WORKPLACE MODELS

Deliverable nº: D3.2

EC-GA Number: 723737
Project full title: HUman MANufacturing

	Work Package:	**WP3**	
	Type of document:	**Deliverable**	
	Date:	**29/03/2018**	
Grant Agreement No 723737			
Partners:	SUPSI, SINTEF, HSZ, HX, IUVO, LMS, SSSA, UCL		
Responsible:	**SUPSI**		
Title:	**D3.2. Workplace models**	Version: 1.0	**Page: 0 /40**

Deliverable D3.2
Workplace models

DUE DELIVERY DATE: M15

ACTUAL DELIVERY DATE: MARCH 2018 (M18)

Document History

Vers.	Issue Date	Content and changes	Author
0.1	30.10.2017	Table of Content	Andrea Bettoni, Marco Cinus
0.3	04.12.2017	Class descriptions and APIs	Marco Cinus
0.4	08.03.2018	Internal Review	Marino Alge
0.5	09.03.2018	Correction based on review	Marco Cinus
0.6	15.03.2018	Internal Review, executive summary, introduction and conclusions	Andrea Bettoni
0.7	16.03.2018	Finalization	Marco Cinus
0.8	28.03.2018	Correction based on reviews	Marco Cinus
1.0	28.03.2018	Final revision	Marco Cinus

Document Authors

Partners	Contributors
SUPSI	Marco Cinus, Andrea Bettoni, Marino Alge
HX	Rabah Djennadi
SINTEF	Felix Mannhardt
LMS	Anna Kourvouniari

Dissemination level: CONFIDENTIAL

Document Approvers

Partners	Approvers
HX	Rabah Djennadi
IUVO	Francesco Giovacchini

Executive Summary

This deliverable reports on the development of the HUMAN Shared Data Model that provides a worker-centric representation of the production environment considering the main entities involved: (i) the worker, as the center of the carried out activities and whose physical and mental well-being is the focus of the HUMAN system; (ii) the factory, with its logical and physical production resources; (iii) the production process, along with the skill and capacity demands it carries, and (iv) the events that take place in the shop-floor dictating its pace.

The SDM acts as a common agreed baseline for data integration and exchange within the modules operating in the HUMAN system. A bridge is so built that allows data providing modules connected to the Sensing Layer of Task 3.1 (environmental sensors, the wearable apparatus, etc.) to arrange the provided information so that it is made available to the all the data consumer modules such as the short- and long-term reasoning, the WP4 services and the insight intelligence.

The model here presented is based on a review of the existing human-centered factory models that previous research and standardization actions, also with the contribution of project members, have developed in the last year as it is reported in § 2.

Follows in § 3 a detailed description of the classes envisioned for the SDM, subdivided per area, and provided with an explanation of the semantic they hold and of the connection they present one another.

In § 4 the problem of access to the data structures of the SDM is introduced by defining a first version of the API that will be further developed in the implementation stage of the work package thus ensuring that the HUMAN services will be able to exploit the model for their own purposes.

Finally some conclusions are drawn in § 5 about the achieved level of interoperability and the validity of the developed models and next implementation steps are briefly presented.

TABLE OF CONTENTS

Contents

Acronyms

Acronym	Explanation
API	Application Programming Interface
EXO	Exoskeleton Service
KIT	Knowledge In Time Service
KSN	Knowledge, Skill & Needs
LTR	Long-Term Reasoning
PIS	Physical, Intellectual and Sensorial capacities
SII	Shopfloor Insight Intelligence
WOS	Workplace Optimization Service

1. Introduction

Goal of Task 3.2 is the digital representation of the workplace from a human-centric perspective with the aim to create a data environment that supports the delivery of services looking for an improved well-being or more comfortable and safer working conditions and, consequently, more quality in the production operations. This means that the elements interacting with the worker (and the worker him/herself) need to be characterized, first by identifying which are the areas of interest in the modeling action and then by deeply investigating what are the needed elements and how they interact.

Several boundary conditions need to be considered when building this kind of shop floor digital representation to ensure it serves the scope of the HUMAN system. First, the perspective of the HUMAN services, that rely on the digital representation to retrieve (and add) data they consume, has to be taken into account. Second, an holistic nature for the representation needs to be pursued to ensure that the different industrial domains included in the HUMAN scenarios can be easily mapped in the developed data model.

In order to meet these requirements the Shared Data Model designed and presented in this document takes four broad areas as reference for its structure:

- in the worker area it includes all the information about the main character of the HUMAN system by classifying his/her skills to be spent in the working activities, the physical, sensorial and intellectual capacities by exploiting the concept of work trait, an abstract quality who can be applied to any semantic field both in its static and dynamic condition;
- in the factory area it describes the cyber-physical nature of the shop-floor by describing the physical machinery and equipment operating in the production system, their arrangement in production lines and their three-dimensional representation for usage in virtual environments;
- in the task area it provides the counterpart to the worker representation by defining the hierarchical structure at the base of the production process representation where jobs, tasks and operation, together with the worker-related demand they bring, concur to make up the production process;
- in the event area it keeps track of all what happens in the shop-floor during the period of activation of the HUMAN solution so that, on the one hand, reasoning on the actual working conditions and post-mortem analysis can be conducted on a reliable data base while, on the other hand, coordination of HUMAN interventions can be easily orchestrated by leveraging on an event-driven situation awareness.

On top of this representation, the HUMAN algorithms, each described in the related service deliverable, implement solutions for enhancing the working conditions, ensuring safety of operations and achieve better performances of the production system.

A particular aspect of the digital representation here presented, is the sensitivity of the data stored in the model repository. Even though this will be addressed during the deployment of the HUMAN system at the end-users by establishing industry-level security standards and not at the level of the data model, it is worth reminding that not all data collected in the shop floor will be made persistent within this digital model. In particular, live data of physiological parameters, that are usually considered as medical data and thus are particularly sensitive, are not stored here where instead it is any event, elaborated by the system starting from those parameters that will be made persistent in the data model with an obvious reduction on the level of sensitivity of the information stored.

Since the data model will evolve during the whole project, an updated version of it will be reported in D3.3 and will be available on the project Redmine site.

2. Review of Existing Human-Centered Factory Models

The emergence of factory data models that consider the human dimension as baseline perspective has taken place in the recent years as more and more companies have understood they need to leverage on the workforce potential and adapt production systems so that they become an extension of that potential (Choi et al., 2000).

These models are also important to support human-machine collaboration, especially in cognitive tasks, in context where automation plays a relevant role. To this regard, Borst (2016) discusses that without a model to share information between operators and the other operators or the automated agents of the environment, it is impossible to reach a shared understanding of the ways task can be accomplished.

In modeling a human-centered factory data model the characterization workers is an inescapable requirement (Bettoni et al., 2014). The main area of modeling are knowledge, skills and personal needs (KSN) and physical, intellectual and sensorial capacities (PIS). In this way a complete worker representation is obtained also offering the interface for providing relevant information to any factory agent or tool for the purpose of delivering functionalities of various nature. KSN and PIS are in fact relevant in the design of workplaces, in human resources management, in the correct design and allocation of jobs based on risk prevention, in the planning of training programs and, last but not least, in impairment conditions that limit those functionalities (Peruzzini & Pellicciari, 2017). Moreover, these data models should guarantee orthogonality with respect to the characterization of both the workers and the working activities, thus characterizing also the job structure with its tasks and the related demand. The two antagonist points of view, worker and factory, are in fact the information space where to perform comparisons aimed to the identification of potential matching between a worker's quality and some corresponding work-activity requirements or constraints. Some concepts embody the core of any human-centered factory model:

- *Worker*, a *Person* employed in a *Company*;
- *Job*, a sequence of *Task*s, to be allocated on a single worker at a workstation;
- *Task*, a sequence of *Operation*s, which compose a meaningful production step;
- *Operation*, an elementary sequence of actions;
- *Workstation*, the physical space where a single *Worker* carries out his/her *Job;*

However, usually companies struggle in the definition of the data characterizing the worker since it is a practice going beyond the traditional domain of human resources. That is probably the reason why most of the existent services, both open and commercial, are oriented to support career management. Consequently their modeling of worker characteristics are strongly dependent on some standardization of common concepts which support the definition of "occupations" on a side,

and “competence”, “skill” and “needs” on the other (Zhao et al., 2015; Fernández-Sanz et al., 2017). Both at EU and non-EU level, there are standardization initiatives (Blázquez, 2014; Ceclan, 2016) that have produced usable results. The ESCO initiative represents an on-going EU effort to integrate most of the existing national standards in Europe targeting in particular the semantic harmonization of the equivalent terms adopted in different EU languages to indicate specific occupations and worker characteristics. Another example is the O*Net initiative (National Center for O*NET Development, 2017), considered the US nation’s primary source of occupational information, whose database contains information on hundreds of standardized and occupation-specific descriptors. In terms of suitability for the HUMAN data modeling solution O*Net prevents several advantages when compared to other standards: first O*Net provides an assessment structure based on the evaluation of level and importance of already available list of manifold skills; second, the level of granularity offered by O*Net is closer to the operational level needed for this work when compared to other more job search-oriented standards; third, the O*Net classification itself is converging to the Standard Occupational Classification (SOC) System which is already adopted, with some variations, by some of the European governments.

Moreover, the O*Net standard has been used by the authors of this document as a starting point for the shared data model within the EU-funded project MAN-MADE aimed at defining new socially sustainable and adaptable workplaces where the human dimension is a key cornerstone. A central concept in the MAN-MADE shared data model is the so-called work-trait, which is an abstraction of some generic “quality” of the working activity. Based on that, both the workers and the working requirements are described in terms of their adherence to specific work-traits. A work-trait is simply described in terms of its coded name, a description and a classification, plus a possible reference to an external authority source, usually an acknowledged standard.

This approach allows a company to define its own set of pertinent work-traits, which could possibly comply with an existing standard or to some local classifications, based on organizational convenience.

In the development of the HUMAN Shared Data Model, the presented standards and, in particular O*Net will be used to relate the developments with an on-going convergence process taking place beyond the project boundaries. Also the experience and outcomes of the previous experience of key partners in the data model formalization and design will be reused and expanded so as to maximize exploitation of funding and effort from the EU and for building on a solid and validated knowledge basis.

3. HUMAN Shared Data Model

The scope of the HUMAN Shared Data Model (SDM) is the representation of the factory from a worker centric perspective meant to achieve an integrated and holistic view of the worker-factory pair. This holistic view will enable to reason considering the whole picture to envision interventions aiming to harmonize workers' activities with factory behavior both in the short- and long-term.

An overall picture of the data model is shown in Figure 1. The presented model can be logically divided in 4 distinct areas. The first area, with the blue background classes, is the representation of the *worker*. The second area, with the pink background classes, represents the *factory* related elements. The third area, with the green background classes, is the design of *jobs*, task and their scheduling. Last but not least, the area with orange background classes, is the formalization of the *events* that happens to the actors inside the system. Classes in red are *common* parts between two or more parts of the data model, like *Trait* and *Measure.*

In the rest of this section we are going to explain, for each of the data model logical segments, the rationale that has driven the modeling and to provide a detailed description of the classes, pointing out which are the fields and the relations to the other classes in the system.

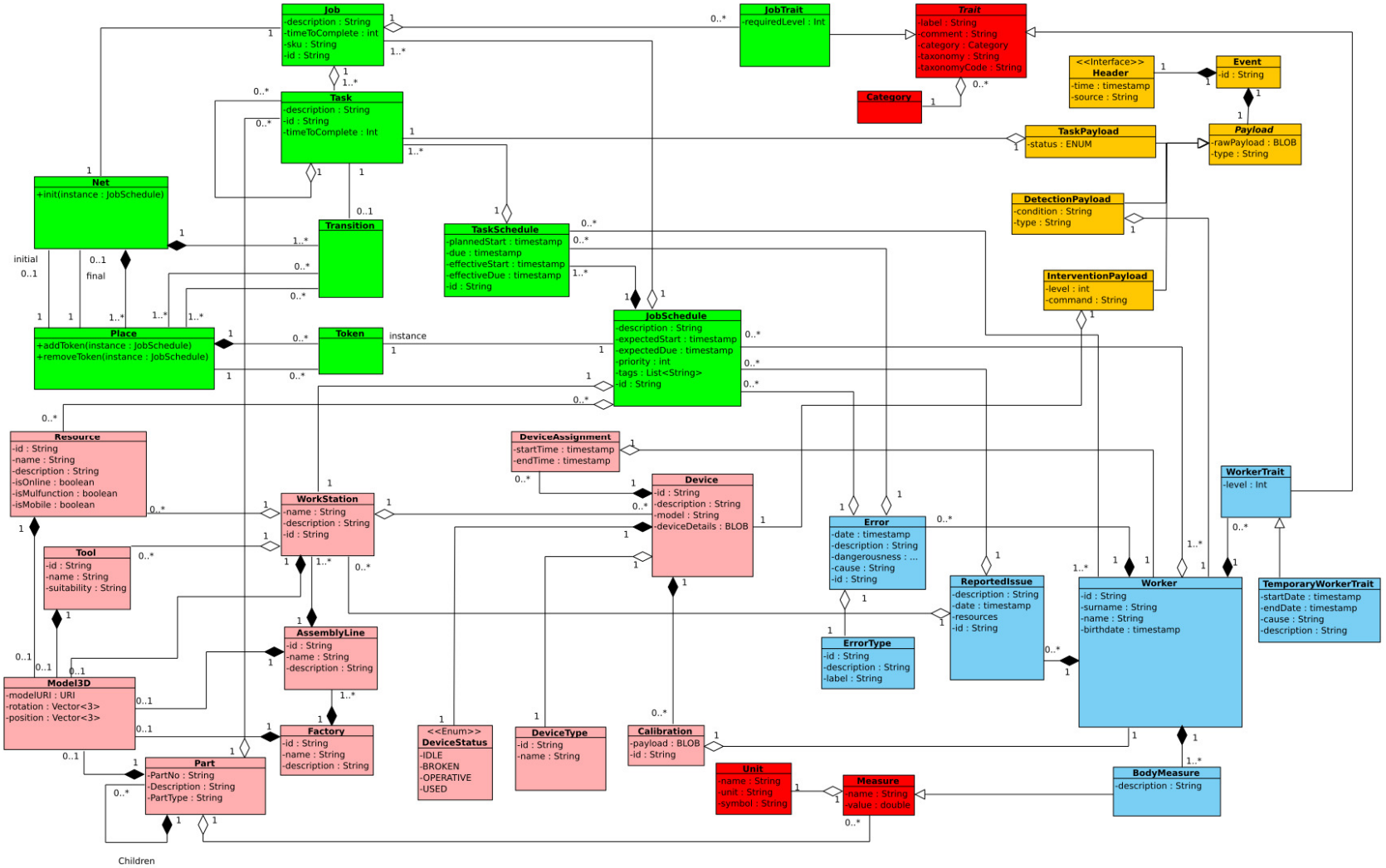

Figure 1 UML Class diagram of the HUMAN Shared Data Model

3.1 COMMON CONCEPTS

This section is going to describe two parts of the data model that cannot be clearly be assigned to one of the parts of the data model.

3.1.1 MEASURE

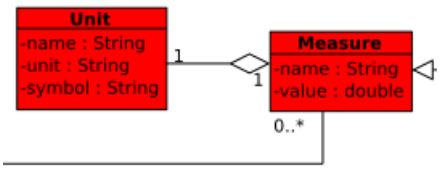

Figure 2 Measure class

A common concept that arises in two parts of the data model is the notion of measure. To facilitate the handling of this concept two classes have been modeled: *Measure* and *Unit*.

A *Measure* is composed by *name* and *value*. It has a reference to a *Unit*.

The *Unit* is composed by a *name*, a *unit* string and a *symbol*. This allows to define every type of measure (e.g.: dimensions, volumes, etc.).

3.1.2 TRAIT

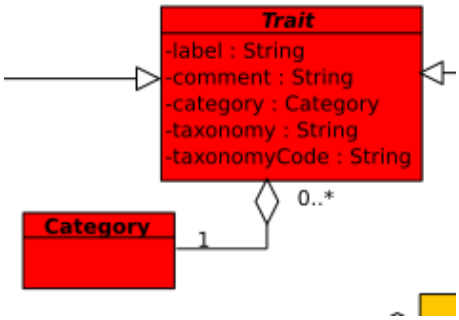

Figure 3 Trait class

A central concept in the current model is the class called *Trait*. A *Trait* is a generic representation of a worker's skill (something the worker can do), a worker's knowledge (something the worker knows) or, when referred to a *Job*, a job demand (a requirement for the job). It is an abstract class which is then implemented by the concrete classes which are dedicated at describing the worker characteristic and the job demands.

The class is abstract and extended by *JobTrait* and *WorkerTrait*. This distinction has been created in order to avoid assignment problems between traits (e.g. the visual acuity trait could be used to better setup the KIT interface).

Name	Type	Comment
Label	String	The name of the trait, it is also its unique identifier
Comment	String	A human readable description of the trait
Taxonomy	String	The taxonomy where the trait belongs to. It is used to related the HUMAN trait with other taxonomy like O*Net or ESCO
TaxonomyCode	String	The unique identifier of the related taxonomy.

Table 1 Description of the attributes of the *Trait* class

Class	Relation Type	Multiplicty
Category	Aggregation	0..* to 1
WorkerTrait	Superclass	
JobTrait	Superclass	

Table 2 Description of the classes with inheritance relationship to the *Trait* class

The class *Category*, needs little explanations, it is just an enumeration defining the type of the trait, that can be Skill, Knowledge, Capacity.

3.2 WORKER MODEL

This section presents the part of the data model dedicated to the worker. It will first highlight the results of the theoretical investigation on the model topics carried out through brainstorming, comparison to literature examples and verification of the validity of the assumptions in collaboration with the project end-users. The core concepts related to the worker are so presented and, following, a detailed description of data model itself is provided, highlighting which are the classes, their fields and the relations between them.

3.2.1 RATIONALE

3.2.1.1 PERSONAL DATA

The personal data, such as name, surname, gender, age and, of course, a unique identifier are mostly needed to let the system recognize who the worker is and to help the user identify the workers. This kind of data is sensitive, so it has to be protected, with particular regards when it is possible to link this data with medical or other sensitive information, as it then permits to made sensitive information available to people who are not legitimated to have access to it. This data will be protected using standard security tools adopted in industry. More details about the security mechanism are elicited and proposed in D3.3.

3.2.1.2 SKILLS, KNOWLDEGE, CAPACITIES

In order to understand what a worker is able to do and what proficiency level it has, the SDM has to formalize in a standard manner the knowledge, skills and capacities of the workers. The next subsections will provide more information about the formalization and standardization process of knowledge, skills and needs. The concept semantic subdivision is an heritage of the authors experience in the previous EU-funded project MAN-MADE.

Even though skills, knowledge and capacities belong to different semantic domains, their use in the services for matching what jobs require (demand) with what workers can offer (supply) is quiet similar. For this reason this abstract quality concept has been condensed in a single *WorkerTrait* class, regardless of the nature of that quality. In terms of measuring scale meaning, a *WorkerTrait* has always a positive connotation, i.e. walking ability of 100 (on a 1 to 100 scale) means that the worker has no walking disabilities. In the same way a welding skill of 100 means that the worker is an expert in the field. Proper reference descriptions of skill levels need to be setup to guide an objective characterization of worker traits.

A best practice in the use of the SDM is to assign worker traits with a one-to-one relationship with the job / task trait it has to be compared against, so that it is possible to perform the matching between the needs of the factory and the worker's skill set. Eventually it is worth noting that some decisions for the activation of one or more services require that a value is defined for some relevant worker traits. This calls for the definition of default values in case a worker trait is missing and/or design of system behavior in this data missing condition.

3.2.1.2.1 SKILLS

The skills of the worker are the ability to do something well, also called expertise. They can be seen as a worker capacity that can be taught and have a direct impact on the work performed by the worker. Examples of skills are: problem-solving, welding, complex-assembling, ability to use a specific tool, etc. Skill levels are used inside the Intervention Manager by the Short Term Reasoning Engine and the Long Term Reasoning Engine to better detect the deviation patterns or to select the most effective intervention . For instance a low skilled worker could receive more instructions from the KIT when he/she has to assemble a highly complex component as a result of the IM selecting a different intervention based on the skills held by the recipient. Another example could take place during the Long Term Action deviation recognition, as the Long Term Reasoning Engine weights differently a repeated error of a skilled worker than a newbie error. This makes very important to keep an historical track of the skill evolution of every worker.

3.2.1.2.2 KNOWLEDGE

Knowledge of a worker is to be intended as awareness or familiarity gained by experience of a fact or situation. Some examples could be the knowledge of a security procedure, First Aid, some specific

knowledge of the production system (e.g. lean principles) or additional languages known by the worker.

3.2.1.2.3 CAPACITIES

The capacity has to be intended as the ability or power to do, sense or understand something. Physical capacities refer to the ability to lift heavy loads, fully extend arms, adopt certain postures, reach elements and so on. Sensorial capacities, like hearing and seeing, can be of high importance since they could impact the way that services interact with the worker. As an example, the KIT could take care of color blindness and use a suitable set of colors. For the EXO the amount of assistance provided could be different based on the worker strength . Intellectual capacities refer to the cognitive functional status (memory, autonomy, reasoning) or mental (emotional stability, personality traits).

The capacity concept is the positive variant of disability. As a result, a person with a disability that prevents him/her to stand on his/her feet for lengthy periods of time, will be modeled with a low score in the standing capacity.

3.2.1.3 ANTHROPOMETRY

Worker anthropometric characterization is at the core of ergonomic design of worker-centric workplace. This is reflected by the existence of several standards dedicated to its formalization such as ISO (2010), "Basic human body measurements for technological design. Body measurement definitions and landmarks", which is used as a reference in other standards as well as the realization of many anthropometric studies and anthropometric data base.

In HUMAN, since the Workplace Optimization Service cares about the optimization of the workplace also in terms of its ergonomic performance, it is necessary to include information about the anthropometric measures of the worker. The needed information can be dependent on the assessment type. For instance one of the most used ergonomics assessment models, called Rapid Upper Limb Assessment (RULA) entails the definition of only a subset of the data used by more extended Rapid Entire Body Assessment (REBA). Difference can be found also in the definition of anthropometric measures defined within workplace design applications (e.g. CATIA from Dassault uses a set of measures that is almost entirely aligned with the previously cited ISO standard). Therefore, in order to widen the HUMAN system applicability in many contexts, the set of stored measures is not statically defined so that new kind of measures can be added as the system evolves.

3.2.1.4 RELATION TO JOB / TASK

The last part of the worker area data modeling is dedicated to the links with the other areas and, in particular, with job area. This links are required to model the relations between a specific worker and one of the active job / tasks within the factory processes.

This allows to model concepts such as the expertise of a worker in performing a certain job or the last time a worker has been assigned to a specific position or job. The definition of these relationships is at the basis of the activation conditions for the HUMAN services (e.g. KIT refreshing job instructions to a worker who is assigned to a task he dropped a long time ago).

Another important factor to be considered in the relation between the worker and the job is the last time the worker did an error and the amount of errors done by the worker. This information is useful for the activation condition of long-term services like the WOS.

3.2.2 CLASS DESCRIPTIONS

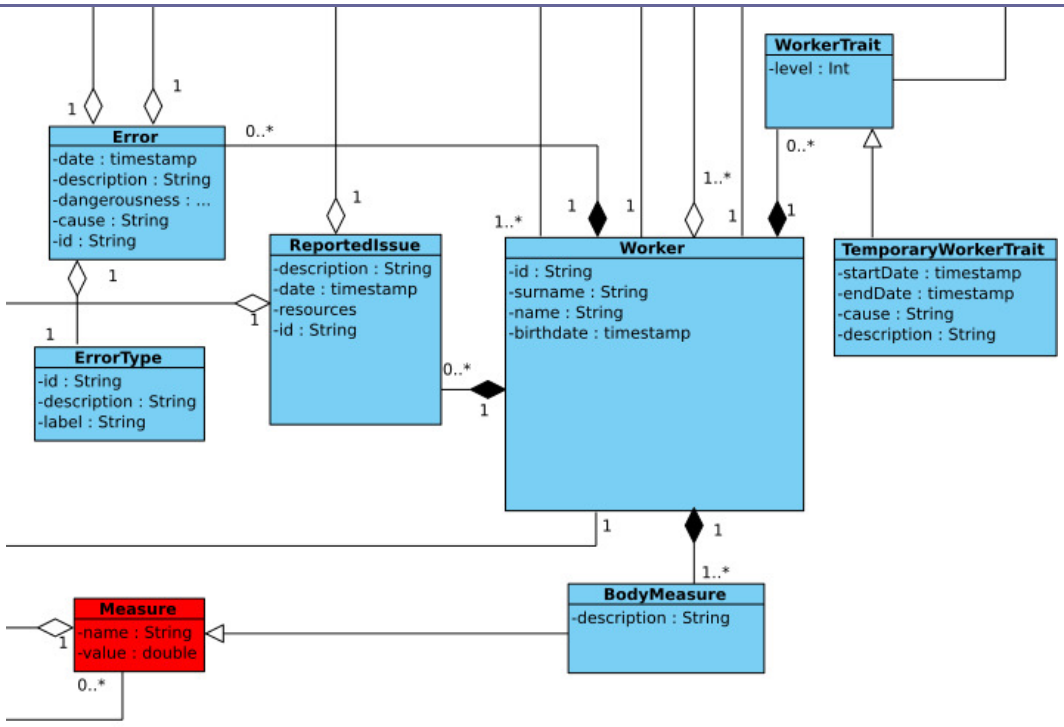

Figure 4 UML Class Diagram of the Worker area of the HUMAN Shared Data Model

Hereafter a description of the classes with their main attributes is proposed while getters and setters are omitted from the description.

3.2.2.1 WORKER

The *Worker* class is meant to hold all the personal data of the worker.

Name	Type	Comment
Id	String / UUID	The unique identifier of the worker.
Name	String	
Surname	String	
Birthdate	Timestamp	

Table 3 Description of the attributes of the *Worker* class

Related Class	Relation Type	Multiplicity
BodyMeasure	Composition	1 to 1..*
Error	Composition	1 to 0..*
ReportedIssue	Composition	1 to 0..*
WorkerTrait	Composition	1 to 0..*
Calibration	Aggregation (End)	1 to 1
DeviceAssignment	Aggregation (End)	1 to 1
DetectionPayload	Aggregation (End)	1 to 1
JobSchedule	Aggregation	1 to 0..*
TaskSchedule	Association	1..* to 0..*

Table 4 Description of the classes related to the *Worker* class

3.2.2.2 WORKERTRAIT

The *WorkerTrait* class is the concrete implementation of class *Trait*. It represents the same concept explained in trait declined from a worker perspective.

Name	Type	Comment
Level	Int	The current "skill" level of the worker. The level is normalized between 0 and 100, where 0 is the lowest level (e.g. is not able to weld) and 100 is the max level (e.g. welding master)

Table 5 Description of the attributes of the *WorkerTrait* class

Related Class	Relation Type	Multiplicity
Worker	Aggregation (End)	0..* to 1
Trait	Subclass	
TemporaryWorkerTrait	Superclass	

Table 6 Description of the classes related to the *WorkerTrait* class

3.2.2.3 TEMPORARYWORKERTRAIT

The *TemporaryWorkerTrait* class is a subclass of *WorkerTrait* which represents a *WorkerTrait* whose validity is limited in time. This can reflect a time-limited trait such as a temporary disability (e.g. a disability which is due to an accident with no long term consequences) or a skill which has to be renewed (e.g. cardiopulmonary resuscitation certification has to be renewed every two years to be valid).

Name	Type	Comment
Start	Timestamp	The starting date and time of the temporary

		trait
End	Timestamp	The ending date and time of the temporary trait
Cause	String	A human readable string illustrating the cause of the temporary trait
Description	String	A human readable description of the temporary trait

Table 7 Description of the attributes of the *TemporaryWorkerTrait* class

Related Class	Relation Type	Multiplicity
WorkerTrait	Subclass	

Table 8 Description of the classes related to the *TemporaryWorkerTrait* class

3.2.2.4 BODYMEASURE

The *BodyMeasure* class responds to the need to model in a generic, scalable and efficient way the anthropometric measures of a worker.

Name	Type	Comment
Description	String	A human readable description of the measure, could also contain a reference to a standard measure
Value	Double	The current value of the body measure

Table 9 Description of the attributes of the *BodyMeasure* class

Related Class	Relation Type	Multiplicity
Measure	Subclass	
Worker	Composition (End)	1..* to 1

Table 10 Description of the classes related to the *BodyMeasure* class

3.2.2.5 REPORTEDISSUE

One of the possibility given to the workers is the ability to report issues to the system. The class *ReportedIssue* holds the information of the worker's reporting activity. The references to the *JobSchedule* and *WorkStation* are optional. They are present if the raised issue has to reference a workstation or a problem occurred during a working shift.

Name	Type	Comment
Description	String	The issue's description
Date	Timestamp	The time and date when the user reported the issue
Resources	List<URI>	A list of URI, where each URI points to a picture,

		video and / or documents further detailing the problem

Table 11 Description of the attributes of the *ReportedIssue* class

Related Class	Relation Type	Multiplicity
Worker	Composition (End)	0..* to 1
JobSchedule	Aggregation	1 to 0..*
Workstation	Aggregation	1 to 0..*

Table 12 Description of the classes related to the *ReportedIssue* class

3.2.2.6 ERROR

The *Error* class stores the information about the errors performed by a worker during his/her shift. This information is not used to evaluate the worker, but instead it is used to offer better support and to detect problems which are common between multiple workers. Particular attention should be given to guarantee that this information cannot be extracted from the system.

Name	Type	Comment
Date	Timestamp	When the error happened
Description	String	A human readable description of the error
Dangerousness	Int	An integer indicating how dangerous is the error. A higher value indicates a more dangerous error
Cause	String	A human readable string detailing the cause of the error.

Table 13 Description of the attributes of the *Error* class

Related Class	Relation Type	Multiplicity
Worker	Aggregation (End)	0..* to 1
JobSchedule	Composition	1 to 0..1
TaskSchedule	Aggregation	1 to 0..1
ErrorType	Aggregation	1 to 1

Table 14 Description of the classes related to the *Error* class

The class ErrorType is an enumeration describing the type of error performed by the worker. This information is used to perform programmatic analyses (e.g. most frequent errors, correlation between working time and error type, etc.) of the errors registered in the system.

3.3 FACTORY MODEL

This section will describe the part of the data model dedicated to model the factory where the HUMAN solution is deployed.

3.3.1 RATIONALE

The modeling of the factory in the context of the HUMAN solution is aimed to characterize the critical features of tasks, workplaces and the workstations from an anthropocentric perspective. This makes area of the SDM here presented focused on the various levels of abstraction of the company production system from the factory high-level representation down to assembly line, workstation, tools and devices. A subsection of the data model is also dedicated to the product that is defined in terms of its bill of materials.

Particular attention is given to the aspects of the factory that are relevant for the HUMAN services. For instance, spatial location of assets and processes is modeled for its usage in virtual reality environments. The next subsections will provide more information about them.

3.3.1.1 FACTORY

A factory is a distinct productive environment within a company, where goods are manufactured or assembled.

3.3.1.2 ASSEMBLY LINE

An assembly line is a specific area in a factory that consists of a group of workstations including a series of workers, dynamically assigned to it, and machines that progressively assemble product parts on the semi-finished product.

3.3.1.3 WORKSTATION

The workstation is an abstraction of work centers that is the dedicated area where work of a particular nature is carried out, such as a specific location on an assembly line. The workplace can contain resource and tools, and its virtual model is used for assessing the workstation and the tasks performed in it.

3.3.1.4 RESOURCE & TOOL

Resources and tools are considered here as physical object belonging to a workstation and utilized in the execution of a process. A workstation can have more than on resource or tool.

The concept is mainly used to describe equipment and specifically machines in terms of their status, geometry and location inside a workstation.

A tool is a device used to carry out a particular operation. One or more tools are usually considered to belong to a specific workstation.

3.3.1.5 BILL OF MATERIALS

A Bill of Materials (BOM) is the back bone of each manufacturing system. BOM constitutes "recipe" for each finished product. Each "recipe" consists of information regarding materials, components and subassemblies composing the finished product, thus representing the product structure. Furthermore, all the standard information about each item, such as part number, description, unit of measure, etc., is to be considered.

3.3.1.6 DEVICE

The device data model is part of the factory model and is utilized to represent the usage of support devices (i.e. not dedicated to the interaction with the product, or the product parts but with the worker) in the shop floor, such as AR glasses, exoskeletons, wearable devices etc. It may be spatially connected to a workstation where it is operating and/or the worker using it.

3.3.2 CLASS DESCRIPTIONS

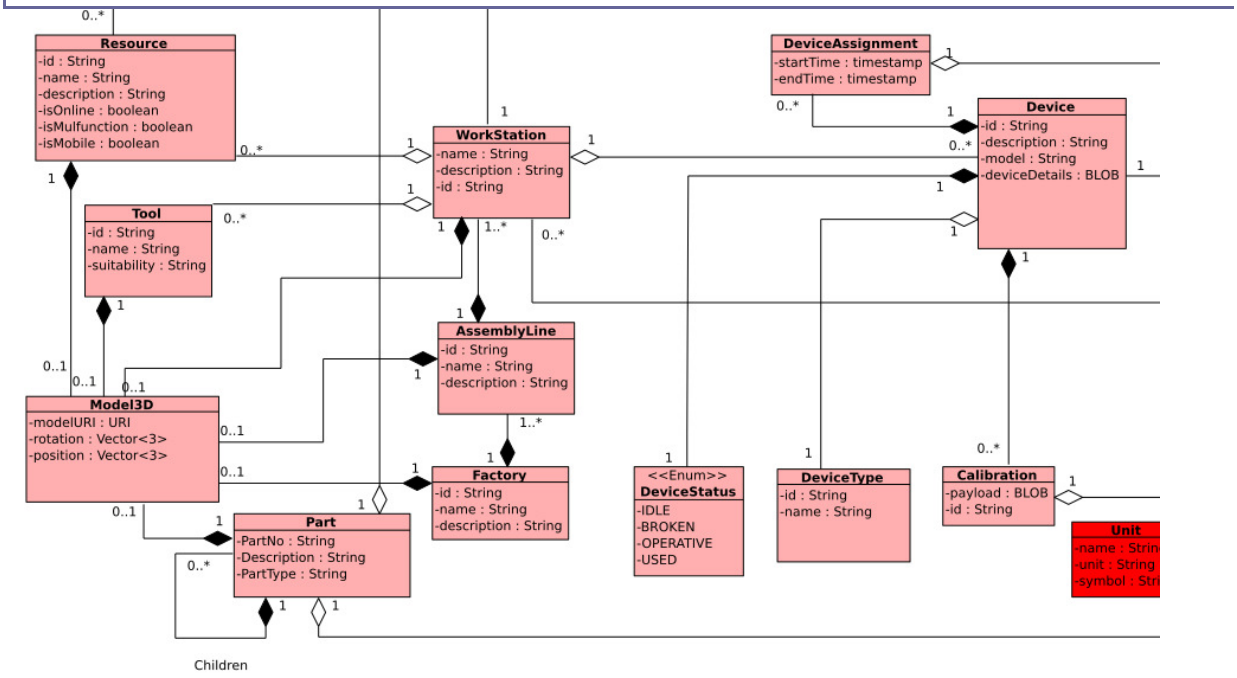

Figure 5 UML Class Diagram of the Factory area of the HUMAN Shared Data Model

Hereafter a description of the classes with their main attributes is proposed while getters and setters are omitted from the description.

3.3.2.1 FACTORY

The *Factory* class is meant to hold identification information for a specific factory.

Name	Type	Comment
Id	String UUID	The unique identifier of the factory
Name	String	
Description	String	A human readable description of the factory

Table 15 Description of the attributes of the *Factory* class

Related Class	Relation Type	Multiplicity
AssemblyLine	Composition	1 to 1..*
Model3D	Composition	1 to 0..1

Table 16 Description of the classes related to the *Factory* class

3.3.2.2 ASSEMBLYLINE

The *AssemblyLine* class holds all the information concerning the *Workstations* it includes. This class is also used to spatially locate the related 3D model within the virtual reality environment.

Name	Type	Comment
Id	String UUID	The unique identifier of the assembly line
Name	String	
Description	String	A human readable description of the factory

Table 17 Description of the attributes of the *AssemblyLine* class

Related Class	Relation Type	Multiplicity
Factory	Composition (End)	1..* to 1
Model3D	Composition	1 to 0..1
Workstation	Composition	1 to 1..*

Table 18 Description of the classes related to the *AssemblyLine* class

3.3.2.3 WORKSTATION

The *Workstation* class represents the fundamental unit where a job is executed.

Name	Type	Comment
Id	String UUID	The unique identifier of the workstation
Name	String	
Description	String	A human readable description of the workstation

Table 19 Description of the attributes of the *Workstation* class

Related Class	Relation Type	Multiplicity
AssemblyLine	Composition (End)	1..* to 1
Model3D	Composition	1 to 0..1
Tool	Aggregation	1 to 0..*
Resource	Aggregation	1 to 0..*

Table 20 Description of the classes related to the *Workstation* class

3.3.2.4 TOOL

The Tool class is aimed at describing devices, normally held in hand, and used to carry out a particular operation. One or more tools may belong to a specific workstation.

Name	Type	Comment
Id	String UUID	The unique identifier of the workstation
Name	String	
Suitability	String	A human readable description of what the tool is used for

Table 21 Description of the attributes of the *Tool* class

Related Class	Relation Type	Multiplicity
Workstation	Aggregation (End)	0..* to 1
Model3D	Composition	1 to 0..1

Table 22 Description of the classes related to the *Tool* class

3.3.2.5 RESOURCE

The *Resource* class is used to describe equipment and specifically machines in terms of their status, geometry and location inside a workstation.

Name	Type	Comment
Id	String UUID	The unique identifier of the resource
Name	String	
Description	String	A human readable description of resource
isOnline	Boolean	
isMulfunction	Boolean	
isMobile	Boolean	This field is used to indicate if a resource can be moved. It can be used to detect if a resource can be reallocated to another workstation

Table 23 Description of the attributes of the *Resource* class

Related Class	Relation Type	Multiplicity
Workstation	Aggregation (End)	0..* to 1
Model3D	Composition	1 to 0..1
JobSchedule	Composition (End)	0..* to 0..*

Table 24 Description of the classes related to the *Resource* class

3.3.2.6 DEVICE

The *Device* class is used to represent the presence of devices in the shop-floor, such as AR glasses, exoskeletons, wearable devices, etc.

Name	Type	Comment
Id	String UUID	The unique identifier of the device
Name	String	
Description	String	A human readable description of device
Model	String	This field serves for identifying the type of device, e.g. if it's a Upper-Limb Exoskeleton, a MS Hololens, etc
DeviceDetail	BLOB	Details in binary format which can be unpack by the controller of the device itself

Table 25 Description of the attributes of the *Device* class

Related Class	Relation Type	Multiplicity
DeviceAssignement	Composition	1 to 0..*
Calibration	Composition	1 to 0..*
DeviceStatus	Composition	1 to 1
DevicetType	Aggregation	1 to 1
Workstation	Aggregation (End)	0..* to 1
InterventionPayload	Aggregation (End)	1 to 1

Table 26 Description of the classes related to the *Device* class

DeviceType is a simple class composed by an id and a string representing the type of devices available in the system.

DeviceStatus is an enumeration of the status of the device, actual values are [IDLE, BROKEN, OPERATIVE, USED]. More values have to be accounted for in the future.

3.3.2.7 DEVICEASSIGNMENT

The *DeviceAssignment* class monitors by whom, for how much time and where a device is used.

Name	Type	Comment
startTime	Timestamp	Time the worker started to use the device
endTime	Timestamp	Time the worker stopped using the device

Table 27 Description of the attributes of the *DeviceAssignment* class

Related Class	Relation Type	Multiplicity
Device	Composition	0..* to 1
Worker	Aggregation	1 to 1

Table 28 Description of the classes related to the *DeviceAssignment* class

3.3.2.8 CALIBRATION

The *Calibration* class stores the calibration information for a particular device and worker. Since the information required and the format are proprietary to each device calibration, the data is stored as a BLOB and left to be unpacked by the device itself.

3.3.2.9 PART

The *Part* class is used to represent either a product, an assembly or a part. The type is defined in the property *PartType*. Each part has general information, such as a unique identifier and a description . The product hierarchy is represented with the use of the Children reflexive association which represents and links all subassemblies and parts that are a level downwards in the hierarchy.

Name	Type	Comment
PartNo	String	
Description	String	A description of the part
PartType	String	

Table 29 Description of the attributes of the *Part* class

Related Class	Relation Type	Multiplicity
Measure	Aggregation	1 to n
Model3D	Aggregation	1 to 0..1
Task	Composition	1 to 0..*

Table 30 Description of the classes related to the *Part* class

3.3.2.10 MODEL3D

The *Model3D* class stores the necessary information to represent an object in a 3-dimensional space.

Name	Type	Comment
ModelURI	URI	The URI of the object's 3d model file
Rotation	Vector<3>	A vector indicating the rotation of the object in 3D space
Position	Vector<3>	A vector indicating the position of the object in 3D space

Table 31 Description of the attributes of the *Model3D* class

This class is referenced through composition by every class that refer to objects that can be inserted in a 3D space.

3.4 TASK MODEL

This section of the document describes the task model, which is the part of the data model dedicated to represent the production activities that are performed in the factory.

3.4.1 RATIONALE

The goal of the task model is to represent the production activities that are performed inside the factory. Two fundamental aspects have to be considered when modelling a job. Firstly the static representation of the job and secondly the job assignment, which worker, when and where.

For the static description of the work, the job is considered as the activity that is carried out in a single workstation (e.g. assembly carcass, rivet tail). This activity can be split into sub-activities which are called tasks (e.g. mount side panel, put single rivet). Further detailing of the task in operations is possible, but is out of the scope in the context of HUMAN.

The task model has to handle also the information on which worker is performing a certain job. This is the concept of job assignment. So, to copy the granularity of job and task and to monitor in-time execution of tasks, two different types of scheduling are introduced, the high level one, which schedules the job and one with finer granularity which schedules the tasks.

Another main request for the task model is to keep track of the progressing of the job. This is why HUMAN uses Petri nets to keep track of the execution of jobs. A Petri net, also known as place / transition net, is a mathematical modeling language for the description and definition of distributed systems. Each job is associated to a Petri net that defines the order in which tasks need to be executed. The current state of a job is determined by the distribution of tokens over the places of its Petri net. Transitions and places in a Petri net form a bi-partite graph, i.e. Places may only be connected to Transitions and vice versa.

By using the Petri net execution semantics, it is possible to track the progress of the instances and determine which tasks may be executed next. Petri nets are represented using 4 entities in the data model.

3.4.2 CLASS DESCRIPTION

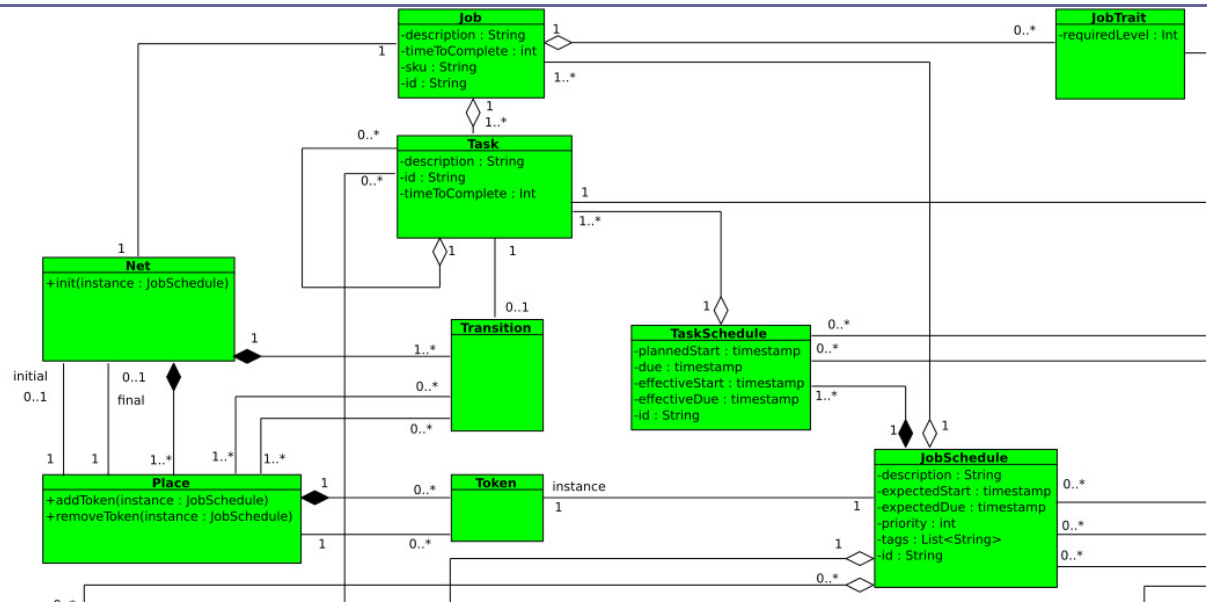

Figure 6 UML Class Diagram of the Task area of the HUMAN Shared Data Model

3.4.2.1 JOB

The *Job* class represents an activity that is carried out on a single workstation. During a shift, more jobs are performed.

Name	Type	Comment
Description	String	
timeToComplete	Int	Theoretical time, expressed in minutes, needed to complete the job in optimal conditions
Sku	String	Stock Keeping Unit of the manufactured product. Can be null.
Id	String	

Table 32 Description of the attributes of the *Job* class

Related Class	Relation Type	Multiplicity
Task	Aggregation	1 to 1..*
JobTrait	Aggregation	1 to 0..*
Net	Association	1 to 1
JobSchedule	Aggregation (End)	1..* to 1

Table 33 Description of the classes related to the *Job* class

3.4.2.2 TASK

The *Task* class represents sub-activities of a *Job*. One or more *Tasks* may be grouped into a *Job*. A *Task* can belong to more than a *Job*. To allow great flexibility in the granularity used to represent the *Task*, each Task has a list of sub-tasks, so it is possible to have a customizable level of granularity.

Name	Type	Comment
Description	String	
timeToComplete	Int	Theoretical time, expressed in minutes, needed to complete the task in optimal conditions
Id	String UUID	Unique identifier of the task

Table 34 Description of the attributes of the *Task* class

Related Class	Relation Type	Multiplicity
Part	Association	0..* to 1
Transition	Association	1 to 0..1
TaskPayload	Aggregation (End)	1 to 1
TaskSchedule	Aggregation (End)	1..* to 1

Table 35 Description of the classes related to the *Task* class

3.4.2.3 JOBTRAIT

The *JobTrait* class represents a trait that is required by a job in order to be performed. It is the concrete counterpart of the *WorkerTrait*. This duality allows the easy match between factory demands and worker abilities.

Name	Type	Comment
Level	Int	The requested skill level. The level is normalized between 0 and 100, where 0 is the lowest level (e.g. no requirements) and 100 is the max level (e.g. master level required)

Table 36 Description of the attributes of the *JobTrait* class

Related Class	Relation Type	Multiplicity
Trait	Subclass	
Job	Composition (End)	0..* to 1

Table 37 Description of the classes related to the *JobTrait* class

3.4.2.4 JOBSCHEDULE

The *JobSchedule* class models the assignment of a job to a worker and the planning of its execution. For this reason, it includes information about when the job has to performed, who has to perform it, where it has to done and some priority concepts.

Name	Type	Comment
Description	String	
expectedStart	Timestamp	Planned starting time of the job
expectedDue	Timestamp	Planned ending time of the job
Priority	Int	Priority for the job, 0 is the job with highest priority
Tags	List<String>	A list of string, called tags which are used to better define the job

Table 38 Description of the attributes of the *JobSchedule* class

Related Class	Relation Type	Multiplicity
Job	Aggregation	1 to 1..*
TaskSchedule	Composition	1 to 1..*
Worker	Aggregation (End)	0..* to 1
ReportedIssue	Aggregation (End)	0..* to 1
Error	Aggregation (End)	0..* to 1
Workstation	Composition	1 to 1
Resource	Composition	0..* to 0..*
Token	Association	1 to 1

Table 39 Description of the classes related to the *JobSchedule* class

3.4.2.5 TASKSCHEDULE

The *TaskSchedule* class models the scheduling of tasks. Since some jobs can span over a long period the *TaskSchedule* allows a finer granularity for the work planning when necessary.

Name	Type	Comment
plannedStart	Timestamp	Planned start date and time of the task
Due	Timestamp	Maximum ending time of the task
effectiveStart	Timestamp	Real start date and time of the task
effectiveEnd	Timestamp	Real end date and time of the task

Table 40 Description of the attributes of the *TaskSchedule* class

Related Class	Relation Type	Multiplicity
Task	Aggregation	1 to 1..*
Error	Composition	0..* to 1
JobSchedule	Aggregation (End)	1..* to 1
Worker	Association	0..* to 1 .. *

Table 41 Description of the classes related to the *TaskSchedule* class

3.4.2.6 PETRI NET

This part will describe the classes necessary to formalize a Petri net that is represented using 4 entities in the data model. Most of the entities are entirely described through the associations in the UML class diagram and do not require additional fields.

3.4.2.6.1 NET

A *Job* has an associated *Net* class, which models the ordering of tasks within a job. The Petri net defines the set of possible orderings using token-game semantics (Reisig, 2012).

3.4.2.6.2 TRANSITION

A *Transition* is associated to a *Task*. Executing a task executes the corresponding transition and removes tokens for the same job instance from the input places and produces tokens for the same job instance on the output places as per Petri net execution semantics.

However, some tasks do not have transitions. These (composite) tasks are used to group sub-tasks together.

3.4.2.6.3 PLACE

A *Place* represent (together with tokens) the state of the Petri net and, thus, the state of the job instances. Each place can hold 0 or more tokens that are produced and consumed by the connected input and output transitions.

3.4.2.6.4 TOKEN

*Token*s are distributed across places (multiset) and define the state space of the Petri net. The state space of the Petri net corresponds here to the set of possible alternative job executions. Tokens are produced and consumed by transitions. The execution of a transition produces tokens on its output places and consumes tokens from its input places. The distribution of tokens over the places defines the current state of a job instance (here *JobSchedule*).

3.5 EVENT MODEL

This section of the data model describes the modeling of the events inside the HUMAN system. The events stored inside this part of the data model are the low frequency data and commands which are used in the system. Examples of low frequency data are the request for intervention made by the Intervention Manager, the outputs of the Short Term Reasoning Engine and of the Long Term Reasoning Engine when a deviation is detected, updates regarding advancements in the task execution, start / end of a working shift. As a disambiguation note, the modeling of the event is not intended to replicate the Message Schema used to exchange data through the Middleware nor to store all the exchanged data on the middleware. It is important to note that the events that are saved are the low frequency, meaning that data coming from the sensors are not stored in this way.

3.5.1 RATIONALE

The choice of saving events inside the HUMAN system arises from the need of some services to perform analyses on the events that happened in the system. An example could be the SII which allows power users of the system to perform post-mortem analyses of what happened inside the system. Another example is the LTRE which decides the trigger of long-term interventions based on events that happened in the system.

The very nature of the events imposes to model a solution which is generic and scalable. This leads to design a structure composed by two parts, the first one is a header which contains some generic information (time, source, id) and a payload, which is dependent on the event itself. High-level analysis tool can then perform their job by looking at the header and the tools, which work with specific events, and by extracting the detailed information from the payload.

3.5.2 CLASS DESCRIPTION

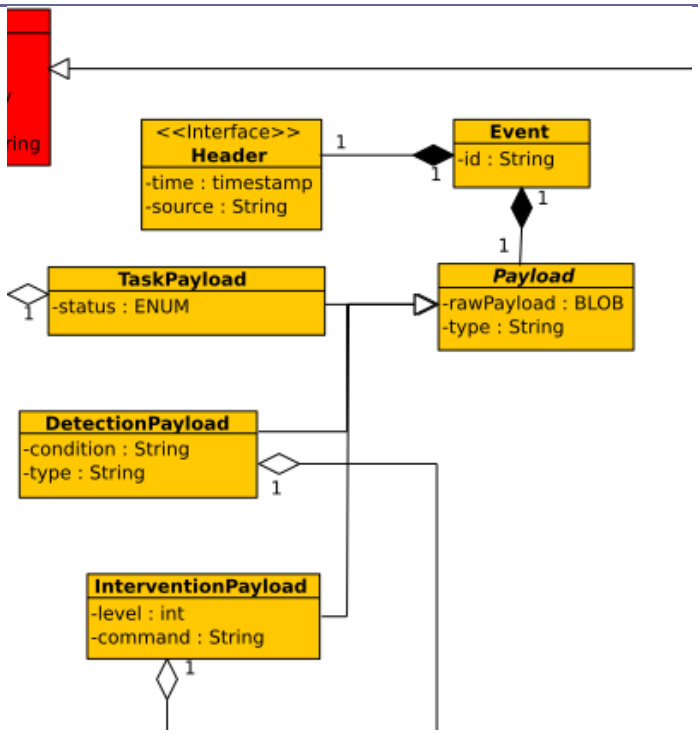

Figure 7 UML Class Diagram of the Events area of the HUMAN Shared Data Model

Hereafter a description of the classes with their main attributes is proposed while getters and setters are omitted from the description.

3.5.2.1 EVENT

The *Event* class is the main entry point in this part of the data model. It represents the low frequency events which happen inside the HUMAN system. It is designed following the principle of a header, which contains generic information and a payload which is the actually relevant information for the event.

Name	Type	Comment
Id	String UUID	The unique identifier of this event

Table 42 Description of the attributes of the *Event* class

Related Class	Relation Type	Multiplicity
Header	Composition	1 to 1
Payload	Composition	1 to 1

Table 43 Description of the classes related to the *Event* class

3.5.2.2 HEADER

The *Header* class is the holder of the generic information of the events, it can be seen as an envelope for the event itself.

Name	Type	Comment
Time	Timestamp	The date and time when the event occurred
Source	String	A string representing the source of the event (e.g.: who fired the event)

Table 44 Description of the attributes of the *Header* class

Related Class	Relation Type	Multiplicity
Event	Composition (End)	1 to 1

Table 45 Description of the classes related to the *Header* class

3.5.2.3 PAYLOAD

In computing and telecommunications, the payload is the part of transmitted data that is the actual intended message. The payload excludes any headers or metadata sent solely to facilitate payload delivery. The *Payload* class in the SDM represents the same concept as the one stated in the definition of payload. It is an abstract class which is then extended by the actual payloads.

Name	Type	Comment
rawPayload	BLOB	The serialized payload, this is provided to allow further processing on data that may not be clearly expressed in the data model.
Type	String	A unique string identifying the type of the payload. It is used to help the unmarshalling process of the *rawPayload*

Table 46 Description of the attributes of the *Payload* class

Related Class	Relation Type	Multiplicity
Event	Composition (End)	1 to 1
TaskPayload	Superclass	

DetectionPayload	Superclass	
InterventionPayload	Superclass	

Table 47 Description of the classes related to the *Payload* class

3.5.2.4 TASKPAYLOAD

The *TaskPayload* class refers to the payload of events which are fired inside the system when the user advances in the activity of performing a job.

Name	Type	Comment
Status	ENUM / String	A string representing the action that triggered this event. An implementation using a String will allow more flexibility, whereas an enumeration will limit the flexibility but will increase the usage of automated system to elaborate the information.

Table 48 Description of the attributes of the *TaskPayload* class

Related Class	Relation Type	Multiplicity
Payload	Subclass	
Task	Aggregation	1 to 1

Table 49 Description of the classes related to the *TaskPayload* class

3.5.2.5 DETECTIONPAYLOAD

The *DetectionPayload* class holds the information of a detection event A detection event is fired by the Short Term Reasoning Engine or the Long Term Reasoning Engine when a deviation is detected. For more information on the detection mechanism, please refer to D3.4.1 and D3.5.1.

Name	Type	Comment
Condition	String	A string, in human readable format, detailing the condition that caused the firing of the event
Type	String	The type of deviation detected

Table 50 Description of the attributes of the *DetectionPayload* class

Related Class	Relation Type	Multiplicity
Payload	Subclass	
Worker	Aggregation	1 to 1

Table 51 Description of the classes related to the *DetectionPayload* class

3.5.2.6 INTERVENTIONPAYLOAD

The *InterventionPayload* class stores the necessary information to trigger an intervention. This type of event is fired by the Intervention Manager when an intervention has to be executed.

Name	Type	Comment
Level	Int	The level at which the intervention should be triggered (e.g.: Exoskeleton level 1-3)
Command	String	The “command” sent to the device

Table 52 Description of the attributes of the *InterventionPayload* class

Related Class	Relation Type	Multiplicity
Payload	Subclass	
Device	Aggregation	1 to 1

Table 53 Description of the classes related to the *InterventionPayload* class

4. HUMAN Shared Data Model Services

In this section the presentation of the HUMAN Shared Data Model is enriched with a draft definition of the needed API to access the envisioned data structures. Actually the project implementation split this two development activities in two tasks: T3.2 aims at formalizing the Data Models while T3.3 focuses on the implementation of the Data Models, the definition and implementation of the APIs. In this section, however, some initial proposal, which will be extended in T3.3, is offered.

This section has been built in collaboration with the users of the SDM reasoning about the API their "clients" are supposed to use to retrieve and publish information into the HUMAN SDM. The APIs are grouped into worker-related, task-related and miscellaneous. This division mimics the separation of the data model.

4.1 WORKER

Return type	Name	Parameters	Comment
List<Worker>	getWorkers	Name: String Surname: String	Retrieves the list of all workers matching the name and surname. Both name and surname could be regex strings to empower the method. If both are null, then return all workers
List<AM>	getAnthropometricMeasures	WorkerId: UUID	Retrieves the list of anthropometric measures of the specified worker
Worker	getWorker	WorkerId: UUID	Retrieves the details about the specified worker
void	startShift	WorkerId: UUID JobId: UUID WorkstationId: UUID	Publishes the information in the system that *Worker WorkerId* started Job *JobId* on *Workstation WorkstationID* (i.e. *SessionStart*)
void	stopShift	WorkerId: UUID WorkstationId: UUID	Publishes the information in the system that Worker "WorkerId" ended his working shift (i.e. *SessionEnd*)

Void	updateShift	WorkerId: UUID newStatus: String	Updates the system with information about the worker (e.g. *Worker* is taking a break, user completed a job)

Table 54 Worker-related API

4.2 TASK

Return type	Name	Parameters	Comment
List<Job>	getJobs	void	Retrieves the list of jobs
List<Job>	getJobSchedule	Start: timestamp End: timestamp	Retrieves the list of job instance scheduled in a certain timeframe
List<Task>	getTaskSchedule	JobScheduleId: UUID	Retrieves the tasks scheduled for a specific job instance
List<Task>	getAllTaskScheduled	Start: timestamp End: timestamp	Retrieves all the tasks schedule in a certain timeframe
PetryNet	getNet	JobId: UUID	Retrieves the Petri net corresponding to a specific job.
Task	getTask	TaskId: UUID	Retrieves the information about a task
Job	getJob	JobId: UUID	Retrieves the information about a job
Task	getWorkingTask	WorkerID	Returns the task on which the specified worker is currently working
Task	advanceInJob	WorkerID: UUID JobId: UUID	Publishes to the system that the worker advanced a step in the job. It returns the new task to accomplish
Task	newJob	WorkerID JobId	Publishes to the system that the worker started a new job. Returns the first task of the new job

Table 55 Task-related API

4.3 EVENTS, INTERVENTIONS, FACTORY

Return type	Name	Parameters	Comment
DeviceCalibration	getCalibration	DeviceID: UUID WorkerID: UUID	Returns the settings / calibration for the specified device and specified worker
List<Event>	getEvents	Start: timestamp End: timestamp Source: String JobId: UUID	Retrieves the events for a certain timeframe. Matching a certain source criteria and job criteria
List<Error>	getErrors	Start: timestamp End: timestamp JobId: UUID	Retrieves the errors recorded for a certain timeframe
List<Issue>	getIssues	Start: timestamp End: timestamp JobId: UUID	Retrieves the reported issues recorded for a certain timeframe
List<Intervention>	getInterventionsDetails	Type: String	Retrieves information about the intervention prompts. For the LTI, which have to be assessed by the engineer, The prompts should include but not be limited to: timestamp of prompt, area affected, task affected, trigger reason, priority / importance For the STI it can include the cause, the affected worker and the effective command string for the device
Intervention	getInterventionDetails	InterventionID: UUID	Retrieves the intervention detail for the specified intervention

Table 56 Miscellaneous API

5. Conclusion

This document has presented a key cornerstone, the envisioning of a Shared Data Model, for the development of a worker-aware HUMAN system that is capable of monitoring the operators' condition in the shop floor, detect deviations from safe working patterns and correspondently trigger adequate interventions to restore well-being and safety.

In this context, the developed Shared Data Model promotes the information sharing by leveraging on a commonly agreed formalization of the data structures and of the semantic behind them. Four areas of interest have been identified in this work and described in detail, namely worker, factory, task and event.

Software developers of the services relying on the data stored according to the models have been involved to ensure relevance and completeness of the result for the purposes of the project. In the same way end-users have provided their contribution to align the developed models to the specific industrial cases that are covered in HUMAN. The diversity of such cases in scale and scope guarantees a wide applicability of the SDM in various context.

The next steps will include, besides a continuous update of the developed SDM as the HUMAN system progresses, a refining of the drafted API for the access to the models and the implementation of the envisioned data structures in Task 3.3. Environment Representation.

6. Reference list

Blázquez, M. (2014). *Skills-based Profiling and Matching in PES*. Publications Office of the European Union, Luxembourg.

Bettoni, A., Cinus, M., Sorlini, M., May, G., Taisch, M., & Pedrazzoli, P. (2014, September). Anthropocentric Workplaces of the Future Approached through a New Holistic Vision. In *IFIP International Conference on Advances in Production Management Systems* (pp. 398-405). Springer, Berlin, Heidelberg.

Borst, C. (2016). Shared mental models in human-machine systems. *IFAC-PapersOnLine, 49*(19), 195-200.

Ceclan, M. (2016). Developing Competence Based Qualification System in the Nuclear Energy Sector. *Atw. Internationale Zeitschrift fuer Kernenergie*, *61*(4), 229-237.

Choi, B. K., & Kim, B. H. (2000, May). A human-centered VMS architecture for next generation manufacturing. In *Proceedings of 2000 International CIRP Design Seminar* (pp. 169-174).

Fernández-Sanz, L., Gómez-Pérez, J., & Castillo-Martínez, A. (2017). e-Skills Match: A framework for mapping and integrating the main skills, knowledge and competence standards and models for ICT occupations. *Computer Standards & Interfaces*, *51*, 30-42.

International Organization for Standardization (2010). *Basic human body measurements for technological design -- Part 1: Body measurement definitions and landmarks*. ISO 7250-1:2010.

Peruzzini, M., & Pellicciari, M. (2017). A framework to design a human-centred adaptive manufacturing system for aging workers. *Advanced Engineering Informatics*, *33*, 330-349.

Reisig, W. (2012). *Petri nets: an introduction* (Vol. 4). Springer Science & Business Media.

Zhao, M., Javed, F., Jacob, F., & McNair, M. (2015, January). SKILL: A System for Skill Identification and Normalization. In *AAAI* (pp. 4012-4018).