INTERACT – Interactive Manual Assembly Operations for the Human-Centered Workplaces of the Future

Grant Agreement Number	:	**611007**
	:	**INTERACT**
Project Start Date	:	**1st October, 2013**
Consortium	:	**DAIMLER AG (DAIMLER)- Project Coordinator** **ELECTROLUX ITALIA S.P.A. (ELECTROLUX)** **INTRASOFT INTERNATIONAL SA (INTRASOFT)** **IMK AUTOMOTIVE GMBH (IMK)** **EMPHASIS TELEMATICS AE (EMPHASIS)** **HADATAP SP ZOO (HADATAP)** **UNIVERSITY OF PATRAS (LMS)** **UNIVERSITAET ULM (IMI)** **DEUTSCHES FORSCHUNGSZENTRUM FUER KUENSTLICHE INTELLIGENZ GMBH (DFKI)**

Title	:	**Deliverable 4.1.1 -** ***"Design of aggregated datastore for planned and sensor data"***
Reference	:	**D4.1.1 of Task 4.1, WP4**
Availability	:	**Public**
Date	:	**2014-09-30**
Author/s	:	**LMS, DFKI, DAIMLER, INTRASOFT, IMK**
Circulation	:	**Consortium**

Summary:

Deliverable D4.1.1 created in Task 4.1 – *"Design of aggregated datastore for planned and sensor data"*

This deliverable covers the design of the semantics based repository that will be used for storing and retrieving information regarding the execution of manual assembly operations. The model that will be used for storing human motions performed in workshops by real assembly workers is provided based on requirements coming from motion recognition and motion synthesis applications.

Contents

1. INTRODUCTION 2
2. APPLICATIONS OF THE MOTIONS RECOGNITION SEMANTIC REPOSITORY 3
2.1. Storage of shopfloor sessions 3
2.1.1. Simulation using data acquired on the shopfloor 3
2.1.2. Recognition of motions in the workshop environment 4
2.1.3. Motion identification for simulation generation 7
2.1.4. Constraints for future simulations 8
2.2. Morphable Graph motion synthesis and constraints 9
2.2.1. Morphable graph construction overview and storage 9
2.2.2. High Level description of the controlled motion synthesis algorithm 10
2.2.3. Morphable Graph Constraints 10
3. COMMON GLOSSARY OF MOTIONS 12
3.1. Two level description of motions 12
3.1.1. Elementary Actions from Controlled Natural Language for motion synthesis 12
3.1.2. Motion constraints reuse in future scenarios 13
3.2. Elementary Actions and Motion Elements 13
3.2.1. Elementary Actions 13
3.2.2. Motion Elements 15
4. CONCEPTS OF MOTIONS RECOGNITION SEMANTIC REPOSITORY 17
4.1. Shopfloor sessions 17
4.2. Elementary Actions and Motion Elements concepts 18
5. INFRASTRUCTURE 20
5.1. Kinematic data post-processing 20
5.2. Sensor and workshop information processing 20
5.3. Motion identification and reuse 20
6. SUMMARY AND CONCLUSIONS 23
ABBREVIATIONS 24
7. APPENDIX 25

1. INTRODUCTION

The objective of this document is the description of the rationale behind the design of the Motions Recognition Semantic Repository that will be used for storing semantic descriptions of motions captured in workshops, as well as the main outcome of the design process. In order to specify the features and content of the repository, the applications that will be using it are described and each one’s requirements are defined in different sections. Special attention is given to the motion recognition application which was developed as a first prototype in MATLAB®.

The common glossary of motions is then presented as a common way to describe the motions for different applications. The motions are divided into different description “layers” in order to satisfy the requirements of different applications.

Finally, the proposed infrastructure that will be implemented, in order to realise the semantic repository, is described in the final sections and includes the relevant middleware that will enable the communication with other INTERACT components.

2. APPLICATIONS OF THE MOTIONS RECOGNITION SEMANTIC REPOSITORY

In this section, the various applications related to the use of the Motions Recognition Semantic Repository (MRSR) are described. The description of the applications will be followed by the identification of the various requirements generated by each one of them.

2.1. Storage of shopfloor sessions

The storage of shopfloor sessions into the MRSR has as purpose the storage of semantic descriptions of the actions that correspond not only to the workers' that performed them but are based on the constraints required to reproduce them using morphable graphs (WP2). Furthermore, the semantic description should allow the identification of similar motions to motions that are planned, i.e. for simulation.

2.1.1. Simulation using data acquired on the shopfloor

Regarding previous efforts, a number of possible issues, concerning the authoring and control of various Elementary Actions, have arisen. For instance, each action has a time constraint that should be reflected in a simulation. An action should not be too fast or too slow, in order to appear natural. Moreover, in a shopfloor environment, various operations include both a subject and one or more objects. In order to identify such an action, process it and reuse the constraints associated to it, object parameters, like dimensions, weight, location and picking and grasping points may be required.

Simulation constraining should respect constraints/ parameters of the related Elementary Action. For a *Pick* action, e.g. "pick a screw driver", the human model should get close to the screw driver, turn towards the object, estimate the next movement regarding the object's, relevant to the subject, position, reach to it, touch it and pick it from the correct grasping point. Picking an object can be performed with one hand or two hands. The object can be picked from any side. The object should not be picked without reaching it, released before the release point or even carried without touching it. From a recorded walking motion, the walking path may be derived, starting from a specific point towards another point, through a collision free path. The manikin should not walk a shorter distance or at a different speed, from the similar ones measured in the shopfloor. For evaluation purposes, possible parameters that may affect the simulation result, for four Elementary Actions, are presented in the following table (Table 1).

Table 1: Elementary Actions' parameters relevant to constraints

Elementary Action	Example parameters that have to be respected
Walk	Walking speed Start and finish point Walking distance
Pick	Number of hands grasping Reach distance Standing position Grasp point(s)
Place	Place position Place surface Reach distance Grasp point(s)
Carry	Carry speed Start position Carry stance Carry surface Grasp point(s)

It is evident that the parameters necessary for constraining an action belong to two categories:

- Scene relevant parameters
- Action relevant parameters

The first category has to do with the objects and environment where the action is performed at and the second has to do with the way that the action was carried out. The first category parameters can be used for identifying similar conditions that the action was performed in, while the second can provide useful information for the configuration/ constraining of similar planned motions in future simulations. In order to obtain these parameters, an identification mechanism should be employed, which, beyond identifying the motion, will also provide the kinematic parameters that it was performed with.

2.1.2. Recognition of motions in the workshop environment

Recognizing motions from motion capture files generated in workshop environments, is important for further processing and validation, before storing them as semantic individuals for future use. In the Motions Recognition Semantic Repository, different motions should be recognized and stored. In order to do this, the data from the shop floor, containing joint displacement values and rotation angles along with resolution and joint information will be analyzed. Processing the information extracted from MoCap files is crucial from an aspect of time as well as for efficiency. The steps of the current version of the Motion recognition algorithm are (Figure 1):

- After the MoCap files have been generated, they are further broken down into relevant labels (values) regarding the motions. The new processed MoCap labels include: Joint name, Frame number and Offset value based on the structure of the MoCap file format used. Separating each of the three labels provides all the related information. Purpose of the different labels is to provide flexibility during processing.

- Afterwards, normalization of the data is performed in order for them to have a common reference point. This enables the statistical analysis of the data for identifying possible similarities or variations.

- Rotation values, stored under the Frames label, provide, after processing, the rotational speed of each joint. The calculation of the rotational speed enables motion recognition, by means of rule based identification. The same can be done for different parameters creating new labels, such as the distance between two joints for all frames.

- Further process actions provide the parameterized motions description as outcome.

For the development of the relevant software, MATLAB® was selected, since it provides a great number of capabilities for data processing.

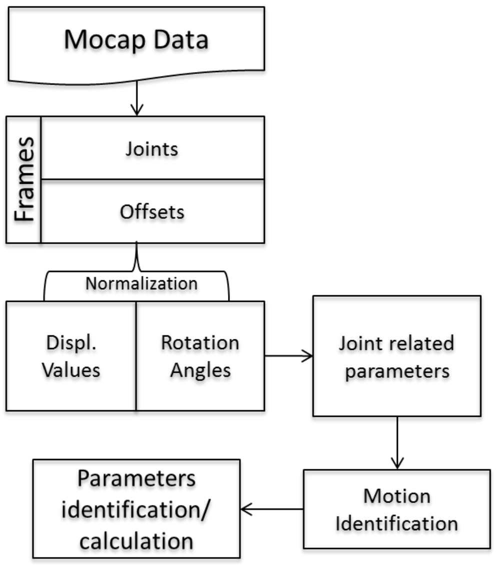


Figure 1: Parameterised Motion Element recognition

For a specific motion, common patterns exist among the various captures. The motion style can vary depending on secondary features such as age, gender, size, etc. This fact, in addition with the dynamic nature of the motions, motivated the use of rules to recognize different motions in accordance with space and time constraints. **The use of rules can overcome small variations in motion styles** since they focus only on key features related to joint relations. Moreover, rules are easily processed, i.e. they require less computational capacity. On the other hand, they require careful adaptation to each motion. As a result, and concerning the first sessions, motions are recognized with a set of rules in combination with statistical validation. The idea behind exploiting rules was that each movement shall activate at least one different joint not used by another motion that could be performed in parallel. For each motion, key joints, with the related constraints, are examined in detail. For example, walking is similar to running but with different velocity, or bend is similar to jump, but hips joint displacement values should have opposite directions of movement on the vertical axis.

At first, the motion for *Walk* with its related features, e.g. steps, distance, velocity, along with displacement and rotation parameters from a motion capture file are examined. Regarding the *Walk* motion, the key joints considered are; left foot joint, right foot joint and hips joint. The rules set for recognizing a step are condition - action rules of the form of "IF… THEN…", e.g. "IF (foot accelerates) and (hips move in the same direction) THEN store one step". Concerning the first evaluation method for recognizing a motion, peak values of rotational speed are processed. Through the rotation angles of the two feet joint, in accordance with a displacement values from the hips joint, one or more steps are recognized. The exact moment, before the skeletal model makes a step, the foot which is about to make the step, has zero rotational speed. During the step motion, speed curve has a peak, which is detected for step recognition. Only statistically significant peak values are considered for further examination, since due to noise, smaller peaks do not correspond to any actual step movement. From a physical point of view, in order to make a step, an amount of force is required to move the mass of the foot. The result of this force is an acquired acceleration, which is depicted as rise and fall of rotation acceleration curves. After the related speed falls to zero and to negative values during the other foot's step. Negative values, are caused due to the reference joint that moves forward in comparison to the examined foot joint. As a result, one step represented by a curve rising from zero to a peak value and then falling to zero again, is detected from the peak values between the two zero points (Figure 2).

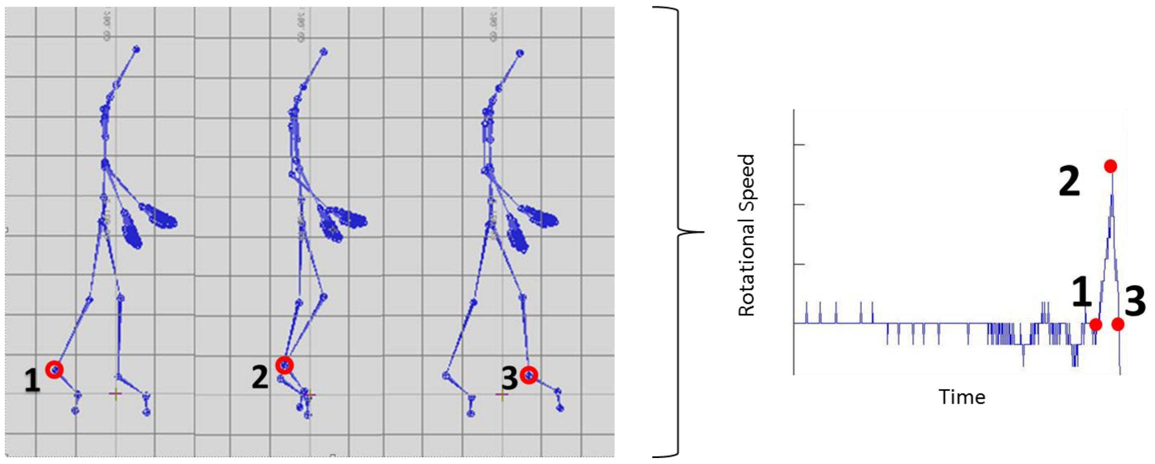
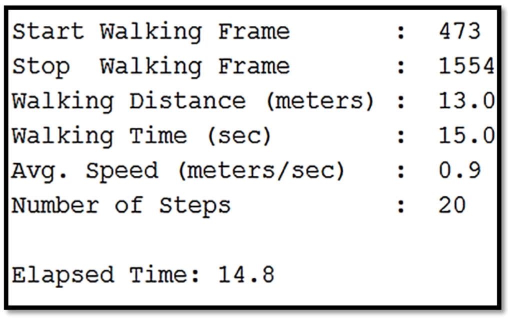


Figure 2: Rule based motion recognition for walking

The first experiments were conducted using different .bvh files (MoCap format) concerning the Elementary Action *Walk*. The recognition process was accomplished, by following the rule-based identification by the first prototype of the software developed in MATLAB®. Different parameters of *Walk* were identified, specifically, the start and stop points were tracked. The results also included walking distance and velocity as well as the related steps. The final outputs from the current script, concerning one motion capture are presented in the following figure (Figure 3).



```
Start Walking Frame        :  473
Stop  Walking Frame        :  1554
Walking Distance (meters)  :  13.0
Walking Time (sec)         :  15.0
Avg. Speed (meters/sec)    :  0.9
Number of Steps            :  20

Elapsed Time: 14.8
```

Figure 3: Experiments' results printed through the MATLAB® console

The method followed for recognizing *Walk* can be used for other motions with different sets of rules. However, most Elementary Actions can't be identified as they are and they need to be broken down into smaller, identifiable motions. These motions will be referred to within WP4 as Motion Elements (MEs). A set of rules can then be translated into an Elementary Action. The Motion Elements (MEs) are very similar to the Motion Primitives used in WP2 for motion synthesis but are not the same.

One or more joints constitute the key joints that can be used for identifying a specific Motion Element. The key joints should be identified before continuing to motion recognition. By adjusting the rules in the existing script each and every Motion Element should be able to be recognized in a similar way.

Additional experiments were conducted using motions recorded through the Kinect for Windows v2 sensor, where specific Motion Elements were recognized using the same algorithm. The Elementary Actions performed were *Walk*, *Pick*, *Carry*, *Place* and *Walk* backwards in that sequence.

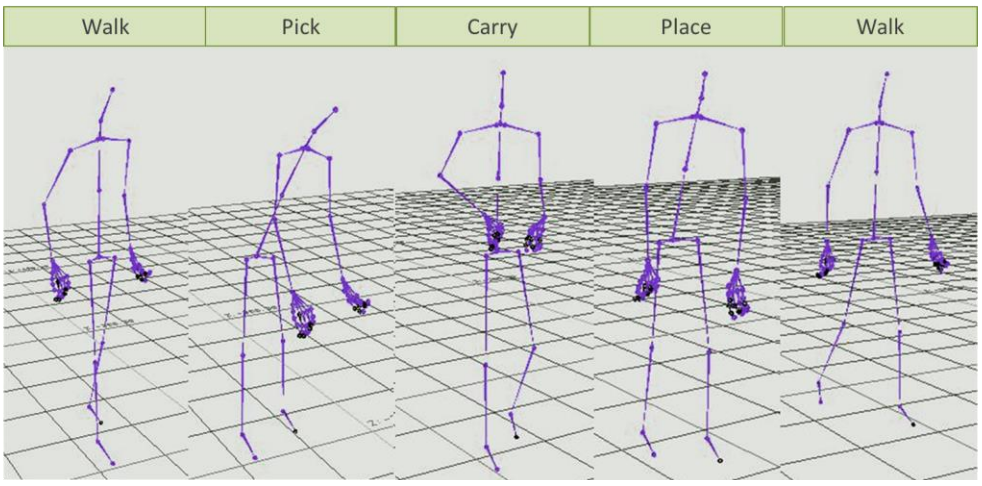


Figure 4: Motions performed in additional experiments

The algorithm was configured to detect an additional Motion Element that is used to construct the Elementary Actions performed, namely *Reach*. All motions can be detected as well as the inverse motions e.g. *Walk* backwards. The configuration of the algorithm utilized the derivative of the distance of the wrist joints from the hip joint. The output of the motion recognition can be seen in Table 2. The Motion Elements recognized can be coupled with additional sensor data showing that the user has grasped an object (*Grasp* Motion Element) in order to derive the relevant Elementary Actions. For example, a *Walk* ME recognized at the same time as a *Grasp* ME will be translated into a *Carry* motion. A *Reach* ME followed by a *Grasp* ME will be translated into a *Pick* motion and so on.

Table 2: Motion Elements recognised in additional experiments

	'Start Frame'	**'Stop Frame'**	**'Start Frame'**	**'Stop Frame'**
Walk	77	141	199	266
Walk Inv.	327	408	-	-
Reach	144	171	270	295
Reach Inv.	172	187	296	323

The output file from the Kinect version 2 sensor was in XEF format which was translated into a TXT format to be further processed by MATLAB®. After the recognition of the motion, relevant information regarding the motion synthesis of the motion can be stored which comprise of joint space and time constraints.

2.1.3. Motion identification for simulation generation

The motions recognized through the methodology presented in the previous sections will be stored in the Motions Recognition Semantic Repository initially as Motion Elements (ME) including the elements translated from additional sensor data. This data fusion process results in specific actions with specific parameters and constraints. The MEs created will be linked to relevant constraints and parameters based on the way a motion was performed and will be already linked through semantic relations to the products and processes they refer to.

In the semantic repository, specific rules will be applied which will be performed in the following order:

1. Motion Elements processing
2. Motion Elements clustering → Elementary Actions definition

The first step includes the actual recognition of the MEs which comprise the Elementary Actions. The methodology that will be used (section 2.1.2) combined with additional sensor data, will provide the necessary parameters in order to constraint new similar motions through simulations (Motion Synthesis – WP2) as correspondingly as possible. Of course, in order to do that the MEs should be clustered which means that beyond the identification of elements that create an Elementary Action (e.g. *Reach-Grasp*) the parameters of the Elementary Actions should be extracted from the Motion Elements (Figure 5). Both MEs and the derived Elementary Actions (EAs) will be stored into the Motions Recognition Semantic Repository in order to make their constraints and parameters reusable in future scenarios where a new simulation will be generated.

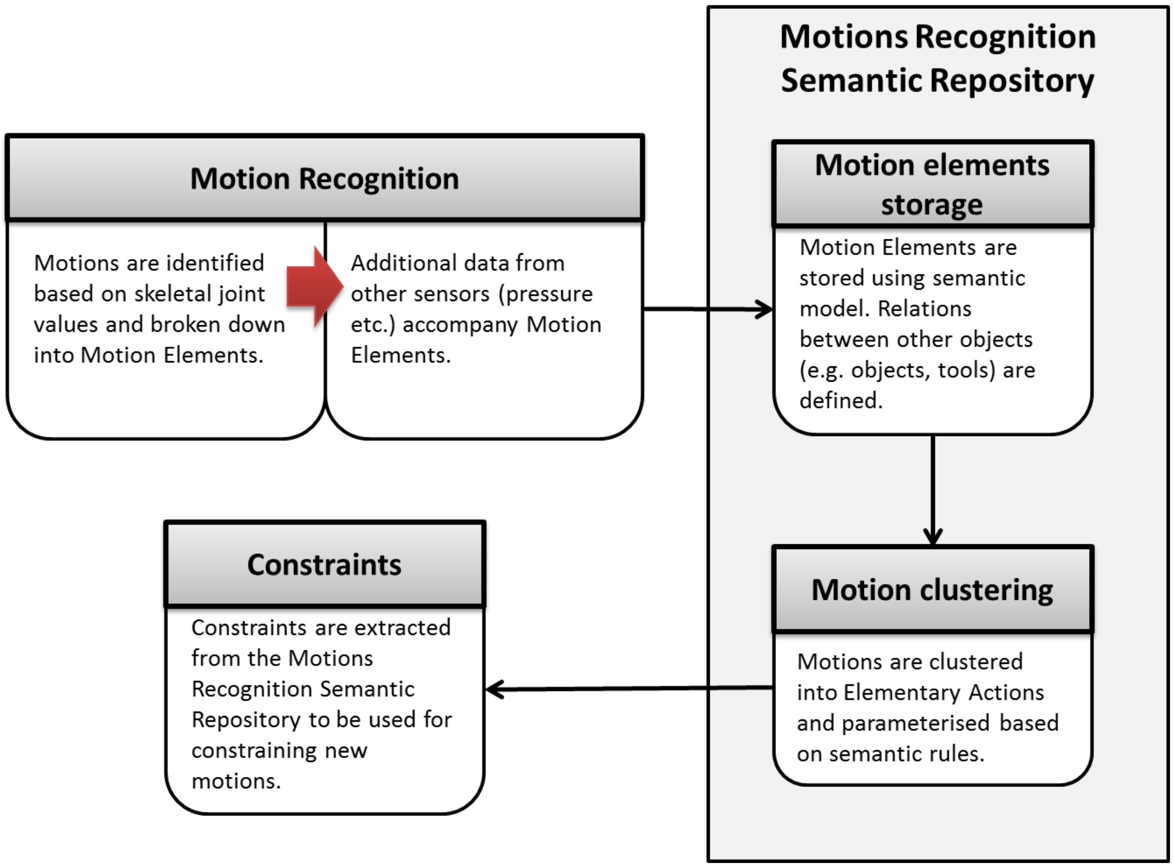

Motions Recognition Semantic Repository
Motion Recognition
Motions are identified based on skeletal joint values and broken down into Motion Elements.
Additional data from other sensors (pressure etc.) accompany Motion Elements.
Motion elements storage
Motion Elements are stored using semantic model. Relations between other objects (e.g. objects, tools) are defined.
Motion clustering
Motions are clustered into Elementary Actions and parameterised based on semantic rules.
Constraints
Constraints are extracted from the Motions Recognition Semantic Repository to be used for constraining new motions.

Figure 5: Motions recognition and storage

2.1.4. Constraints for future simulations

A very important aspect of each project is robustness and flexibility. The constraints of the motions recorded in one workshop should be reused in similar motions simulated in future scenarios. As already stated, each generic Elementary Action should be recognized appropriately, accompanied by the necessary constraints in order to generate a more complex activity. With the INTERACT approach, simulation modeling time and effort should be minimized and at the same time the realism of motions should be enhanced through knowledge acquired from the shop floor. Moreover, the developed models should provide use cases for a number of different scenarios and changes, e.g. different tools and assemblies.

Since Elementary Actions can be performed differently by different people, this should not constraint their identification. This means that although *Walk* can be performed in a different way by two individuals it should be described in the same way using the same parameters and constraints to describe it. In order for the information and knowledge extracted from the shop floor to be reusable, proper and detailed parameterization should be done for each and every Elementary Action as described in the previous sections. Such information should be available and easily processed when required. The related data will be stored into a repository from where the required information including the constraints could be extracted for similar cases. Simulation reproduction results should be included with the related information and parameters. The creation of such a repository will enable the reuse of motion constraints regardless of the scenario.

2.2. Morphable Graph motion synthesis and constraints

This section provides a high level description of the morphable model graph motion synthesis algorithm that is developed in the context of WP2 in order to provide an overview of the input required for constraining different Elementary Actions.

2.2.1. Morphable graph construction overview and storage

Morphable graphs use a directed graph of parametric models each representing a short motion primitive, e.g. left step, to enable parametric motion synthesis.

Kinematic and time variation, which together form the “motion space” of a motion primitive, are parameterized separately by registering a set of example motion clips of a motion primitive into a reference timeline using Dynamic Time Warping. After this registration a motion clip is represented by a single high dimensional vector of a fixed length and a time function, a vector that relates frame indices from the reference timeline to frame indices of the example motion clip.

Principal Component Analysis is then applied on the motion and time vectors to perform a dimension reduction of the motion representation, by identifying the mean and the Eigen vectors. A new motion in the motion space of a morphable model graph can then be synthesized using the mean motion and a linear combination of the Eigen vectors, which forms a morphable function that takes the weights of the Eigen vectors as input.

For each morphable primitive, a statistical model of the parameters in the form of Gaussian Mixture Model (GMM), is trained using the projection of the original motion examples in the dimensionally reduced representation. The GMM can be used to generate natural-looking motions by sampling and evaluating the naturalness of motions. Additionally, Gaussian Process (GP) regression is used to find optimal parameters at transitions between models. Transition models are created by finding correspondences between start and end frames of example motions and using the dimensionally reduced representation of the parameters as training samples for the GP. The storage of a morphable model graph requires the storage of a skeleton hierarchy, as used for the storage of a normal motion file, e.g. in the BVH format. Additionally the parameters of the motion models representing the nodes of the graph and the parameters of the transition models representing the arcs need to be stored:

A morphable model contains as described above a morphable function and a statistical model. The morphable function is defined by the Eigenvectors and the mean of the kinematic vectors and time vectors respectively. In the current implementation two separate statistical models are used for the kinematic and time parameters of a morphable model, each represented by a list of covariance matrices, mean vectors and a list of weights. With the currently used GP-library, transition models between morphable models are stored in the form of training data for the GP, i.e. parameter correspondences. However, this is supposed to be replaced by the parameters of the trained GP.

2.2.2. High Level description of the controlled motion synthesis algorithm

Controlled motion synthesis is formulated by the Morphable Graph algorithm (WP2) as an optimization problem, in which user and environmental constraints together form an error function, which needs to be minimized, while maximizing the likelihood of the motion parameters in the statistical model. Based on a task description in CNL as input this optimization problem is supposed to be solved in three steps:

In the first step the breakdown from the CNL-task description to a description in a Elementary Action vocabulary that can be interpreted by the Morphable Graph algorithm is performed as described in section 2.2. This vocabulary is being developed by DFKI in the context of WP2. Additionally, constraints are derived at this step from the complete task description, the environment and the parameters of motions already stored in the semantic repository.

In the second step, a graph algorithm uses the motion description and the derived constraints to select an optimal sequence of motion primitives using random motion parameter samples from the statistical models. In the last step the optimal parameters for this sequence of motion primitives are found by a gradient based optimization algorithm using the best of the sampled parameters from the previous step as a first guess.

At this step, additional constraints can be retrieved from the semantic repository, in order to configure the new, planned motions based on historical data. This is the most important requirement for the information that should be stored in the Motions Recognition Semantic Repository.

2.2.3. Morphable Graph Constraints

There are three kinds of constraints that are used by the Morphable Graph algorithm developed in WP2: Kinematic constraints, time constraints and semantic annotations.

Kinematic constraints restrict the position or orientation of a joint or end effector of the skeleton and are defined in the global coordinate system of the scene at a point or interval in time. The association of a kinematic constraint with time can be done by applying the constraint on a specific frame of the reference time line of a motion primitive, e.g. for grabbing, or in the form of a spline that defines the change for the whole duration of a motion. An example of joint constraints that are used for synthesizing a sequence of Elementary Actions (*Walk* → *Pick* → *Place*) can be seen below:

```
walk:
   (Hips, targetPosition)
two hands pick:
   (RightHand, RightHandGripPoint, <contact:true>),
   (LeftHand, LeftHandGripPoint, <contact:true>) two hands carry:
   (Hips, targetPosition),
   (RightHand, <contact:true>),
   (LeftHand, <contact:true>)
two hands place:
   (RightHand, RightHandTargetPosition, <contact:false>),
   (LeftHand, LeftHandTargetPosition, <contact:false>)
```

Constraints are provided both as a space and time constraints (referring to one or more joints) and can be accompanied by Semantic Annotations (e.g. for *Pick* → Contact). The Motion Recognition's goal is to detect how the movement of the worker corresponds to the constraints that are used to generate similar Elementary Actions, in order to be more realistic. At the same time, the semantically stored motions should be identifiable in terms of similarity to planned motions.

3. Common glossary of motions

In the following sections, the reasoning behind the description and storage of motions in the Motions Recognition Semantics Repository is provided. Based on the requirements of the applications that are involved in the creation and usage of the motions, the modeling of the motions is presented which will be used in order to create the motion's repository.

3.1. Two level description of motions

Incorporating the dynamic features of a 3D model is difficult, because of the amount of information to be stored and processed. Basic analysis is required, since INTERACT is not only focusing on visualization, but also performance, time and space optimization. The basic motions represent simple tasks, e.g. *Pick*, *Place*, *Carry* or *Walk*.

After selecting the tracking targets and methods, as previously discussed in D1.2.1, recognising and extracting parameters and constraints of human motions should be neither time nor storage consuming. Only the vital information should be kept – which includes information related not only to the motion itself but to everything that can be useful for its identification. To provide sufficient resolution, predefined values of the movement are filtered and used, as to identify the related motion's constraints.

Thus, for a two level model the first level will be used for constraining simulated motions while the second level for the recognition of actual motions.

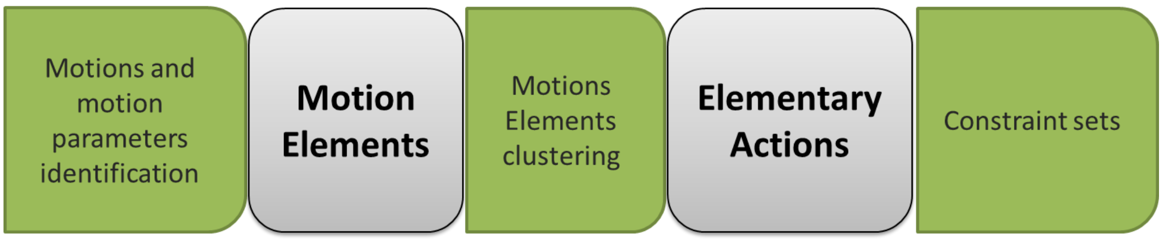

Motions and motion parameters identification
Motion Elements
Motions Elements clustering
Elementary Actions
Constraint sets

Figure 6: Two level motion description

The reasons for adding a separate description of motions (Motion Elements) instead of using Motion Primitives has to do with the requirements of the Motion Recognition algorithm and the classification of Motion Elements based on the sensor data they derive from. For example, a walking motion does not have to be broken down into left and right stance motions in order to be translated into an Elementary Action. Furthermore, a number of Motion Elements will not be recognised by optical data. For example the *Grasp* motion will be recognised through sensor data coming from pressure sensors.

3.1.1. Elementary Actions from Controlled Natural Language for motion synthesis

The breakdown of CNL descriptions (cf. D1.1.1) leads to EA descriptions with associated parameters and constraints. The same kind of vocabulary will be used and the usage of condition - action rules has been proposed for computing the breakdown (cf. D2.1.1). However there are different viewpoints. The CNL breakdown aims at specifying a sequence of motions corresponding to the level described in D.1.1.1, i.e. *Walk*, *Pick*, *Place* and *Carry* at first. Since CNL is usually not explicit about any of these motions, they must be derived from the task description (e.g. “tighten middle console” may involve all four of these motions in some particular order, some of them multiple times). The parameters used are static (belonging to the objects in question) or dynamic (depending on locations). They are specified to the extent needed to distinguish between the appropriate elements of the motion graph.

This means that collision avoidance during *Walk* is not dealt with since Morphable Graph will be able to handle this.

The simulation approach described in section 2.1 aims at more detailed motions. The CNL perspective requires less fine-grained resources, but they are of the same general types (condition-action rules, objects, motions, and parameters).

3.1.2. Motion constraints reuse in future scenarios

The most important requirement for the motion description, as defined earlier, is formed by the reusability of the motion constraints and parameters (annotations) which includes the identification of relevant parameters related to the conditions under which the motions will be performed. Therefore, beyond the storage of motions, the relevant information has to be available in order to identify the similarities of the abovementioned conditions:

- Product: the product and more specifically the components that were processed/ assembled per task.
- Resource: the resources used for the processes requiring tools.
- Human: the characteristics of the human that performed the tasks (e.g. height).

Beyond the actual information that has to be stored for each of the above elements, the relationships between them will also have to be available in order to enable inference mechanisms to correlate stored motions to new conditions.

3.2. Elementary Actions and Motion Elements

The following sections provide an overview of the conceptual description of the motions that will be stored in the semantic repository. Based on the discussion over a common glossary, the two level descriptions are provided in separate sections.

3.2.1. Elementary Actions

The Elementary Actions as described in D1.1.1 (as Elementary Motions), will be the first level description of motions, satisfying the requirements set by motion synthesis, i.e. space / time constraints. This means that each motion that will be stored in that way will be linked to the necessary control information for the Morphable Graphs algorithm to synthesise the relevant simulation. Further parameters of the motions have to be recognised and stored in order to make them identifiable when similar motions are planned. These parameters will not be used for motion synthesis but only to identify the stored motions. The current list of motions with a first list of information that has to be available can be seen below:

Table 3: Elementary Actions with exemplary motion information to be stored

Elementary Action	**Exemplary motion information to be stored**	
Walk	Time constraints Space constraints	
Carry	Time constraints Space constraints Hands used	Carried object characteristics

Pick	Time constraints Space constraints Hands used	Picked object characteristics
Place	Time constraints Space constraints Hands used	Placed object characteristics
Side Step	Time constraints Space constraints	
Press with Finger	Time constraints Space constraints	
Bolt Connect/Disconnect	Time constraints Space constraints Hands used	Connection event
Turn head to look around	Time constraints Space constraints	
Hand screwing	Time constraints Space constraints Rotation side	Fastener characteristics
Push/pull object (extend / retract arms)	Time constraints Space constraints Hands used	Object characteristics Push or Pull
Press/Lift (move object down/up)	Time constraints Space constraints Hands used	Object characteristics Press or Lift
Regrasp (Change grasp without relinquishing control)	Time constraints Space constraints Start orientation	End orientation
Transfer (control transfer grasp from one hand to other)	Time constraints Space constraints Start orientation	End orientation Initial hand
Clipsing	Time constraints Space constraints	

The list of information will be increased or reduced or modified based on the implementation of the technologies in workshops. Additional information about the objects used during the actions that the motions were performed in, will be described through relations to instances of the actual objects.

Regarding the Time and Space constraints, they will be used to reproduce the motions using motion synthesis. Other parameters will be mainly used in order identify similarities between recorded and planned motions.

3.2.2. Motion Elements

The need for using Motion Elements derives from the sub-motions that exist in one or more of the motions and can be identified through motion recognition. As stated earlier, the need for using Motion Elements instead of Motion Primitives (used in motion synthesis – WP2) is the requirements set by the Motion Recognition algorithm. For example the walking motion doesn't have to be broken down into left or right stances in order to be identified as an Elementary Action. However motions such as *Pick* or *Bolt Connect* comprise of motions such as *Reach* and *Grasp* as seen below, which can be similar to the motion primitives that construct these motions:

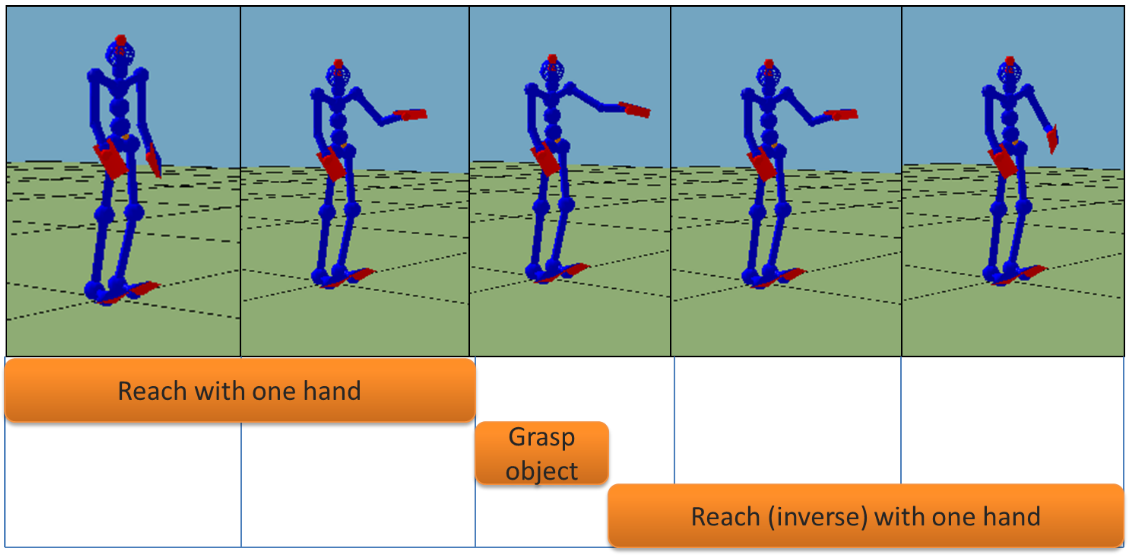


Figure 7: Pick object from shelf close to the shoulder height

The MEs are recognised by the Motion Recognition algorithm in order to be translated into Elementary Actions. MEs carry information that can be transferred to the EAs. The linking of MEs to EAs is done after the MEs have been stored semantically through clustering. This is previously referred to as Motion Elements clustering. An example can be seen in the figure below:

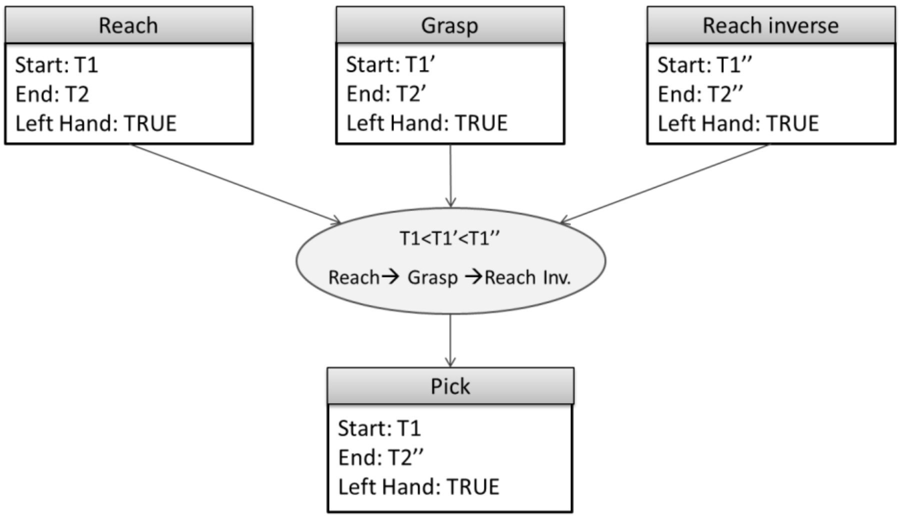


Figure 8: Motion Elements clustering to form a Pick Elementary Action

In order to enable the clustering of Motion Elements into EAs, the end and beginning frame of each Element should be stored. This will allow the effective sequencing of motions. Furthermore, each additional parameter / attribute of a Motion Element should enable its identification for future use.

4. Concepts of Motions Recognition Semantic Repository

This section is dedicated to the description of the first design of the semantic concepts (classes) of the repository as well as their relations. The two main context spaces identified are the information relevant to the sessions performed on workshops and the information only relevant to the motions.

4.1. Shopfloor sessions

The relevant parameters that will be stored for each session will be an ID as well as the date that the sessions took place. All the other parameters will be stored separately in order to allow their connection to different sessions. One of the most important elements will be the product components that certain tasks where performed on. The relevant information about these components will be their dimensions and weight, since these parameters will be used to identify the behaviour (in terms of motions) of a user regarding different types of parts or subassemblies. The availability of this information will enable the case based reasoning on the conditions that motions were performed under. The characteristics of the human that performed the actions will also be stored for the same reason.

In addition, the engineering relevant information regarding the resources used will be stored. Both resources and components will be linked to a specific session and will be also linked to the relevant Elementary Actions that they relate to. The product components can be the various sub-assemblies such as the tail light or the middle console of a car (main product) or single parts. Further to that, the subcomponents of an assembly will also be described through the same class. These parameters are applicable both to parts to assemble and resources such as screwdrivers. The resources will be described by a main resource concept and will be categorised through a type attribute to the main tool and auxiliary equipment types such as screwdrivers, wrenches and pulleys.

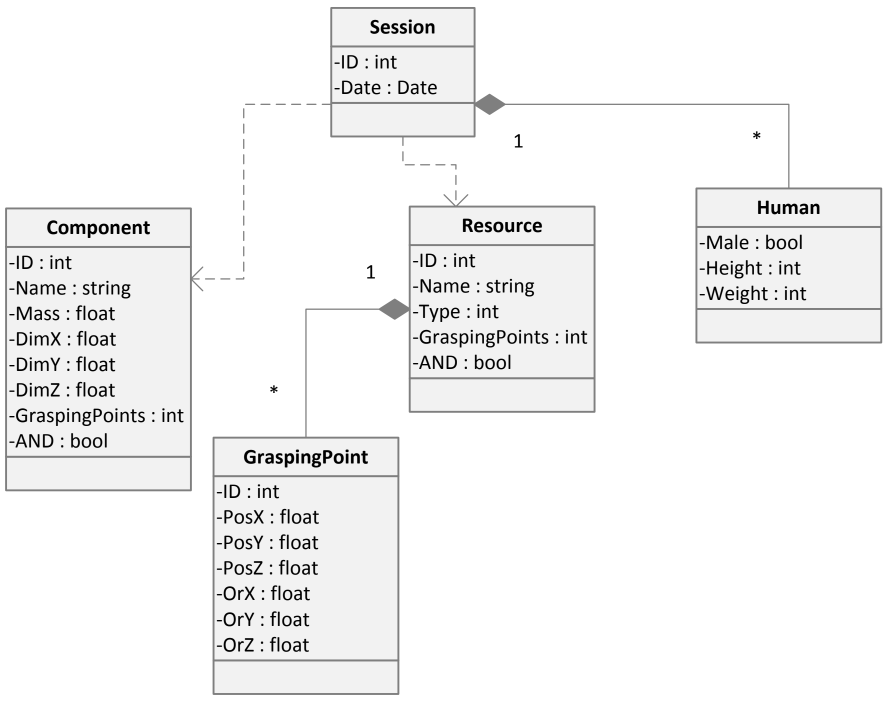


Figure 9: Relevant information regarding the Shopfloor sessions

4.2. Elementary Actions and Motion Elements concepts

As described previously, the motions will be divided into Elementary Actions and Motion Elements. During motion recognition, Motion Elements will be recognised and stored in the semantic repository. From there, the Motion Elements will be clustered based on their sequence and type. The clustering of motions will be performed through semantic rules. The relevant concepts of the motions are sub-categorised into concepts containing the specific parameters of the motions. Additionally, starting, ending and intermediate points (position and orientation) of the relevant joints of the motions will also be stored in order to be later used as constraints in motion synthesis (WP2). The joints will be identified by a single number which will correspond to a single joint of the RocketBox skeleton.

The frame of a kinematic constraint will be stored, in order to have time constrains as well. As an example, *Walk* will have a starting point, intermediate points and an end point for the hips (joint number 1) which will also be used for later defining the trajectory of the motion (through the orientation). The following figure shows the main classes of that will be used to create the concepts of the motions. The classes for four example actions are provided, namely *Walk*, *Pick*, *Carry* and *Place*.

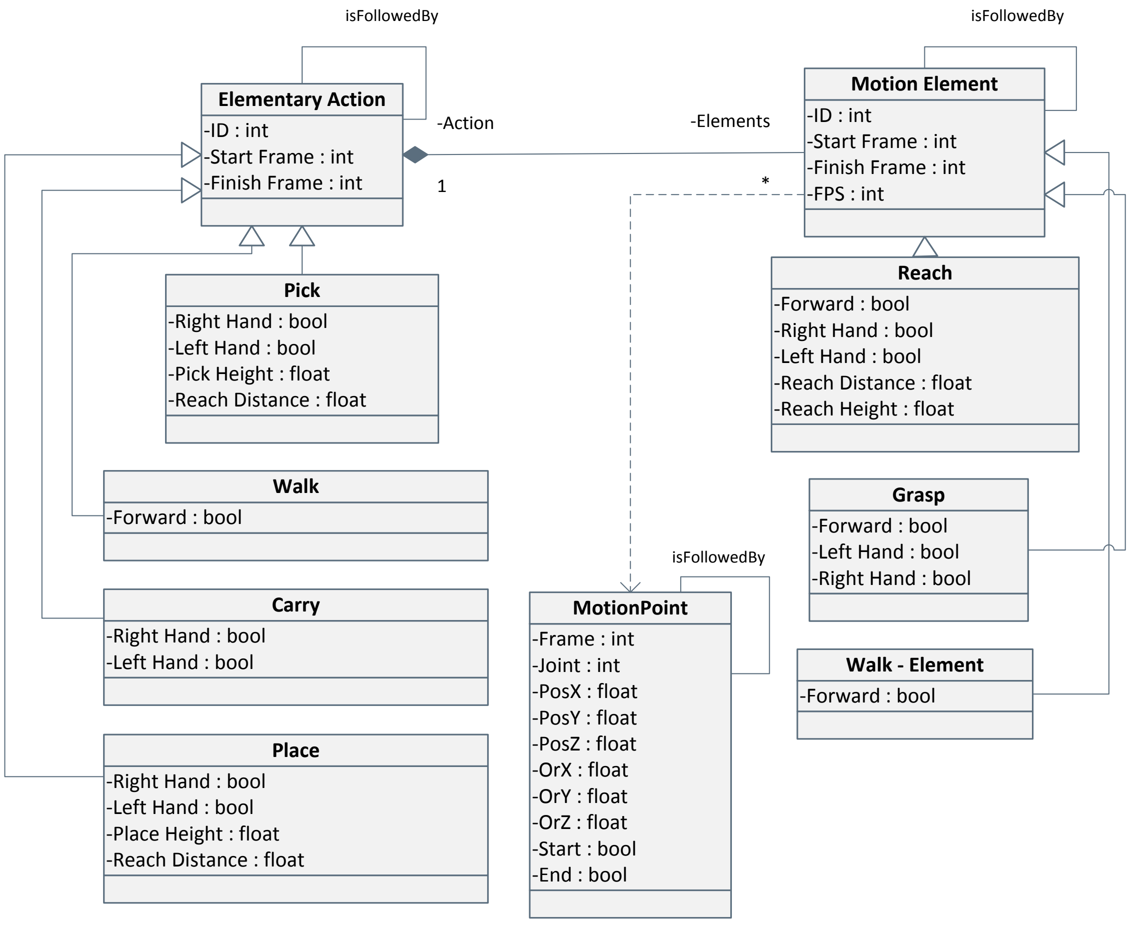


Figure 10: Elementary Actions and Motion Elements model

Different joints will be used for different motions; as another example, a *Reach* ME will be linked to motion points regarding the wrist joints. In addition, the starting and ending frames of all motions will be stored as well as the frame-rate that they were recorded with. This will be a sufficient time reference which, after clustering will also be translated as the duration of the motion in seconds for faster reasoning. The clustering of motions will be performed as soon as the MEs for one session are created and provided through the relevant module as individual data elements linked to a specific session. After the MEs have been stored as semantic individuals they will be clustered and translated (new individuals' creation).

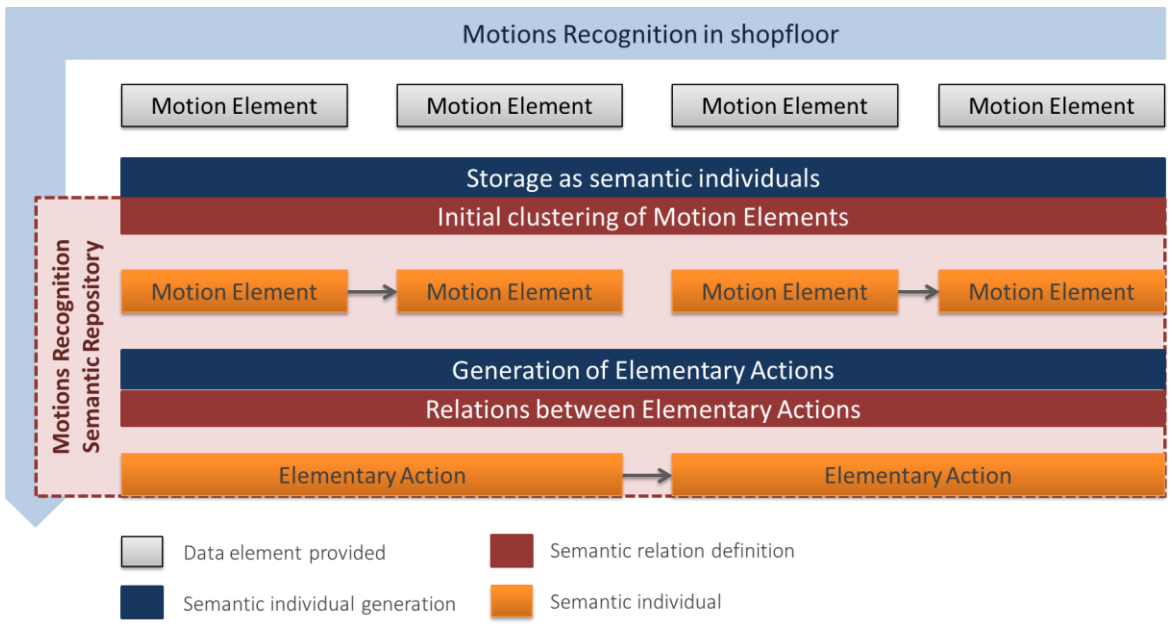

Motions Recognition in shopfloor
Motion Element
Motion Element
Motion Element
Motion Element
Storage as semantic individuals
Initial clustering of Motion Elements
Motion Element
Motion Element
Motion Element
Motion Element
Generation of Elementary Actions
Relations between Elementary Actions
Elementary Action
Elementary Action
Motions Recognition Semantic Repository
Data element provided
Semantic relation definition
Semantic individual generation
Semantic individual

Figure 11: Pipeline of Elementary Actions storage

5. INFRASTRUCTURE

The Motions Recognition Semantic Repository that will be developed by LMS, will get input and provide output to different applications. The main input will be obtained through experiments performed in the workshop and will be provided by the Sensor Management Platform as described in D3.1.1. After the processing of the input from the shopfloor, the translation of the data into semantic information will be performed and additional applications will need to identify the stored knowledge using specific mechanisms. The goal will be to retrieve the necessary motion parameters for motions in new simulation scenarios.

5.1. Kinematic data post-processing

The data generated in the workshop are kinematic and raw sensor data. The kinematic data will be stored directly after they have been recorded, together with the recording specifications. The sensor data coming from the rest of the sensors (IMU, pressure sensors etc.) will be fused as described in D3.1.1 and will be provided separately from the kinematic data. All data will be referring to the same timeline in order to allow the correlation of events coming from different sensors.

The kinematic data will be post-processed prior to their final storage (see deliverable D3.1.1) where the various optical signals coming from the optical sensing systems (multiple Kinect v2 for Windows) will be merged. Additional post processing of the kinematic data will be done after their fusion and will include the following processes:

- Peak removal to decrease jittering
- Interpolation to eliminate jittering

5.2. Sensor and workshop information processing

After the data have been stored to the data repository together with information regarding the workshop (provided by the engineer) they will be processed and translated into semantic individuals with specific relations. As described previously a specific model will be used which is provided also in the Appendix of this document. The main processing will be the storage of the Motion Elements and then the application of semantic rules for the interpretation into Elementary Actions with specific relations to the product components and resources. This will allow the further identification of motion parameters and associated constraints for their reuse in future scenarios. The main application that will process the data has been currently developed as a MATLAB® standalone application.

5.3. Motion identification and reuse

During planning and simulation generation, the planned motions will be used as input by a specific web service that will identify similar motions as well as their parameters. This service will communicate with the Best Fit simulation pipeline after the CNL breakdown in order to identify similar motions in terms of conditions and other input.

The specific components that will take part in the Motion identification pipeline include both the Translation modules as well as the Knowledge identification and reuse mechanisms. All the components will be part of the Recognised Motions Management System that will be developed by LMS. The specific components of the system are:

- Motions Management Interface (MMI): The MMI will provide the input for both storing motions from sensors, as well as identifying similar motions based on new, planned motions. The MMI will also provide output to other applications, mainly the parameters/ constraints of similar motions that had been performed in recorded sessions. The specification on how the constraints will be provided will be based on the input required by the WP2 developments.

- Ontology Creator (OC): The OC will organise and create ontology individuals based on the input from the Sensor Management Platform.

- Knowledge Management Abstraction Layer (KMAL): The KMAL will be the main component that will control the ontology and the applied rules. The KMAL will get specifications on specific actions based on input from the OC or directly from the KMI.

- Motions Recognition Semantic Repository (MRSR): The MRSR will contain all the motions and relevant information as described earlier.

The following sequence diagram shows the interactions between the specific components:

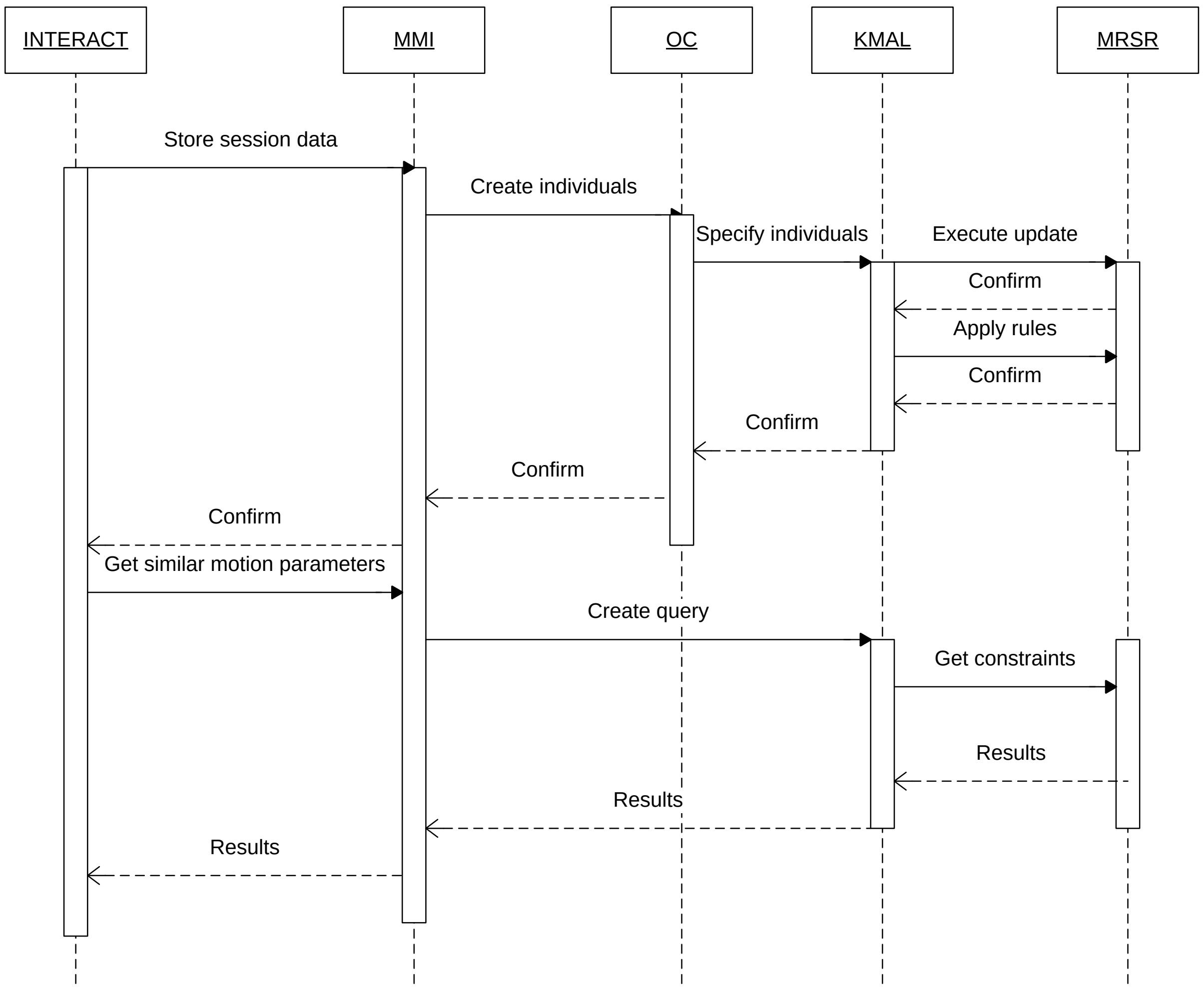


Figure 12: Recognised Motions Management System sequence diagram

The semantic model of the MRSR will be in OWL format and SPARQL Protocol will be used for scripting queries. The rules and queries will be configurable scripts containing placeholders that will be filled by the MMI in order to obtain only the relevant information based on the application or the motion that needs to be identified. The MMI and all relevant Web Services will be exposed over HTTP/SOAP protocols. The main framework that will be used for handling the ontology either by the

OC or KMAL will be the Jena framework[1] which is an open source framework that provides facilities for handling OWL ontologies. More specifically, the Jena Framework includes a general purpose rule-based reasoner. This reasoner supports rule-based inference over RDF graphs (core graphs used to construct OWL models) and provides forward chaining, backward chaining and a hybrid execution model. In particular, there are two internal rule engines, one forward chaining RETE[2] engine and one tabled datalog[3] engine - they can be run separately or the forward engine can be used to prime the backward engine which in turn will be used to answer queries.

[1] https://jena.apache.org/
[2] http://www.cis.temple.edu/~giorgio/cis587/readings/rete.html
[3] http://tinman.cs.gsu.edu/~raj/8710/f05/datalog.pdf

6. SUMMARY AND CONCLUSIONS

The main purpose of this deliverable was to provide the output as well as the perceptive behind the design of the model that will be used to store information regarding the execution of manual assembly operations. The main challenge was the definition of the motions' model that will be used to store specific Elementary Actions that can be reused by the rest of the INTERACT platform and mainly by the components developed in the context of WP2.

The definition of the motion model was based on the requirements of other components, mainly regarding the synthesis of motions after they have been stored. A two level motion model is proposed which can address requirements coming from both motion synthesis and motion recognition and reuse. Furthermore, the rest of the information that should be stored to make similar motions identifiable for reusing constraints in new simulations was defined and contains the necessary information for identifying motions performed to or with similar objects on the shopfloor. Finally, the first prototype of the motion recognition algorithm that will be used for recognising motions through kinematic data was described.

The first activities beyond the design will be the initial application and configuration of the motion recognition algorithm for different motions, including the creation of the MRSR using the first results of the algorithm. The motion recognition algorithm will be expanded to process additional sensor data (starting with finger pressure sensors). Based on the model, the first rules regarding motion clustering and the first queries regarding previous motion identification will be developed in order to validate the relevant technologies.

ABBREVIATIONS

BVH	Biovision Hierarchy character animation
EA	Elementary Action
HTTP	Hypertext Transfer Protocol
KMAL	Knowledge Management Abstraction Layer
ME	Motion Element
MMI	Motions Management Interface
MoCap	Motion Capture
MRSR	Motions Recognition Semantic Repository
OC	Ontology Creator
OWL	Web Ontology Language
RDF	Resource Description Framework
SOAP	Simple Object Access protocol
SPARQL	SPARQL Protocol and RDF Query Language
TXT	Text file

7. APPENDIX

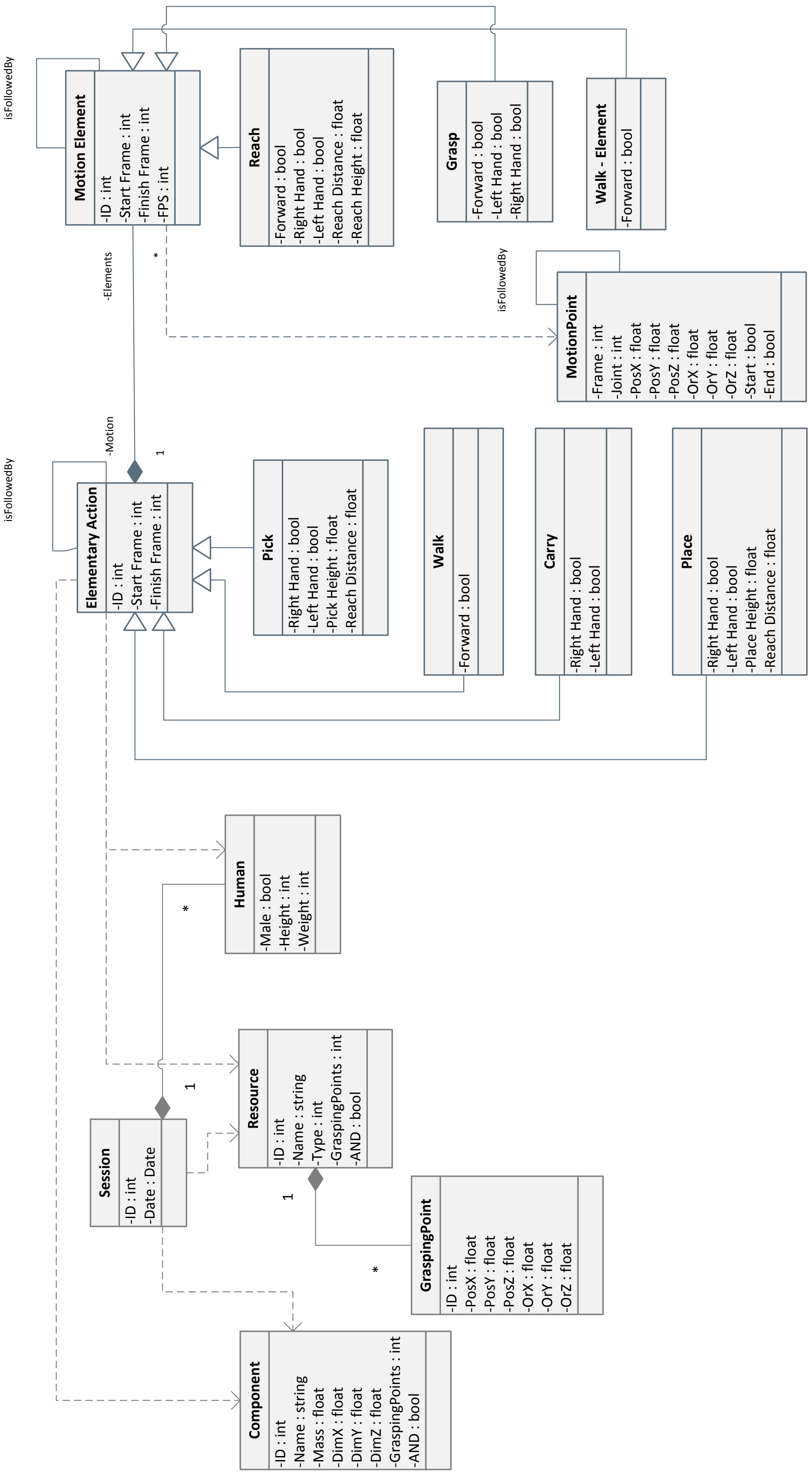


Figure 13: Model used for motion storage and reuse