Natural Language Based Virtual Prototyping on the Web

Bernhard Jung, Thies Pfeifer, Jure Zakotnik

Knowledge-Based Systems Group, Faculty of Technology,
University of Bielefeld
http://www.TechFak.Uni-Bielefeld.DE/techfak/ags/wbski/

Abstract

This contribution describes a WWW-based multi-user system for concurrent virtual prototyping. A 3D scene of CAD parts is presented to the users in the web browser. By instructing the system using simple natural language commands, complex aggregates can be assembled from the basic parts. The current state of the assembly is instantly published to all system users who can discuss design choices in a chat area. The implementation builds on an existing system for virtual assembly made available as a web service. The client side components are fully implemented as Java applets and require no plugin for visualization of 3D content. Http tunneled messaging between web clients and server ensures system accessibility from any modern web browser even behind firewalls. The system is first to demonstrate natural language based virtual prototyping on the web.

1 Background and Motivation

A current trend in CAx technology – historically one of the major driving forces in the development of 3D based human-computer interfaces – is the move towards the Internet. For example, 3D PartStream.NET[1] offers internet based enabling technology that allows manufactures to publish 3D CAD content to online catalogs and e-commerce solutions. Users of such systems can *view* CAD models over the web, *rotate* the parts, and *zoom in and out*. Building on PartStream technology, Technicon[2] has developed a system with additional *configuration* capabilities. Customers can choose their desired product configurations by selecting predefined values for parameters such as width, height, and diameter. Then the specific CAD model is generated on the fly from the specified parameters. The generated 3D model can also be *downloaded* in different file formats for use in own designs. These examples represent novel types of applications which add value to existing, off-line CAD solutions. They do not, however, provide functionalities traditionally offered by CAD systems, such as the free form modeling of novel parts or the assembly of complex aggregrates out of a given set of base parts. One likely reason for these limitations is rooted in the menu-overloaded, notoriously complex user interfaces of current CAD systems whose transfer into the web browser is hard to imagine.

[1] www.3dpartstream.net
[2] www.technicon.com

We have developed a web-based system that addresses some of these shortcomings. Its goal is to enable the interactive *assembly* of complex digital product models (*virtual prototypes*) out of a given set of CAD parts. Instead of relying on time-consuming menus and input fields for one-by-one entry of part mating constraints or numerical values for part transformations, high-level *natural language* instructions are used to build and modify the assembly. The system further offers *multi user* and *chat* capabilities, thus supporting applications in concurrent engineering. A critical design goal was to ensure system access even behind firewalls.

The web-based solution builds on an existing system, the Virtual Constructor, described in more detail in section 2, which already provides for natural language input but was so far restricted to the off-line case. Section 3 describes the web browser based user interfaces and the overall client-server architecture of the internet version of the Virtual Constructor. Section 4 describes the underlying http-based framework for two-way message transport between web clients and the web server. Finally, we conclude and discuss the proposed system, especially the natural language based approach to virtual prototyping within the larger context of so called *post-WIMP* interfaces (windows, icons, menus, pointers) advocated in virtual environments research.

Figure 1: In the Virtual Constructor, complex aggregates can be assembled from 3D visualized parts using direct manipulation or natural language instructions.

2 The Virtual Constructor

The Virtual Constructor is a knowledge-based system that enables an interactive assembly of 3D visualized mechanical parts to complex aggregates. The user can both directly manipulate the virtual scene using the mouse or similar input devices and instruct the system using simple commands in natural language. Various operations such as assembly, disassembly, and rotation of sub-assemblies are supported by a knowledge-based description of the objects' mating possibilities. A key feature of the system is that the current state of the assembly is dynamically conceptualized by step-keepingly matching the geometry scene against a structured model of the target aggregate. Dynamic knowledge representations are created when constructed aggregates are recognized as assembly groups of the target aggregate. The internal representations are further modified when, according to their use, the specific functions of single parts in the target aggregate are determined. Therefore, verbal instructions can always refer to the current state of the assembly.

The Virtual Instructor incorporates several methods from Artificial Intelligence that allow for a natural, task-level, and closed-loop human-computer interaction:

- Knowledge-Based, Real-Time Assembly Simulation: Knowledge-based descriptions of the visualized mechanical objects' connection ports enable the simulation of part assembly and disassembly as well as aggregate modifications along rotational and translational degrees of freedom. Top-level concepts our port taxonomy include extrusion ports for modeling peg-in-hole-like insertions, plane ports for modeling connections between co-planar faces, and point ports for modeling point-like connections that induce no translational degrees of freedom.

- Dynamic Scene Conceptualization: Geometry scene objects are represented in the frame-based language COAR [6]. Inferences over COAR representations include aggregate conceptualization, by which constructed aggregates are recognized as subassemblies of the target aggregate, and role assignment, by which components are reclassified w.r.t. the underlying concept hierarchy according to their use in larger assemblies. COAR representations also integrate spatial informations, such as position, size, distance, or orthogonality, which are inferred on need from the geometry scene.

- Multimodal Input

  – Natural language instructions: Typed and spoken input is supported. Verbal instructions may refer to spatial and visual properties of objects and – due to dynamic conceptualization – to currently assembled aggregates and functional roles of objects.

  – Direct manipulation: Parts can be assembled, disassembled, and rotated by using the mouse or similar input devices. Direct manipulation operations build on a knowledge-based snapping mechanism. E.g., for object assembly, the user moves an object close to another object; the snapping mechanism then completes the fitting process in a collision-free manner.

The Virtual Constructor is implemented as a distributed system consisting of various software processes or *agents* that communicate via message passing. The agents can be divided into logic and presentation layer agents (see Figure 2). The logic level agents provide the core functionality such as simulation of various assembly operations, processing of natural language instructions, and dynamic scene conceptualization. The presentation layer agents visualize the 3D scene and provide forms for input of natural language instructions.

The Virtual Constructor has so far been tested on desk-top as well as large screen interfaces, such as an interactive wall [7, 8]. The high-level instructability using natural language and modular system architecture make the Virtual Constructor a suitable basis for the design of an Internet-based system for virtual prototyping.

3 A Web Based Platform for Concurrent Virtual Prototyping

Virtual prototyping is concerned with the design and evaluation of CAD-based product models (virtual prototypes) with the objective of reduced the product development cycle times. Virtual prototyping often involves concurrent activities of diverse groups of individuals who may be located at geographically different places. Through its task-level instructability using natural language commands, the Virtual Constructor supports the rapid design of virtual prototypes

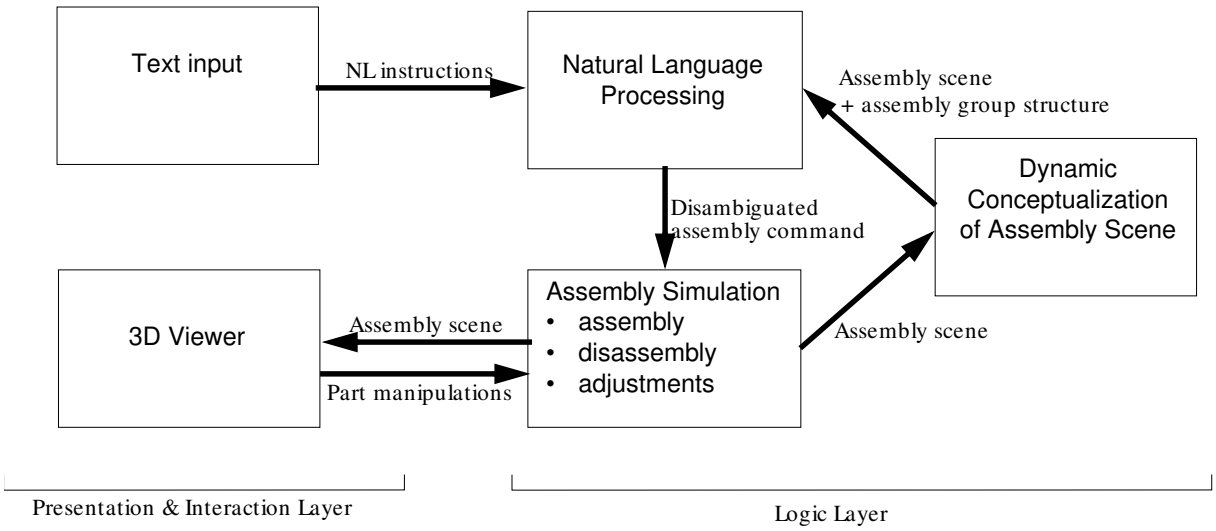

Figure 2: Functional architecture of the Virtual Constructor. In the web-based version of the Virtual Constructor, the logic modules reside on the web server. The presentation modules are realized as Java applets and run on the client side.

assembled from pieces of standardized construction kits [8]. However, previous versions of the Virtual Constructor were restricted to operation by a single user and assumed access to more or less powerful graphics workstations. Our goal was to add support for concurrent design activities by geographically widely distributed development teams.

As the Virtual Constructor is implemented as a distributed system, *in principle*, a multi-user version could be realized by simply running its user interface modules at the client side which at system startup would connect to some central server running the logic modules (cf. Figure 2). While this solution would work in principle, in practice several difficulties could arise. First, the 3D presentation modules would need to be installed and maintained at the client side. Although the C++ coded presentation modules are available for several Unix platforms (Irix, Solaris, Linux) a large subset of PC users would still be excluded. And second, communication between presentation and logic modules in the Virtual Constructor is based on plain TCP/IP connections which are often inhibited by restrictive firewall configurations. This would exclude even more potential system users. For these reasons, we opted for a somewhat more radical approach for the realization of the internet platform for virtual protoyping: System access should require only a web browser.

In sum, the following design choices were made for extending the Virtual Constructor into an internet platform for concurrent virtual protoyping:

- System access should be possible from anywhere. No assumptions on the users' computing environment are made except for internet access and a modern web browser with Java support. Especially, no browser plugins for display of 3D content should be required and no assumptions on firewall setups made (except for WWW access over standard http connections).

- The system should provide multi user and chat capabilities. The internet faciltitates system access by distributed design teams. Interactively assembled virtual prototypes can be reviewed and discussed by all participants.

- Furthermore, concerning the user interface, we decided to strictly rely on natural language instructions to initiate various simulated assembly operations (assembly, disassembly, adjustments) on the 3D parts. Menu-based interactions for assembly modeling in conventional CAD systems are unnecessarily complex to transfer to the web browser. On the other hand, task-level natural language commands already have proven feasible in desktop and virtual reality versions of the Virtual Constructor.

As constraint on the implementation, the existing code base should be reused to the largest extent possible.

3.1 System Architecture

The web-based multi-user platform for virtual prototyping builds on the Virtual Constructor by making its core modules available on a web server. The presentation layer components for visualization of 3D content and input of natural language command are implemented as Java applets running in the web browser.

The server-side components essentially consist of existing logic layer agents of the Virtual Constructor. These agents are responsible for the processing of natural language instructions, the simulation of assembly operations, and the dynamic conceptualization of constructed assemblies. The only addition to existing agents is a simple gatekeeper agent that keeps track of users being logged into the system. The gatekeeper also maintains a history of assembly instructions and chat contributions of the individual users. Further, the gatekeeper effectively synchronizes scene changes requested by the multiple users, processing instructions one at a time. Every time the current assembly scene changes as result of a scene manipulation, and every time a constructed assembly group matches a predefined model in the Virtual Constructors knowledge base, the new assembly state is published to the presentation applets at the client side.

The client-side components comprise a 3D visualization of the current assembly state and a chat forum including a text input field for natural language instructions (Figure 3). All client side components are implemented within one Java applet.

The 3D visualization extends shout3d's java applet[3] for interactive display of 3D content which avoids the need for VRML plugins. As usual for 3D visualizations, the 3D scene can be freely navigated. As a restriction, when compared to the desktop version of the Virtual Constructor, mouse based direct manipulations of the parts are not realized in the web-based presentation applets, as they require fast communication channels and visualization frame rates not necessarily achieved in Internet situations. However, all part manipulations can be triggered over natural language instructions.

The chat forum consists of an input field and a text area showing past contributions of all users. Text input to the system can be of two kinds: A contribution to a discussion about the assembly state or an instruction to change the assembly scene. To differentiate between the two types of input, assembly instructions must be addressed to "Max", an invisible user interface agent present in the chat forum who represents the logic layer modules of the Virtual

[3] www.shout3d.com

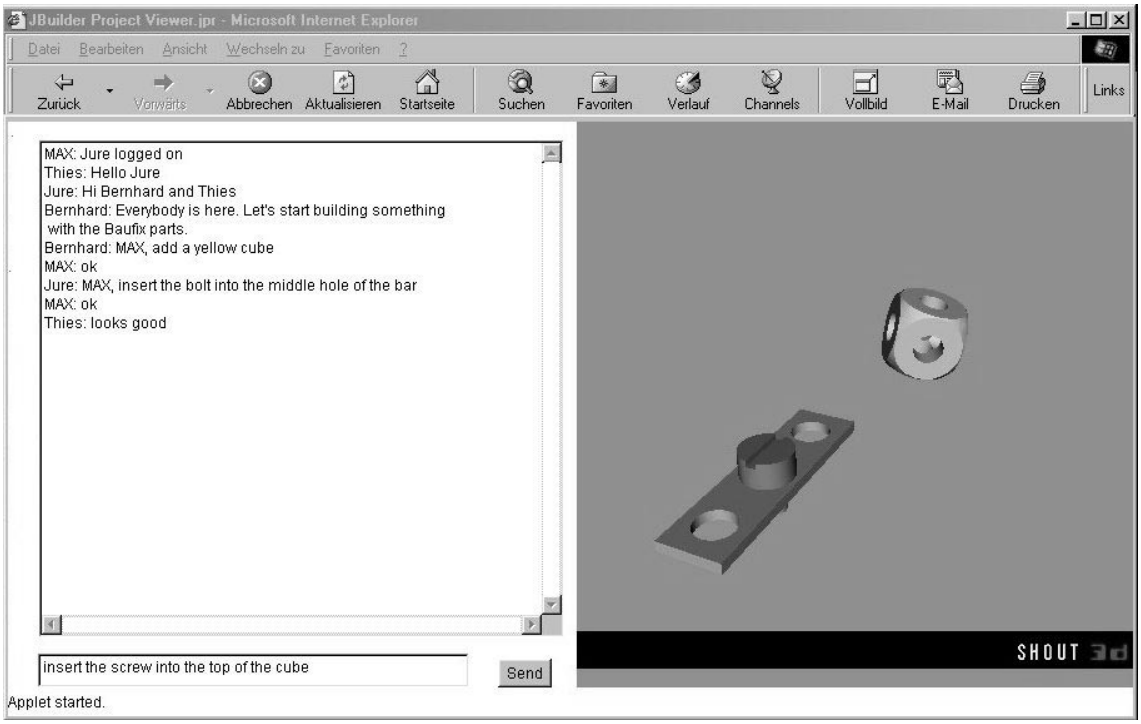

Figure 3: The web browser interface of the virtual prototyping system consists of a 3D view of the assembly scene and a chat area. Assembly instructions are addressed to MAX, a user interface agent who is always present in the chat forum. Note: The picture shows a software prototype. The final version will include further interface elements, such as buttons for selection of different modes for 3D navigation.

Constructor[4]. Max also acknowledges the successful completion of assembly operations and will notify the users if assembly instructions could not be performed, e.g. due to mispelled worlds or unconnectablilty of two given CAD parts.

The assembly instructions are passed on to the natural language processing component of the Virtual Constructor, so the full range of instruction types for assembly, disassembly, and adjustments of assemblies is supported. More specifically, natural language instrutions can make reference to visual (such as color, size) and spatial attributes (orientation of parts w.r.t. camera, w.r.t. each other) of single parts as well as constructed assemblies. Some examples from the Baufix construction kit domain are:[5]

- Max, put the bolt from below into the middle hole of the bar
- Max, attach the propeller to the airplane
- Max, rotate the lower bar crosswise to the tail unit
- Max, disconnect the right axle from the undercarriage
- Max, give me a long red screw

[4]MAX is the acronym for the Multimodal Assembly eXpert, an anthropomorhpic agent present in a virtual reality application of the Virtual Constructor [9]. A visual representation of MAX might be added to the web based system described here in the future.

[5]Instructions are translated from German. The Virtual Constructor (or Max) is currently monolingual.

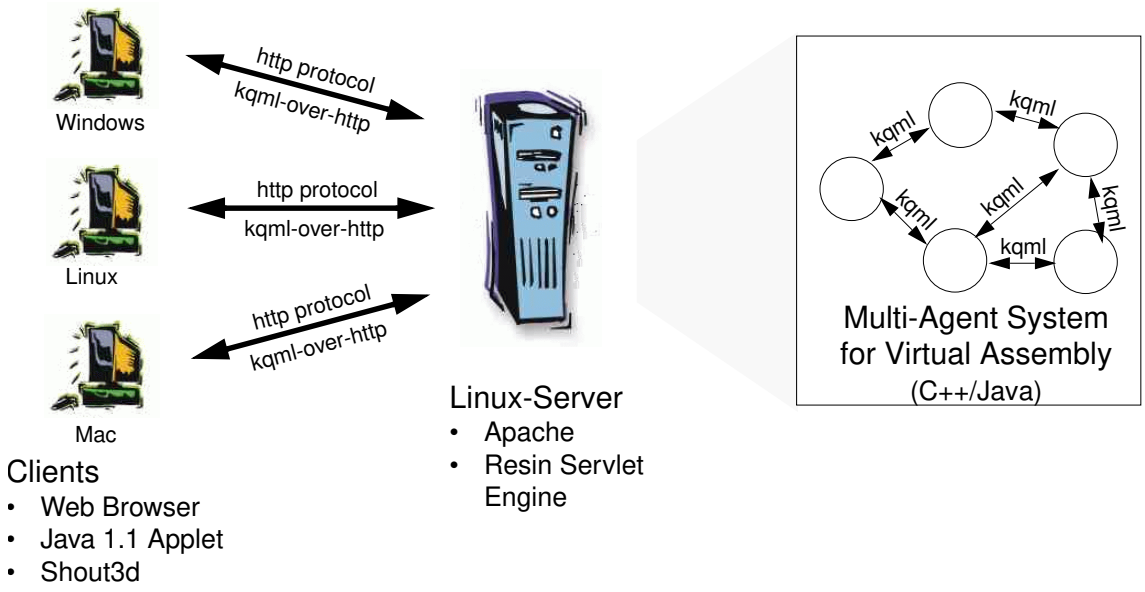

Figure 4: The web-based multi-user platform for virtual prototyping builds on the Virtual Constructor, a multi agent system for virtual assembly which is made abvailable on a Linux web server. At the client side, platform independent Java applets present the assembly scene to the user.

Together with the natural language instruction, also the user's current location is transmitted to the Virtual Constructor. This is necessary for the resolution of spatial adjectives such as *"left"*, that can only be successfully evaluated if the user's current location (camera viewpoint and point of interest) in the 3D environment is known.

4 Agent Communication over the WWW: KQML-over-HTTP

The software agents that make up the Virtual Constructor communicate with each other by passing KQML-based messages. KQML (the *knowledge query and manipulation language*) defines a textual message format and message exchange protocol with well-defined semantics [5]. The logic-layer agents of the Virtual Constructor provide the high-level protocols for interacting with presentation-layer agents responsible for visualization of 3D content and textual input of natural language instructions. Thus, the message contents generated by the logic-layer agents can be reused for the web-based visualization clients. However, the presentation-layer agents developed so far aim at applications in local area networks, e.g. in immersive Virtual Reality settings, and utilize (TCP/IP) socket-based communications at the transport level. In wide area networks such as the WWW communications based on plain socket connections or similarly CORBA or RMI are problematic, as client side firewalls might be configured in too restrictive ways. Therefore, a messaging framework was implemented that relies on http connections only: KQML-over-HTTP. In addition to the request-response messaging of standard http, the framework also supports publish-and-subscribe type messaging, where messages are streamed from the server to the clients. The messaging framework is integrated into a web-capable agent model that hides all details of the communication layer from the

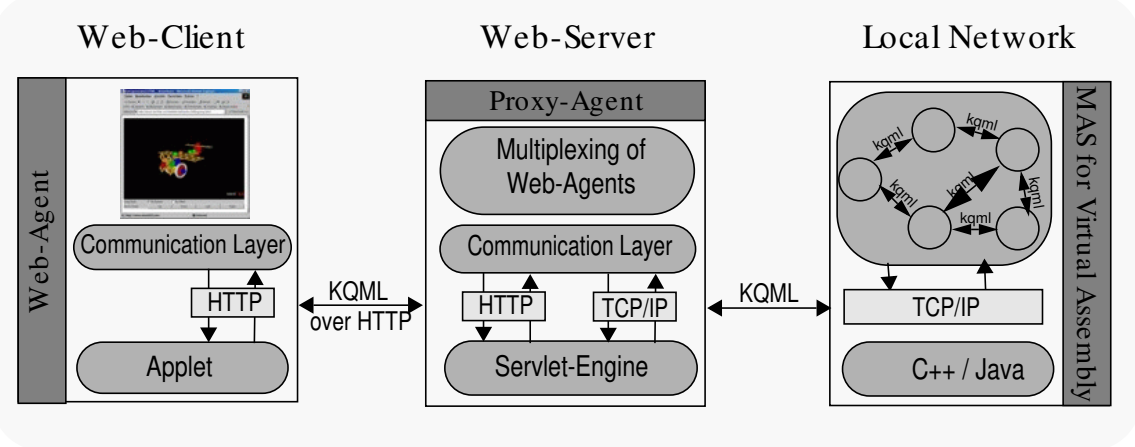

Figure 5: The agents communicate by passing of KQML based messages. Communication with applets on the client side tunnels http, avoiding problems with restrictive firewall set-ups.

application.

Conceptually, the web-based visualization and chat applets that run at the client side should be considered as first-class agents being able to communicate with any other agent, no matter if this agent is hosted on the web server, on some other computer in the same local network as the web server, or as an applet on the computer of some other system user. A straightforward implementation of applets as agents is however prohibited by the applet security model as the applets are running inside a sandbox that places certain restrictions on the establishment of communication channels. Our solution to the agentification of the visualization applets into *web-agents* builds on a server-side *proxy agent* that manages the message exchange between applets and the other agents (Figure 5).

The proxy agent resides on the web server and is implemented as a Java servlet. On the one side, it implements the same communication protocols as the logic layer agents on the web server. I.e. the proxy agent opens TCP/IP based connections to the local agents and is thus able to exchange messages with these agents. On the other side, the proxy agent keeps persistent http connections to the applets. Technically, this is achieved in the following way: The applet initiates the communication by opening an URL connection to the servlet. The servlet responds to the applet's request in its doPost() method where – to keep the connection persistent – it enters an infinite loop, waiting for messages from the logic layer agents to be forwarded to the applet. The establishment of a persistent connection between applet and servlet is necessary to implement the publish-and-subscribe messaging required by KQML. For example, the visualization applets subscribe a service offered by one of the logic layer agents that always publishes the current transformations of the scene parts. Whenever the scene changes, the new part positions are sent to the proxy agent and automatically forwarded to the applet. In other words, the publish-and-subscribe-type messaging ensures that each system user always is presented the current state of the evolving assembly without having to repeatedly press the reload button in the web browser.

5 Conclusion

This contribution described a web-based multi-user environment for virtual prototyping. Unlike other web based virtual prototyping environments currently advertized as innovative add-ons to CAD systems, the proposed system supports the interactive design of novel virtual prototypes assembled from CAD-based parts. A prerequisite for the real-time assembly of the 3D visualized parts is of course, that the parts' mating features are represented in the system. The little extra effort for modeling of mating features is however more than justified in situations where many products are assembled from the pieces of the same construction kit. In other work, we have developed a feature recognizer that is able to automatically detect the mating features of arbitary CAD parts [3]. Also, unlike other web based systems in the context of virtual prototyping, no plugins for viewing and manipulating of 3D content and no special firewall configurations are necessary. The only requirement for system access is a Java-enabled web browser.

The implementation of the multi-user platform for virtual prototyping is based on an existing system, the Virtual Constructor, whose functionality is made available as a web service. This design choice – as opposed to providing all of the virtual prototyping functionality as client-side applets – allows the reuse of the large existing C++ code base to the largest extent possible. As welcome side-effect, far less program code needs to be transferred to the client thus minimizing download time. The web-based version of the Virtual Constructor is hosted on a Linux system running the Apache web server and Caucho's Resin servlet engine[6]. The viewing and chat applet is implemented in Java 1.1 as newer Java versions are typically not supported by common web browsers.

A further noteworthy feature of the described system is that it abondons conventional menu-based interactions and replaces them by natural language instructions. In Virtual Environments research, such 3D interfaces with non-conventional human-computer interaction methods have been labeled as *post-WIMP* (windows, icons, pointers, menus) [11] or even *SILK* (speech, image, language, vision) [2] interfaces. Natural language interfaces have been proposed and successfully applied for control of various virtual environments, especially for instruction of animated characters, e.g. [1, 4, 12, 10]. The use of natural language at the human-computer interface is doubtlessly appealing as it exploits the most intuitive communication modality of the human user. Our work on the Virtual Constructor demonstrates that task-level, natural language based interactions techniques can also be applied in virtual prototyping domains.

References

[1] N.I. Badler, B.L. Webber, J. Kalita, and J. Esakov. Animation from Instructions. In N.I. Badler, B.A. Barsky, and D. Zeltzer, editors, *Making Them Move. Mechanics, Control, and Animation of Articulated Figures*, pages 51–93. Morgan Kaufman, San Mateo, CA, 1991.

[2] W. Barfield and T.A. (Eds.) Furness. *Virtual Environments and Advanced Interface Design*. Oxford University Press, 1995.

[6] www.caucho.com

[3] P. Biermann. Interaktives VR-System zur halbautomatischen Generierung von Wissen über Verbindungsmerkmale CAD-basierter Bauteil-Modelle. Diplomarbeit, Universität Bielefeld, 2000.

[4] B. Blumberg and T. Galyean. Multi-level direction of autonomous creatures for real-time virtual environments. In *Computer Graphics Proceedings, SIGGRAPH-95*, 1995.

[5] T. Finin, Y. Labrou, and J. Mayfield. KQML as an agent communication language. In J. Bradshaw, editor, *Software Agents*. MIT Press, 1995.

[6] B. Jung. Reasoning about objects, assemblies, and roles in on-going assembly tasks. In *Distributed Autonomous Robotic Systems*, volume 3, pages 257–266. Springer, 1998.

[7] B. Jung, M. Latoschik, S. Kopp, T. Sowa, and I. Wachsmuth. Virtuelles Konstruieren mit Gestik und Sprache. *Künstliche Intelligenz*, 2000/2:55–11, 2000.

[8] B. Jung, M. Latoschik, and I. Wachsmuth. Knowledge-based assembly simulation for virtual prototype modeling. In *IECON'98 - Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society*, volume 4, pages 2152–2157. IEEE, 1998.

[9] S. Kopp and B. Jung. An anthropomorphic assistant for virtual assembly: Max. In *In Workshop Communicative Agents in Intelligent Virtual Environments, Autonomous Agents*, 2000.

[10] Jan-Torsten Milde. Action-centered communication with an embedded agent. In *FLAIRS, Special Track on natural language processing and Human Computer Interaction*, 1998.

[11] A. van Dam. Post-WIMP user interfaces. *Communications of the ACM*, pages 63–67, 1997.

[12] I. Wachsmuth, B. Lenzmann, T. Jörding, B. Jung, M. Latoschik, and M. Fröhlich. A virtual interface agent and its agency. In W.L. Johnson, editor, *Proceedings of the First International Conference on Autonomous Agents*, pages 516–517. ACM Press, 1997.