University of Pennsylvania
ScholarlyCommons

Center for Human Modeling and Simulation | Department of Computer & Information Science

July 2002

Virtual Humans for Validating Maintenance Procedures

Norman I. Badler
University of Pennsylvania, badler@seas.upenn.edu

Charles A. Erignac
Boeing Phantom Works, USA

Ying Liu
University of Pennsylvania

Follow this and additional works at: http://repository.upenn.edu/hms

Recommended Citation
Badler, N. I., Erignac, C. A., & Liu, Y. (2002). Virtual Humans for Validating Maintenance Procedures. Retrieved from http://repository.upenn.edu/hms/31

Postprint version. Copyright ACM, 2002. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in *Communications of the ACM*, Volume 45, Issue 7, July 2002, Pages 56-63.
Publisher URL: http://doi.acm.org/10.1145/514236.514264

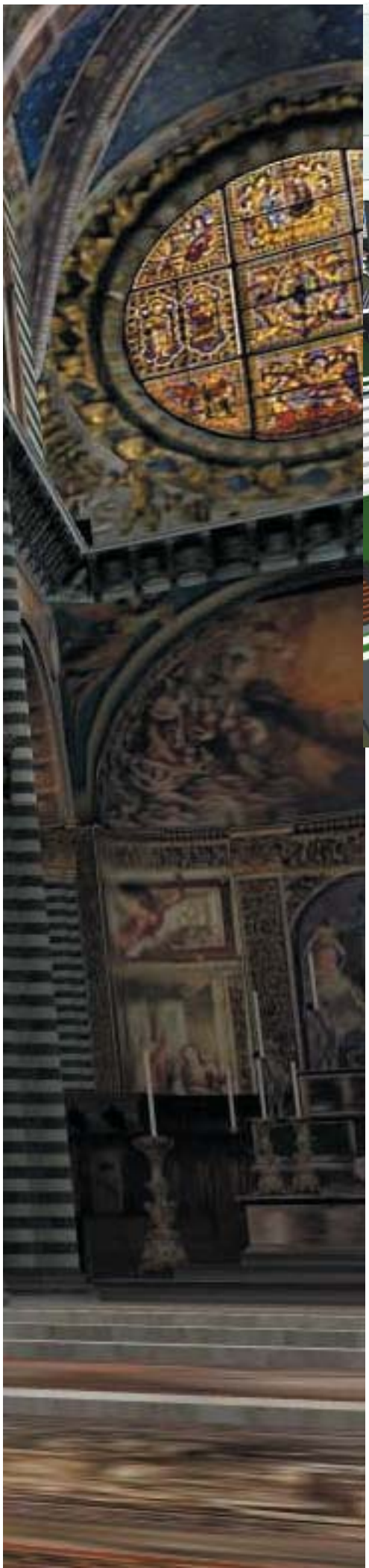
This paper is posted at ScholarlyCommons. http://repository.upenn.edu/hms/31
For more information, please contact libraryrepository@pobox.upenn.edu.

Virtual Humans for Validating Maintenance Procedures

Abstract
They can be sent to check the human aspects of complex physical systems by simulating assembly, repair, and maintenance tasks in a 3D virtual environment.

Comments
Postprint version. Copyright ACM, 2002. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in *Communications of the ACM*, Volume 45, Issue 7, July 2002, Pages 56-63.
Publisher URL: http://doi.acm.org/10.1145/514236.514264

This journal article is available at ScholarlyCommons: http://repository.upenn.edu/hms/31



[BY NORMAN I. BADLER, CHARLES A. ERIGNAC, AND YING LIU]

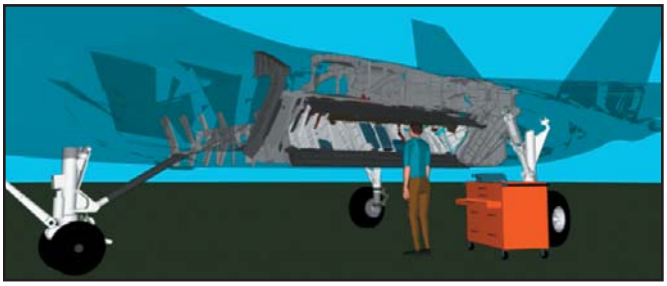
VIRTUAL HUMANS FOR VALIDATING MAINTENANCE PROCEDURES

They can be sent to check the human aspects of complex physical systems by simulating assembly, repair, and maintenance tasks in a 3D virtual environment.

(top) *Jack* virtual humans as part of the manufacturing analysis of air conditioner units (iSiD, Japan). (left) Virtual tour guide Luigi shows visitors around a photorealistic computer model of the cathedral of Siena, providing information about one of Europe's most beautiful gothic churches; the virtual environment was developed as an exhibit for the World Exhibition EXPO 2000 in Hanover, Germany (Fraunhofer Institute for Computer Graphics, Darmstadt, Germany).

The design of complex physical systems must accommodate the human technicians who assemble and maintain them. Technicians need instructions guaranteeing successful and safe task performance. By integrating computer-graphic human models, 3D geometric environments, and a system's functional models, designers and instruction authors gain a computational tool for automating task validation (ATV). Language-level instructions are interpreted as parameterized procedures that control embodied agent models to execute the tasks and report success and failure conditions. Interpreting these instructions requires integration of spatial (geometric), visual, and functional reasoning. Here, we demonstrate a form of task simulation originating in instructions, revealing action execution failures when validating a component-removal process in a representative aircraft.

Figure 1. General view of the F-22 upper-left weapons bay.



Instructions guide us to do things we have not done before or remind us of forgotten tasks, supporting our efforts to assemble, operate, and service a multitude of technological artifacts. But no computational procedure can guarantee a particular instruction will be executed correctly or successfully. Indeed, jokes abound concerning ambiguous, impossible, complex, or simply unintelligible instructions. Instructions translated across natural languages are especially prone to mistakes and unintentional humor. We read the words, and the words make sense; we know how to manipulate the tools and see the parts; but connecting what is asserted to explicit actions is often formidable. Understanding instructions fundamentally challenges computational paradigms of language understanding, spatial representation, and human action, yet demands their interaction, integration, and visualization.

The Center for Human Modeling and Simulation at the University of Pennsylvania has a longstanding interest in connecting language instruction and human action [2, 3, 5]. While a significant set of commercial and experimental interactive human modeling and task analysis tools is available today [6], development lags in endowing them with autonomy, intelligence, planning, and reasoning. Our work has been directed at interpreting language-level instructions as human-model animations. Action representations are not new in artificial intelligence [11], but the recent emergence of powerful graphical humans with significant behavioral capacity drives recent interest in making virtual humans behave like real people.

Large engineered systems typically require lifetime maintenance and repair. Military fighter aircraft, for example, are highly complex, and maintenance costs over the lifetime of the aircraft can easily exceed their acquisition costs [4]. Design analysis for human maintainability and maintenance procedures lower costs by reducing errors, task complexity (time), and instruction manual updates.

Validating a task instruction means the instruction can be realized with known actions, which can indeed be performed in context by a suitable range of human maintenance technicians. A computational realization of the validation process we've developed uses computer-graphic human models, a representation for task steps, geometric models of the components, and functional system models. Automating the validation process should result in nominal, efficacious, and safe maintainer actions, as well as graphical animations suitable for visualization, explanation, and task training.

Checking the Human Aspects

ATV is a computational process designed to check the human aspects of a physical design by simulating task execution in a 3D virtual environment. The virtual human is programmed to use stored knowledge to generate appropriate actions to carry out its instructions, such as selecting tools and sequencing subactions, according to accessibility conditions. If the instructions are incorrect or the design flawed, the virtual humans need to report failures, such as those involving reach, clearance, strength, and visibility failures, as well as those involving hazards created by contact with moving parts or unacceptable temperatures, fluids, pressures, or electrical currents. A procedure is valid if no failures occur across a sufficiently large range of anthropometric body sizes.

Here we examine a software system—built on the Parameterized Action Representation (PAR) [1] developed at the University of Pennsylvania—to support these capabilities, showing how it works in a real-world maintenance instruction setting. Although the related scenarios involve airframe examples, they readily generalize to other environments. The image on page 57 shows a factory where instructions could be

Figure 2. *Jack* reaching for the connectors on top of the power supply.

A procedure is valid if no failures occur across a sufficiently large range of anthropometric body sizes.

used to guide a number of virtual workers simultaneously. The examples cover typical maintenance tasks, including reaching into confined spaces, disconnecting electrical and hydraulic lines, releasing fasteners, and extracting assemblies.

Instruction Representation

A maintenance procedure is a sequence of actions performed by one or more technicians with the explicit goal of replacing a component or testing a function. Each action changes the system structure or state. The procedure should be designed to keep the system in a safe state, that is, without allowing (potential) hazards or unnecessary damage to its components. U.S. Air Force Technical Orders, or instructions, are textual maintenance procedures for aircraft and support equipment. They include schematics and drawings to complement the text. Although originally produced as printed manuals, they are now electronically accessible through portable workstations.

An instruction starts by describing the initial system state, indicates the human technician's functions and location relative to the system, and lists any special support equipment. The instructions then describe the procedure as a tree of actions with execution proceeding depth-first. Actions can be either elementary (a leaf) or complex (a nonleaf). Elementary actions come in three types:

Structural. A component is removed, installed, positioned, connected, or disconnected;
State changing. A tank is drained, a valve opened or closed, a switch turned, or a battery loaded; and
Cognitive. A sensing action reads a dial or observes fluid flow.

Intermediary. PAR is an intermediary between language instructions and computer animations of virtual humans. In general, a PAR is defined for each action an agent can perform. PARs are stored in an action dictionary, or Actionary. Each PAR contains both high- and low-level parameters describing an action. High-level parameters include the performing agent and the physical objects involved in the action; low-level parameters include motion qualities and spatial values, such as position and orientation. Each PAR also contains conditions describing the action context. Applicability conditions specify the conditions that need to be true for the action to be executed. Preparatory specifications are a list of `<condition, action>` statements preparing the environment for the required action. If a condition is not satisfied, its corresponding action is executed. A PAR can describe either an elementary or a complex action. For an elementary action, the underlying action code (such as walk, reach, direct attention, grasp) is executed. A complex action lists subactions to be executed in sequence or in parallel. Parameterization mitigates the explicit storage of all possible actions; for example, `reach(agent_A, object_site)` becomes a single elementary action. Termination conditions specify when an action is completed and often require sensing actions. When an action is completed, post assertions update the state of the world.

Each virtual human in the environment is associated with an agent process responsible for executing PAR actions. In order to perform an action, the parameters of the PAR are bound to participating agents,

objects, locations, orientations, directions, and forces. Many of the parameters have default values (such as walking to change location, reaching to contact a part, and moving with nominal velocity). Once instantiated, the PARs are placed in an action queue inside an agent process. Actions are popped from the queue and sent to a process manager where their conditions are tested and subactions expanded. Ultimately, each PAR is associated with a motion generator that moves the 3D geometry of the human figure and objects in the environment to perform the action. During an action's performance, the process manager continually monitors action execution, checking for termination conditions, sequencing subactions, managing agent resources (such as attention), and handling failures.

While human models are usually embedded in interactive graphics tools, task analysis may be automated through procedural controls. For example, accommodation analysis tests exemplar human models representing a population's significant anthropometric variability to find the range of fit or reach of, say, an operator or technician within the geometry of a workplace. A task that can be performed by 90% of the human aircraft maintainer population is preferred to a design requiring only an individual of one particular size. With ATV, users work less with direct manipulation and more with instructions using parameterized procedures to search and optimize task execution.

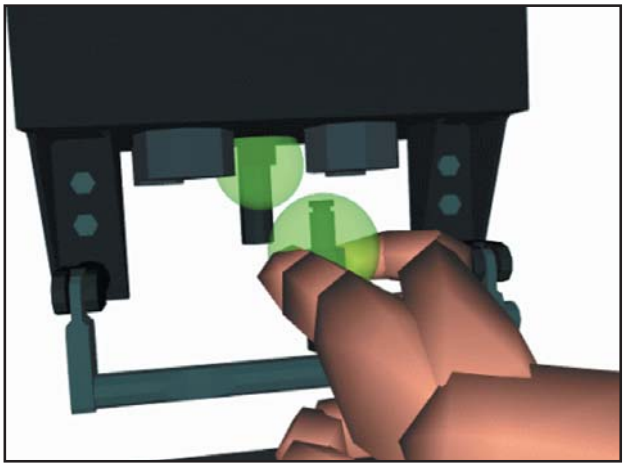
Failure detection and handling. Action failures in PAR execution can be used for sequence control and for recognizing hazards. Agents experiencing failures attempt to recover by trying an alternative low-level motion strategy or attempting a new action to produce a state of the world in which the failure would not occur. Based on failures, a planner could generate new paths or actions. For example, we studied possible access and movement failures in removing an F-16 fuel tank assembly with three fluid connections. While the connections were mechanically independent, human access constraints forced one order from the six possible disconnection orders (see the sidebar "Disassembly Sequencing").

Integrated system. A maintenance procedure is generally expressed as a hierarchical plan spelled out in text. Using natural language as given is highly desirable for the sake of not having to translate operator instructions into some computational form. We therefore use the Actionary and natural language processing to parse verbs into PAR and bind other words to objects and spatial directions.

Simulating a maintenance procedure with PAR requires defining the tree of corresponding PAR actions. The top-level (root) action represents the goal, such as `replace_valve_x`. The PAR lists the subaction PARs and their associated applicability conditions. Because PAR is parameterized, its subactions and executable elementary action procedures can be invoked in a wide range of spatial contexts.

The PAR simulation engine is a plug-in to the human animation system *Jack* (see www.ugs.com/products/efactory/jack/) linking executable actions (such as locomotion, reach, attention, and grasp) to real-time animation procedures. As the task is simulated, success is documented and visualized by the action animation and task failures explicitly detected. A human observer can check a failure to determine whether the task is truly impossible at that point or the program or planner needs to take an alternative approach. The benefit of ATV is that human-analyst effort for validating the accomplishment of procedures focuses only on the most difficult cases, rather than being required for every elementary task.

Figure 3. Warning spheres indicating contaminated hydraulic disconnects.



AUTOMATING task validation tries to catch situations where the instruction is not humanly executable.

Implementation and Demonstration

We've demonstrated the ATV concept in a scenario based on an F-22 aircraft maintenance task involving the removal of an avionics component (an electronic

power supply) from the aircraft's upper-left weapons bay (see Figure 1). As technical documentation, we used computer-aided design geometry and the task's logistic support analysis (LSA) record provided by Lockheed-Martin (the aircraft's manufacturer). (The LSA record is essentially the precursor to the actual instruction manual given to the human maintenance technician.) The demonstration shows how PAR can be used to model a virtual human's knowledge and how an agent uses that knowledge to execute a maintenance task. It also shows PAR's ability to detect various failures.

System architecture. The ATV components consist of an agent controlling the virtual human, motion generators for the elementary actions, geometry observers for detecting joint rotations and object states, collision-detection code, and the user interface. Most are written in C++, with the exception of the graphical user interface, which is a Tcl/Tk script and controls the agent and monitors its progress. PAR modules include the PAR engine executing the agent's actions and a PAR simulator running the environmental model with physics and hazard models.

Physical objects in PAR and associated status.

Name	Quantity	Status
Power_Unit	1	Locked / In Place / Supported / Extracted
Handle	1	Stowed / Transit / Extended
Connector	9	Connected / Disconnected
Fastener	8	Locked / Released

Agents and physical objects. The virtual human is a PAR agent—an object that can perform actions. We created a PAR model for each physical object interacting with the virtual technician. Task execution affects various object attributes, as reflected in a status field (see the table); depending on status, an object might trigger a number of subactions.

The LSA record includes five maintenance steps:

- Rotate the handle at the base of the unit;
- Disconnect the five top electrical connectors;
- Disconnect the four bottom electrical connectors;
- Disconnect the two coolant lines; and
- Unbolt the height bolts retaining the power supply to the airframe, support it, then remove it.

We created a set of PAR actions corresponding to these instructions that can be performed by the virtual human, including: disconnect the connector; reach from point to point; release fasteners; and extract the power unit. The disconnections can all be done by hand; the action for releasing fasteners involves a socket wrench. Figure 2 shows *Jack* reaching for connectors on top of the power supply. Note that the instructions are general, purposely not including specialized details like: explicit orderings of subtasks (which connectors to remove among the sets); specifics on tools required; attention instructions (where to look); and hazards other than a mention of contents (electricity or coolant). Instruction preambles cite general cautions or warnings about hazards throughout the procedure.

Failures. Tasks can fail for many reasons; for example, a human maintenance technician might be unable to reach something, lack the strength to move or hold a part, or lack the proper tools. The instructions might be inherently ambiguous; spatial referents, such as objects, might be imprecisely or incompletely specified; or an action might actually be applied in different ways. A natural language instruction needs to be bound to specific PARs and objects, and multiple parses may be disambiguated in the geometric environment [10]. Assuming a single parse, ATV tries to catch situations where the instruction is not humanly executable. We therefore model three kinds of failures:

Out-of-reach. The virtual human cannot reach a particular object;
Collision. The hand of the agent or the solid thing it is holding collides with the environment while moving; and
Failed applicability conditions. Conditions might include the freeing of known mechanical restraints, so failure is inevitable on any attempt to extract the power supply while it is still attached.

The out-of-reach and applicability failures bring all agent activities to a halt. Collision, however, may cause an action to terminate with either success (a grasp) or failure (thwarted). An operator or task planner uses failure to choose a different item from the set, or, if accessible ones are not available, abort the procedure.

Environmental model. The Air Force instruction preamble warns the human technician to use adequate protection against the toxic cooling fluid and wipe off any spills. An environmental model with green transparent spheres represents areas contaminated by the fluid (see Figure 3). Visualization techniques (such as temperature color-coding and transparent surfaces) may be used to show potentially hazardous parts.

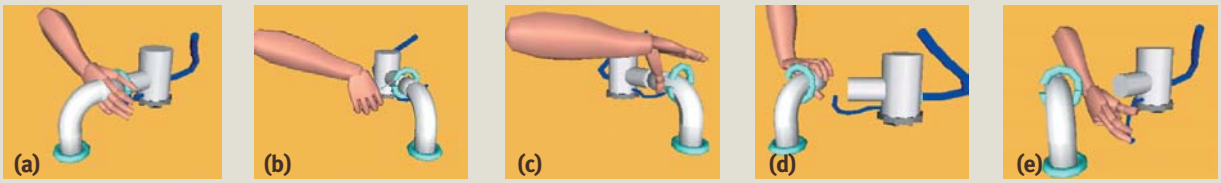
We designed four failure scenarios to demonstrate ATV. The first three showcase a specific type of failure, representing a designer's or instruction-author's successive attempts to simulate and correct a maintenance procedure until it completes successfully in the fourth scenario:

Disassembly Sequencing

Disassembly sequencing depends on human access. One of our case studies involves an F-16 internal fuel tank vent situated behind a rather small access panel inhibiting the direction of a human technician's approach and clearance. Allowable and necessary actions are limited to: open elbow coupling; slide sleeve on elbow; rotate elbow; disconnect pressure sense tube; open pressurization tube coupling; slide sleeve on pressurization tube; and disconnect pressurization tube. Some maintenance actions succeed, others fail due to collisions between the virtual human's arm and the part geometry or due to connections between geometry segments. For example, if human maintenance technicians attempt to disconnect the pressure sense tube without first creating sufficient clearance, their arms collide with the pipe elbow. Figure (a) shows the attempted action and its result. During interaction, whenever a technician's arm collides with the assembly geometry, the colliding segments are temporarily highlighted.

Designers of maintenance procedures find a feasible ordering of actions by trying various actions. They first create clearance around, say, the pipe elbow in order to reach the pressure sense tube. Trying to rotate the pipe elbow without, say, first detaching the elbow coupling, generates an error message, indicating the elbow coupling is still attached and must still be removed. The series of actions to provide clearance to the pressure sense tube are illustrated in the figure sequence. The user first detaches the pipe elbow coupling (b), then slides the coupling sleeve on the pipe elbow (c), and finally rotates the elbow (d).

When the geometry reflects collision-free access to the pressure sense tube, the action of disconnecting the pressure sense tube succeeds; an example of the successful reach and disconnect action is shown in (e). C



Attempted actions and results, leading to collision-free access and movement.

Scenario 1 (out-of-reach failure) as in Figure 3. The agent begins the procedure, but since it's standing on the ground it fails to reach the first connector on top of the power supply. This scenario ends with a reach-error message displayed via the user interface.

Scenario 2 (collision error). The virtual agent is standing on an elevated platform, but its first reach fails again. This time the PAR selected an arbitrary connector that happened to be toward the rear of the unit; the agent's hand collides with the connector in front, and a collision error is reported. A spatial planner should have been able to predict this outcome, electing one of the front connectors for the reach instead.

Scenario 3 (incomplete instructions). After the electrical disconnection order is corrected for accessibility, the procedure runs until the technician tries to extract the power supply. But the extract action lacks the preparatory specifications requiring it to release the fasteners. The user interface reports this condition, and the error is corrected by adding applicability conditions to release any mechanical constraints.

Scenario 4 (errors corrected). The extract action completes successfully.

During the third and fourth scenarios, the functional model flags the hydraulic connectors with visible contamination markers, as in Figure 3.

Conclusion

ATV is both a useful concept and a challenging research area. Adapting simulation, visualization, and validation of maintenance procedures, an ATV system can predict the possible outcomes of a maintenance task, including what may go wrong and what should be corrected before a prototype is built or the physical system tested. ATV could greatly reduce errors in design and maintenance instructions, along with the costs of design and instruction modification.

Although these validation experiments are preliminary, we've already derived several useful principles of how to computationally analyze maintenance procedures:

Noninteractive and interactive techniques play different roles in task validation. As graphical human models

have evolved as interactive tools, the burden of imagining a design's maintainability has shifted to iteratively and visually testing whether the design is viable with a range of human technicians. The number of possible body configuration spaces is large, and human-factors engineers have to interactively test a range of body sizes to determine plausible reaches. A task's failure might be due to a design flaw or simply to forgetting to exhaustively check for solutions. Obstacle avoidance and task-sequence planning are the obligation of the people writing the maintenance instructions. If robust search procedures are available to check human pose and reach, a simulation could reveal problematic situations requiring refined instructions or even redesign. (One of the authors (Liu) is developing fast heuristic algorithms for reaching into confined spaces to account for spatial search, body structure, and strength limits.)

Explicit sequences of actions are not always expressed in task instructions. Planning may be needed to establish the essential steps needed to fulfill instruction intentions. PAR serves this function by listing subactions and giving preparatory specifications to force actions establishing the necessary conditions. Instruction execution also depends on the constraints holding parts to other parts; constraints on attached parts need to be broken to permit removal. Because this is such a challenging geometric reasoning and planning process, only a few functional disassembly planners have been built [7, 8]. Planners also need to consider part extraction paths [9], as well as the presence of the human body. Yet another challenge remains: how to integrate disassembly order, part extraction, and human access to produce a maintenance procedure that can be performed by a virtual, and subsequently, human technician.

Understanding part functional behaviors is important in creating safe action sequences. Just because a connection is accessible does not mean it can be broken. The connection may be fragile, slippery, hot, or contain a hazardous substance, a fluid under pressure, or electrical current. This information may not be available in a geometric model. System schematics may exist but are unlikely to connect directly with the model. Designers need to build better databases of such information, and planners have to use them, so any lack of appropriate cautions and warnings are detected and asserted during task validation.

The ATV framework can be used to simulate and analyze procedures for technician reach, visibility, strength, and potential hazards. ATV can provide designers of complex physical systems an instructable virtual human agent to assess a physical system's assembly, repair, and maintenance tasks. Since an animation results from the simulation, ATV can also provide visualization services for training technicians in the safe execution of these tasks. Computer science can therefore play a fundamental role in design, analysis, and training for human operator effectiveness and safety for the people performing physical maintenance tasks under difficult or potentially dangerous conditions. C

References

1. Badler, N., Bindiganavale, R., Allbeck, J., Schuler, W., Zhao, L., and Palmer, M. Parameterized action representation. In *Embodied Conversational Agents,* J. Cassell, J. Sullivan, S. Prevost, and E. Churchill, Eds. MIT Press, Cambridge, MA, 2000.
2. Badler, N., Palmer, M., and Bindiganavale, R. Animation control for real-time virtual humans. *Commun. ACM 42,* 8 (Aug. 1999), 64–73.
3. Badler, N., Phillips, C., and Webber, B. *Simulating Humans: Computer Graphics, Animation, and Control.* Oxford University Press, New York, 1993.
4. Barron, M. and Abshire, K. *Design Maintainability Demonstration For F-22 Avionics.* Tech. Rep., Lockheed-Martin Aeronautical Systems Co., U.S. Air Force, 1997.
5. Bindiganavale, R., Schuler, W., Allbeck, J., Badler, N., Joshi, A., and Palmer, M. Dynamically altering agent behaviors using natural language instructions. In *Proceedings of Autonomous Agents* (Barcelona, Spain, June 3–7). ACM Press, New York, 2000, 293–300.
6. Chaffin, D. *Digital Human Modeling for Vehicle and Workplace Design.* Society of Automotive Engineers, Warrendale, PA, 2001.
7. Homem de Mello, L. and Lee, S. *Computer-Aided Mechanical Assembly Planning.* Kluwer Academic Publishers, Boston, MA, 1991.
8. Jones, R., Wilson, R., and Calton, T. Constraint-based interactive assembly planning. In *Proceedings of the IEEE International Conference on Robotics and Automation* (Albuquerque, NM, Apr. 20–25). IEEE Computer Society Press, Los Alamitos, CA, 1997, 913–920.
9. Schroeder, W., Lorensen, W., and Linthicum, S. Implicit modeling of swept surfaces and volumes. In *Proceedings of Visualization* (Arlington, VA, Oct. 19–21). IEEE Computer Society Press, Los Alamitos, CA, 1994, 40–45.
10. Schuler, W. Computational properties of environment-based disambiguation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL'01)* (Toulouse, France, July 6–11). Morgan Kaufman, San Francisco, 466–473.
11. Winston, P. *Artificial Intelligence, 3rd Ed.* Addison-Wesley, Reading, MA, 1992.

Norman I. Badler (badler@seas.upenn.edu) is a professor of computer and information science, Director of the Center for Human Modeling and Simulation, and Associate Dean for Academic Affairs of the School of Engineering and Applied Science at the University of Pennsylvania, Philadelphia.
Charles A. Erignac (charles.a.erignac@boeing.com) is a technologist in Boeing Phantom Works, Seattle, WA.
Ying Liu (liuying@gradient.cis.upenn.edu) is a Ph.D. candidate in the Department of Computer and Information Science at the University of Pennsylvania, Philadelphia.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

© 2002 ACM 0002-0782/02/0700 $5.00