Production harmonizEd Reconfiguration of Flexible Robots and Machinery

Horizon 2020 – Factories of the Future, Project ID: 680435

Deliverable 3.3

Human Expert Knowledge Capture and Interpretation

Lead Author: LBORO

Version: 0.3
Date: 14.06.2017
Status: Draft
Dissemination level: PUBLIC

Version	Date	Content
0.1	08.05.2017	Draft table of contents
0.2	31.05.2017	Compilation of the contributions from partners
0.3	14.06.2017	First draft of the deliverable. Refined text from partners
1.0	30.06.2017	Final version

Author List:

Phil Ogun (LBORO)
Niels Lohse (LBORO)
Paola Fantini (POLIMI)

Abstract

The increasing need to make products that fulfil customer demands is driving manufacturing towards shorter and more flexible production cycles. Short production cycles lead to more frequent changeovers, so many manufacturers are interested in reducing the overall time. Operators approach changeover in different ways depending on their experience level and it has been established that the lack of knowledge capturing and sharing is a major cause of variable and prolonged changeover.

The work reported in this deliverable (D3.3) is focused on the development of an effective method for capturing and interpreting human expert knowledge during changeover. The work draws from both the outputs of past European research projects and the PERFoRM project. Previous research projects have focused mainly on the development of scientific concepts but the focus in PERFoRM Task 3.3 is on the development of a system that can be deployed on the shop floor.

Firstly, a formal model of a changeover process is created. The model is then used to formulate the requirements a human observation capturing and interpretation system. The PERFoRM data model defined in D2.3 is extended in order meet the specified requirements. A decision support system, which consists of a database, machine learning logic and a web user interface has been developed. During changeover, events, actions and observations are recorded by operators in a contextual, structured and machine interpretable format. The goal is to use the self-adjusting policies derived from the gathered data to guide the operators in making decisions during future changeover operations.

One of the main reasons for embarking on this work at this stage is to create a solution that aligns with the vision of plug-and-produce concept. The idea is for existing and newly introduced machines on the shop floor to communicate information on how to set them up, their observable states and the possible adjustments that can be made to them. The exchanged data and the additional product-specific data gathered from experienced operators can be used to assist less experienced operators during set-up and ramp-up. This will guarantee a much more consistent and efficient changeover process.

Table of Contents

List of Figures

List of Tables

Acronyms

Abbreviation	**Meaning**
SPA	Single Page Application
JWT	JSON Web Token
RDBMS	Relational Database Management System
SOP	Standard Operating Procedures
UML	Unified Modelling Language
WP	Work Package
XML	Extensible Markup Language

1. Introduction

1.1 Structure of the Report

This report contains the outcome of Task 3.3, titled “Human Expert Knowledge Capture and Interpretation”. There are five sections in the report. Section 1 is the introductory section, which contains problem definition, scope of work and the objectives of this report. Section 2 contains requirements specification and system modelling. The details of the software application development are presented in section 3. Class diagrams and state diagrams are used to explain the details of the system. The details of the implementation of the front end (user interface) and the back end (business logic and data storage) of the human machine interface software are also presented in the section. A study was conducted in Whirlpool on operator knowledge capturing, this is documented in section 4.

1.2 Problem Definition

In today’s manufacturing, the proliferation of product variants necessitates shorter production runs, which leads to more frequent changeovers. Rapid increase in product quality and quantity demand increases the pressure to maximise capacity utilisation. Therefore, a quick changeover process is very critical to the production of small batch sizes of a large diversity of products without loss in productivity. As manufacturers are working to get products to market much faster and cheaper, self-learning solutions are needed to enable quick adaptations and flexibility without the need for many highly-trained engineers and operators.

Studies have shown that the effectiveness of a changeover process depends largely on the knowledge of the person carrying out the changeover on the shop floor (technicians, operators, inspectors etc). In the absence of standard procedural guidelines, operators use their judgment and experience to select the best sequence of actions for the changeover process. Human involvement in the decision making process introduces variablities, which make the process to be stocahastic and unpredictable. A new operator can be trained by observing a more experienced colleague. However, great deal of information and clarity can be lost during the training process, which will lead to poor practices. The changeover process will vary from person to person and time to time.

In most cases, human expert knowledge about production systems and resources is often tacit and is at best expressed in natural language format. Knowledge about a system that is not represented in a structured data format cannot be fully exploited. Unstructured data is difficult to analyse and current data mining and learning techniques often miss substantial amount of useful information embedded in such data. This restriction has been a problem in real industrial environments particularly during changeover and other related activities such as new equipment introduction and maintenance.

Self-adaptation has become an increasingly important capability of many systems especially the ones operating in dynamic manufacturing environments [1] - [12]. Such systems must be able to make data-driven decisions and predictions from existing causal relationships through the building of various machine learning models. The Fostering Human Rights Among European Policies (FRAME) project funded under the Seventh Framework Programme has shown that historical data can be used to support ramp-up process through self-adjusting policies derived from past ramp-up cases [3] - [5]. Similar machine learning decision support models for production ramp-up are also available [1], [2], [6], [7].

In product changeover situation, sensory feedback data from machines is often limited because the operations are mainly carried out by human operators and it has been established that lack of knowledge capturing and sharing is a major cause of variable set-up and prolonged ramp-up process [4], [8], [12]. Therefore, the focus of this task is on the development of an effective method for capturing and interpreting human expert knowledge during product changeover process. This involves documenting events, actions and observations in a structured and machine interpretable format. The goal is to leverage the data by using it to formulate more effective strategies that will help in reducing total changeover time. As machine learning models are built on sample input data, the captured human expert knowledge becomes a valuable source of training, testing and exploration data.

1.3 Scope of Work

In the literature, there are some confusion in the meanings of terminologies such as changeover, set-up, start-up, run-up and ramp-up. In some cases, changeover and set-up are used interchangeably and in some other cases, set-up is viewed as a component of changeover. Therefore, it is necessary to define the terms as used in this report.

Changeover: Changeover is defined as the total process of converting a production line or machine from one product run to another, which comprises of clean-up, set-up and ramp-up operations [8]. The changeover process is illustrated in Figure 1.

- **Clean-up**: This is the removal of all previous materials, products or components from the line. It may involve minor or major operations such as cleaning of line components followed by sterilizing and disassembly of the equipment

- **Set-up**: This is the physical conversion and configuration of the equipment for the next product run. It involves adjustments or replacements of product-specific parts.

- **Ramp-up:** Ramp-up is also known as start-up or run-up**.** This is the process of fine-tuning the machine after it begins production but before settling down into normal operation or steady-state production. It is often characterised by frequent interruptions, jams, defective or marginal product and other things that prevent it from reaching full production capacity. The main causes of ramp-up time are variability in product or its components, and variability in clean-up or set-up.

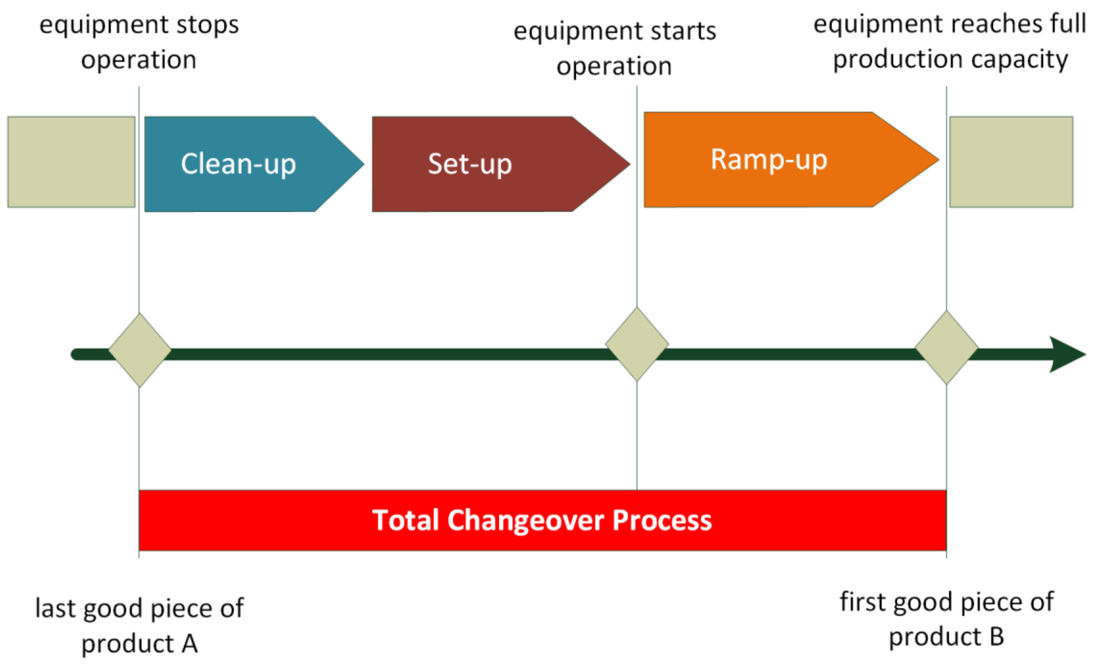

equipment stops operation
equipment starts operation
equipment reaches full production capacity
Clean-up
Set-up
Ramp-up
Total Changeover Process
last good piece of product A
first good piece of product B

Figure 1 Total changeover process

Apart from product changeover situation, ramp-up is also performed when a new equipment is introduced. Under new equipment installation situation, ramp-up is the process of bringing the equipment up to a desired production capacity. Ramp-up may also be necessary during the lifecycle of a production system, for instance when significant changes are required or more substantial breakdowns need to be recovered [9]. The set-up phase is also a component of new equipment introduction process. It is often referred to as the build phase.

Significant research efforts have been put into ramp-up time reduction during the introduction of new or adapted manufacturing systems [1] - [6]. However, studies have shown that although ramp-up occurs due to variations in product or materials, it is more frequently due to variations in the set-up and occasionally in clean-up phases [10]. Since set-up has a significant impact on ramp-up time, it needs to be controlled also. Whenever there is excessive variation in set-up and it is not controlled, it is difficult to determine whether ramp-up time variation is due to the variation in set-up or variation in product materials. Therefore, this task will focus on ramp-up and driving out variation in set-up. The clean-up phase is not addressed in this work as it does not relate directly to production machines.

Considering the number of machine learning and data mining applications to ramp-up that have been reported in the research community in recent years, the need for further work in this area seems unjustifiable at first. Although many research papers have identified the need to capture and analyse the decision-making process of experienced operators, none has developed a practical or tangible solution for shop floor use. The main aim of this task is to industrialise the outputs of past research projects by creating a decision support system for operators use during changeover.

1.3 Objectives of Task 3.3

The objectives of the task are as follows:

- Formalise product changeover process
- Create structural data and behaviour models for capturing and representing the functional requirements of the product changeover process
- Create a software for capturing human expert knowledge data
- Implement a self-learning inference engine that incorporates the human expert knowledge
- Conduct a case study of one of the industrial partners

2. Requirements Specification and System Modelling

The changeover process modelling involves creating the static structure and the behavioural diagrams of the system. The static structure diagrams describe the information elements that exist in the system, their internal structure and how the elements relate to one another. The behavioural diagrams describe the dynamic behaviour of the system, such as the sequence of actions that a user can perform and the possible states of the system elements as events occur. The model design is based on the following considerations.

2.1 Changeover Process Formalisation

2.1.1 Set-up

Scheduling problems involving set-up times are generally divided into two classes, namely sequence-independent and sequence-dependent set-up times. Set-up is sequence-dependent if the duration of setting up a machine for the next job depends on the job and the immediately preceding job, whereas sequence-independent set-up duration depends only on the next job. As shown in Figure 1, set-up is preceded by clean-up. Since the clean-up phase is not covered in Task 3.3, the set-up is formalised as a sequence-independent process. In this case, the process of setting up a machine for the next product run is dependent on the product only since everything relating to the immediately preceding product would have been cleaned up.

The physical conversion of a machinery for the next product run is achieved through adjustments or by replacing the product-specific parts of the machinery. Set-up may also include other operational tasks such as preparation of documentation, material movement from warehouse to production and quality inspection. However, our focus in Task 3.3 is on the tasks that are directly related to the production machinery.

In the industry, coordination is achieved during set-up by using standard operating procedures (SOP) and checklists. The SOP describes in details the actions to be performed, how to perform them and when they should be performed. A checklist is an abridged version of the SOP, which contains only a list of the actions without the details of how to perform them. The SOP is normally used by new operators with no special training, who will follow the step by step instructions to perform the set-up successfully. As operators acquire more experience, their reliance on the SOP will reduce and they will work more from memory. Under this situation, there is likelihood of deviating from standard practices. The purpose of the checklist is to ensure that all actions have been carried out in the proper order.

One of the goals in Task 3.3 is to create an electronic version of the SOP and checklists so that operators can access them on any computer and other similar devices on the shop floor. The electronic version provides ease of navigation, ease of updating, real-time recording of operator actions and higher level of security.

2.1.2 Ramp-up

A model for ramp-up process has already been developed in previous research work [2], [5], [9]. Although this model has been developed for new equipment introduction scenario, it is applicable to a changeover process. The model represents ramp-up as a finite state transition process consisting of a sequence of actions and observations. The actions are the specific adaptations and adjustments that are applied in order to move the machine from an initial state to a desired target state. The observations are the observable or measurable conditions of the machine, which may change in response to the applied actions. The state of the machine is the combination of all discrete parameter sets that are used to define the performance of the machine at any point in time. A schematic overview of the transition process is shown in Figure 2, where is s_i the state before an action is applied and s_{i+1} is the new state of the system. s_{ck} and s_{ck+1} are sub-states reflecting internal state transitions.

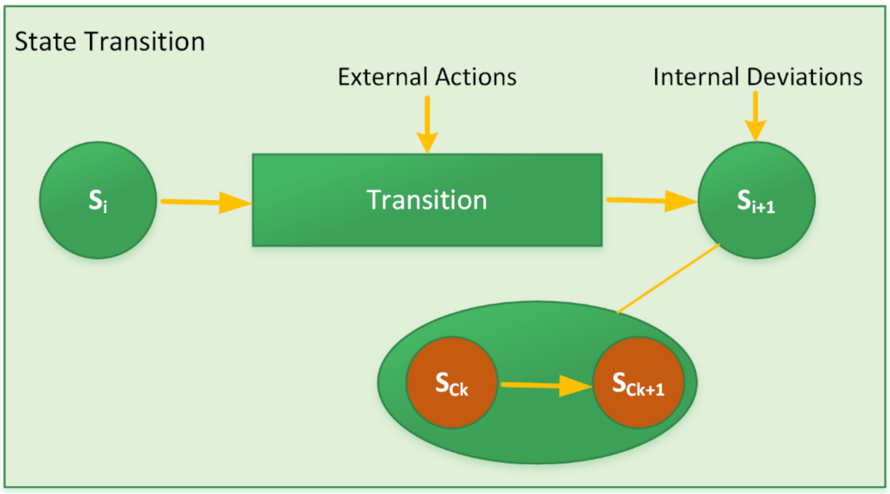


Figure 2 Finite state transition [5]

In order to capture the relationship between observed states and actions, the ramp-up process P_R is represented as a set of change sessions called experiences. The experiences are assumed to occur in a sequential order, .i.e. observations and actions are recorded in chronological order by an operator.

$$P_R = \{e_1, e_2 \dots \dots \dots \dots e_i\} \tag{1}$$

where *i* is the number of change sessions. The experience relates the state before (s_i) and after a change (s_{i+1}) with the action (a_i) carried out to effect the state transition. An experience is a tuple expressed as follows:

$$e_i = \langle s_i, a_i, s_{i+1} | \, S, A \rangle \tag{2}$$

This means that the system was in state s_i, the operator performed action a_i, and the system moved to state s_{i+1}. S is a set of all possible states of the machine ($s \in S$) and A is a set of all possible actions that can be performed ($a \in A$). A state s_i is expressed as follows:

$$s_i = \{p_1^i, p_2^i \dots \dots .. p_m^i \, |P\} \tag{3}$$

where P is the set of all the parameters that describe the system state ($p \in$ P) and p_j^i is the value of parameter *j* in state *i*. The capturing of all the different types of information requires parameter classification and value discretisation. In this work, the state parameters are classified into functionality, quality and optimisation information. A parameter-based state definition can result in an increasingly large number of unique states. For a system with *n* parameters and *m* discrete values for each parameter *p*, the dimension of the state-space model can be derived as follows:

$$State\ Space\ Dimension\ (n, m) = \prod_{i=1}^{n} p_i\ (m) \tag{4}$$

2.2 Descision Support System

Decision support and self-learning methods designed specifically for total product changeover process are almost non-existence in the literature. However, a few studies have been conducted on the development of various self-learning models to support decision making during production ramp-up [2] - [11]. The more recent approaches formulate ramp-up as a Markov Decision Process (MDP), which is solved using a reinforcement learning (RL) algorithm known as Q-learning [2], [9] and [10].

The Q-learning algorithm is applied in this task. The technique works by learning an action-reward function that generates the most optimal policy for ramping-up a machine based on previous ramp-up experiences. One of the advantages of Q-learning is that it is model-free, so it does not use a model of the environment to learn the actions that give the best rewards. Q-learning model is composed of the following:

S: Set of all possible states of the environment

A: Set of all possible actions that can be taken in order to change the environment state

R: Reward function, which provides a numerical reward for an action taken in a specific state

Q: State-action value, which is the memory of what the agent has learned through experience. It is expressesd as:

$$Q(s_i, a_i) = R(s_i, a_i) + \gamma \left(max_{a_{i+1}} Q(s_{i+1}, a_{i+1}) \right) \qquad (5)$$

where

- *R(s, a)* is the immediate reward received for taken an action in a given state
- γ is the discount factor, which represents the difference in importance between future rewards and immediate rewards ($0 \leq \gamma < 1$)

The state-action value is updated iteratively for all episodes and experiences during learning to obtain an optimal state-action value (Q^*). A policy is derived by simply following the actions with the highest values of Q^* in each state.

The ramp-up learning process is illustrated in Figure 3. In the first loop, an operator performs ramp-up actions without any decision-making support. The operator observes the state of the machine (system performance) and then decides on the next action to be performed based on the observed state. During this period, the RL system (2nd Loop) monitors the decision-making process of the operator. The data collected from the process are used to learn a model and the model’s state values. Based on that, state-action policy that targets reward maximisation is generated. The policy can then be used to suggest actions to operators based on the observed state during subsequent ramp-up operations.

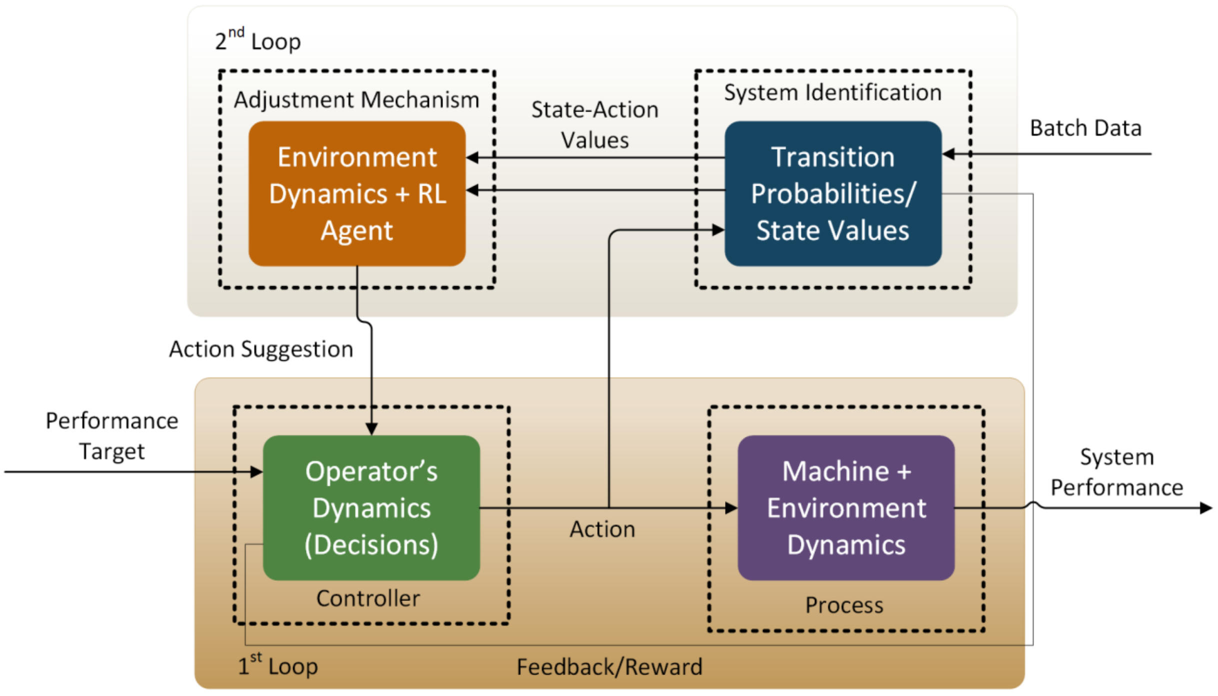


Figure 3 Reinforcement learning during ramp-up [2]

The goal of the decision support system is to search through past state-transition paths to find the most rewarding combination of state-actions that provide an optimal path (Figure 4).

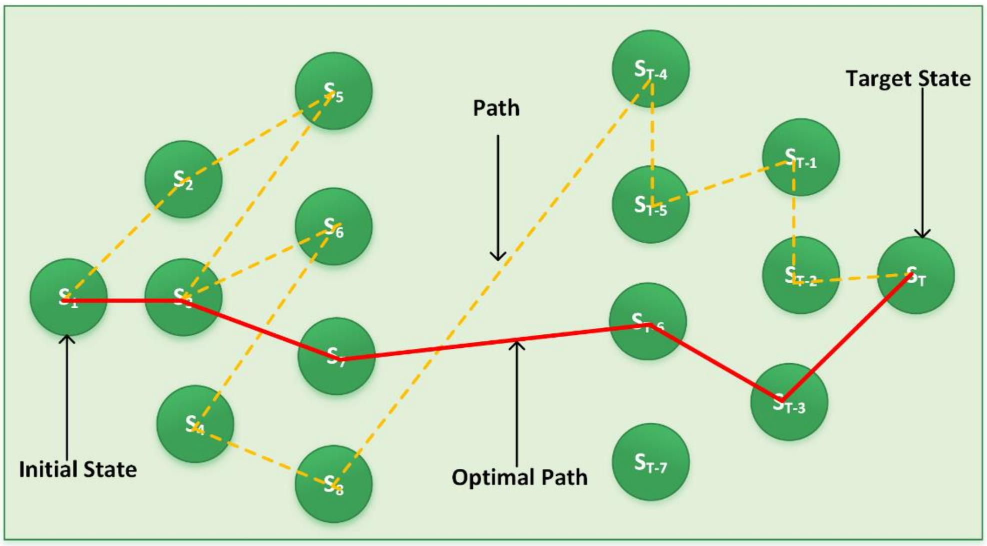


Figure 4 State transition optimisation during production ramp-up [9]

2.3 Context Mapping

The human knowledge capturing and interpretetation system will act on two types of data. The first sets of data relates to the equipment, products and processes records. These records are preloaded into the database and are presented to the operator during a changeover operation. The second sets of data relates to the actions and observed parameters during operation. The two main sources of parameters are machine sensors and human operator observations [3]. In order to establish causality between events, actions, and observations, in relation to the products, equipment and processes, releveant contextual information needs to be captured. The context mapping is illustrated in Figure 5.

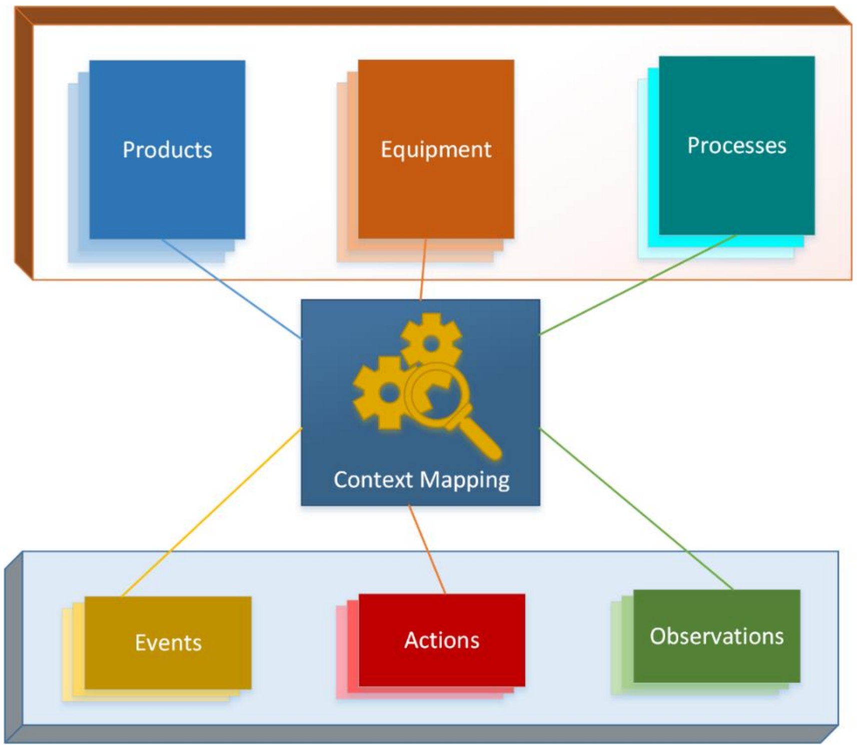


Figure 5 Data context mapping

2.4 UML Structural Class Diagrams

A baseline data model for PERFoRM has already been created in WP2 based on use cases covering a wide spectrum of the European industrial force. The use cases range from home appliances to aerospace and from micro electrical vehicles to large compressor production [10]. However, the baseline model is not adequate for the requirements of Task 3.3. Therefore, the PERFoRM data model has been extended by introducing additional classes. The class design is informed by the concepts, data structures and flows presented in section 2.1, 2.2 and 2.3. Instead of a single class diagram, domain concepts describing separate aspects of the overall system have been introduced. The classes defined in the model are presented as follows:

Manufacturer: The *Manufacturer* class is a representation an equipment manufacturer

- *ID*: A unique identifier for a manufacturer
- *Name:* Name of the manufacturer
- *OtherInformation:* Relevant information about the manufacturer

Equipment: The *Equipment* class is a representation of an actual machine, its modules, components and subsystems

- *ID*: A unique identifier for an equipment instance
- *Name*: Name of the equipment
- *Manufacturer*: Equipment manufacturer
- *Location*: Location of the equipment on the shop floor
- *AdjustableParameters*: List of all the parameters that can be adjusted on the equipment. e.g. temperature, pressure, timer setting, speed
- *RampupStates*: List of all the discretised observable states (state space) of the equipment during ramp-up operation
- *RampupActions*: List of all actions that can be taken on the machine during ramp-up operation

AdjustableParameter: The *AdjustableParameter* class is a representation of a parameter that can be adjusted on a machine

- *ID*: A unique identifier for a parameter instance
- *Name*: Name/description of the parameter
- *Unit*: Unit of the parameter

***RampupState*:** The *RampupState* class is a representation of a machine observable state during ramp-up

- *ID*: A unique identifier for an observable state
- *Description*: Description of the state
- *StateVariables*: List of all variables that define the state
- *IsGoalState*: A true/false value indicating if the state is the goal state

***StateVariable*:** The *StateVariable* class is a representation of a machine state parameter and its value

- *Parameter*: The state parameter
- *Value*: The value of the state parameter (chosen from the list of possible values)

***StateParameter*:** The *StateParameter* class is a representation of a state parameter

- *ID*: A unique identifier for a state parameter
- *Name*: Name or description of the parameter
- *Type*: The performance metric type of the parameter
- *PossibleValues*: A list of the possible discrete or categorical values that can be assigned to the parameter

***StateParameterType*:** The *StateParameterType* class is a representation of a type of state parameter

- *ID*: A unique identifier for a type
- *Name*: Name or description of the type e.g. Cycle time, Product Quality, State description

***RampupAction*:** The *RampupAction* class is a representation of a ramp-up action

- *ID*: A unique identifier for a ramp-up action
- *Description*: Description of the action

Product: The *Product* class is a representation of a given product variant and its core-defining characteristics

- *ID*: A unique identifier for a product instance
- *Name*: Name of the product
- *Description*: Description of the product including defining characteristics

SetupProcedure: The *SetupProcedure* class is a representation of standard operating procedures (SOP) for set-up phase of a changeover process

- *ID*: A unique identifier for an SOP
- *Description*: Description of the SOP including statement of purpose
- *WrittenBy*: The person who created or last modified the SOP
- *ApprovedBy*: The person who reviewed the SOP before putting it into use
- *ApprovedDate*: The date and time of SOP approval
- *Equipment*: The equipment the SOP is associated to

- *Tools*: List of tools required during set-up
- *ChangeParts*: List of product-specific replaceable parts. i.e. a list parts that will require replacement during set-up for the next product
- *Actions*: A detailed list of step by step actions that should be performed during set-up
- *AdjustmentMatrix*: A list of product-specific parameter settings

Person: The *Person* class is a generic representation of a shop floor personnel

- *ID*: A unique identifier for a person
- *Name*: Name of the person
- *Role*: Role of the person e.g. Operator, Engineer, Technician, Mechanic, Supervisor
- *Password*: Login password
- *StartDate*: User account creation date
- *EndDate*: User account expiry date

FileMetadata: The *FileMetadata* class is a representation of a file information

- *ID*: A unique identifier for a file
- *FilePath*: Location of the file

SetupTool: The *SetupTool* class is a representation of a tool used during set-up

- *ID*: A unique identifier for a tool
- *Name*: Name of the tool
- *Description:* Description of the tool
- *Photo:* File definition of attached graphical information (if any)

SetupAction: The *SetupAction* class is a representation of hierarchical actions performed during set-up. The first-level action is a brief description of what is to be done and it is used to generate a checklist for experienced operators. The simple first-level instructions may not be adequate for less experienced operators, so the sub-actions provide more step-by-step instructions on how to perform an action

- *ID*: A unique identifier for an action
- *Description*: Description of the action
- *Photo:* File definition of attached graphical information (if any)
- *SequenceNo:* A number indicating the order of the action
- *SubActions:* A list of sub-actions to be performed

ChangePart: The *ChangePart* class is a representation of a product-specific replaceable part

- *ID*: A unique identifier for a part
- *Description:* Description of the part
- *Photo:* File definition of attached graphical information (if any)

AdjustmentSetting: The *AdjustmentSetting* class is a representation of a product-specific adjustment/setting to be made to an equipment during set-up

- *Parameter*: Adjusted parameter
- *Product:* The product the adjustment is associated to
- *SetPointValue:* The setpoint value for the parameter
- *Photo:* File definition of attached graphical information (if any)
- *SequenceNo*: A number indicating the order of the setting (if required)

SetupEvent: The *SetupEvent* class is a representation of a set-up process

- *ID*: A unique identifier for a set-up process
- *Equipment*: The equipment the set-up process is associated to
- *Product*: The product the set-up process is associated to
- *ContextMap*: The context of the set-up process
- *Persons:* A list of persons involved in the set-up process
- *SetupActions:* A record of actions carried out during the set-up phase of the changeover process
- *SetupObservations:* A record of all the discrepancies or other observations made during set-up
- *AdjustmentActions:* A record of all the adjustments made during set-up

Context: The *Context* class concept provides a general context to events, actions and observations

- *ID*: A unique identifier for a context instance
- *DateTime*: A timestamp for the context instance

***SetupActionRecord*:** The *SetupActionRecord* class is a representation of an action performed during set-up

- *Action*: The performed action (from a list of actions defined in the SOP)
- *ContextMap*: The context of the action

SetupObservationRecord**:** The *SetupObservationRecord* class is a representation of an observation made during set-up

- *Description*: Description of the observation
- *ContextMap*: The context of the observation

AdjustmentActionRecord**:** The *AdjustmentActionRecord* class is a representation of an adjustment made during set-up

- *Parameter*: Adjusted parameter
- *SetPointValue*: The setpoint value of the adjusted parameter
- *ContextMap*: The context of the adjustment record

RampupEvent**:** The *RampupEvent* class is a representation of a ramp-up process

- *ID*: A unique identifier for a ramp-up process
- *SetupEvent*: The set-up event to which the ramp-up is associated
- *ContextMap*: The context of the ramp-up process
- *RampupStates*: List of all the states observed during the ramp-up process
- *RampupActions*: List of all the actions taken during the ramp-up process

RampupStateRecord**:** The *RampupStateRecord* class is a representation of the observed state of a machine at any point in time

- *ObservedState*: The observed ramp-up state
- *ContextMap*: The context of the state record

RampupActionRecord**:** The *RampupActionRecord* class is a representation of a ramp-up action performed on a machine at any point in time

- *RampupAction*: The performed ramp-up action
- *ContextMap*: The context of the action record

The changeover data class diagrams are divided into three domain models as follows:

- Equipment class diagram
- Set-up procedure class diagram
- Set-up event class diagram
- Ramp-up event class diagram

The class diagrams for the four domain models are shown in Figure 6, 7, 8 and 9 respectively.

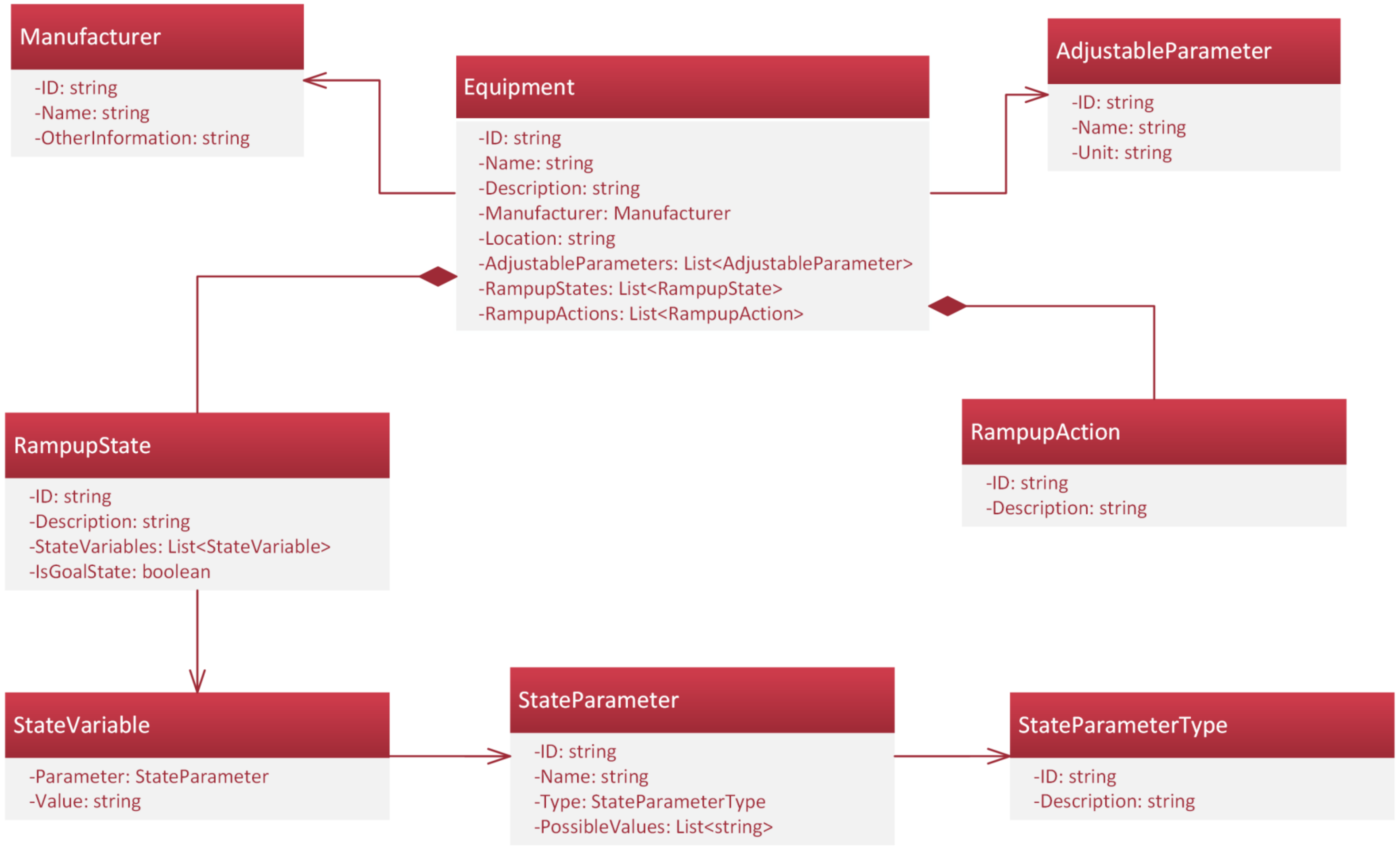


Figure 6 Equipment class diagram

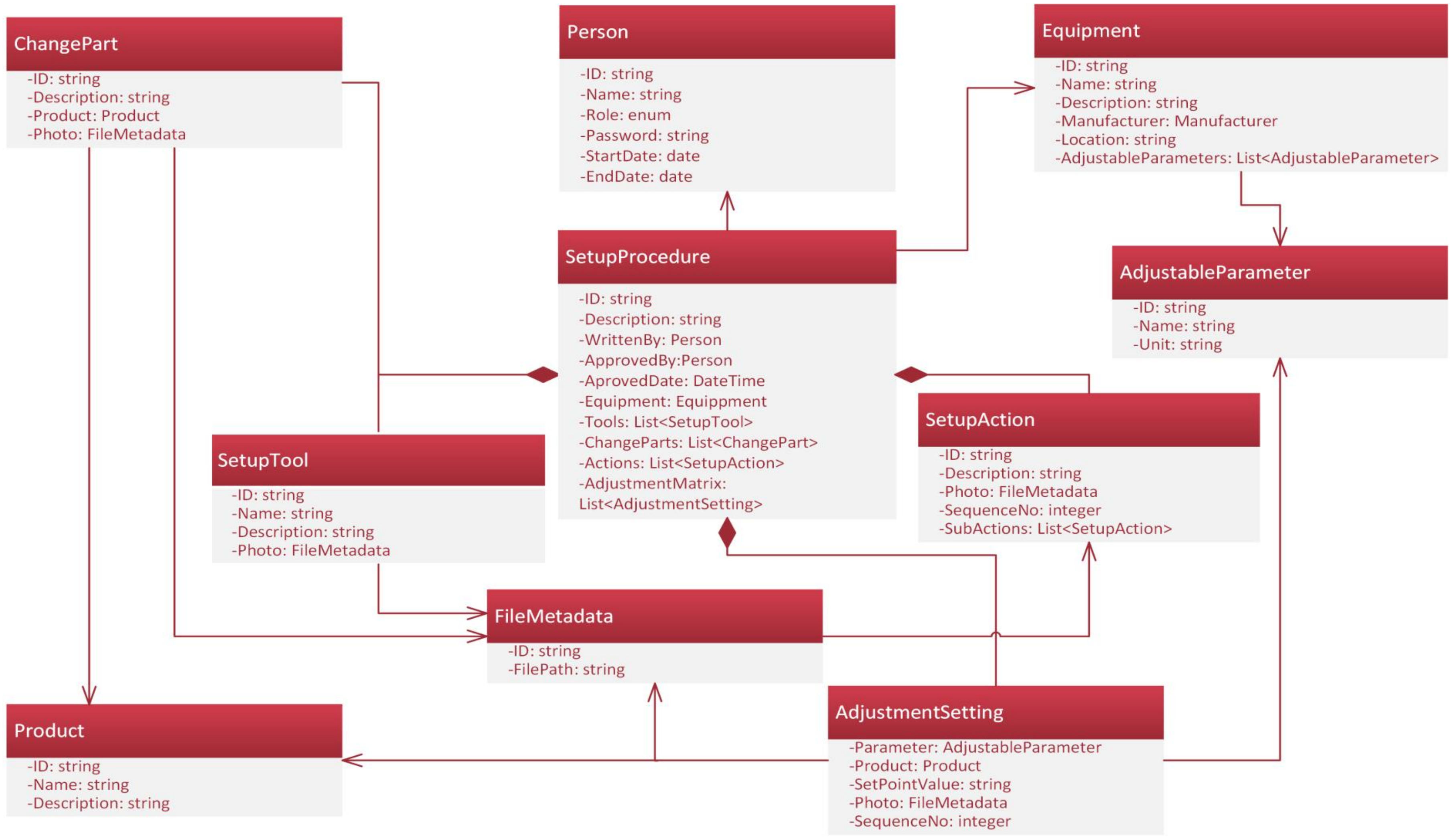


Figure 7 Set-up procedure class diagram

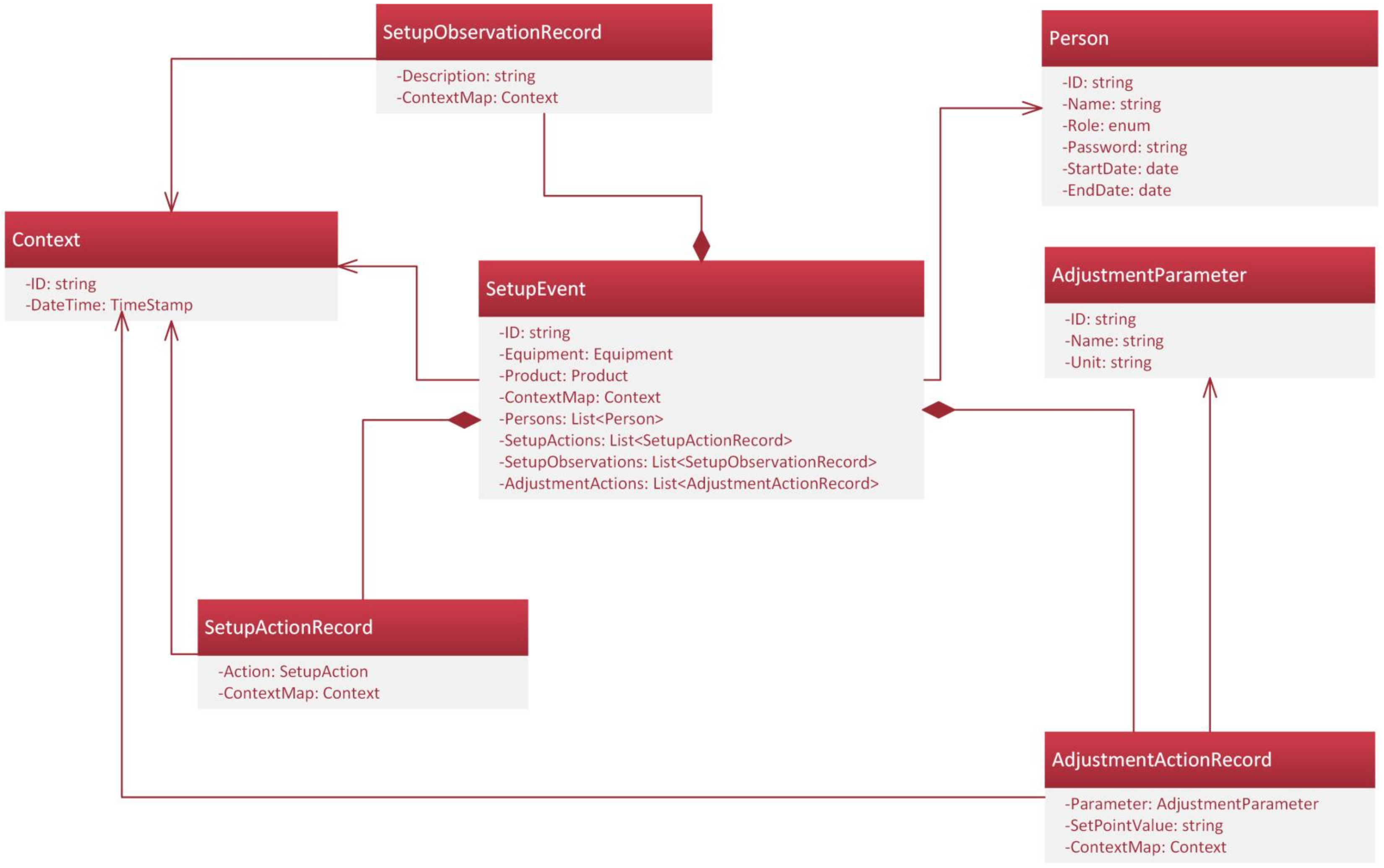


Figure 8 Setup event class diagram

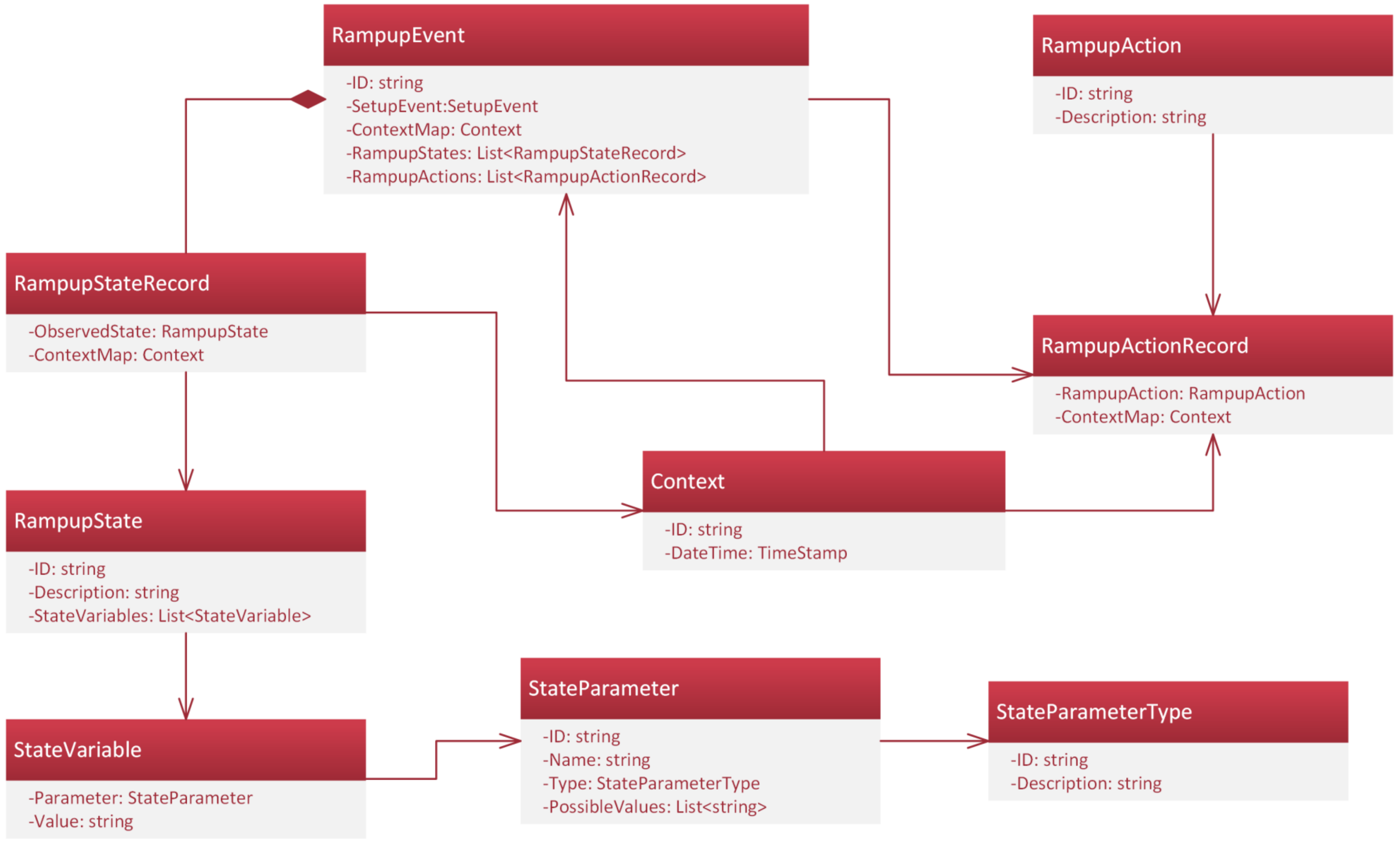


Figure 9 Ramp-up event class diagram

3 Software Application Development

3.1 Functional Requirements Specification

The software is an interactive application that is intended to be used by shop floor operators during a changeover process. The functions of the application are divided into two:

- *Operational*: The software will be used to manage entities, record events, actions and observations in a contextual and structured format
- *Informational*: The software will also function as a decision support system that will assist operators in their decision-making process by presenting them with sorted, group or ranked lists of possible actions that can be performed at machine state

One of the operational requirements of the software is to provide functionality for creating and managing changeover SOPs. Manufacturers will usually provide generic instruction manuals on how to setup an equipment as well as setpoints developed during the testing of the equipment. Manufacturers' instruction manuals are good starting point for creating SOPs. The goal is to put all the required knowledge for set-up in the electronic SOP. Annotated drawings and photographs can be attached to actions, changeparts and tools.

At the beginning of a changeover process, an interface is provided for the operator to select the associated equipment and product. The selections are then used to guide the operator during the process according to the SOP and the decision support system. The first-level set-up actions describing what to do (checklist) are presented to the operator in the pre-defined sequence. This information will usually be adequate for an experienced operator. However, less experienced operators can navigate to the second-level instructions for more detailed guidance on how to perform the actions. As the software guides the operator, all actions taken by the operator are recorded with the relevant contextual information. Provision is also made for the operator to record any discrepancies observed during the process.

In an ideal situation, it should be possible to start a machine and have it run at full production efficiency after clean-up and set-up. However, when a machine resumes production, it may not always run optimally at the specified setpoints due to several reasons. Setpoints developed during equipment testing may not always reflect actual operational conditions Also, some machines could have been modified on the shop floor or worn out after certain period. Moreover, some product variants may require slight adjustments to the specified settings. All these discrepancies necessitate a ramp-up process during which the setpoints are adjusted until the machine reaches a target performance level.

At the beginning of the ramp-up process, the operator observes and records the state of the machine through the software user interface. Necessary action is taken based on the operator's experience and the action is also recorded. The state of the machine after the action is observed and recorded. This process creates a data structure that contains the state of the machine prior to the action, a record of the action and the machine state after the action. The whole process is repeated until the machine achieves the target performance state as shown in Figure 10.

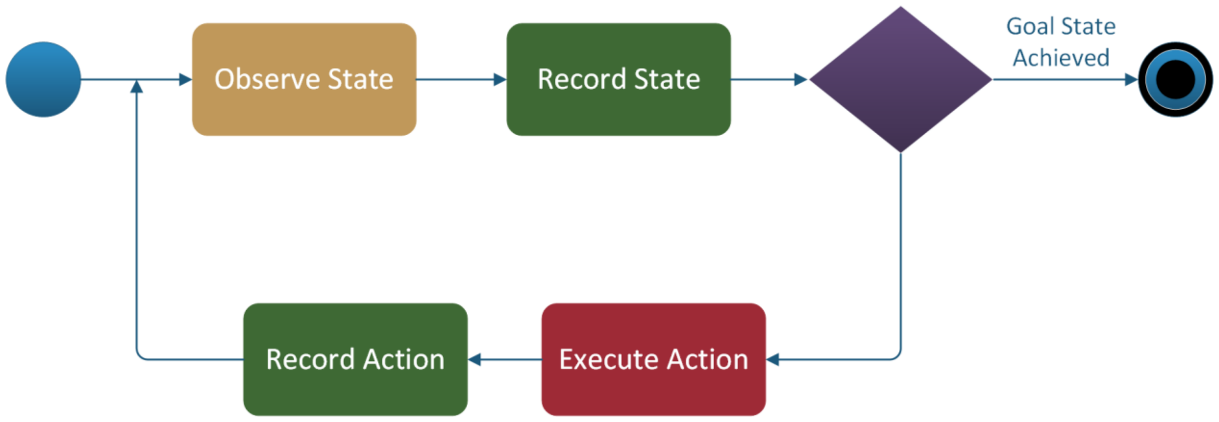


Figure 10 Ramp-up activity diagram

The captured data is converted into knowledge that is used to support future decision making via a reinforcement learning system (section 2.2). When an operator presents the current state of a machine to the decision support system, the system responds with a ranked list of possible adjustments that can be made to the machine to reach the target state quicker. The operator can choose to perform any of the recommended adjustments or ignore them.

3.2 Technical Requirements

The technical requirements of the software are as follows:

- Accessible on any device (mobile phones, tablets, desktops)
- Cross-platform compatibility (Windows, Linux, Mac OS, Android, iOS etc.)
- Separation of concerns between data resources and user interfaces
- Advanced interoperability with other web-based services such as PERFoRM Middleware.

In fulfilling the cross-platform and multiple device accessibility requirements, HTML5/CSS3/JavaScript web application is the favoured choice because web browsers are available on every platform and device. Native cross-platform application development frameworks such as Qt/C++ and Xamarin are alternative options but their open-source licences are not currently suitable for mobile device deployment. Also, installation and maintenance is less complicated with the web-based approach. Once a new version or upgrade is installed on the host server, it is immediately available to all users and devices without activating any update process.

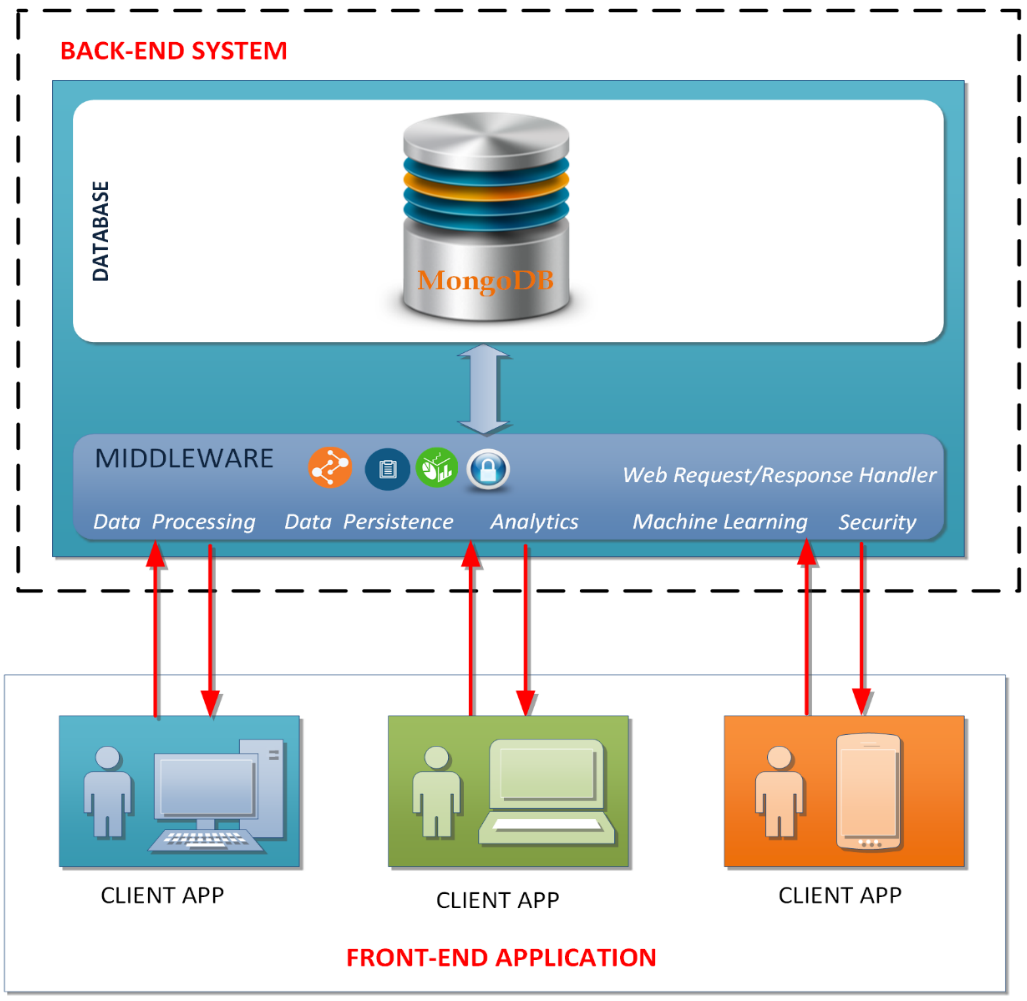


Figure 11 Service-oriented architecture

To fulfil the separation of concern and services interoperability requirements, a service-oriented architecture in which the application is composed by integrating self-contained and separately deployed software components has been adopted (Figure 11).

3.3 Back-end System Development (Web Service)

The backend of the system is a RESTful Web Service, which comprises of a database and a middleware. The web service creates a separation of concerns between data resources and resource consumers. The service manages the data resources but does not address the user interfaces implementation. It only deals with maintaining application state between the client and the server. The functions of the middleware are as follows:

- Handles clients' requests and responses. Maps incoming requests to route handlers and generates responses
- Handles data persistence logic and operations (*Create*, *Read*, *Update*, *Delete*). The data persistence logic is built into a data access layer that separates the database

operations from the rest of the application. This provides the flexibility of migrating from one database technology to the other by changing the data access layer only.

- Handles security using token-based authentication and authorisation
- Handles business logic, including machine learning and analytics.

The web service has been created using ASP.NET Core. It is a new open-source and cross-platform framework for building modern web applications. Other suitable frameworks include Node.js, Java Jersey (JAX-RS) and PHP.

MongoDB has been choosing for storing data. MongoDB is a free and open-source cross-platform document-oriented NoSQL database. In contrast to relational database management systems (RDBMS), it stores records as JSON documents instead of table-based structured storage where a single object may be spread across several tables. Prior to the advent of JSON and XML data stores, most web services are backed by RDBMS. However, MongoDB has been chosen because it does not require an explicit schema definition. Instances of a given object defined in section 2 can be stored in a single collection and every stored instance can have different fields. This allows direct mapping of the PERFoRMML data model created in AutomationML without using any object-relational mapper. Other open-source JSON-based NoSQL data stores that are suitable for this project include CouchDB and Couchbase Server.

An overview of the web service is shown in Figure 12.

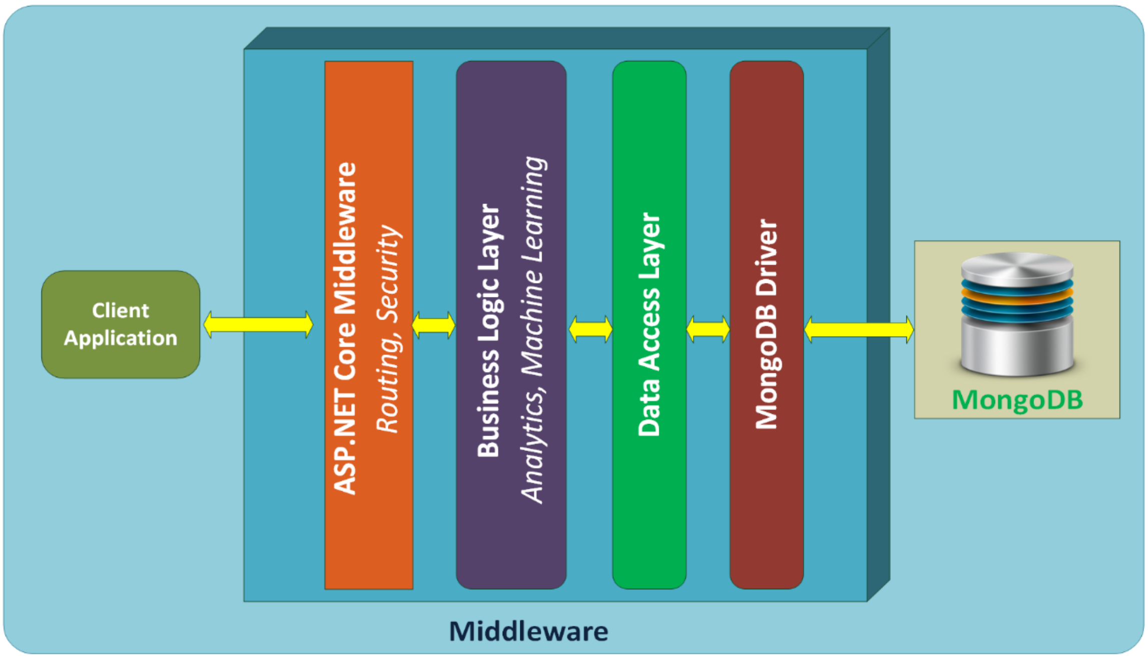


Figure 12 Server-side RESTful web service overview

3.3.1 Reinforcement Learning System Implementation

The Q-learning agent is built into the web service. The agent tries to learn the optimal policy from the histories of operators' interactions with an equipment during ramp-up. A history is a sequence of state-action-rewards, which is derived from the contextual ramp-up action and state records. The history of interaction is treated as experiences, which is the data from which the agent learns what to do. In building the reward function *R*, any state-action pair that is not captured in the experiences are assigned -1 and experiences in which the agent moved to the goal state gets a reward of 10. Since the goal is to minimise ramp-up time, no reward is given to any state transition that does not bring the system to the desired production performance (goal state). This strategy penalises every step taken and forces the learning algorithm to choose a set of actions (known as policy) that will bring the system to the goal state much quicker.

The system is implemented using the QLearning functionality in AForge.NET Framework.

3.3.2 Security: Authentication and Authorisation

Authentication is an essential component of any meaningful application. Although there is no intention of granting access to third party consumers, it is best practice to address the issue of security. The service is secured using JSON Web Token (JWT), which is an industry standard method for token-based authentication for modern single-page web and mobile applications.

The authentication system is implemented using ASP.NET Core Identity provider with MongoDB to store membership, login and user account data (Figure 13).

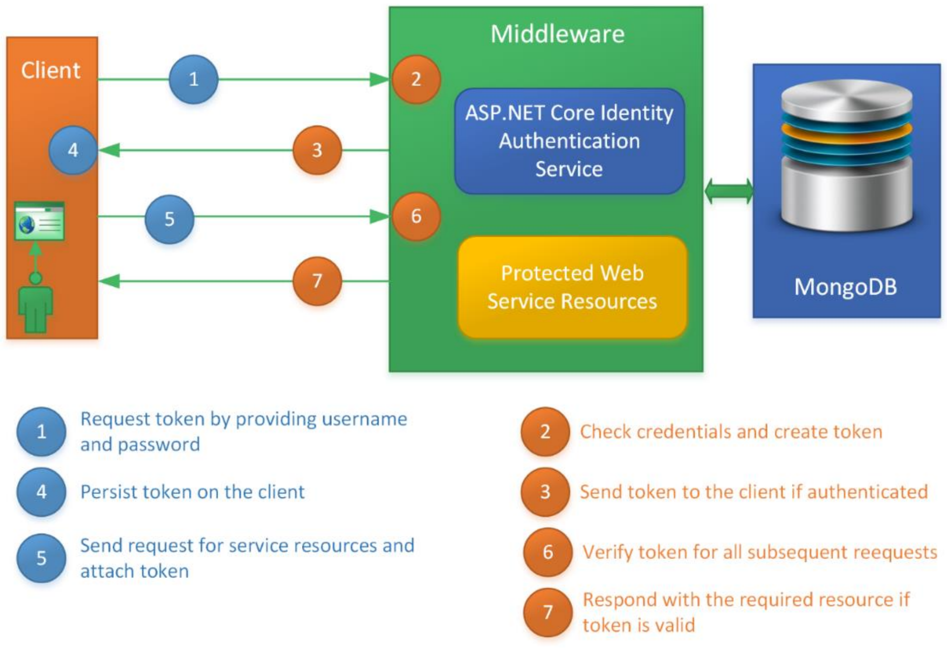

Client
Middleware
ASP.NET Core Identity Authentication Service
Protected Web Service Resources
MongoDB
1
2
3
4
5
6
7
1 Request token by providing username and password
4 Persist token on the client
5 Send request for service resources and attach token
2 Check credentials and create token
3 Send token to the client if authenticated
6 Verify token for all subsequent reequests
7 Respond with the required resource if token is valid

Figure 13 Token-based authentication process

3.4 Front-end Application Development

The changeover software user interface is a web application, which is powered by the ASP.NET Core backend service. The application was developed using the latest patterns and the following tools:

Angular 4.0

Angular is an open-source front-end web application framework for developing single-page applications (SPA) that can be deployed across many devices and platforms such as native web and mobile phones. The main reason for choosing Angular is because it allows the user interface (UI) and the business logic of the application to be separated (separation of concerns).

TypeScript

TypeScript is a free and open-source programming language that transpiles to JavaScript. It adds optional static typing and object-oriented programming concepts such as classes, properties, modules, interfaces and inheritance to JavaScript. This makes is a better fit for large-scale applications development than plain JavaScript.

Bootstrap

Bootstrap framework is used to design the user interface to make it responsive. Being responsive in this case means that the user interface will automatically adjust itself to look good on all devices, from small phones to large desktop monitors. Other responsive design front-end frameworks such as Foundation exist but Bootstrap appears to be more popular and has more commercial ready-made templates.

3.4.1 Angular 4.0 Application Structure

The application consists of a tree of components organised into cohesive but loosely coupled blocks of functionalities known as feature modules. The feature modules encapsulate the required functionalities of the system. The modules are described as follows:

- *EquipmentModule*: Contains all the components for managing equipment
- *SetupProcedureModule*: Contains all the components for managing setup operating procedures
- *SetupEventModule*: Contains all the components for managing set-up process
- *RampupEventModule*: Contains all the components for managing ramp-up process

The services block contains user access control and http services communications functionalities. The functions of the services are described as follows:

- *AuthService:* This service is used for user authentication http calls and logic
- *EquipmentHttpService:* This service is used to manage all http data calls in the *EquipmentModule*
- *SetupProcedureHttpService:* This service is used to manage all http data calls in the *SetupProcedureModule*
- *SetupEventHttpService:* This service is used to manage all http data calls in the *SetupEventModule*
- *RampupEventHttpService:* This service is used to manage all http data calls in the *RampupEventModule*

The services block is the link between the PERFoRM Middleware and the web service. Modifications can be made to the services block without affecting other areas of the application. The overall system overview is shown in Figure 14 and the components tree is shown in Figure 15.

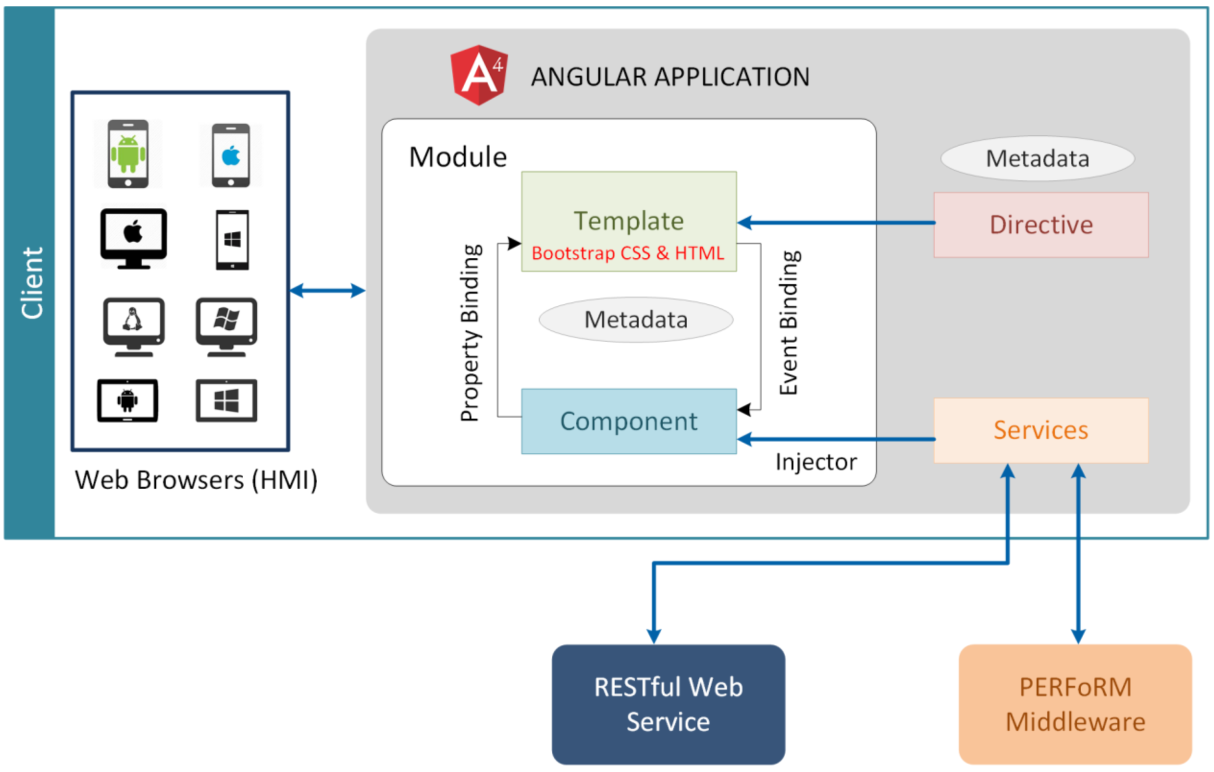


Figure 14 **Client-side Angualr 4.0 application overview**

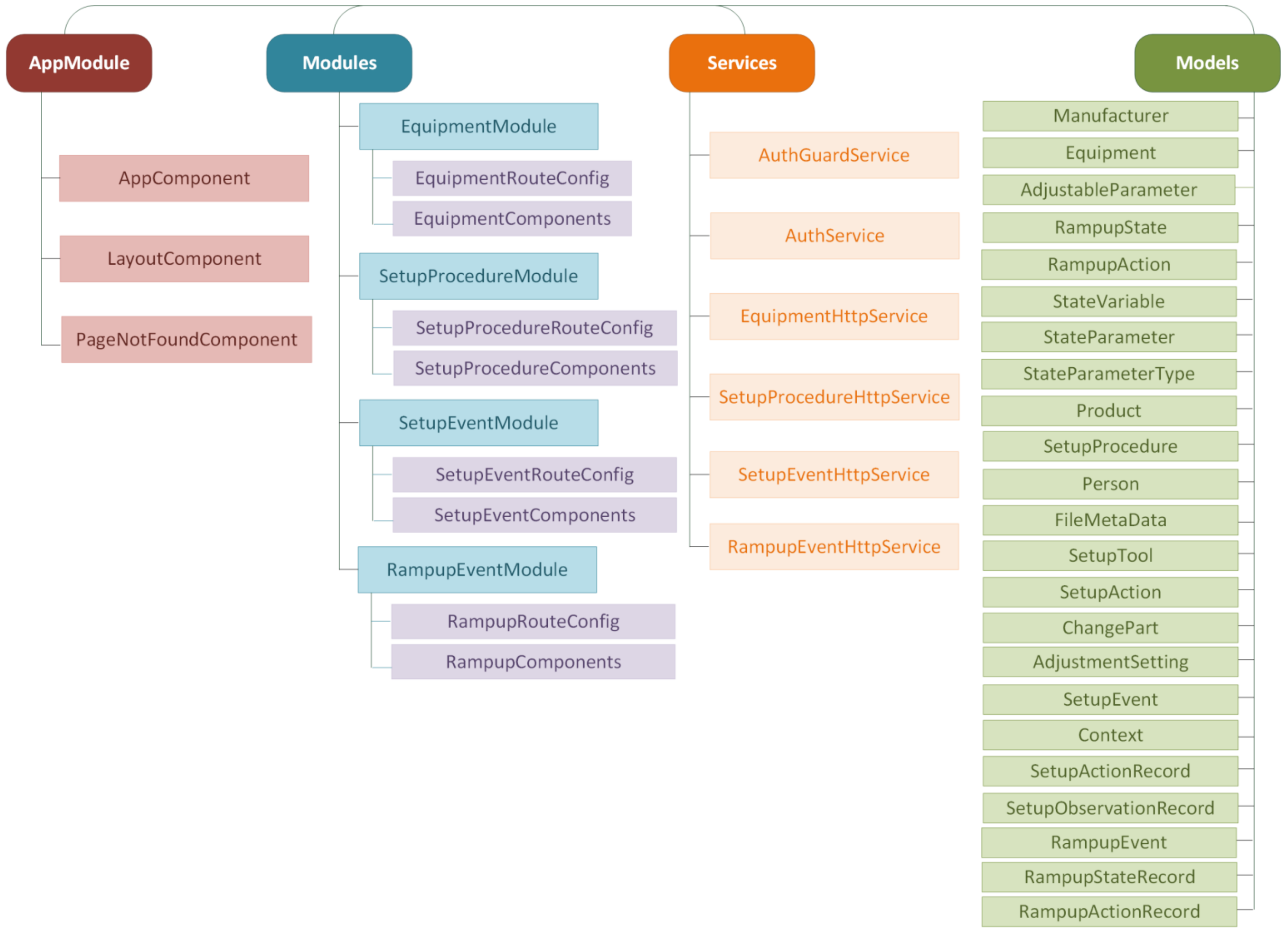


Figure 15 Client application design structure

Whirlpool Industrial Case Study

4.1 Introduction

4.1.1 Collaborative robots

Collaborative robots have emerged as a novel phenomenon to perform tasks and operate in environments that were not possible for traditional robots. They are becoming a practical and convenient solution to be massively employed in the manufacturing lines. Whether they are employed in truly collaborative tasks or not, novel questions emerge regarding how they can be effectively and efficiently handled on the shop floor.

The first installation of the collaborative robots is carefully designed and implemented by the Industrial Engineering Departments and by external suppliers such as cobots manufacturers or, more frequently, systems integrators. Main changes are expected to be handled analogously, but production must deal with frequent adjustments, related to the increasing number of product’s variants.

As the collaborative robot’s intensity becomes higher, the task of reprogramming the robot’s job to accommodate minor product changes is no longer sustainable buy the Industrial Engineering staff and need to be handed over to the plant and line staff.

The Maintenance Technician and the “Operator 4.0” are the candidates to develop the competences and take over the task of reprogramming the collaborative robot.

While robotics or programming-related skills might be easily found on the shop floor in large or highly automated enterprises, SMEs need to find methods and practices to develop these capabilities, taking into account economic and production constraints.

In this section, a method is presented to extract knowledge from the observation of an expert and of an inexperienced operator.

Section 4.1.2 provides an overview of the state of the art with relevant reference to the challenges of building capabilities for the manufacturing staff to reprogram cobots, by leveraging the expert’s knowledge.

The rest of the section is organized as follows: section 4.1.3 introduces the industrial case by describing the context and the needs; section 0 presents the methodological approach proposed; section 0 illustrates the results of empirical study based on the industrial case.

4.1.2 State of the Art

The most relevant technological trends in robotics encompass simplified, lightweight and collaborative applications [14]. The sales of industrial robots are growing fast and according to the analysts [15] - [16] and the manufacturers [17], collaborative robots will have a breakthrough. Research has been having a parallel boost, having started at the end of the last millennium, the indexed papers increased steadily, and with a sharp acceleration in the last three years.

As collaborative robots are a recent phenomenon, research is highly focused on the design, engineering and the demonstration of novel functionalities and interaction modes with humans. Also, studies addressing the topics of collaborative robot programming or training, such as [18], [19] assume the same perspective with the aim of developing robots more easily programmable or trainable.

In order to investigate how cobots are programmed and skills developed, references are available on the manufacturers' web sites, such as the Universal Robots Academy (Universal Robots) or Kuka Teaching by Demonstration [20]. However, as technology enables to instruct the robot through different modes [21] which allows multiple paths to achieve the same objectives, the task of efficiently and effectively programming a cobot is not straightforward.

Experts perform this task by exploiting operational knowledge, which is considered as a part of tacit knowledge, are more difficult to capture than other forms [22].

According to the scientific literature, observation, together with apprentice, is considered the most suitable method to capture and transfer operational knowledge, through the fusion and analysis of different data sources and the use of a comprehensive framework that captures also the intent behind the actions [22].

Approaches to acquire tacit knowledge tend to blend different methods for capturing, analysing and categorizing the content, and for placing it into interpretative frameworks, like scenarios or if-then-because statements applicable to specific application domains.

4.1.3 The Industrial Case

The collaborative robot

The robot is equipped with safety a mechanism, which detects obstacle and stops to prevent possible damages, as illustrated in Figure 16.

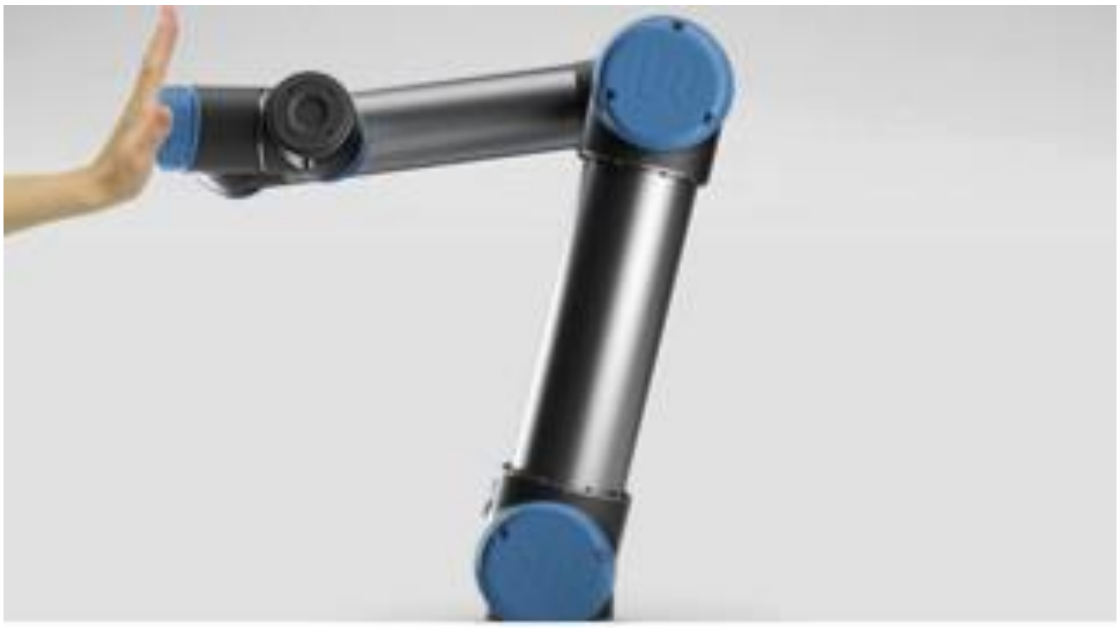
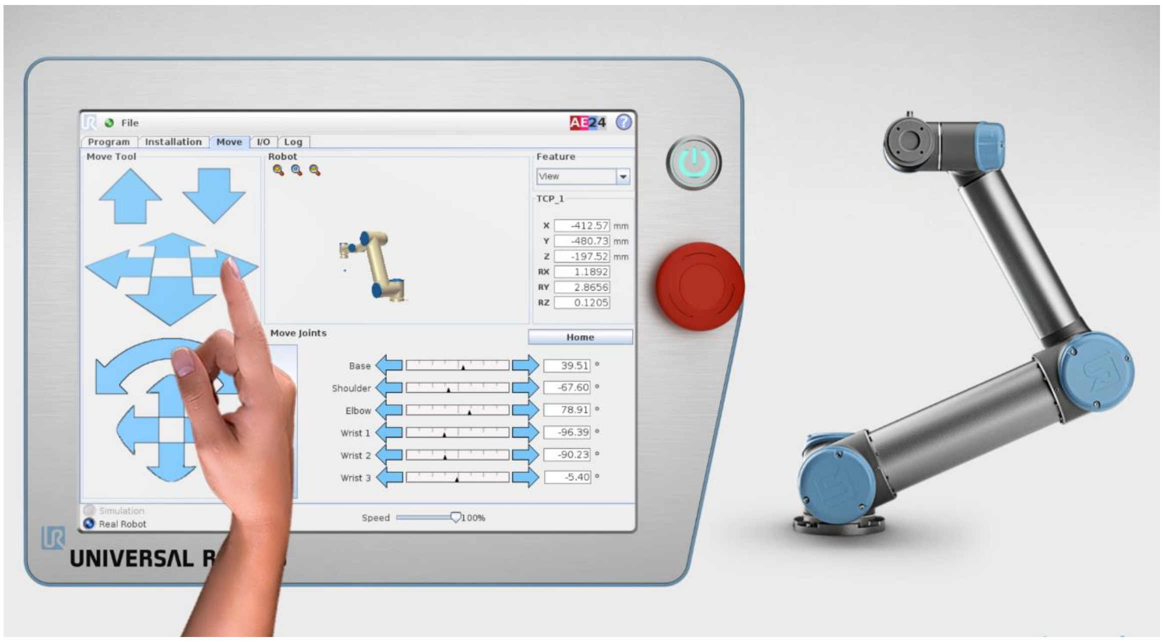


Figure 16 The robot safety features

The teach pendant is the human-machine interface through which the robot is programmed and controlled. Some of the actions that can be performed are as follows:

1. Move the tool, by holding down translate and rotating rows, as illustrated in Figure 17;
2. Teach the robot, by holding down the teach button and physically grabbing the robot arm and pull it to the desired position
3. Change the tool position, by manually editing the coordinates (Figure 18).



Feature: View

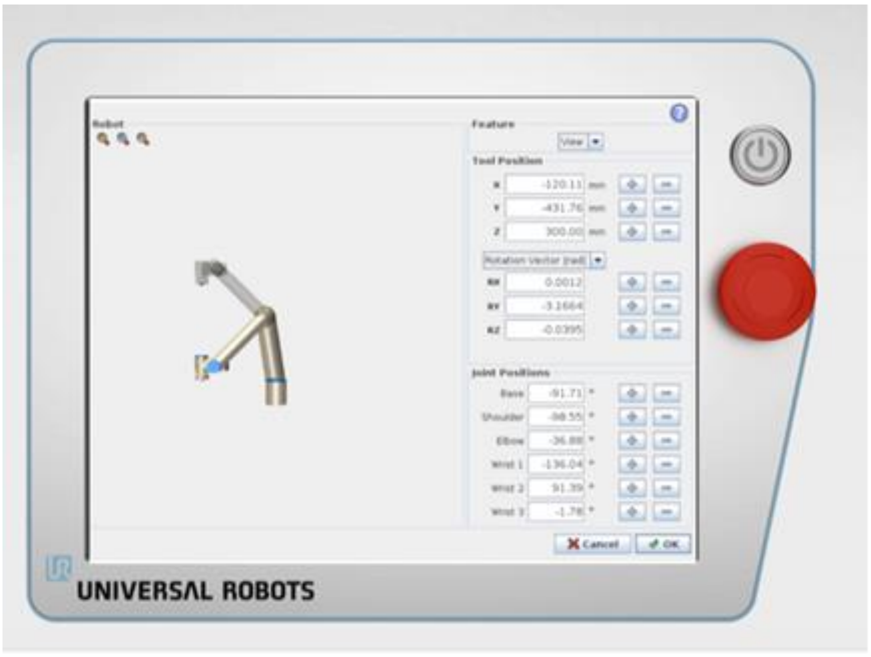
TCP_1

X	-412.57	mm
Y	-480.73	mm
Z	-197.52	mm
RX	1.1892	
RY	2.8656	
RZ	0.1205	

Base	39.51 °
Shoulder	-67.60 °
Elbow	78.91 °
Wrist 1	-96.39 °
Wrist 2	-90.23 °
Wrist 3	-5.40 °

Speed 100%

Figure 17 The teach pendant command to move robot



Feature: View

Tool Position

X	-120.11	mm
Y	-431.76	mm
Z	300.00	mm

Rotation Vector [rad]

RX	0.0012
RY	-3.1664
RZ	-0.0395

Joint Positions

Base	-91.71 °
Shoulder	-98.55 °
Elbow	-36.88 °
Wrist 1	-136.04 °
Wrist 2	91.39 °
Wrist 3	-1.78 °

Figure 18 The teach pendant in editing mode

The task executed by the robot

The robot is used in a testing workstation, at the end of an assembly line producing a family of three burners cooktops products.

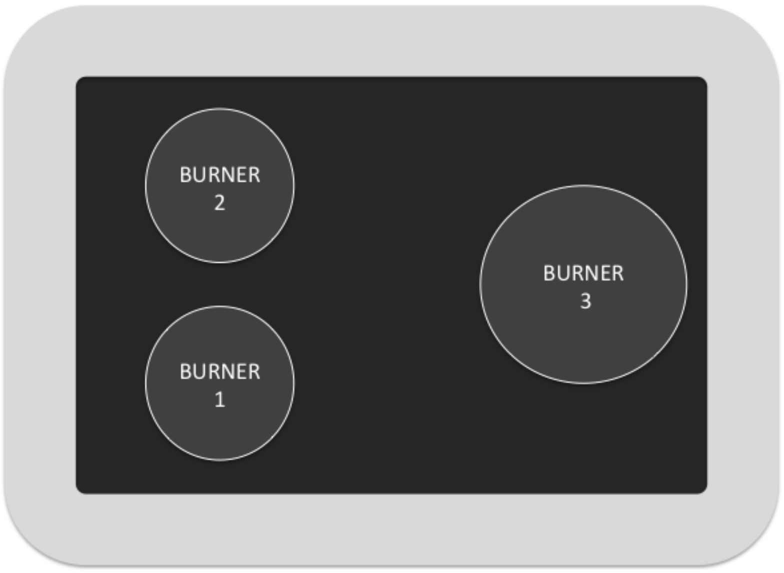


Figure 19 Three burners cooktop

The task of the robot is to place the tool, simulating a pot, on each of the three burners and wait for the special equipment of the station to electrify the burner and check that it behaves according to specifications (Figure 19).

The structure of the testing programs is the same for all the products in this family and it is represented in Figure 20, abstracting from the details that are not relevant in this context.

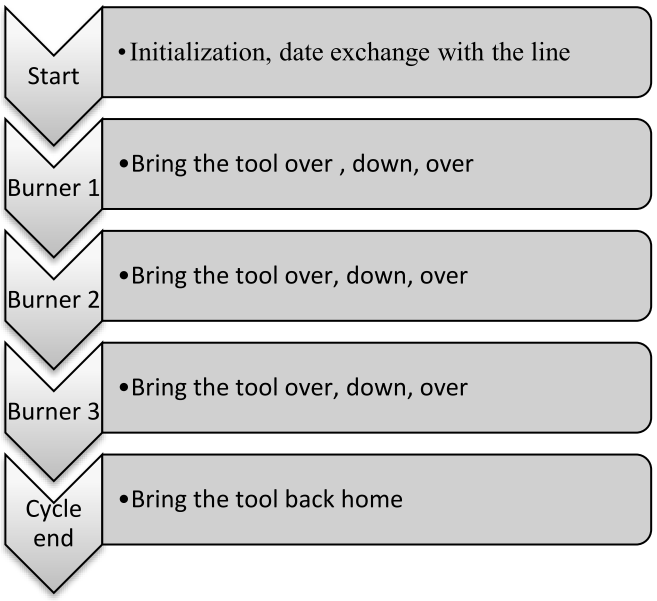

Start
• Initialization, date exchange with the line
Burner 1
• Bring the tool over , down, over
Burner 2
• Bring the tool over, down, over
Burner 3
• Bring the tool over, down, over
Cycle end
• Bring the tool back home

Figure 20 Robot program structure for testing

The case of new products

The three burners cooktop encompasses products with different lengths, width, height and burners disposition. Each time a new product is produced for the first time, a new program for the robot is released.

The new program maintains the same structure of the product family but requires an adjustment of the coordinates of the six waiting points, included in the trajectories of the tool:

1) Burner 1 over
2) Burner 1 down
3) Burner 2 over
4) Burner 2 down
5) Burner 3 over
6) Burner 3 down

The increasing demand for customized product entails frequent introduction of new products within the same family.

The need for human observation

The introduction of new products into the production line implies the frequent need to generate slightly different programs for the robot. The capabilities to generate these new programs need to be developed in the plant where the robot is installed, so that the personnel can handle the introduction of new products by themselves. In order to build those capabilities, it is useful to extract the knowledge from the expert and investigate what should be transferred and provided to the inexperienced personnel.

4.2 Methodological approach

In order to perform the above-mentioned extraction of the expert knowledge and development of the capabilities at the production plant, an approach based on human observation has been proposed. It is illustrated in Figure 21.

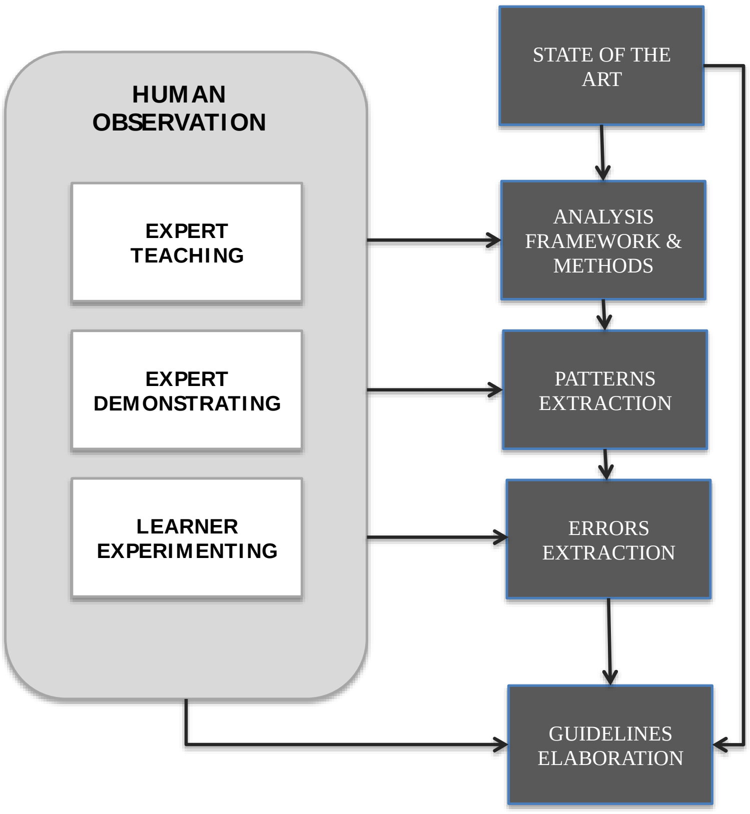


Figure 21 **Methodological apporach**

The approach leverages the observation of the expert and of the learner or inexperienced personnel. The expert is observed while teaching how to re-program the robot and while demonstrating the task. The learner or inexperienced personnel is observed while experimenting the robot re-programming.

At the core of the approach is the definition of a framework of analysis. This framework has been built based on previous knowledge coming from the scientific literature and technical documentation, and has been adapted to the specific context derived from the observation of the expert teaching. The framework for the analysis of human behaviour is based on five dimensions: position, posture, gaze, gesture and interaction with the robot.

Figure 22 presents the framework, with the dimension and the related values in the specific context of analysis.

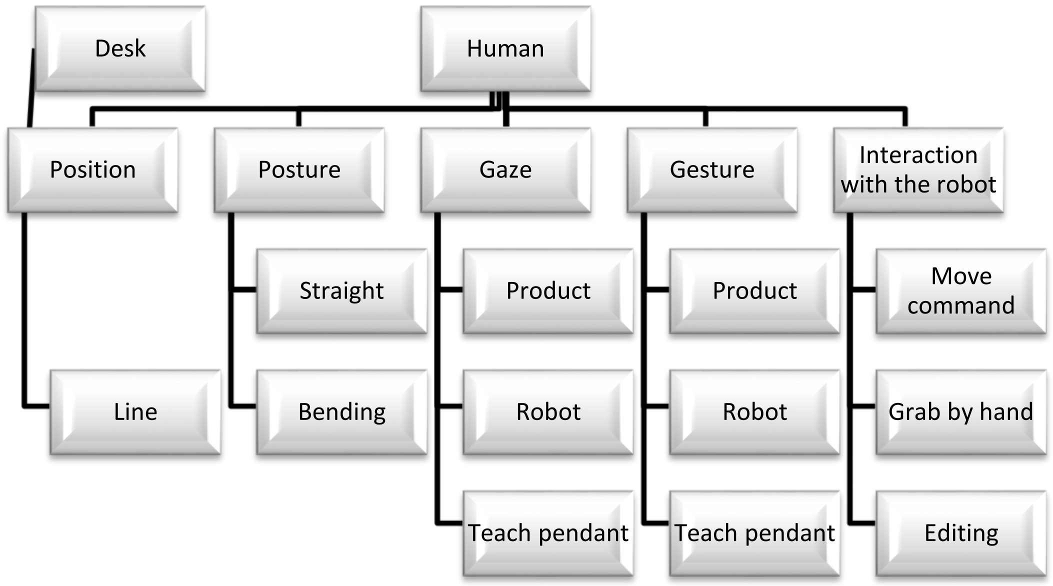


Figure 22 The analysis framework

The methods for the observation and analysis of human behaviour encompasses:

Teaching:

- Video and audio recording of the field activity
- Decomposition of the video into individual action items

Demonstration:

- Denomination and characterization of the individual activities, according to the framework
- Identification of the patterns

Experiment:

- Isolation and annotation of the exemplary situations
- Identification of the errors

Section 0 describes the application of the methods; the framework used in the field observations in order to extract patterns and errors; the guidelines elaborated on the basis of the observation and analysis activities.

4.3 Results

4.3.1 Expert observation

The expert demonstrating the task was video recorded. The video was split into action items. Each action has been shortly described and time referenced, as reported in Table 1.

Table 1 Expert's demonstration action items

N. ITEM	TIME	ACTION	ON COORDINATES	
			XY	Z
0	0	Opens rename program		
1	0,18	Sets the alignmnt of the product vs. the references		
2	0,23	Command robot to burner 1 over		
3	0,33	Observe robot moving to burner 1over		
4	0,5	Reduces speed and adjusts burner1 over	adjusts D	ajusts D
5	0,55	Saves burner 1 over		
6	1,08	Change burner 1 down with coordinates of burner 1 over	modifies D	
7	1,09	Command robot down and adjusts vertically		adjusts D
8	1,11	Saves burner 1 down		
9	1,15	Commands robot to burner 2 over		
10	1,18	Observes robot moving to burner 2 over		
11	1,25	Check the alignment and adiusts burner 2 over	adjust D	
12	1,36	Move robot down	moves D	adjusts D
13	1,39	Saves burner 2 down		
14	1,48	Change burner 2 over by changing z+50		modifies D
15	1,5	saves burner 2 over		
16	1,52	Commands robot to burner 3 down		
17	2,02	Observes robot moving to burner 3 down		
18	2,11	Adjusts burner 3 down	adjust D	adjust D
19	2,14	Saves burner 3 down		modify D
20	2,24	Change z +50mm		
21	2,17	Check height		
22	2,25	Save burner 3 over		
23	2,26	Command robot moving burner 3 over		
24	2,27	Check		
25	2,3	Command robot moving to burner 2 over		
26	2,34	Observe robot moving		
27	2,36	Check		
28	2,37	Save		
29	2,39	End		

Some preliminary considerations confirm that the sequence of the activities has a correspondence with the structure of the program, as reported in Figure 18.

In fact, the expert executes some initial activities, then, he modifies the coordinates of the six waiting points. As highlighted in the last two columns on the right side of the table, three different strategies are adopted for each of the three burners:

For the first burner, the point Over is adjusted first relative to the three coordinates. Then the robot’s tool is lowered and the *z* of the point Down adjusted, while maintaining the same *x* and *y* of the point Over.

1. For the second burner, the *x* and *y* coordinated are adjusted with the tool Over. Then the tool is lowered and the point Down is saved. Finally, the point Over is defined by maintaining *x* and *y* and decreasing the coordinate *z* by 50 mm.

2. The third burner is handled in a different way. The coordinates *x*, *y* and *z* Down are adjusted first, then the *z* of the Over point is determined by decreasing *z* by 50 mm.

For the first two burners, the horizontal adjustment has been executed from an Over position, while for the third burner from a Down position.

This is due to the size of the marks of the burner. In fact, they are too small to be visible in the down position, for the first two burners, but they are wide enough for the third.

For the first burner, the *z* has been determined by adjusting the tool position both for the Over and Down points, For the other burners, only the *z* of the Down point has been adjusted, the z of the Over has been determined by decreasing the value of the coordinate.

This is because the same vertical distance defined for the first burner has been left for the other two.

Furthermore, in order to progress with the analysis, each activity has been characterized according to the analyses framework, as illustrated in Table 2.

Table 2 Experts demonstration activities characterization

N. ITEM	ACTION	POSITION		POSTURE		HUMAN SENSING-ACTUATING		ROBOT INTERACTION		
		desk	line	bending	straight	glaze	gesture	coding	manual command	guide by hand
0	Opens rename program	x		x		teach pendant	teach pendant	x		
1	Sets the alignmnt of the product vs. the references		x		x	product	product			
2	Command robot to burner 1 over	x		x		teach pendant	teach pendant		x	
3	Observe robot moving to burner 1 over		x		x	robot	teach pendant			
4	Reduces speed and adjusts burner1 over		x		x	teach pendant-robot	teach pendant		x	
5	Saves burner 1 over		x			teach pendant	teach pendant	x		
6	Change burner 1 down with coordinates of burner 1 over		x			teach pendant	teach pendant	x		
7	Command robot down and adjusts vertically		x	x		robot	teach pendant		x	
8	Saves burner 1 down		x	x		teach pendant	teach pendant	x		
9	Commands robot to burner 2 over					teach pendant	teach pendant		x	
10	Observes robot moving to burner 2 over		x	x		robot	teach pendant		x	
11	Check the alignment and adiusts burner 2 over		x		* (leaning)	robot	teach pendant			
12	Move robot down		x	x		teach pendant-robot	teach pendant		x	
13	Saves burner 2 down		x	x		teach pendant	teach pendant	x		
14	Change burner 2 over by changing +50		x	x		teach pendant	teach pendant	x		
15	saves burner 2 over						teach pendant	x		
16	Commands robot to burner 3 down		x	x		teach pendant	teach pendant		x	
17	Observes robot moving to burner 3 down		x		x (leaning)	robot	teach pendant			
18	Adjusts burner 3 down		x	x		teach pendant-robot	teach pendant		x	
19	Saves burner 3 down					teach pendant	teach pendant	x		
20	Change +50mm		x	x		teach pendant	teach pendant	x		
21	Check height					robot	teach pendant			
22	Save burner 3 over		x	x		teach pendant	teach pendant	x		
23	Command robot moving burner 3 over		x	x		teach pendant	teach pendant		x	
24	Check		x	x		robot	teach pendant			
25	Command robot moving to burner 2 over		x	x		teach pendant	teach pendant		x	
26	Observe robot moving		x	x		robot	teach pendant			
27	Check		x	x		robot	teach pendant			
28	Save		x	x		teach pendant	teach pendant	x		
29	End	x				teach pendant	teach pendant			

As highlighted in green in the table, the expert's position is close to the **line**, in the activities of adjustment: s**traight, even leaning**, immediately before or when regulating the coordinates *x* and why; **bending** when regulating the coordinate *z*.

Which is perfectly coherent with the objective of horizontal and vertical alignment of the robot's tool with the top surface and the burners' axis.

During the horizontal alignment, the **glaze** switches quickly **from the teach tenant to the robot**.

When **coding**, the glaze is directed to **the teach tenant**.

The **gestures** consisted in interacting with the **teach tenant** all the time, but in the second activity, when

In the timeframe of the demonstration, the "**guiding by hand**" modality has never been used; mainly because the learner works on programs already prepared by the expert. However, the expert had employed that modality to bring back the robot in the "home" position, just before starting the demonstration.

4.3.2 Learner Observation

The observation of a learner experimenting the re-programming of the robot, allowed the identification of several errors that typically occur when a less experienced person undertakes this task.

Table 3 Learner’s errors

N. ITEM	ACTION	POSSIBLE ERRORS/PROBLEMS
0	Opens rename program	fingers too thick
1	Sets the alignmnt of the product vs. the references	difficulty in finiding correspondence between the coordinates of the robots and the line
2	Command robot to burner 1 over	
3	Observe robot moving to burner 1over	
4	Reduces speed and adjusts burner1 over	
5	Saves burner 1 over	not sure to have saved the burner
6	Change burner 1 down with coordinates of burner 1 over	confuses the instruction: command to move the robot to burner down 1 instead of change burner 1 down
7	Command robot down and adjusts vertically	
8	Saves burner 1 down	
9	Commands robot to burner 2 over	
10	Observes robot moving to burner 2 over	
11	Check the alignment and adiusts burner 2 over	posture uncorrect
12	Move robot down	
13	Saves burner 2 down	
14	Change burner 2 over by changing z+50	difficulty in remembering the correspondence between the direction and increase/decrease of the coordinate
15	saves burner 2 over	
16	Commands robot to burner 3 down	
17	Observes robot moving to burner 3 down	
18	Adjusts burner 3 down	
19	Saves burner 3 down	burner 3 over saved before problem of posture
20	Change z +50mm	plus versus minus
21	Check height	
22	Save burner 3 over	
23	Command robot moving burner 3 over	
24	Check	
25	Command robot moving to burner 2 over	
26	Observe robot moving	
27	Check	
28	Save	
29	End	

4.3.3 Guidelines

Overall from the observation of the expert teaching, of the expert demonstrating and of the learner experimenting, and on the basis of the analysis performed, a set of guidelines have been elaborated.

Preparation

Prior to starting the activity of reprogramming the robot, the inexperienced personnel follows these instructions:

- Check if it is convenient to interact with the teach pendant with or without a touch screen pen
- Get familiar the robot interface and, in particular, with the move command and with the programming functionality
- Get familiar with the view on the teach tenant, the x, y, and z-axis and their direction.

- Check the alignment of the product with the line and the reference sensors (photocell)
- Copy an existing program and change the name and codes to be exchanged with the control of the line.

It is worth noticing that some simple artefacts or technological devices might support the preparation activities. For example, by adding physical or created by applications of Augmented Reality arrow-shaped marks on the assembly line, aligned with the view on the teach tenant, might be helpful to prevent issues and possible errors and streamlining the activity.

Change of the burners[1]

The main activity in reprogramming consists of changing the coordinates of the waypoints for updating the trajectories of the cobot to the geometrical characteristics of the new product. In order to better perform it, the inexperienced personnel should follow these indications:

- Get close to the line
- Command the robot to burner over

If the marks on the cooktop are wide enough, apply the strategy **HORIZONTAL ALIGNMENT (X, Y) DOWN** on the cooktop

[1] The assumption is that the original program has to be adapted in order to accommodate a new product of the same family. In this case, a cooktop with a different geometry and disposition of the burners.

- Lower the tool and **bend** to **observe** that the **tool** has the same inclination as the cooktop.
- When the tool is placed on the top, stand up **straight**, leaning on the product **to adjust the too**l along x and y until it is centred
- Save the point burner Down
- Command the robot to lift the tool 50 mm or edit the z coordinate – 50 mm
- Save the point burner Over

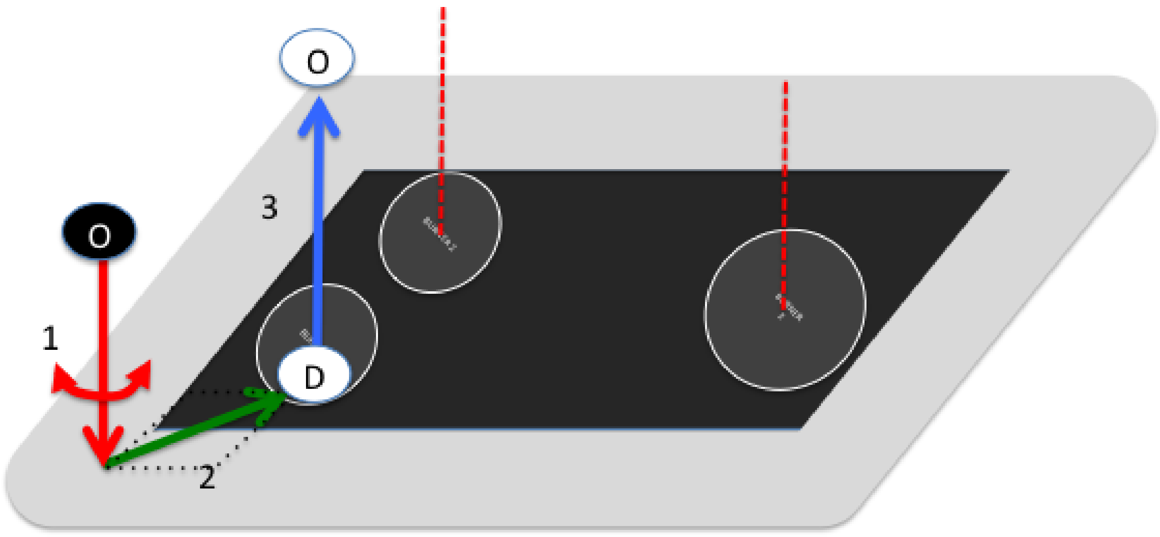


Figure 23 Horizontal alignment Down on the cooktop

Otherwise, if the marks are not visible when the tool is placed down, apply the strategy **HORIZONTAL ALIGNMENT (X, Y) OVER** the cooktop**,** illustrated in Figure 23.

- Stand up **straigh**t leaning over the product
- Move the tool horizontally and **align the x and y coordinates** until the tool is **Over** the burner (green arrow 1)
- Lower the tool (red arrow 2) and **bend** to **observe** that the **tool** has the same inclination as the cooktop.
- When the tool is placed on the top, save the point burner **Down**
- Command the robot to lift the tool 50 mm or edit the z coordinate – 50 mm (blue arrow 3)
- Save the point burner **Over**

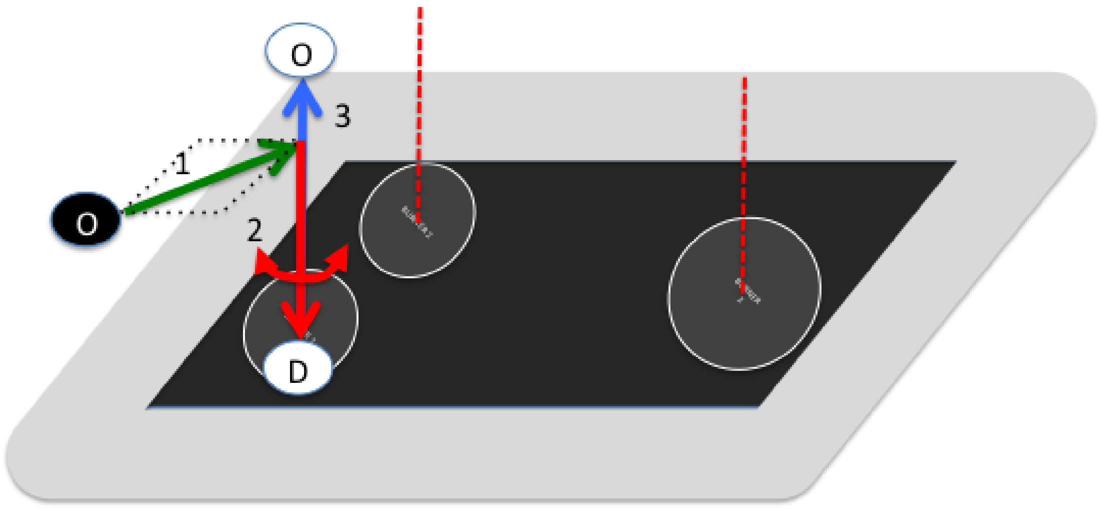


Figure 24 Horizontal alignment Over the burner

Some additional suggestions may be useful in the following cases:

- If not sure of having saved the point, command the robot to move to the point and check if it moves.
- If not sure if increasing or decreasing the z coordinate, command the robot to move up and check if the z coordinate increases or decreases.

Finally, also this activity may be supported by signals, such as pending stripes, light beams or applications of augmented reality, to make the burners axis and contours always visible and to intuitively illustrate the correct sign, minus or plus, for editing the z coordinate.

Position and posture

As mentioned while illustrating the modification of burners points, the position and posture is important.

Especially when the points located towards the back end of the product, such as burner 3, are concerned. Figure 25 illustrates incorrect and correct positions and postures.

It may be useful to prepare signals (footprints, beams, augmented reality) to guide the learners to assume the right position and posture. Potentially, by adding wearable sensors, ergonomic guidance could be added.

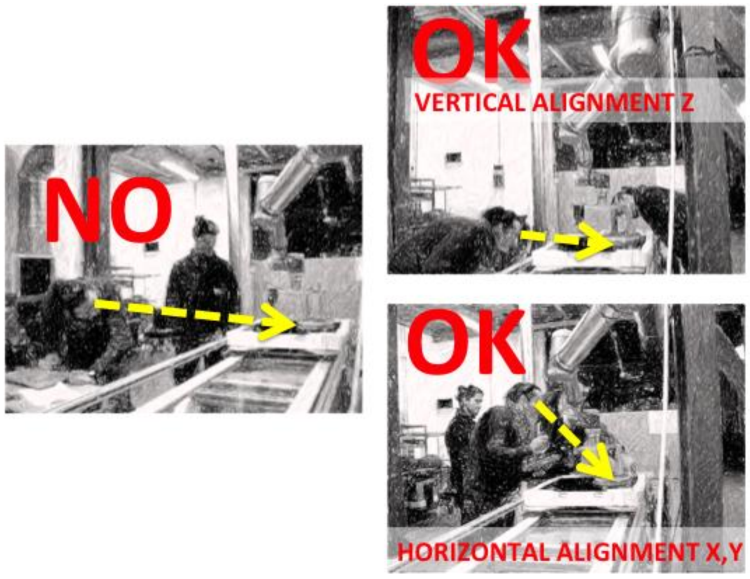


Figure 25 Incorrect and correct position and posture

4.4 Concluding remarks on the Whirlpool industrial case study

The observation of a human expert while teaching and demonstrating and of a learner while experimenting has allowed the extraction of knowledge about the task of reprogramming a collaborative robot in order to adapt it to a novel product.

In particular patterns for changing the waiting points have been identified and some mistakes that a beginner might make have been spotted. On this basis, a set of guidelines have been elaborated to support the learners in acquiring the reprogramming skill and avoiding the commonest errors.

Additionally, some suggestions have been provided about possible tools and technologies that may guide the reprogramming activities both in training and on the job (Figure 26). Future work might extend this analysis to a wider range of cases, on the one hand, and experiment the effect of the guidelines on the learning performances of the manufacturing workers.

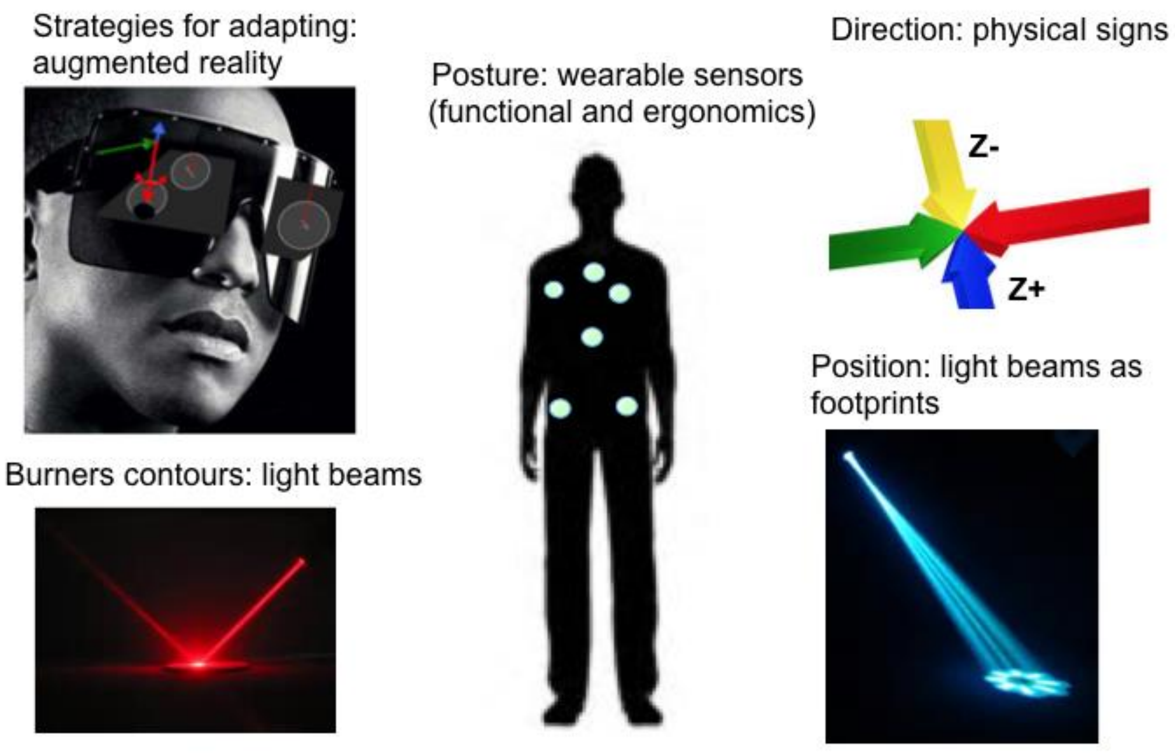

Strategies for adapting: augmented reality
Posture: wearable sensors (functional and ergonomics)
Direction: physical signs
Z-
Z+
Position: light beams as footprints
Burners contours: light beams

Figure 26 **Examples of supporting technologies**

5 Conclusions

A practical system for capturing and interpreting human expert knowledge in a flexible manufacturing environment has been developed in this task. Although the system has been developed for product changeover, the overall concept is applicable to other related scenarios such as new equipment introduction, reconfiguration and maintenance.

The core of the system is an intelligent decision support system, which proposes possible courses of actions to operators based on observed machine states, and also evaluates the benefits of the proposed actions. The suggestions are made according to machine and product specific data and the data gathered from operators in past changeover operations.

A study on how to extract knowledge from expert and inexperienced operators was conducted in Whirpool and it is reported in this deliverable. Althouth there is no link between the intelligent decision support system and the study at this stage, there is a great potential for actual implementation and evaluation of concepts. Experimental work to determine the effectiveness of the developed system will be explored in WP 7 or WP 9.

References

[1] Terwiesch, C., and Bohn, R.E. (2001) Learning and process improvement during production ramp-up. *International Journal of Production Economics*, 2001, 70 () 1–19.

[2] Doltsinis, S. C., Ferreira, P., and Lohse, N. (2014). “An MDP Model-based Reinforcement Learning Approach for Production Station Ramp-up Optimization: Q-learning Analysis.” IEEE Transactions on Systems, Man, and Cybernetics: Systems 44 (9): 1125–1138

[3] Oates R., Scrimieri D., Ratchev S. Accelerated ramp-up of assembly systems through self-learning (2012). *In: Ratchev S. (eds) Precision Assembly Technologies and Systems. IPAS 2012. IFIP Advances in Information and Communication Technology*, Springer, Berlin, Heidelberg, 2012, 371, 175-182.

[4] Konrad, K., Hoffmeister, M. Zapp, M., Verl A., and Busse J. (2012). Enabling fast ramp-up of assembly lines through context-mapping of implicit operator knowledge and machine-derived data. *6th International Precision Assembly Seminar*, Chamonix, France, February 12-15, 2012.

[5] Doltsinis S. C., Ratchev S., and Lohse N. (2013). A framework for performance measurement during production ramp-up of assembly stations. *European Journal of Operation Research*, 2013, 229 (1) 85-94.

[6] Glock, C. H., Jaber M. Y. and Zolfaghari, S. (2012). Production planning for a ramp-up process with learning in production and growth in demand. *International Journal of Production Research*, 2012, 50 (20), 5707-5718.

[7] Glock, C. H. and Grosse E. H. (2015). Decision support models for production ramp-up: A systematic literature review. *International Journal of Production Research*, 2015, 53 (21), 1-15.

[8] Fjallstram,S. K. Safsten, K., Harlin, U., and Stahre, J. (2009).Information enabling production ramp-up. *Journal of Manufacturing Technology Management*, 2009, 20 (2), 178–196.

[9] Doltsinis, S. C. and Lohse N. (2012). A Model-Free Reinforcement Learning Approach Using Monte Carlo Method for Production Ramp-Up Policy Improvement - A Copy Exactly Test Case, *Proceedings of the 14th IFAC Symposium on Information Control Problems in Manufacturing* Bucharest, Romania, May 23-25, 2012

[10] Doltsinis, S., Ferreira, P. and Lohse N. (2012). Reinforcement learning for production ramp-up – A Q-batch learning approach, *11th International Conference on Machine Learning and Applications*, Boca Raton, USA 1, 12-15 Dec, 2012.

[11] Ennen, P., Reuter, S. Vossen, R. and Jeschke, S. (2016). Automated production ramp-up through self-learning, *Procedia CIRP*, 2016, 51, 2016, 57-62

[12] Henry, J. R. (2013). *Achieving lean changeover: putting SMED to Work*. Boca Raton: CRC Press, 2013, 9781466501751

[13] PEFoRM Project Deliverable D2.3 - *Specification of the Generic Interfaces for Machinery, Control Systems and Data Backbone*. 2017.

[14] IFR International Federation of Robotics. (2016). *World Robotics Report 2016: European Union occupies top position in the global automation race .*
IMC-AESOP Project. (s.d.). Tratto da http://imc-aesop.org

[15] Tobe, F. (2017). *Collaborative robots are broadening their marketplaces*. Tratto da The robot report: https://www.therobotreport.com/news/collaborative-robots-are-broadening-their-market-spheres

[16] Spiegel, R. (2017). *Why Collaborative Robots Are Spiking in Sales*. Tratto da DesignNews: https://www.designnews.com/automation-motion-control/why-collaborative-robots-are-spiking-sales/159112324752523

[17] Edwards, D. (2016). *Collaborative robots are driving the global market, says Universal Robots*. Tratto da Robotics & Automation News: https://roboticsandautomationnews.com/2016/09/30/collaborative-robots-are-driving-the-global-market-says-universal-robots/7508/
EMC2-Factory. (s.d.). Tratto da http://www.emc2-factory.eu

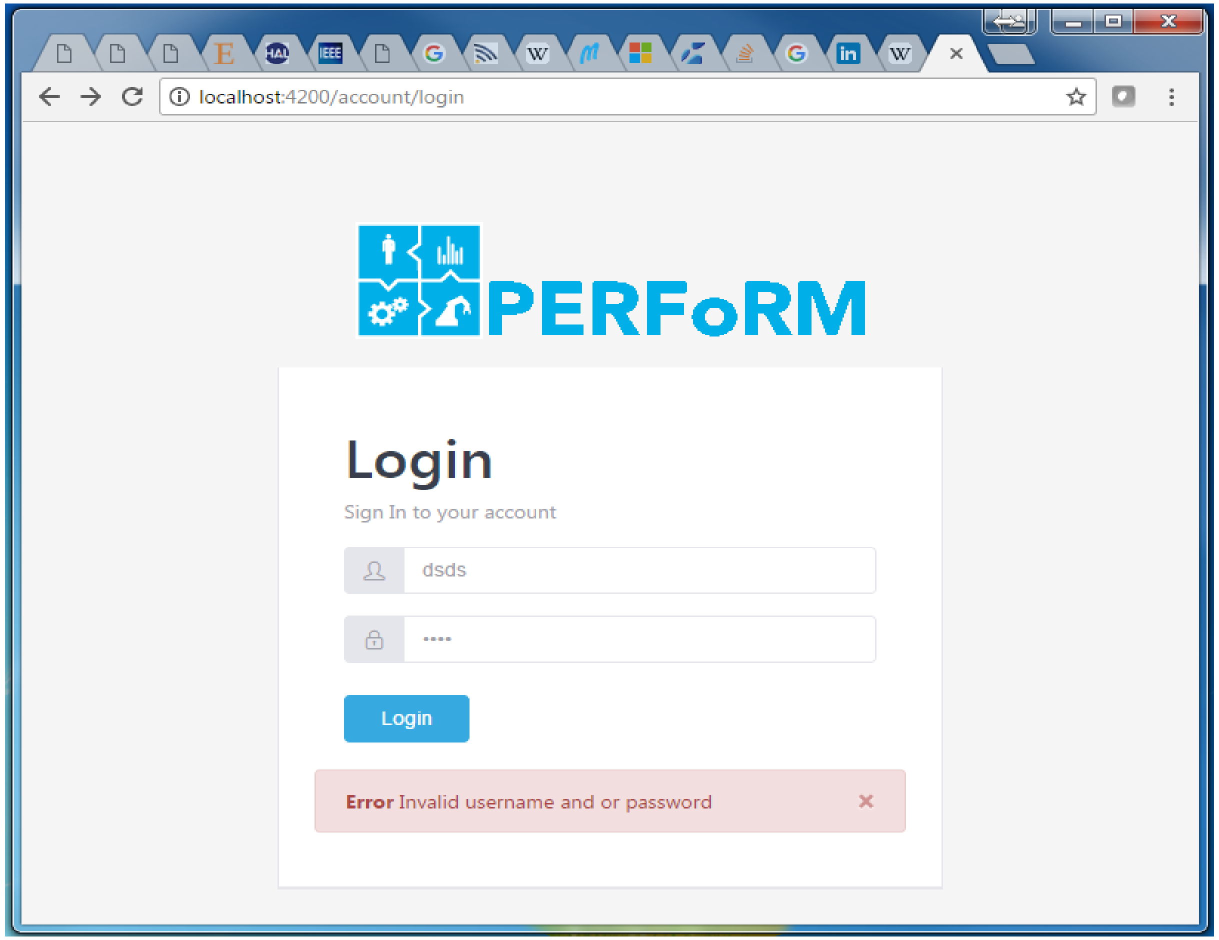
[18] Rozo, L., Sylvain, C., Caldwell, D. G., Jimenez, P., & Torras, C. (2016). Learning Physical Collaborative Robot Behaviors from Human Demonstrations. *IEEE TRANSACTIONS ON ROBOTICS.*

[19] Andersen, R., Schou, C., & Damgaard, J. (2016). Using a Flexible Skill-Based Approach to Recognize Objects in Industrial Scenarios. *ISR 2016 (June 21 – 22, 2016, Munich, Germany).*
ARUM. (s.d.). Tratto da http://www.arum-project.eu

[20] KUKA. (2014). *Teaching by demonstration*. Tratto da YouTube: https://www.youtube.com/watch?v=PEft1BTHBQU

[21] Biggs, G., & MacDonald, B. (2003). A Survey of Rob ot Programming Systems. *Proceedings of the Australasian conference on robotics and automation.*

[22] Liu,, L., Jiang, Z., Song, B., Zhu, H., & Li, X. (2016). A Novel Method for Acquiring Engineering-Oriented Operational Empirical Knowledge. *Mathematical Problems in Engineering* .



localhost:4200/account/login

Login

Sign In to your account

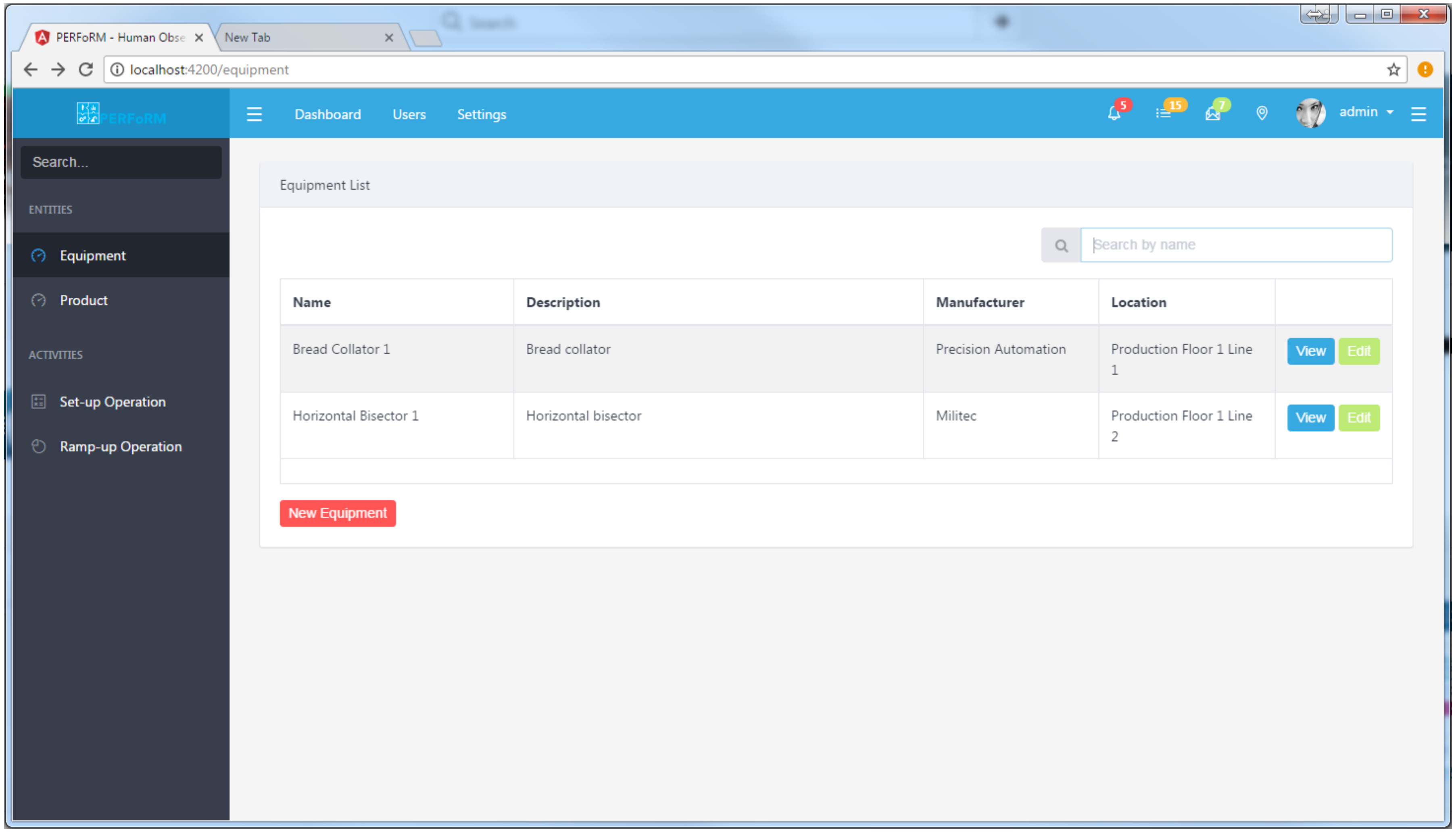
dsds

••••

Login

Error Invalid username and or password ×

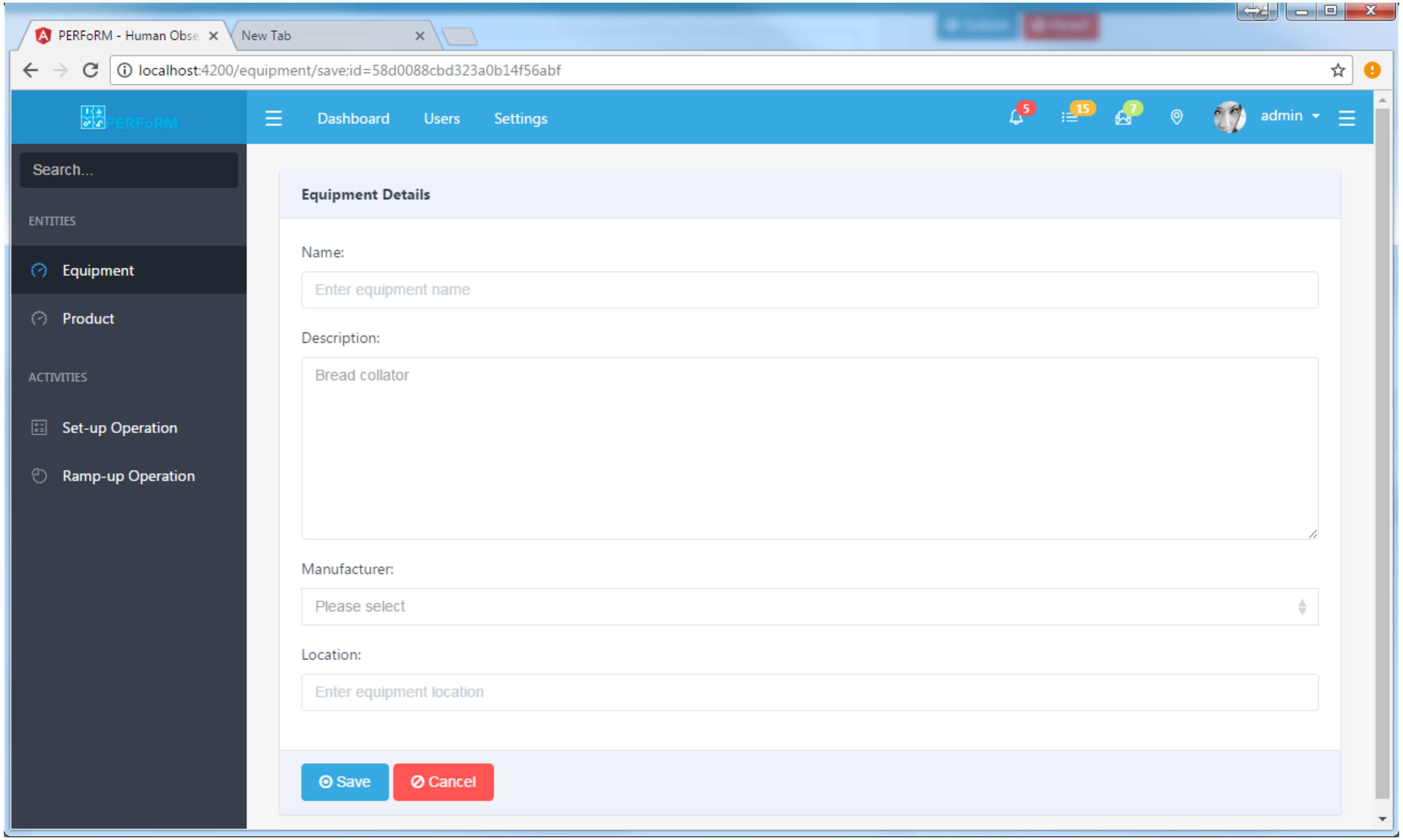
Login page



Equipment List

Name	Description	Manufacturer	Location	
Bread Collator 1	Bread collator	Precision Automation	Production Floor 1 Line 1	View Edit
Horizontal Bisector 1	Horizontal bisector	Militec	Production Floor 1 Line 2	View Edit

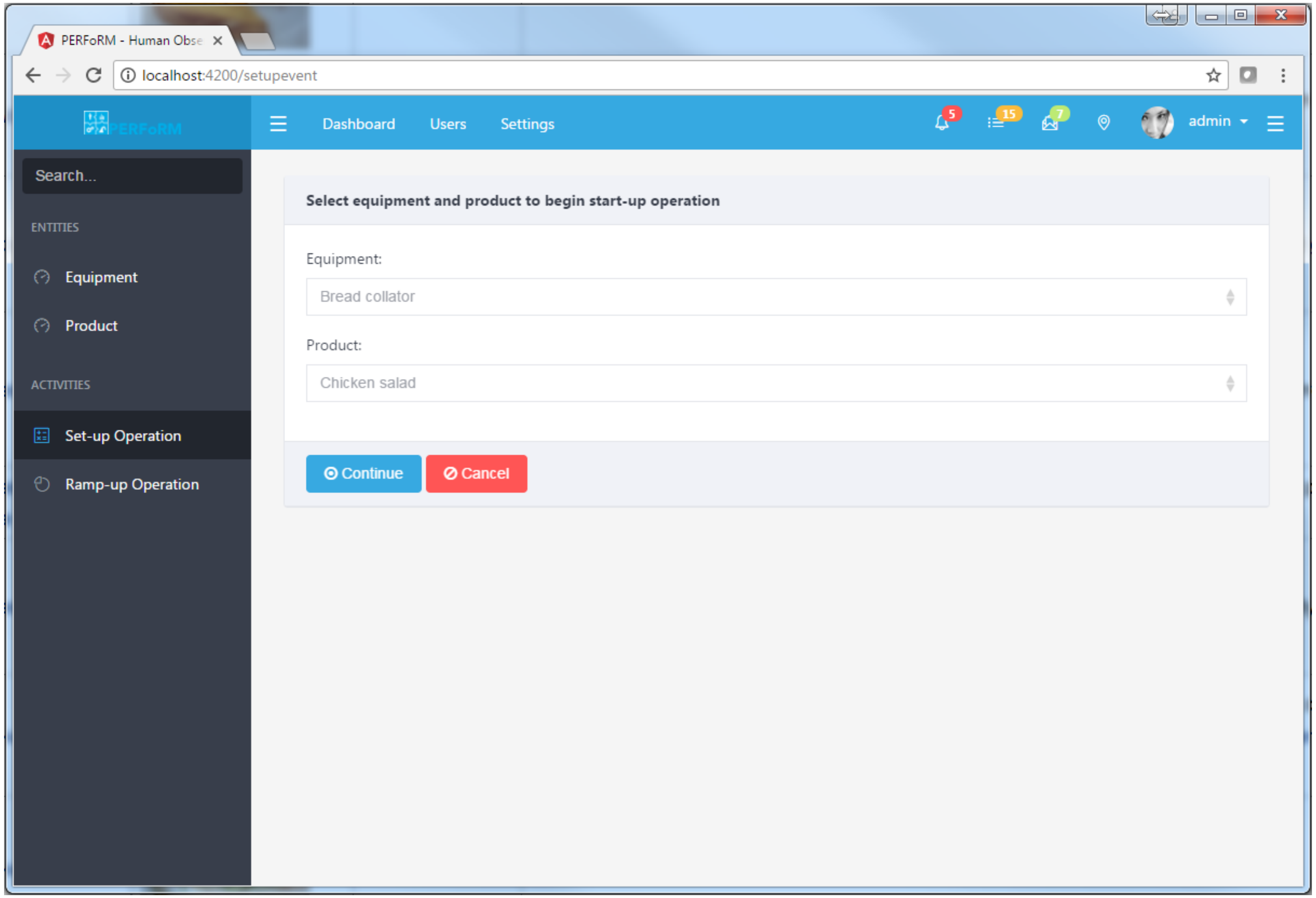
Equipment list page



Equipment Details

Name:	Enter equipment name
Description:	Bread collator
Manufacturer:	Please select
Location:	Enter equipment location

Equipment creation and update page

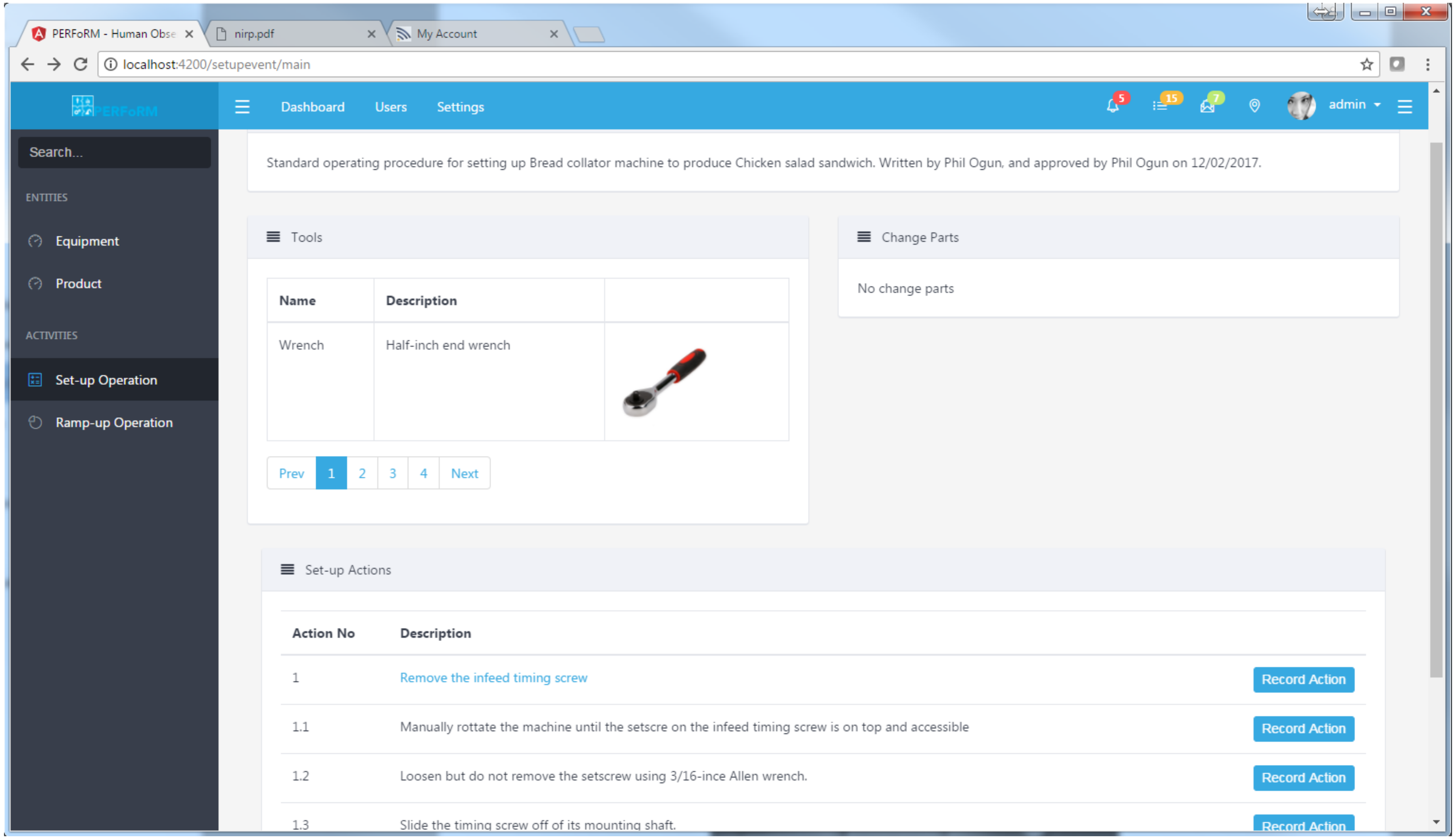


Select equipment and product to begin start-up operation

Equipment: Bread collator

Product: Chicken salad

Set-up operation equipment and product selection page



Standard operating procedure for setting up Bread collator machine to produce Chicken salad sandwich. Written by Phil Ogun, and approved by Phil Ogun on 12/02/2017.

Tools

Name	Description	
Wrench	Half-inch end wrench	

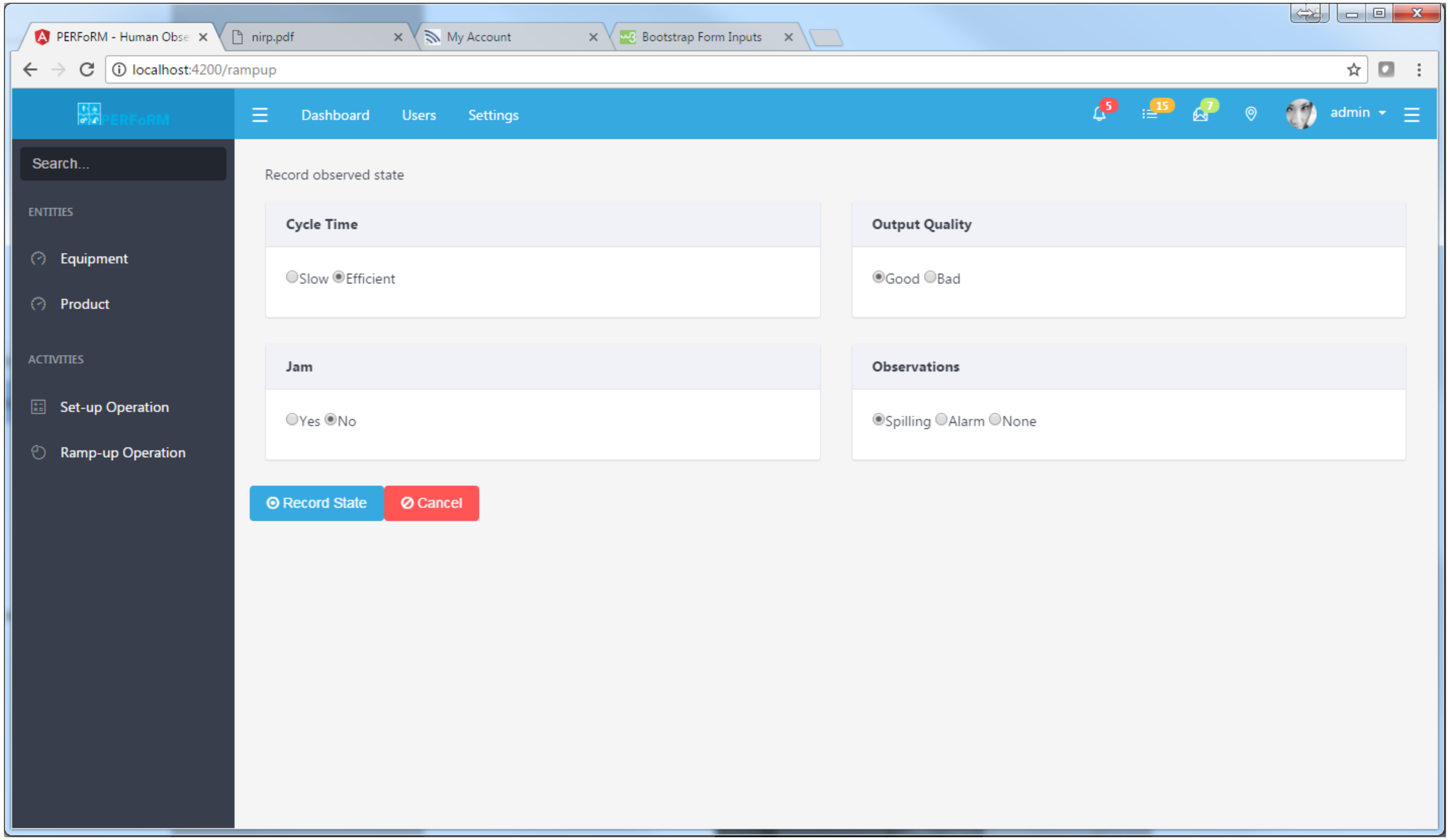
Change Parts

No change parts

Set-up Actions

Action No	Description
1	Remove the infeed timing screw
1.1	Manually rottate the machine until the setscre on the infeed timing screw is on top and accessible
1.2	Loosen but do not remove the setscrew using 3/16-ince Allen wrench.
1.3	Slide the timing screw off of its mounting shaft.

Set-up procedure and operator action recording page



Record observed state

Cycle Time: Slow [✓ Efficient]

Output Quality: [✓ Good] Bad

Jam: Yes [✓ No]

Observations: [✓ Spilling] Alarm None

Ramp-up state recording page