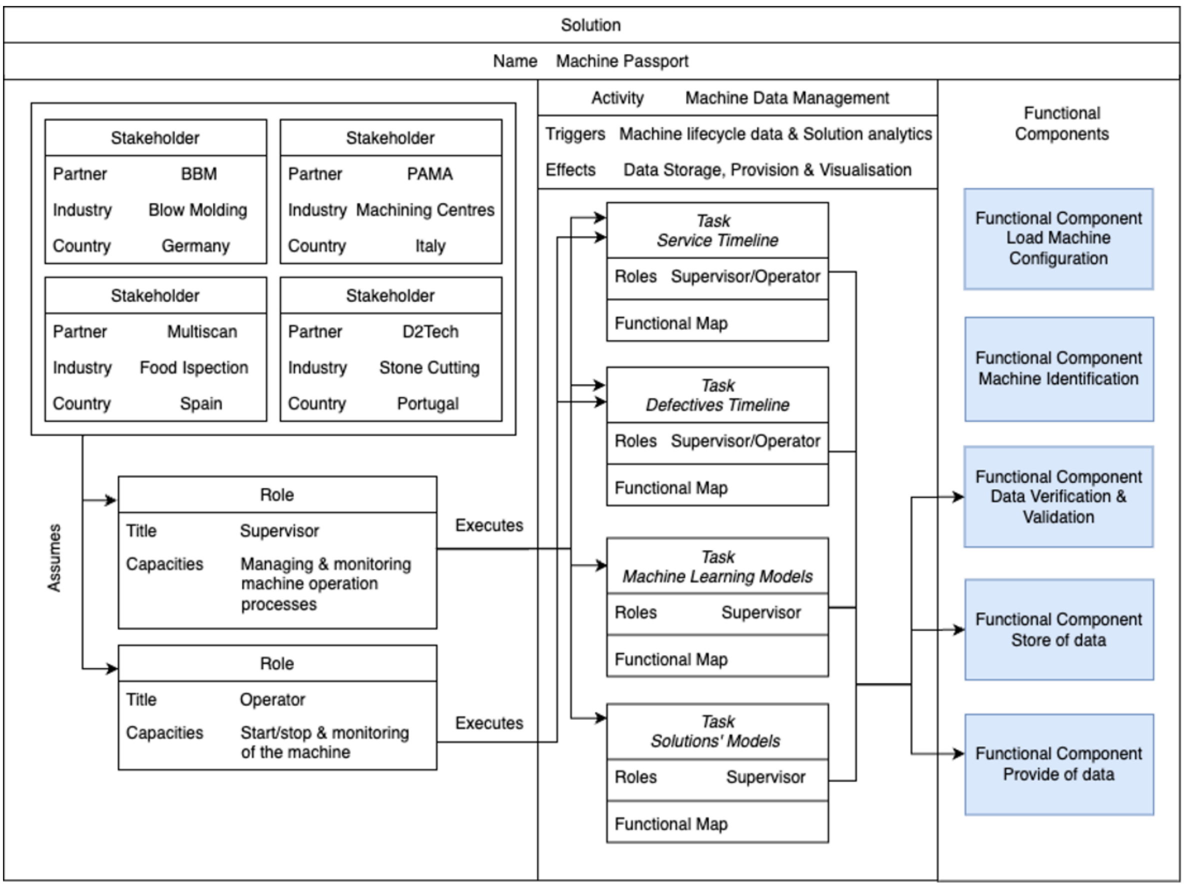


Figure 71. AI^{MP} Activity Diagram (Machine Data Management)

4.16.2 Functional Viewpoint

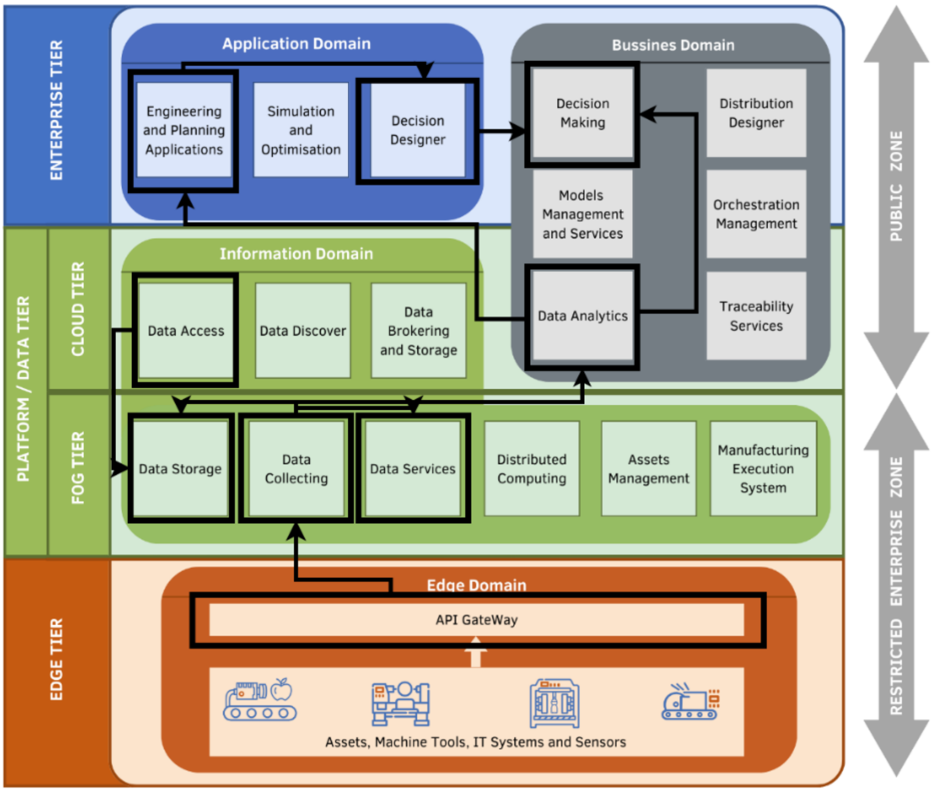


Figure 72. Data Flow AI[MP]

It all starts in the Edge Domain, where physical assets, machine tools, IT systems, and sensors are directly connected to an API Gateway. The gateway serves as the data ingress point, channeling information from the hardware up to the higher levels of the system. The data collected by the edge devices is first sent to the Fog Tier. Here, it undergoes initial processing and might be temporarily stored.

This layer is characterized by its distributed computing capabilities, which means data can be processed closer to where it's collected, reducing latency and bandwidth use. As data moves up from the Fog Tier, it is accessed and discovered by the Information Domain to control how data is retrieved from storage and ensures that data is accessible to authorized applications and services. The processed data is then used by various applications within the enterprise for specific purposes. The data is used to assist in engineering tasks, to plan future activities and to simulate scenarios and optimize processes.

The data is also employed for business-oriented tasks such as data analytics for business decisions, planning and optimizing the distribution of goods and services, managing and servicing business and data models, turning data into insights through analysis and coordinating and managing various business processes. All this data flow occurs within defined security zones, from a public zone where more general data is handled, to a restricted enterprise zone where sensitive or critical data is processed.

4.16.2.1 Data structure of AI^MP

Format	Input/Output	Example
CSV	input	UDI, Product ID, Type, Air temperature [K], Process temperature [K], Rotational speed [rpm], Torque [Nm], Tool wear [min], Target, Failure Type 1, M14860, M, 298.1, 308.6, 1551, 42.8, 0, 0, No Failure 2, L47181, L, 298.2, 308.7, 1408, 46.3, 3, 0, No Failure 3, L47182, L, 298.1, 308.5, 1498, 49.4, 5, 0, No Failure 4, L47183, L, 298.2, 308.6, 1433, 39.5, 7, 0, No Failure
JSON	output	[{ "UDI": 1, "Product ID": "M14860", "Type": "M", "Air temperature [K]": 298.1, "Process temperature [K]": 308.6, "Rotational speed [rpm]": 1551, "Torque [Nm]": 42.8, "Tool wear [min]": 0, "Target": 0, "Failure Type": "No Failure" }, { "UDI": 2, "Product ID": "L47181", "Type": "L", "Air temperature [K]": 298.2, "Process temperature [K]": 308.7, "Rotational speed [rpm]": 1408, "Torque [Nm]": 46.3, "Tool wear [min]": 3, "Target": 0, "Failure Type": "No Failure" }]
JSON	input	{ "UDI": 1, "Product ID": "M14860", "Type": "M", "Air temperature [K]": 298.1, "Process temperature [K]": 308.6, "Rotational speed [rpm]": 1551, "Torque [Nm]": 42.8, "Tool wear [min]": 0, "Target": 0, "Failure Type": "No Failure" }

Table 174. Input / Output Data Format AI^MP

4.16.3 AI^MP Hardware Requirements

Hardware Element	Importance and Explanation	Data of Element
Storage	For the storage of knowledge coming from the solutions and data from different sources	CSV file, JSON messages, APIs
Server	To be used as a smart data management platform for manipulating industrial data	APIs
Connectivity	Connectivity with SQL databases, NoSQL databases and APIs	APIs

Table 175. Hardware required AI^MP

4.16.4 AI^{MP} Software Requirements

Software Component	Description/Role	Required Version/Configuration	Dependencies
LINUX	Operating system to use the specific tool	22.04	N/A
NodeJS	To run the component in the server	20.10	N/A
Docker	To deploy the tool in the server	20.10+	N/A

Table 176. Software requirements AI^{MP}

4.16.4.1 AI^{MP} Lifecycle

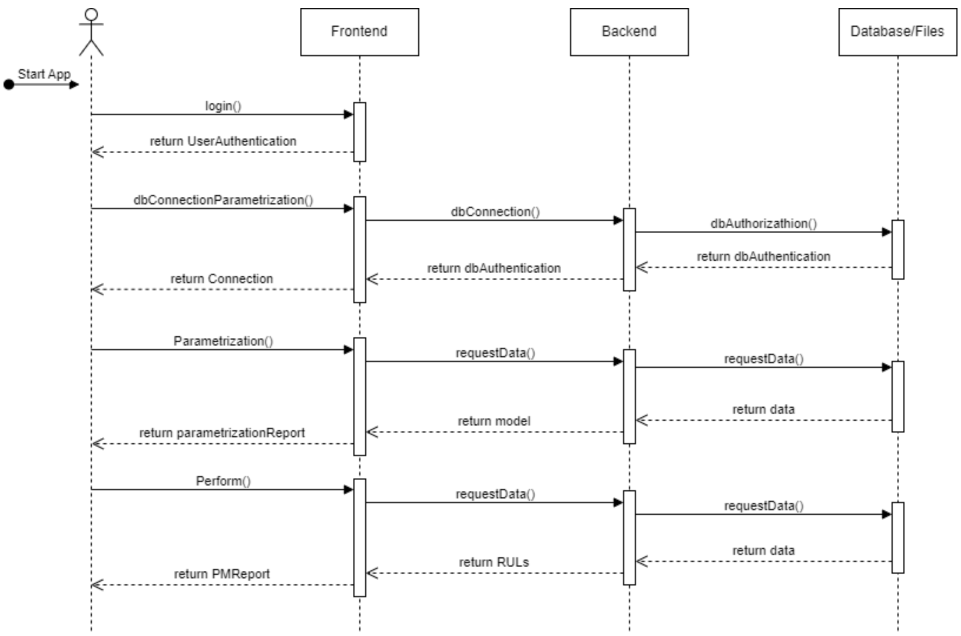


Figure 73. AI^{MP} Lifecycle

The AIDEAS Machine Passport component, as depicted in the sequence diagram, acts as a centralized platform for orchestrating the flow of data across the lifecycle of industrial equipment. It begins with user authentication through the frontend, ensuring secure access to the system. Subsequently, the dbConnectionParameterization() function is called to establish a secure connection to the backend, where dbAuthorization() verifies access rights, and upon successful authentication, a connection is established.

In the backend, the Machine Passport interfaces with the database/files system to request and retrieve data. The Parameterization() process is a critical step where specific parameters for data acquisition and processing are set. Upon completion, a report is generated and sent back to the frontend, indicating the successful configuration of parameters.

The Perform() function appears to trigger a more complex data processing or operational action, where data is once again requested from the backend. This request initiates a sequence of data retrieval from the database/files, which is then processed according to the specified business rules or logic (denoted by return RULs), and the resultant model or decision-making insight is returned to the frontend.

The Machine Passport's role in this workflow is to ensure that data from various stages of the product life phases are not only collected and stored but also made readily available for advanced analysis. The data flow facilitated by the Machine Passport allows for smart, trustful data integration, reflecting the exchange protocols, standards, and interfaces developed for seamless interaction between computer-aided systems and manufacturing stages. By managing large-scale data acquisition and sharing, the Machine Passport enables real-time, informed decision-making backed by explainable AI, ensuring that the insights provided to the frontend for user consumption are both rational and interpretable. This comprehensive approach empowers stakeholders to make decisions that enhance the design, manufacturing, and end-of-life management of industrial equipment.

4.16.5 Objects

- **User:** Acts as the authenticated gateway to the Machine Passport, initiating secure data transactions and interactions.
- **Frontend:** Delivers a user-centric interface for precise data parameterization and seamless interaction with the Machine Passport.
- **Backend:** Serves as the intelligence hub, processing data exchanges and integrating multifaceted manufacturing data within the Machine Passport framework.

4.16.6 Description – Database Connection:

Objects	Description
Start	The initiation of the Machine Passport component is triggered by the User seeking to engage with the industrial equipment lifecycle data.
User – Frontend Interaction	User authentication is facilitated through the frontend, securing access to the Machine Passport system.
Frontend - Backend Interaction	The frontend requests backend access through dbConnectionParameterization(), ensuring parameters for data exchange are precisely established.
Backend - Database/Files Interaction	Upon successful authorization, the backend engages with the database/files, retrieving and sending necessary data for the Machine Passport operations.
Database/Files - Backend Response	Data, once fetched, is processed and returned to the backend, adhering to predefined operational rules and parameterizations.

Backend - Frontend Response	The backend evaluates the data against the Machine Passport's analytical model and sends the processed information back to the frontend.
Frontend - User Interaction	The frontend presents the user with comprehensive insights and reports derived from the Machine Passport's data analysis, aiding in decision-making.
Completion	The Machine Passport component concludes its current sequence and stands by for subsequent user-initiated actions, ready to facilitate further data-driven operations.

Table 177. Life-Cycle description Database Connection AI[MP]

4.16.7 Implementation Viewpoint

The architecture of the Machine Passport is a comprehensive framework designed to facilitate the flow and processing of manufacturing data across different stages of the product lifecycle in a secure, efficient manner.

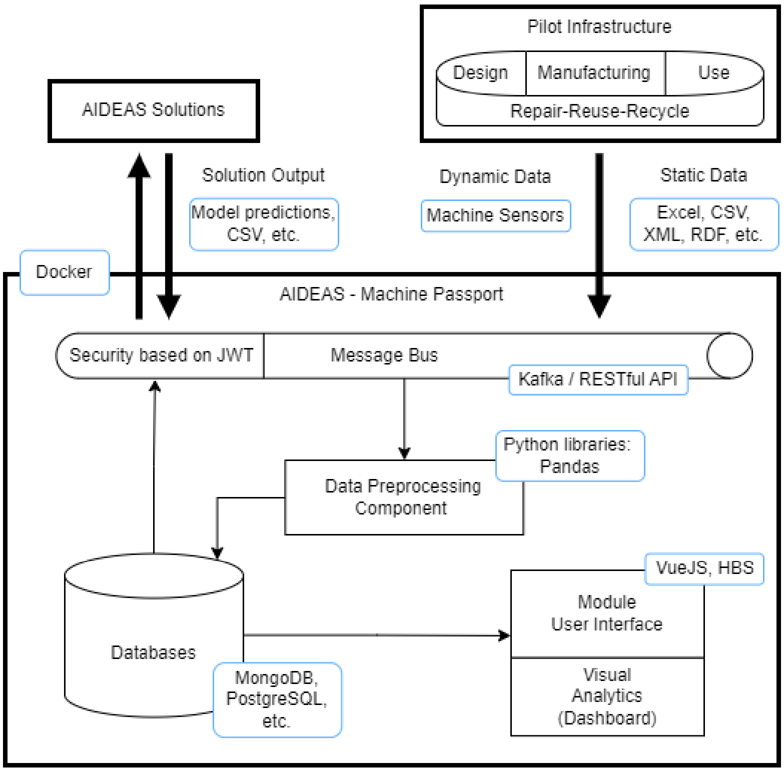


Figure 74. AI[MP] Implementation Architecture

At the core of the Machine Passport lies the AIDEAS Solutions, which are modular services running in a Docker environment, ensuring portability and scalability. These services generate valuable outputs like model predictions, which are essential for decision-making processes.

Data is at the heart of the Machine Passport, with dynamic data sourced from machine sensors during the equipment's operational phase and static data such as Excel, CSV, XML, RDF files, encompassing the Design, Manufacturing, Use, and Repair-Reuse-Recycle stages of the product lifecycle within the Pilot Infrastructure. The architecture emphasizes security with JWT (JSON Web Tokens) to safeguard communications. An intermediary message bus, possibly a Kafka stream or

RESTful API, orchestrates the data flow, allowing for asynchronous data processing and integration.

The Data Harmonizer will be introduced as middleware between the AIDEAS Machine Passport Backend and the AI Solutions to ensure that communications follow the predefined data schema for each solution and adhere to the standard. The Data Preprocessing Component is a pivotal element that utilizes Python libraries, notably Pandas, to clean, transform, and prepare data for analysis, ensuring that the data fed into the AIDEAS Solutions is of high quality and ready for processing.

Data storage is managed by robust, scalable databases such as MongoDB and PostgreSQL, which store the vast amounts of manufacturing data securely. Lastly, the architecture is rounded off with a Module User Interface, implemented with NodeJs and HBS, providing an intuitive visual analytics dashboard for users to interact with the Machine Passport. This user interface is not just for data visualization but also for interacting with the AIDEAS Solutions, allowing users to input parameters, start data processing tasks, and view the results of analyses in an accessible format.

Overall, the Machine Passport architecture is designed to be a secure, robust, and flexible ecosystem that supports the complex data requirements of modern industrial manufacturing environments, leveraging smart data integration and advanced analytics to deliver actionable insights throughout the entire product lifecycle.

4.16.7.1 AI^{MP} Implementation Components

Implementation Components	Description
Front-End	This facilitates the User Interface of the MP and all the appropriate actions.
Back-End	This facilitates all the endpoints, connections, security and models to communicate with the available components.
Database	Local storage of the Machine Passport to store valuable knowledge coming from solutions or other data sources.
Data Harmonizer	This facilitates data validation for the communication between the AI Solutions and the Machine Passport Backend

Table 178. AI^{MP} Implementation Components

4.16.7.2 Technical Description of AI^{MP} Components

Following the identification and specification of the implementation components in the previous section, the technical description of each component is provided in the following tables. It details the set of technologies required to implement each of the implementation components.

Implementation component	Front-End
Description of implementation	User interface of Machine Passport
Used technologies	NodeJS, HTML, JavaScript. CSS

Technical Description of the Component	Dependencies
	Development Language: JavaScript Libraries: HBS, Bootstrap, JQuery, PlotlyJS, DataTables, XLSX Container: Docker Database: MongoDB
	Interfaces
	User Interface: Yes Synchronous/Asynchronous Interface: Asynchronous Network/Protocols: HTTP/HTTPS Data Repository: MongoDB

Table 179. Technical Description of AI^{MP} "Front-End" Implementation Component

Implementation component	Back-End
Description of implementation	Back-End of Machine Passport
Used technologies	NodeJS, Javascript
Technical Description of the Component	Dependencies
	Development Language: JavaScript Libraries: ExpressJS, Mongoose, jsonwebtoken, Nodemon Container: Docker Database: MongoDB
	Interfaces
	User Interface: Yes Synchronous/Asynchronous Interface: Asynchronous Network/Protocols: HTTP/HTTPS Data Repository: MongoDB

Table 180. Technical Description of AI^{MP} "Back-End" Implementation Component

Implementation component	Database
Description of implementation	Local repository of Machine Passport
Used technologies	NodeJS, JavaScript
	Dependencies
	Development Language: JavaScript Libraries: Mongoose, ExpressJS, Nodemon

Technical Description of the Component	Container: Docker Database: MongoDB	
	Interfaces	
	User Interface: Yes Synchronous/Asynchronous Interface: Asynchronous Network/Protocols: HTTP/HTTPS Data Repository: MongoDB	

Table 181. Technical Description of AI^{MP} "Database" Implementation Component

Implementation component	Data Harmonizer
Description of implementation	Middleware between the AIDEAS Machine Passport Backend and the AI Solutions
Used technologies	NodeJS, JavaScript
Technical Description of the Component	Dependencies
	Development Language: JavaScript Libraries: ExpressJS, Mongoose, jsonwebtoken, Nodemo, Ajv, axios, MinIO, Multer Container: Docker Database: MongoDB, MinIO
	Interfaces
	User Interface: Yes Synchronous/Asynchronous Interface: Asynchronous Network/Protocols: HTTP/HTTPS Data Repository: MongoDB

Table 182. Technical Description of AI^{MP} "Data Harmonizer" Implementation Component

5. Conclusions

Throughout this deliverable, the methodology and approaches used in AIDEAS have been examined in detail. From the business perspective to the technical implementation, every aspect of the project has been designed to maximise the value and effectiveness of the solutions developed.

In particular, the business-focused approach has been crucial in ensuring that the AIDEAS solutions are aligned with real market needs and objectives. The early incorporation of business requirements and needs has helped to mitigate the risk of technology-centric development, thereby ensuring value creation for both organisations and society. Close collaboration with experienced business partners has further enriched this approach, providing valuable insights and real-world perspectives.

On the other hand, the user-centred approach has enabled the definition of effective implementation strategies for AIDEAS solutions. By clearly identifying tasks, roles, activities and components within AIDEAS, the creation of functional and implementation maps has been facilitated, ensuring a smooth and efficient execution of solutions. In addition, the use of UML activity diagrams has provided a clear visual representation of system usage and activities, improving understanding and communication between all stakeholders.

From a functional perspective, a careful breakdown of the AIDEAS solutions into distinct functional components within different functional domains was performed. This perspective allowed the identification of key interfaces and the understanding of complex interactions between different solution components. Furthermore, the definition of operational requirements, data structures and hardware/software requirements has laid the foundation for effective solution design and implementation.

Finally, the implementation-focused approach provided a detailed technical representation of the AIDEAS solutions. By identifying the necessary implementation components and specifying the required technologies, the successful and efficient implementation of the solutions in the real world was ensured. The detailed architecture of each solution, together with the technical description of the associated components and technologies, has laid the foundation for the development of robust and scalable solutions.

As the project progresses into the implementation and deployment phase, maintaining this comprehensive and collaborative approach is critical to addressing emerging challenges and seizing future opportunities in the business and technology landscape.

Appendix I

Survey for the Pilots/Value Proposition

SURVEY INSTRUCTIONS

Dear pilots,

As mentioned during the plenary in Valencia we will be organising a Value Proposition Workshop in the coming weeks (tentatively the week of December 11).

To help this process we need your insights as the first testers of the AIDEAS solutions. We need to help the technology providers understand if what they are developing correctly addresses your needs, problems, and expectations. Doing this will help us maximise the chances of success when bringing the AIDEAS solution to the market.

Therefore, we need you to:

Try to forget what you already know about the individual solutions that are part of each KER/Suite, and tell us what your real problem is at each phase of the lifecycle (Design, Manufacturing, Use, RRR) and also with regards to the Machine Passport. We would like to emphasize on this aspect, even though abstract, to try and focus on a simple way of understanding your needs.

You can have a look at this very short video to understand more about the Value Proposition. We want you to help us fill in the right side of the canvas (the Customer Profile), so the partners together can work on the left side of it (the Value Map) during the workshop in December. Put your first insights into this survey. You'll find questions related to your jobs-to-be-done, pains, and gains in the context of the AIDEAS KERs. There is guiding information to help you complete it.

The survey consists of 5 sections; 1 for each KER/Suite + the Machine Passport. Please fill out each segment specific to each Suite, in order for us to understand the pains, gains and jobs-to-be-done. Since not all Pilots are interested in all KERs/Suites, if you consider a certain Suite is not for you, just indicate it with N/A.

IMPORTANT: Please fill it before Friday, December 1st

To fill this appropriately make sure the person that is confronted with the problems that the AIDEAS solutions are trying to solve is the person that answers to this survey.

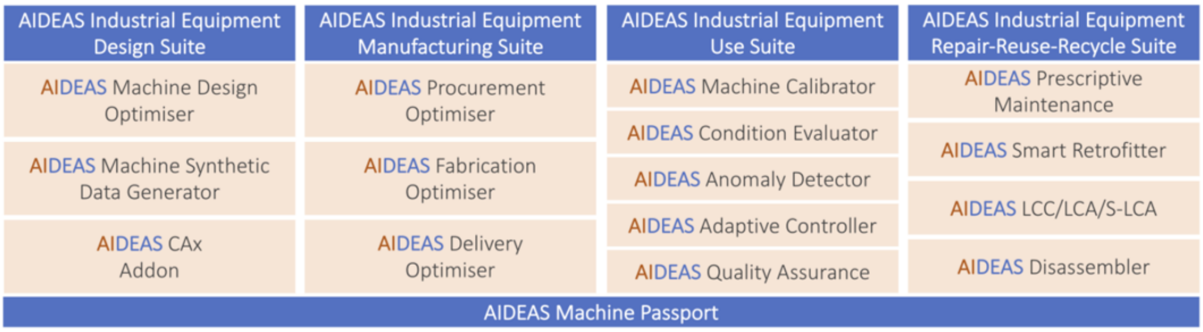
Don't hesitate to contact me if you have any questions: alejandro.dominguez@fundingbox.com

To bring the most market driven solution to the market, meaning with high potential, we need to understand the perspective of the users: YOU. Your feedback is essential to support the technology providers in shaping the best solution possible. This survey is just a preparation before the Value Proposition Workshop.

Note: at the bottom of each part to be answered the are a set of questions or ideas that could serve as an inspiration for you.

Name	
Which Pilot do you represent?	Multiscan/D2Tech/BBM/PAMA

For each Suite:



Question	Open Answer
Jobs to be done	
Pains	
Gains	
Alternative Solution	

Jobs-to-be-done: what are you trying to do? What need or problem are you and other bad experiences that you may experience before, during, or after the jobs to be done.

trying to satisfy? (always from your point of view as a user).

Please write them in the order of importance⚠ Job#1:

Job#2: Job#3: Job#...

- What functional tasks are you trying to accomplish? (eg, problems you need to solve, tasks you need to complete, ...)
- What social tasks are you trying to accomplish? (eg, look good on tape, gain status, be promoted, ...)
- What emotional tasks are you trying to accomplish? (eg aesthetics, feel good, safety, ...)
- What basic needs do you want to satisfy? (eg communication, hygiene, ...)

Pains: Describe the negative emotions, costs and unwanted situations, risks,

Please write them in the order of importance⚠ Pain#1:

Pain#2: Pain#3: Pain#...

- What do you think is very expensive? (ex: it takes a lot of time, it costs a lot, it requires a lot of effort, etc.)
- What makes you feel bad? (e.g. frustrations, annoyances, things that give headaches, etc.)
- How current solutions are leaving to be desired for you? (e.g. lack of functionality, performance, defects, etc.)
- What are the main difficulties and challenges that you encounter? (e.g., understanding how certain things work, difficulties in performing tasks, resistances, etc.)
- What are the negative consequences? What negative social consequences does you encounter or fear? (e.g., loss of face, power, trust, or status, ...)
- What risks are you afraid of? (e.g. financial, social, technical or what could go very wrong?)
- What is keeping you awake at night? (e.g. major issues, concerns, challenges, etc.) What are the common mistakes you make? (e.g., errors in use, understanding, expectation, etc.)
- What barriers are preventing you from adopting solutions (e.g., initial investment, learning curve, resistance to change, etc.)

Gains: Benefits that you expect, desire or would be positively surprised if they* existed. This includes, among others, functional utility, social gains, positive emotions, and cost reduction.

Please write them in the order of importance⚠ Gain#1:

Gain#2: Gain#3: Gain#...

- What ways to save money would make you happy? (eg in terms of time, money, effort, etc.)
- What results do you expect to have and what would go beyond their expectations? (eg, quality level, more than something, less than something else, etc.)
- What current solutions enchant you? (e.g. specific functionalities, performance, quality, etc.)
- What would make your tasks easier? (e.g., lower learning curve, more services, lower cost of ownership, etc.)
- What positive consequences do you want? (e.g., increase power, status, etc.) What are you looking for? (e.g., beautiful design, guarantees, more specific functionalities, etc.)
- How does your client measure success and failure? (e.g. cost, performance, likes on social networks, etc.)
- What would increase your client's chances of adopting a solution? (e.g. lower cost, lower investment, more guarantee, performance, design, etc.)

Alternative Solution: What are the current solutions on the market that solve the *

problem you have? Or how are you solving this problem now?

Survey for External Stakeholders

SURVEY INSTRUCTIONS

Dear AIDEAS Stakeholder,

We would like to invite you to give us feedback on your vision and point of view of the solutions we are developing within our AIDEAS project.

About the project:

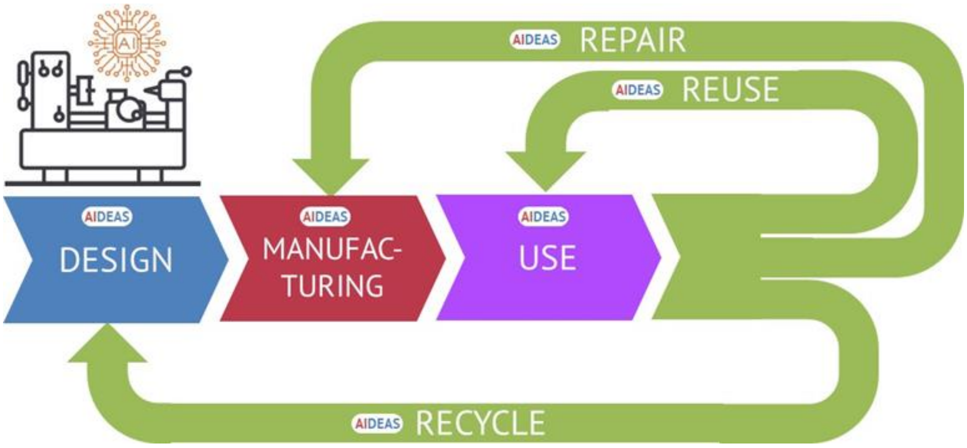
Within the AIDEAS Project, we are developing AI technologies for supporting the entire life cycle of industrial equipment (design, manufacturing, use and repair/reuse/recycle) as a strategic instrument to improve sustainability, agility and resilience of the European machinery manufacturing companies.

A few words about this survey:

In a few months, we will launch a "Beta Testing Campaign" to potentially interested users, like you, who would like to get a closer look at What is done within our project, and even engage in discussions to understand How it can directly solve your daily problems. But before we reach that, we would like to get your opinion to try and steer the development of our solutions to match your needs as much as possible. To do so, we kindly ask you to complete this survey.

In any case, if you have any doubts, do not hesitate to contact us at: nikoskourk@iti.gr, ines.dinten@fundingbox.com, or alejandro.dominguez@fundingbox.com

Thank you very much, The AIDEAS Team



A few details about you and your organization

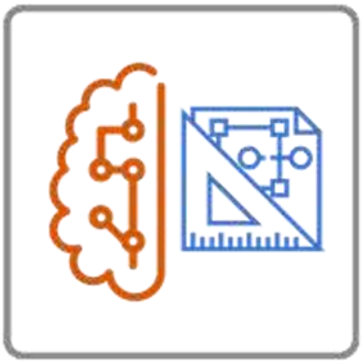
First of all, we would like to know just some basic information about you. It will help us later on with the analysis of the results of the survey and will allow us to clarify anything, if necessary. This information will not be shared outside the AIDEAS consortium and will solely be used for the purpose here indicated.

Your Name	
Your Organization	
Sector of your organization	
Your Position/role within your organization	
Your email address	

AIDEAS Industrial Equipment Design Suite

Transform the way you design industrial equipment with our cutting-edge, AI-powered tools. Our solution revolutionizes the traditional, labor-intensive design process, enabling your business to effortlessly generate innovative product designs that are not only cost- effective but also highly resilient. With this toolkit, you'll experience a significant reduction in waste and an enhanced ability to adapt to the ever-changing needs of your customers.

Our toolkit simplifies and streamlines the design process, ensuring that even manufacturers facing a skills gap or resource shortage can produce reliable, customer- tailored solutions with ease. By integrating our AI-assisted technology into your workflow, you'll overcome the challenges of manual design methods and set a new standard in machine construction. This is more than just a design tool; it's a pathway to greater efficiency, resource optimization, and a future where your products are perfectly aligned with market demands.

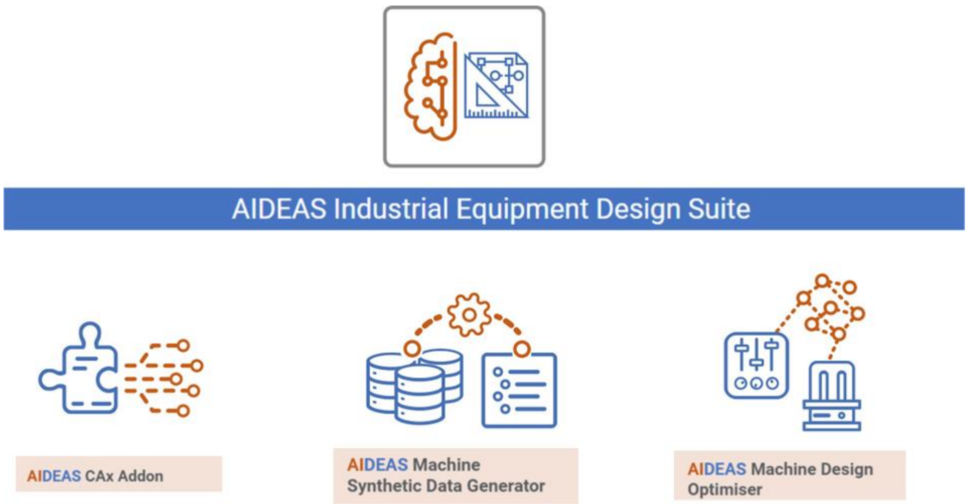


Are you interested in this Suite?

Yes
No
At the moment I am still not sure. I need more insights and information

Design Suite Solutions

Please, have a look at the individual technical solutions that make part of this Suite, and tell us if you are interested in them.



CAx Addon:
*A set of APIs and UIs supporting the integration of AI-assisted optimisation modules into CAx systems.

Are you interested in this solution?

Yes
No
At the moment I am still not sure. I need more insights and information

Machine Synthetic Data Generator:

Toolkit for synthesising large high-quality datasets by simulations for the analysis of the machine design and for the training of the optimisation algorithms that will propose optimal design parameters.

Are you interested in this solution?

Yes
No
At the moment I am still not sure. I need more insights and information

Machine Design Optimizer:

Toolkit to assist designers in optimally defining the key design parameters in multi-physical systems, enhancing machine performance as required for each scenario, through AI.

Are you interested in this solution?

Yes
No
At the moment I am still not sure. I need more insights and information

We want to understand your problem better regarding the Design Suite

Please, tell us a bit more about your problem concerning the Design Suite. * Every indication you will provide us, such as the intended % of improvement, current situation or ideal KPIs that you would like to achieve will help us adapt our solution to better fit your needs.	

AIDEAS Industrial Equipment Manufacturing Suite

Our advanced set of tools is designed specifically for manufacturing companies seeking to revolutionize their production processes. This suite empowers your business to make informed decisions in real-time or near real-time, optimizing every stage from procurement to delivery.

Experience unprecedented agility in your operations, even amidst extreme complexity. Our AI-driven solutions enable a more predictable and efficient manufacturing phase, ensuring that product delivery schedules are accurately forecasted and significantly accelerated.

This leads to a faster market reach, giving your company a competitive edge.

Unlike conventional commercially available software, our toolkit is tailored to streamline procurement, fabrication, and delivery plans, using intelligent algorithms that adapt to your unique manufacturing challenges. This is not just about keeping pace with industry trends; it's about setting new standards in efficiency and speed, ensuring that your business stays ahead in a fast-evolving market.

Are you interested in this Suite?

Yes
No
At the moment I am still not sure. I need more insights and information

Manufacturing Suite Solutions

Please, have a look at the individual technical solutions that make part of this Suite, and tell us if you are interested in them

Delivery Optimiser:

AI-based toolkit that is capable of optimising the storage and delivery of products. This optimisation will target storage space, storage conditions and product transportation. Additionally, this optimiser will provide optimisation for logistics scheduling and planning.

Are you interested in this solution?

Yes
No
At the moment I am still not sure. I need more insights and information

Fabrication Optimiser:

Toolkit for optimising production scheduling and resource allocation by predicting production and set-up times, operations dependencies, etc. allowing a near real time response to environment changes like machine breakdowns, last minute customer orders and raw materials delays, through AI.

Are you interested in this solution?

Yes
No
At the moment I am still not sure. I need more insights and information

Procurement Optimiser:

Toolkit for optimising the inventory and purchase of materials and components that are required to build a machine and meet customer delivery dates using AI.

Are you interested in this solution?

Yes
No
At the moment I am still not sure. I need more insights and information

We want to understand your problem better regarding the Manufacturing Suite

Please, tell us a bit more about your problem concerning the Manufacturing Suite. * Every indication you will provide us, such as the intended % of improvement, current situation or ideal KPIs that you would like to achieve will help us adapt our solution to better fit your needs.	

AIDEAS Industrial Equipment Use Suite

Our suite marks a new era in industrial technology by ensuring optimal installation, calibration, and adaptive control of industrial equipment. Combining precision with efficiency, our offerings revolutionize machine performance and initial setup. This toolkit provides a level of quality assurance that surpasses traditional methods with minimal data input, automating visual inspections and guaranteeing zero defects.

It also delivers rapid machinery condition evaluation and early anomaly detection, surpassing conventional monitoring systems.

This comprehensive approach not only streamlines manufacturing processes but also supports a guarantee of product quality and defect-free production, ensuring heightened operational efficiency and reliability.

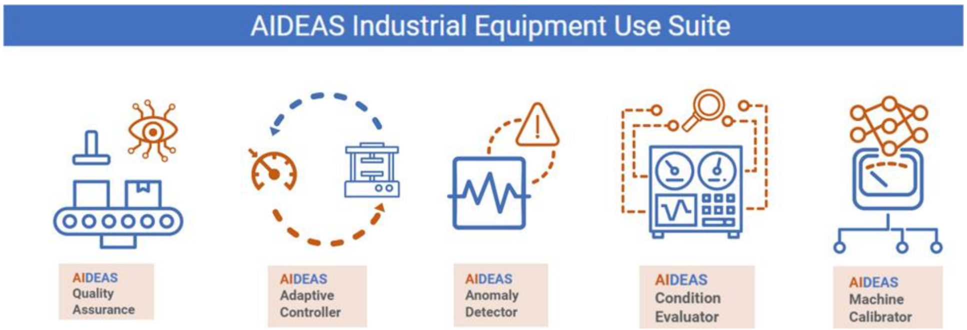
Are you interested in this Suite?

Yes

No
At the moment I am still not sure. I need more insights and information

Use Suite Solutions

Please, have a look at the individual technical solutions that make part of this Suite, and tell us if you are interested in them.



Quality Assurance:

Toolkit comprising a set of AI-enabled features for manufactured product quality monitoring.

Are you interested in this solution?

Yes
No
At the moment I am still not sure. I need more insights and information

Adaptive Controller:

Toolkit to train models with measurement data and then train machine controllers with said models to accommodate the machine condition and requirements.

Are you interested in this solution?

Yes
No
At the moment I am still not sure. I need more insights and information

Anomaly Detector:

Toolkit that will allow detecting anomalies at a component level or of the machine as a whole when it is in working conditions in the factory where it is being used.

Are you interested in this solution?

Yes
No
At the moment I am still not sure. I need more insights and information

Condition Evaluator:

Toolkit for determining the condition of the machine as a whole or of some of its components when it is in working conditions in the factory where it is being used.

Are you interested in this solution?

Yes
No
At the moment I am still not sure. I need more insights and information

Machine Calibrator:

Toolkit for the fast calibration of industrial equipment when installed for the first time in a factory or when a re-calibration is needed. It uses AI techniques to provide the most well-suited calibration parameters.

Are you interested in this solution?

Yes
No
At the moment I am still not sure. I need more insights and information

We want to understand your problem better regarding the Use Suite

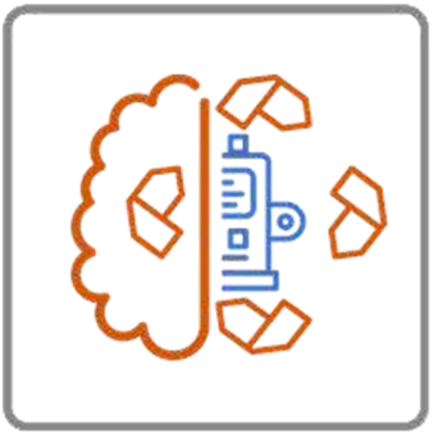
Please, tell us a bit more about your problem concerning the Use Suite. * Every indication you will provide us, such as the intended % of improvement, current situation or ideal KPIs that you would like to achieve will help us adapt our solution to better fit your needs.	

AIDEAS Industrial Equipment Repair-Reuse-Recycle Suite

Our integrated solution offers a comprehensive approach to enhancing the lifecycle and efficiency of industrial equipment, targeting maintenance and production managers. This system optimizes machine parameters and resource usage (e.g., water or energy) around the machine. It enables precise control and behavior monitoring of the equipment, effectively extending its operational life and reducing the need for new purchases.

Additionally, our solution supports data-driven decision-making, allowing managers to proactively maintain and optimize machinery, rather than passively allowing wear and deterioration.

Complementing these capabilities, our AI-based solutions foster a circular production model, focusing on repairing, reusing, and recycling industrial equipment. This approach not only maximizes productivity and the residual value of materials but also emphasizes recycling over downcycling, aligning with sustainable and efficient manufacturing practices. Together, these features create a powerful tool for businesses looking to enhance equipment longevity, optimize resource usage, and support environmentally responsible production methods.



Are you interested in this Suite?

Yes
No
At the moment I am still not sure. I need more insights and information

Repair-Reuse-Recycle Suite Solutions

Please, have a look at the individual technical solutions that make part of this Suite, and tell us if you are interested in them

Disassembler:

AI-based toolkit for modelling the disassembly/recycle processes to help streamline the infrastructure needed to circulate materials focusing on the ability for AI algorithms to recognise and identify objects using cameras and other sensors.

Are you interested in this solution?

Yes
No
At the moment I am still not sure. I need more insights and information

LCC/LCA/S-LCA:

Toolkit that combines AI and Life Cycle methodologies (LCC, LCA, S-LCA) for identifying the best machine end-of-life by devising a multi-objective optimisation strategy to strike a balance between economic, social and environmental benefits.

Are you interested in this solution?

Yes
No
At the moment I am still not sure. I need more insights and information

Smart Retrofitter:

Toolkit for smart retrofitting old machine tools to give them a second life by improving working conditions and product quality, developing a communication system and collaboration, enhancing productivity, efficiency, flexibility, and agility.

Are you interested in this solution?

Yes
No
At the moment I am still not sure. I need more insights and information

Prescriptive Maintenance:

Toolkit for predicting remaining useful life and identifying maintenance requirements with the target of extending the overall machine remaining life.

Are you interested in this solution?

Yes
No
At the moment I am still not sure. I need more insights and information

We want to understand your problem better regarding the Repair-Reuse-Recycle Suite

Please, tell us a bit more about your problem concerning the Repair-Reuse-Recycle Suite. * Every indication you will provide us, such as the intended % of improvement, current situation or ideal KPIs that you would like to achieve will help us adapt our solution to better fit your needs.	

AIDEAS Machine Passport

Our innovative platform features a comprehensive digital data repository supported by a smart management system, catering to all stakeholders in the machine life-cycle. This unique solution offers seamless access to machine data and AIDEAS solutions knowledge, enhancing AI capabilities and enabling interoperable data access across various phases of industrial equipment's life. Unlike traditional paper-based or current siloed digital data sources, our platform supports large-scale data acquisition, management, and sharing among different devices and parties involved.

From the design phase, where it offers suggestions for optimal machinery construction, to the manufacturing phase, where it aids the supply chain (suppliers, manufacturers, and customers), our solution is multifaceted. During the use phase, it allows for optimal calibration of CNC parameters and monitors the functionality of machine components and product quality. Finally, in the crucial repair-reuse-recycle phase, it facilitates data exchange between end-of-life parties, including consumers, repair shops, and waste management companies. This integrated approach not only streamlines processes but also ensures a more sustainable and efficient lifecycle for industrial equipment.

AIDEAS Machine Passport

Are you interested in this Suite?

Yes
No
At the moment I am still not sure. I need more insights and information

We want to understand your problem better regarding the Machine Passport

Please, tell us a bit more about your problem concerning the Machine Passport Suite. * Every indication you will provide us, such as the intended % of improvement, current situation or ideal KPIs that you would like to achieve will help us adapt our solution to better fit your needs.	

Feedback on the way of presenting the solutions.

What do you think is the easiest way of understanding the value AIDEAS could bring to you?

I understand it better when it is presented by use case / suites (e.g. Design, Manufacturing, Use, Repair-Reuse-Recycle or Machine Passport)
I prefer to see it by individual technical solution (e.g. Machine Calibrator, Smart Retrofitter, etc.)
In my case, a combination of different technical solutions from different stages of the lifecycle (suites) is what better addresses my problems
It is still complex, and I would prefer to have assistance by one of the AIDEAS technical experts to better understand how this can solve my problems

Anything else that you would like to add or share with us?	