

---

# **A Specification for Human Action Representation**

**John D. Ianni**

Air Force Research Laboratory

SAE routinely stocks printed papers for a period of three years following date of publication. Direct your orders to SAE Customer Sales and Satisfaction Department.

Quantity reprint rates can be obtained from the Customer Sales and Satisfaction Department.

To request permission to reprint a technical paper or permission to use copyrighted SAE publications in other works, contact the SAE Publications Group.



**GLOBAL MOBILITY** DATABASE

*All SAE papers, standards, and selected books are abstracted and indexed in the Global Mobility Database*

**ISSN 0148-7191**

Positions and opinions advanced in this paper are those of the author(s) and not necessarily those of SAE. The author is solely responsible for the content of the paper. A process is available by which discussions will be printed with the paper if it is published in SAE Transactions. For permission to publish this paper in full or in part, contact the SAE Publications Group.

Persons wishing to submit papers to be considered for presentation or publication through SAE should send the manuscript or a 300 word abstract of a proposed manuscript to: Secretary, Engineering Meetings Board, SAE.

**Printed in USA**

# A Specification for Human Action Representation

John D. Ianni

Air Force Research Laboratory

## ABSTRACT

This paper will propose a specification that will allow any Human Figure Model (HFM) to be controlled by external sources such as Human Performance Process Models (HPPMs). The specification includes an enumerated set of actions – each with a set of parameters that describe how the action is performed. Return codes and queries are also supported to update the calling program on the status of the action. The specification will be proposed as a Society of Automotive Engineers (SAE) standard.

## INTRODUCTION

If someone talks about simulating a human, what comes to mind? Is it a highly accurate representation of the physical human? Or do you think of simulating cognitive activity? Perhaps you think of both. Until recently, it was inconceivable to simulate realistic human activity because of the computational requirements. We need to prepare ourselves for simulations that far exceed many of our expectations. Computer hardware and software are advancing so fast that we need to ensure that a systematic approach is maintained.

In order to prepare ourselves for advancements in human simulation, we need to speak a common language. Several terms are used to describe simulations of humans but each has its own meaning. For example, a human figure model (HFM) or mannequin is a model of the physical human. Loosely speaking, an avatar is an HFM controlled with virtual reality equipment.

But simulations of humans should not be limited to physical tasks. Simulating a human's task performance sometimes must account for cognitive processes as well. Human Performance Process Models (HPPMs) are models of the cognitive human. Therefore for this paper we can depict these relationships as follows:

Mannequin = HFM

Avatar = HFM controlled with virtual reality equipment (loosely speaking)

HPPM = cognitive model (sometimes referred to as agents)

Given the maturity of HFM and HPPM software, we should soon see simulations that are highly accurate

depictions of both the physical and cognitive human. For this type of simulation, the term virtual human (VH) seems most appropriate. Thus we can think of this relationship as:

$$VH = HFM + HPPM$$

HFM's are used for various engineering and training applications. They can simulate actions of a driver, pilot, maintainer, manufacturer or other roles performed by humans. These models are comprised of numerous software functions and data to accurately represent anthropometry, biomechanics and other ergonomic factors.

HPPMs come in many varieties. Some allow a user to construct a task network in which various situations can be simulated. Although there may be advanced logic underlying these networks, they actually do not simulate how people think. Other software can make *decisions* based on beliefs, experiences, and heuristics formulated over its "life." Some of these can even simulate human factors like fatigue, anger, and mental overload.

Many applications can benefit greatly through a marriage of mind and body models. One of the simplest benefits is adding a realistic looking human to HPPMs. Some HPPM software offers a stick figure that represents humans, while others represent humans with unarticulated icons.

Human figure models will benefit from a standard action representation as well. By linking with HPPMs, HFMs should become easier to use for task analysis because they will be able to perform tasks autonomously. In addition, a standard user interface can be created using the standard interface.

Beyond simulation, a standard representation of tasks can be used to control any articulated figure such as a robot. After all, HFMs can be considered simulations of robots with mechanics similar to a humans.

## PREVIOUS WORK

The Air Force Research Laboratory, Sustainment Logistics Branch (AFRL/HESS) has conducted research in maintenance simulation on the DEPTH (Design Evaluation for Personnel, Training and Human Factors) program. Early in the program, an interface was created between the University of Pennsylvania (Penn) Jack model and

the Human Operator Simulator (HOS). HOS was an HPPM developed by MicroAnalysis and Design, Inc. for the Army Research Laboratory.

Although some success was achieved, several implementation shortfalls were noted. Probably the most damaging was the lack of robustness in the socket interface. The system frequently segment faulted during a simulation. Second, HOS was used to control every minute movement rather than controlling at the task level. In addition to excessive message traffic, the clarity of task networks was diminished.

The DEPTH program then developed a set of reusable, parameterized task primitives (called motion models) using Jack's built-in functionality that could be used to create more complex tasks. This demonstrated how tasks could be constructed in a logical hierarchy. The goal was to allow compound motion models to be reused as primitives for even more complex simulations. Dialog boxes were created for the pre-defined motion models that provided a friendly method to enter customization parameters. This functionality was useful but still had shortcomings. One shortcoming was that tasks could only be assembled sequentially – parallel tasks were not supported.

Although this demonstrated how the creation of tasks could be greatly simplified, complex simulations were still not simple to create. So AFRL investigated how an HPPM could be used to automate HFMs. BBN, Inc. and Penn were contracted to create an interface between Jack and the Operator Model Architecture (OMAR). Some success was achieved, but the interface was specific to OMAR and Jack.

Penn has also developed a specification for an action application programmer's interface (API). The API, titled Parameterized Action Representations (PARs), was developed under Air Force contracts to generate technical manuals automatically.

PARs provide a structure for action representation but the actual list of actions were not defined. The parameters for each specific action was obviously not defined either. Thus I reviewed various task analysis methods and specifications of HFMs and HPPMs to develop a streamlined list of actions, inputs and outputs.

Standard methods to control human motion are surfacing in groups such as the Motion Pictures Engineer's Guild (MPEG). MPEG-4 contains a standard for HFM anthropometry and joint motion control. Controlling human activity by specifying changes in joint angles is probably the most flexible method to control an avatar. However, a task-based approach offers several advantages:

1. Bandwidth. A seemingly simple transition from sitting to standing requires numerous joint transitions. Thus significantly fewer task-oriented instructions are required than joint-oriented instructions. This lower bandwidth can be critical if segments of the virtual

human are distributed over the World Wide Web or other wide area network.

2. Task analysis. If the HFM knows the task it is performing, analyses can be performed that otherwise may not be possible. For example, in lifting an object, a lift strength algorithm can be invoked to determine if the object was too heavy to lift to the designated height.
3. More intuitive. In performing a task, humans usually don't think about how their joint angles need to change. They are better able to think in terms of tasks and subtasks. A task-level interface should promote more intuitive creation of HFM simulations.

## THE SPECIFICATION

In coming up with this proposal, I examined various methods of task description including Methods-Time Measurement (MTM). I found that certain actions had considerable overlap because they were essentially accomplishing the same task. For example, carrying, lifting, pushing and pulling are methods used to move an object. They all can be accomplished with either or both hands and terminate once the object reaches a different location. With this in mind, I found that seven basic actions listed in Table 1 were sufficient to describe most work-oriented tasks. They allow for an HFM to move itself (Position), touch an object (Touch), get an object (Get), move an object to another location (Put), look somewhere (LookAt), use an object to act on another object (UseTool), or operate a machine (Operate).

The actions are intentionally broad, allowing them to be as flexible as possible without losing the action's identity. All input parameters are optional, allowing the invocation to be as simple as possible. If insufficient information is passed, the appropriate failure code (Table 2) is returned.

The inputs listed in Table 1 are unique to that action. In addition to the inputs listed, most actions share the following input parameters:

- Agent – Who will perform the action?
- Object – What object or point will be acted upon?
- Duration – How long should it take? This consists of two positive numbers, namely min and max. This allows a range (if min<max) or a specific time (if min=max) to be specified in milliseconds. Either or both numbers may be omitted to make the time segment open ended.
- Conditions – What states are required before (pre) and after (post) the action takes place? Known as "applicability conditions" in PAR description.

In addition to passing information to describe the action, it is important to inform the caller what happened. For example, was the action successfully completed? If so, how long did it take? Of not, what went wrong? Thus two variables are returned for each action invocation:

- Failure code – These codes are enumerated in Table 2. Despite the title, it is also used to inform of success and provide warnings. Failures can result in erroneous or insufficient information being passed, or in the simulated action.
- Duration – How long did the action take (in real, not simulation time)? If the simulated action fails, how much time was spent before the failure occurred?

The actions in Table 1 can be the basis for more complex actions. As an example, consider the removal and replacement of a heavy rack-mounted component. The steps involved as described in natural language are:

1. Remove 6 cables at the rear of the unit.
2. Raise the table platform to the front of the unit so the unit can slide onto it.
3. Remove the 4 rack fasteners.
4. Pull the unit out of the rack.

In the above example, many of the details of the task are omitted. Notice that there is no guidance for the human to move anywhere because the location and requirements of each subtask dictate the location and posture of the human. But since most human models cannot make such interpretations on their own, we will specify the first position change in the code below. The human name and duration information are also omitted in all except the first statement. Comments are italicized.

*The following instruction instructs the human model to walk to and face the first connector (back of unit). The human is to remain standing at the completion of this task in a position to grasp the connector.*

```
Position ( agent=human1,
           object=connector1,
           minimum_duration=1000,
           maximum_duration=2000,
           posture=stand,
           movement_type=walk,
           orientation=facing,
           purpose=grasp )
```

Notes:

- *human1 must be predefined or omitted if no others*
- *connector1 must be predefined*
- *The action must take between 10 and 20 seconds in estimated duration or be considered a failed step.*
- *The posture of the agent at completion of this step*
- *The agent will walk as opposed to run, crawl, etc.*
- *The agent's torso will face the connector*

*Reach for and grasp the first cable's connector.*

*Since purpose=disconnect it is actually unnecessary to indicated grasp\_type=precision.*

*If the HFM software is sophisticated enough, it should actually be unnecessary to perform this "Get" command*

*since it can be presumed given it's followed by a "Use-Tool" command.*

*Also notice human1 is omitted since there are no other humans in the environment.*

```
Get ( object=connector1,
      purpose=disconnect,
      grasp.hand=right,
      grasp.type=precision )
```

*Now disconnect the cable. This is handled with the Use-Tool command but the tool is the right hand.*

*Since a single connector is only disconnected once, repetitions is assumed to be 1.*

*The calling program could have also defined a function "Disconnect (connector1, right)"*

```
UseTool ( object=connector1,
          tool=hand,
          action=disconnect,
          grasp.hand=right,
          grasp.type=precision )
```

```
Put ( object=connector1,
      action=carry,
      base=connector1.center,
      target=right_target )
```

*Note: connector1.center is a predefined site.*

*Note: right\_target is a predefined point out of the removal path*

*\*\* The above commands are repeated for the remaining 3 connectors.*

*Now unscrew the first fastening bolt.*

*Notice the Position and Get functions were omitted since the HFM can presume these actions.*

*The calling program could have also defined a function "Screw (fastener1, unscrew, right)"*

*Since a single connector is disconnected only once, repetitions is assumed to be 1.*

```
UseTool ( object=fastener1,
          tool=screwdriver,
          action=unscrew,
          grasp.hand=right )
```

*Hand tools should have predefined grasp types.*

*\*\* The above command is repeated for the remaining 3 fasteners.*

*Pull out the rack unit using the two handles.*

```
Put ( object=rackunit,
      action=pull,
      grasp.hand=both,
      target=rackunit.handles )
```

Table 1. Proposed Actions and Parameters

Action	Action-Specific Inputs	Comments
<b>Position</b>	posture, movement_type, orientation, purpose, base	This is intended to move and/or re-posture the <i>agent</i> . <i>posture</i> values include stand, sit (straight/slumped), squat, kneel (one/two knees), lay (prone/supine/side), crawl, and bend. <i>orientation</i> can be specified to change the direction that the human model faces. <i>movement_type</i> describes whether the agent will walk, run, crawl, roll, scoot, or ride. <i>object</i> specifies the goal location which depends on <i>purpose</i> which indicates what the agent plans to do with the object. <i>base</i> is a point that is to align with (snap to) the <i>object</i> .
<b>Touch</b>	end_effector	This directs an agent to touch the <i>object</i> with the <i>end-effector</i> . Any point on the body can be used as the <i>end-effector</i> , even elbow, knee or forehead.
<b>Get</b>	purpose, grasp_data	This directs an <i>agent</i> to reach for and grasp an <i>object</i> . The <i>purpose</i> parameter will inform the <i>agent</i> what will be done with the <i>object</i> after it is grasped – moved, turned, carried, etc. <i>grasp_data</i> is specified in Figure 1.
<b>Put</b>	action, grasp_data, base, target	The <i>object</i> is used to specify what is transported and partially how it is transported. <i>action</i> codes are listed in Table 4. <i>grasp_data</i> is specified in Figure 1. <i>base</i> is a point on the agent or object that is used to align with the <i>target</i> . If <i>base</i> is a point on the <i>agent</i> , then the <i>object</i> will continue to be held at completion (i.e., carry and hold) <i>target</i> specifies the surface or point where the <i>base</i> is to be aligned.
<b>LookAt</b>	max_vis_field, focus	<i>object</i> specifies where the agent will look. <i>focus</i> is “yes” if a specific location must be focused on.
<b>UseTool</b>	tool, action, hand, repetitions	<i>tool</i> is comprised of the category code and specs. The categories are described in Table 5. The specs (not yet defined) will define the tool size, accessories, etc. <i>action</i> can be remove, replace, loosen, tighten, connect, disconnect, or hammer. <i>object</i> specifies where the <i>tool</i> is to be applied, such as a bolt, screw or point on a surface. <i>hand</i> indicates the hand(s) used with the tool (right, left, both, right with left used to guide, left with right guide) <i>repetitions</i> indicates how many times the action will be repeated.
<b>Operate</b>	action, unit1, unit2	This instructs the <i>agent</i> to use a machine's controls (specified by <i>object</i> ) including steering wheels, break pedals, switches, keyboards, mice, and other devices (Table 3). <i>unit1</i> and <i>unit2</i> (also described in Table 3) describe the resulting displacement of the control. <i>unit2</i> is only used in cases where a two-dimensional action takes place.

Table 2. Failure Codes

Failure Code	Description
<b>Success states</b>	
0	Not a failure; action was successfully completed with no problems identified
1	Not a failure; action completed but unspecified problem occurred
2	Not a failure; action completed but possibly unsafe
3	Not a failure; action completed but human fatigued
4	Not a failure; action completed but possible strain may result
5	...
<b>Warning &amp; Danger Messages</b>	
128	Undefined warning state (other than those listed below)
129	Multiple warnings occurred (need to query for more clarification)
130	Task is unsafe
131	Object is too hot
132	Object is too cold
133	Noise level too high
134	Too close to sharp object
135	Radiation
136	...
<b>Programming errors</b>	
256	Undefined program error (other than those listed below)
257	Multiple program errors occurred (need to query for more clarification)
258	Feature not supported by this software package
259	Tool not available
260	Insufficient information: other data needed
261	Insufficient information: agent not specified
262	Insufficient information: target not specified
263	...
<b>Task failures</b>	
384	Undefined task failure (other than those listed below)
385	Maximum duration exceeded
386	Required position could not be achieved
387	Multiple task failures occurred (need to query for more clarification)
388	Strength exceeded
389	Insufficient accessibility
390	Human cannot get into position to complete task
391	Target visually obscured
392	Target out of visual range
393	Insufficient lighting
394	Human fatigued
395	...

Table 3. Operate Action Codes

Code	Units	Description
<b>Toggle devices</b>		
0	off=0, on=1	Move toggle switch
1	mm	Press button (units signify the amount the button is depressed)
<b>Sliding/shifting devices</b>		
8	-1 = reverse 0 = neutral 1-7 = actual gears 8 = auto transmission drive 9-11 = low gears 1-3	Shift gear - either on steering wheel or floor (HFM system must have gear settings pre-defined).
9	unit1 = proximal-distal in mm unit2 = left-right in mm (proximal/left are negative)	Move free floating gear shift
10	Mm (proximal/left are negative)	Move slider/fader
<b>Rotating devices</b>		
32	Degrees	Turn dial
33	Degrees	Turn wheel - one hand
34	Degrees	Turn wheel - two hands
<b>Foot controls</b>		
48	Mm (depress positive)	Pedal - linear (e.g., auto breaks and accelerators)
49	Degrees	Pedal - swiveling (e.g., pivoting)
50	Revolutions	Pedal - rotating (e.g., bike pedals)
<b>Computer input devices</b>		
64	Number of characters	Type random characters on computer keyboard
65	ASCII character	Type character on computer keyboard
66	unit1 = proximal-distal in mm unit2 = left-right in mm (proximal/left are negative)	Use mouse

Table 4. Object Manipulation Action List

<b>Grasp</b>	Attach hand(s) to object
<b>Release</b>	Detach hand(s) from object
<b>Push</b>	Extend arms to move object away
<b>Pull</b>	Retract arms to move object closer
<b>Press</b>	Push arms down to move object down
<b>Lift</b>	Pull arms upward to move object up
<b>Insert</b>	Place a component inside another
<b>Extract</b>	Remove a component from inside another
<b>Attach</b>	Connect objects
<b>Detach</b>	Disconnect objects
<b>Latch</b>	Movement to secure an object
<b>Unlatch</b>	Movement to unsecure an object
<b>Open</b>	Open an access panel
<b>Close</b>	Close an access panel
<b>Turn</b>	Physical manipulation of an object
<b>Hold</b>	Keep an object grasped in hand
<b>Carry</b>	Hold on to an object while walking

Table 5. Hand Tool Codes

Code	Tool Description
0	Hand(s) - bare or gloved
1	Screwdriver
2	Pliers
3	Hammer
4	Fixed wrench
5	Torque wrench
6	Socket wrench
7	Allen wrench
8	Ratchet wrench
9	Other wrench
10	Drill
11	Powered screwdriver
12	Other power tool

Code	Tool Description
13	Knife
14	Wire cutter
15	Breaker bar
16	
17	
18	
19	Soldering iron
20	Gages/testers
21	Nut runner
22	Hoist/jack
23	Cleaning tool
24	Lighting unit
25	EVA tool

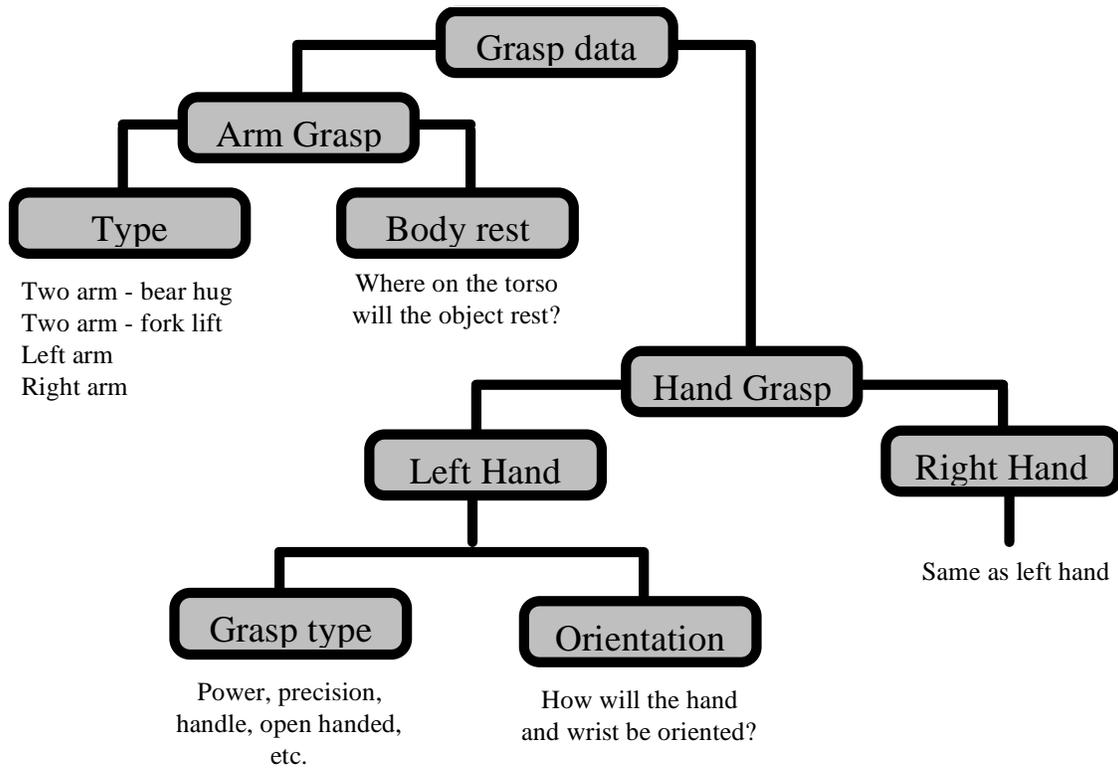


Figure 1. Grasp Description

## CONCLUSIONS

Defining a specification that meets everyone's needs may be impossible. However, I attempted to define a set of actions and parameters that should address most engineering applications. The variable list of parameters allows the caller to be ambiguous if desired, but the HFM must be able to make reasonable judgments.

## ACKNOWLEDGEMENTS

Thanks for comments on this proposal from Dr. Michael Biferno and his team at Boeing, Sylvain Marie of Genicon and Mr. John Quinn formerly of the University of Dayton Research Institute. Special thanks to Dr. Joseph McDaniel for sharing his expertise through the years.

## REFERENCES

1. Perlin, Ken; Goldberg, Thomas (1996). *Improv: A System for Scripting Interactive Actors in Virtual Worlds*, 1996 SIGGRAPH Proceedings
2. Boyle, E. (1991). "Human-Centered Technology: Ends and Means," *Human Centered Technology for Maintainability: Workshop Proceedings*, AL-TP-1991-0010, Brooks Air Force Base, TX: 38-39.
3. Douville, B. J. (1995). *PaT-Net User's Guide*. University of Pennsylvania, Philadelphia, PA: 1-6.

4. Granieri, J.; Crabtree, J.; and Badler, N. (1995). *Production and Playback of Human Figure Motion for 3-D Virtual Environments*. University of Pennsylvania, Philadelphia, PA.
5. Ianni, J. (1991). "Crew Chief: Present and Future," *Human Centered Technology for Maintainability: Workshop Proceedings*: 32-36.
6. Karger, Delmar W., (1977). *Engineered Work Measurement*, Industrial Press.
7. Vujosevic, R. and Ianni, J. (1996). "Maintenance Motion Models," *CALS Expo International '96 Conference Proceedings*.

## CONTACT

John Ianni has been involved in human modeling and simulation since 1985 when he started working on the Crew Chief project. He acted as program manager of the DEPTH project that was completed in 1997. You can reach John at:

AFRL/HESS  
2698 G Street  
Wright-Patterson AFB, OH 45433-7604  
Phone: (937)255-1612 Fax: (937)255-6555  
E-mail: [john.ianni@he.wpafb.af.mil](mailto:john.ianni@he.wpafb.af.mil)