XSTEP: A Markup Language for Embodied Agents

Zhisheng Huang, Anton Eliëns and Cees Visser
Intelligent Multimedia Group,
Vrije University Amsterdam, The Netherlands
{huang,eliens,ctv}@cs.vu.nl

Abstract

In this paper we propose an XML-based markup language, called XSTEP, for embodied agents, based on the scripting language STEP. Thus, XSTEP is the XML-based successor of STEP. STEP is developed on the formal semantics of dynamic logic, and has been implemented in the distributed logic programming language DLP, a tool for the implementation of 3D web agents. In this paper, we discuss the issues of markup language design for embodied agents and several aspects of the implementation and application of XSTEP.
Keywords: embodied agents, avatars, humanoids, H-anim, STEP, XSTEP, XML

1 Introduction

Embodied agents are autonomous agents which have bodies by which the agents can perceive their world directly through sensors and act on the world directly through effectors. Embodied agents whose experienced worlds are located in real environments, are usually called *cognitive robots*. *Web agents* are embodied agents whose experienced worlds are the Web; typically, they act and collaborate in networked virtual environments. In addition, *3D web agents* are embodied agents whose 3D avatars can interact with each other or with users via Web browsers[6].

Embodied agents usually interact with users or each other via multimodal communicative acts, which can be verbal or non-verbal. Gestures, postures and facial expressions are typical non-verbal communicative acts. One of the main applications of embodied agents are virtual presenters, or alternatively called *presentation/conversation agents*. These agents are designed to represent users/agents in virtual environments, like virtual meeting spaces or virtual theaters, by means of hypermedia tools as part of the user interface.

These kinds of applications appeal for human markup languages for multimedia presentations. These markup languages should be able to accommodate the various aspects of human-computer interaction, including facial animation, body animation, speech, emotional representation, and multimedia. In [3], we outline the requirements for a software platform supporting embodied conversational agents. These requirements encompass computational concerns as well as presentation facilities, providing a suitably rich environment for applications deploying conversational agents.

The avatars of 3D web agents are typically built in the Virtual Reality Modeling Language

(VRML)[1]. These avatars are usually humanoid-like ones. The humanoid animation working group[2] proposes a specification, called H-anim specification, for the creation of libraries of reusable humanoids in Web-based applications as well as authoring tools that make it easy to create humanoids and animate them in various ways. H-anim specifies a standard way of representing humanoids in VRML. We have implemented STEP for H-anim based humanoids in the distributed logic programming language DLP[2].[3] DLP is a tool for the implementation of 3D intelligent agents [8, 9][4].

STEP introduces a Prolog-like syntax, which makes it compatible with most standard logic programming languages, whereas the formal semantics of STEP is based on dynamic logic [5]. Thus, STEP has a solid semantic foundation, in spite of a rich number of variants of the compositional operators and interaction facilities on worlds.[5]

In this paper, we propose an XML-based markup/scripting language for embodied agents, called XSTEP, based on the scripting technology STEP. Thus, XSTEP is the XML-based successor of STEP. In this paper, we discuss the issues of markup language design for embodied agents and several aspects of the implementation and application of XSTEP.

This paper is organized as follows: Section 2 discusses the general requirements on a markup language for embodied agents, and examines to what extent the scripting language STEP satisfies these requirements. Section 3 gives an overview of the language XSTEP and discusses a number of examples. Section 4 discusses the components of XSTEP and its current implementation. Section 5 compares XSTEP with other markup or scripting languages, discusses future work, and concludes the paper.

2 Design of Markup Languages for Embodied Agents

2.1 What XSTEP wants to have

We consider the following requirements for the design of the markup language for embodied agents.

Declarative specification of temporal aspects The specification of communicative acts, like gestures and facial expressions usually involve changes of geometrical data with time, like ROUTE statements in VRML, or movement equations, like those in computer graphics. A markup language for the presentation of embodied agents should be designed to have a solid temporal semantics. A good solution is to use existing temporal models, like those in temporal logics or dynamic logics. The scripting language STEP, and therefore, the markup language XSTEP, is based on the semantics of dynamic logics. Typical temporal operators in STEP are: the sequential action *seq* and the parallel action *par*, in XSTEP we have the corresponding tags <par> and <seq>.

Agent-orientation Markup languages for embodied agents should be different from those markup language for general multimedia presentation, like SMIL. The former have to consider the expressiveness and capabilities of their targeted agents. However, It's not our intention to

[1] http://www.vrml.org
[2] http://www.h-anim.org
[3] http://www.cs.vu.nl/∼eliens/projects/logic/index.html
[4] http://wasp.cs.vu.nl/wasp
[5] http://wasp.cs.vu.nl/step

design a markup language with fully-functional computation facilities, like other programming languages as Java, DLP or Prolog, which can be used to construct a fully-functional embodied agents. We separate external-oriented communicative acts from internal changes of the mental states of embodied agents because the former involves only geometrical changes of the body objects and the natural transition of the actions, whereas the latter involves more complicated computation and reasoning. The markup language is designed to be a simplified, user-friendly specification language for the presentation of embodied agents instead of for the construction of a fully functional embodied agent. A markup/scripting language should be interoperable with a fully powered agent implementation language, but offer a rather easy way for authoring. This kind of interaction can be achieved by the introduction of high-level interaction operators, like those in dynamic logic. Typical higher level interaction operators are the execution operator <do> and the conditional <if_then_else>.

Prototypability The presentation of embodied agents usually consists of some typical communicative acts, say, a presentation with greeting gesture. The specification of the greeting gesture can also be used for other presentations. Therefore, a markup language for embodied agents should have re-usability facilities. XML-based markup languages offer a convenient tool for the information exchange over the Web. Thus, an inline hyperlink in the markup language is an easy solution for this purpose. That would lead to the design of prototypability of markup languages, like the internal/external prototypes in VRML. The scripting language STEP is designed to be a rule-based specification system. Scripting actions are defined with their own names. These defined actions can be re-defined for other scripting purposes. XSTEP uses a similar strategy like STEP for prototypability. One of the advantages of this kind of rule-based specification is *parametrization*. Namely, actions can be specified in terms of how these actions cause changes over time to each individual *degree of freedom*, which is proposed by Perlin and Goldberg in [13]. Another method of parametrization is to introduce variables or parameters in the names of scripting actions, which allows for a similar action with different values. That is one of the reasons why STEP introduces Prolog-like syntax. Thus, XSTEP uses a similar method of parametrization like the one in STEP.

2.2 What XSTEP does not want to become

XSTEP is designed to be a markup language for the presentation of embodied agents and offers a lot of functionality on relevant topics, like those for 2D/3D avatars, multimedia, and agents. A lot of work has been done in these areas. Most of them are quite mature already. XSTEP does not want to overlap these existing work; such languages/specifications can be embedded into XSTEP in some degrees. Here are several examples:

- **2D/3D graphical markup languages** The specification of the scalable vector graphics (SVG)[6] is a typical XML-based language for describing two-dimensional graphics. SVG drawings can serve as XML-based 2D avatars for embodied agents. The X3D[7], the next generation of VRML, is a typical XML-based language for 3D object specification. X3D can be used as a tool for the design of XML-based 3D avatars. XSTEP code can be used to manipulate these avatars specified by SVG/X3D. Moreover, SVG/X3D code can also be

[6]http://www.w3.org/Graphics/SVG/
[7]http://www.web3d.org/x3d.html

embedded into XSTEPcode. Thus, it is not necessary for XSTEP to overlap the functionality of languages like SVG and X3D.

- **XML-based multimedia markup languages** The Synchronized Multimedia Integration Language (SMIL)[8] is an XML-based multimedia specification language, which integrates streaming audio and video with images, text or any other media type. Again, XSTEP does not want to replace or overlap the functionality of the language SMIL. The SMIL code can also embedded into XSTEP ones.

- **Humanoid markup languages** H-anim[9] is typically used to be a specification for humanoids based on VRML. The body references in H-anim are well suitable to be used as an ontology of the body parts for 3D embodied agents. Therefore, the body reference based on h-anim becomes a typical ontological specification in STEP and XSTEP. This will be discussed in detail in subsection 2.3. The X3D extension of H-anim can be considered as a typical XML-based Humanoid markup specification. The HumanMarkup specification (HumanML)[10] is another example to represent human characteristics through XML.

- **Agent specification languages** Embodied agents can be constructed in different ways. They can be directly built by programming languages like Java, Prolog, or DLP. Agent specification languages offer indirect ways to build embodied agents, in the sense that they are built with high-level abstraction. The Foundation for Intelligent Physical Agents (FIPA)[11] produces standards for heterogeneous and interacting agents and agent-based systems. XSTEP does not want to replace any existing work on agent specification languages. The existing version of XSTEP is able to interact with the internal states of embodies agents which are directly built via the high-level interact operators/tags. The approach to interact with embodied agents built by agent specification languages is one of the further work for XSTEP.

2.3 A Reference System for XSTEP

H-anim is a specification for 3D avatars based on VRML. It can serve as a point of departure for the design of a reference system in XSTEP. In this paper, we consider this typical reference for XSTEP.

2.3.1 Direction Reference

Based on the standard pose of the humanoid, we can define the direction reference system as sketched in figure 1. The direction reference system is based on these three dimensions: front vs. back which corresponds to the Z-axis, up vs. down which corresponds to the Y-axis, and left vs. right which corresponds to the X-axis. Based on these three dimensions, we can introduce a more natural-language-like direction reference scheme, say, turning left-arm to 'front-up', is to turn the left-arm such that the front-end of the arm will point to the up front direction.

[8] http://www.w3.org/AudioVideo/
[9] http://www.h-anim.org
[10] http://www.oasis-open.org/committees/humanmarkup/index.shtml
[11] http://www.fipa.org/

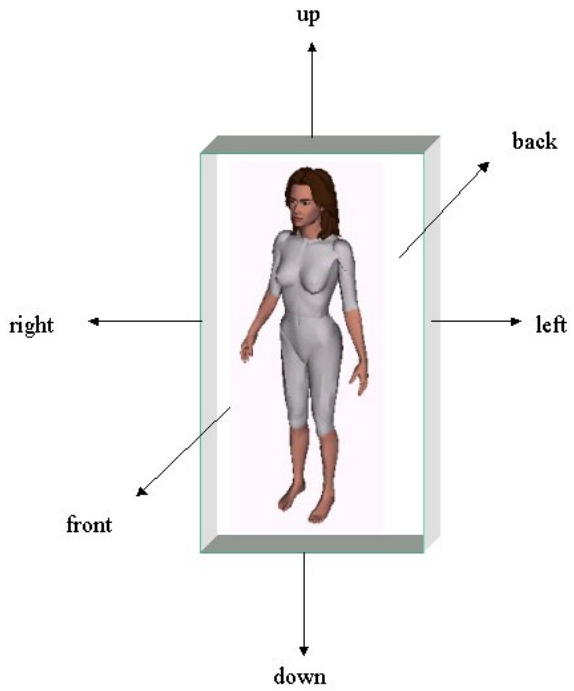

Figure 1: Direction Reference for Humanoid

Figure 2 shows several combinations of directions based on these three dimensions for the left-arm. The direction references for other body parts are similar. These combinations are designed for convenience for non-professional authors. However, they are not sufficient for more complex applications. To solve this kind of problem, we introduce interpolations with respect to the mentioned direction references. For instance, the direction 'left_front2' is referred to as one which is located between 'left_front' and 'left', which is shown in Figure 2. Natural-language-like references are convenient for authors to specify scripting actions, which does not require the author have a detailed knowledge of reference systems in VRML. Moreover, STEP and XSTEP also support the original VRML reference system, which is useful for experienced authors. Directions can also be specified to be a four-place tuple $\langle X, Y, Z, R\rangle$, say, $rotation(1, 0, 0, 1.57)$. In XSTEP, the directions are represented either the tag 'dir', like <dir value="front"/> or the tag 'rotation', like <rotation x="1" y="0" z="0" r="1.57"/>.

2.3.2 Body Reference

An H-Anim specification contains a set of *Joint nodes* that are arranged to form a hierarchy. Figure 3 shows several typical joints of humanoids. Therefore, turning body parts of humanoids implies the setting of the relevant joint's rotation. Body moving means the setting of the HumanoidRoot to a new position. For instance, the action 'turning the left-arm to the front slowly' is specified as:

```
<turn actor="Agent" part="l_shoulder">
      <dir value="front"/><speed value="slow"/>
</turn>
```

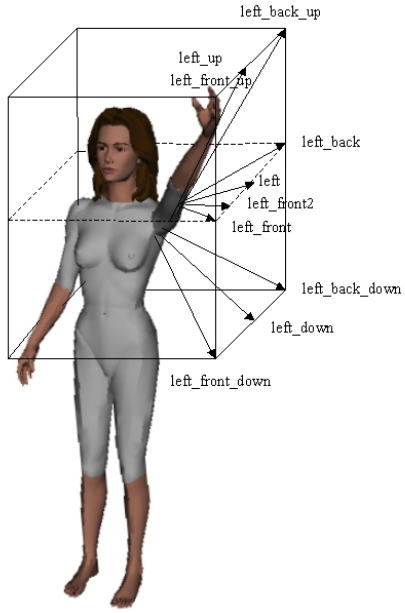

Figure 2: Combination of the Directions for Left Arm

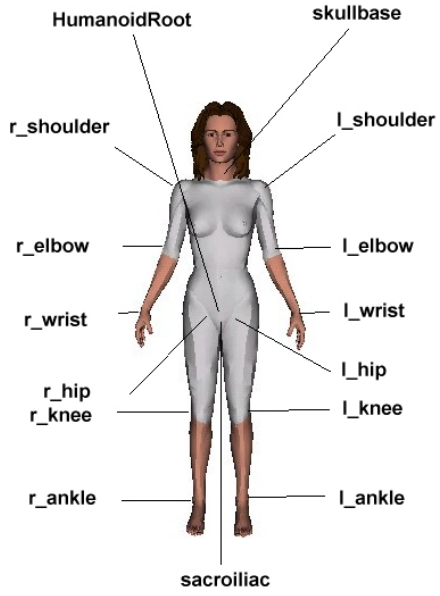

Figure 3: Typical Joints for Humanoid

2.3.3 Time Reference

The proposed scripting language has the same time reference system as in VRML. For example, the action *turning the left arm to the front in 2 seconds* can be specified in STEP as:

```
<turn actor="Agent" part="l_shoulder">
      <dir value="front"/><time unit="second" value="2"/>
</turn>
```

This kind of explicit specification of duration in scripting actions does not satisfy the parametrization principle. STEP introduces a more flexible time reference system based on the notions of beat and tempo. A *beat* is a time interval for body movements, whereas the *tempo* is the number of beats per minute. By default, the tempo is set to 60. Namely, a beat corresponds to a second by default. However, the tempo can be changed. Moreover, we can define different speeds for body movements, say, the speed 'fast' can be defined as one beat, whereas the speed 'slow' can be defined as three beats. In STEP, the set of this kind of the time reference are $\{fast, slow, intermedia, very\_fast, very\_slow\}$.

3 XSTEP: XML-encoded STEP

3.1 Actions Operators

Turn and move are two main primitive actions for body movements. Turn actions specify the change of the rotations of the body parts or the whole body over time, whereas move actions specify the change of the positions of the body parts or the whole body over time. For instance, a turn action in STEP like this,

```
turn(Agent, l_shoulder, front, fast)
```

is expressed in XSTEP as follows:

```
<turn actor="Agent" part="l_shoulder">
      <dir value="front"/><speed value="fast"/>
</turn>
```

A move action in STEP like this:

```
move(Agent, increment(1.0,0.0,0.0), fast)
```

is expressed in XSTEPas follows:

```
<move actor="Agent">
    <increment x="1.0" y="0.0" z="0.0"/><speed value="fast"/>
</move>
```

Similar with SMIL, XSTEP has the same temporal operators/tags: sequence action 'seq' and parallel action 'par'. Extended with the action operators in dynamic logics, XSTEP has the following operators:

- non-deterministic choice operator 'choice': the action <choice> $Action_1, ..., Action_n$ </choice> denotes a composite action in which one of the $Action_1$, ...,and $Action_n$ is executed.
- repeat operator 'repeat': the action <repeat action="Action" times="T"/> denotes a composite action in which the *Action* is repeated T times.

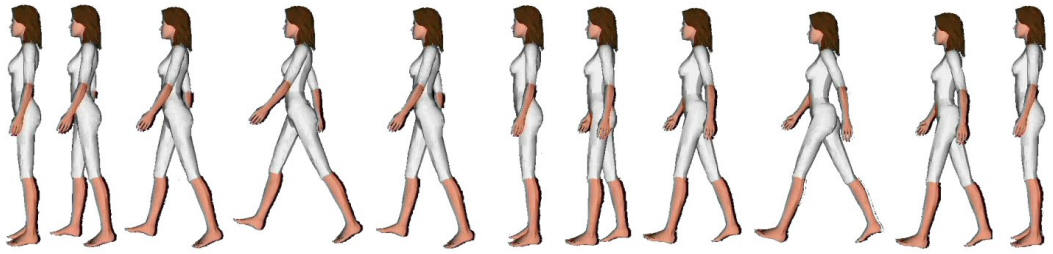

Figure 4: Walk

3.2 High-level Interaction Operators

When using high-level interaction operators, XSTEP can directly interact with internal states of embodies agents or with external states of worlds. These interaction operators are based on a meta language which is used to build embodied agents, say, the distributed logic programming language DLP. In the following, we use lower case Greek letters ϕ, ψ,χ to denote formulas in the meta language. Examples of several higher-level interaction operators:

- execution: <do state="ϕ"/>, make the state ϕ true, i.e. execute ϕ in the meta language.
- conditional: <if_then_else cond="ϕ" then="$action_1$" else="$action_2$"/>: if ϕ holds, then execute $action_1$ else execute $action_2$.

3.3 Example: Walk and its Variants

A walking posture can be simply expressed as a movement which changes the following two main poses: a pose in which the left-arm/right-leg move forward while the right-arm/left-leg move backward, and a pose in which the right-arm/left-leg move forward while the left-arm/right-leg move backward. The main pose and the interpolations are shown in Figure 4. The walk action can be described in XSTEP as follows:

```
<action name="walk(Agent)">
   <seq><par><turn actor="Agent" part="r_shoulder">
            <dir value="back_down2"/><speed value="fast"/></turn>
          <turn actor="Agent" part="r_hip">
            <dir value="front_down2"/><speed value="fast"/></turn>
          <turn actor="Agent" part="l_shoulder">
            <speed value="fast"/><dir value="front_down2"/></turn>
          <turn actor="Agent" part="l_hip">
            <dir value="back_down2"/><speed value="fast"/></turn></par>
        <par><turn actor="Agent" part="l_shoulder">
            <dir value="back_down2"/><speed value="fast"/></turn>
          <turn actor="Agent" part="l_hip">
            <dir value="front_down2"/><speed value="fast"/></turn>
          <turn actor="Agent" part="r_shoulder">
            <dir value="front_down2"/><speed value="fast"/></turn>
          <turn actor="Agent" part="r_hip">
```

```
        <dir value="back_down2"/><speed value="fast"/></turn>
    </par></seq></action>
```

Thus, a walk step can be described to be as a parallel action which consists of the walking posture and the moving action (i.e., changing position) as follows:

```
<action name="walk_forward_step(Agent)">
  <par><script_action name=walk_pose(Agent)/>
    <move actor=Agent part="humanoidRoot">
    <dir value="front"/><speed value="fast"/></move></par></action>
```

The step length can be a concrete value. For example, for the step length with 0.7 meter, it can be defined as follows:

```
<action name="walk_forward_step07(Agent)">
  <par><script_actionn name="walk_pose(Agent)"/>
    <move actor="Agent" part="humanoidRoot">
    <increment x=0 y=0 z=0.7/><speed value="fast"/></move></par></action>
```

Alternatively, the step length can also be a variable like:

```
<action name="walk_forward_step0(Agent,StepLength)">
  <par><script_action name="walk_pose(Agent)">
    <move actor="Agent" part="humanoidRoot">
    <increment x="0" y="0" z="StepLength"/><speed value="fast"/></move></par>
</action>
```

Therefore, the walking forward N steps with the *StepLegnth* can be defined in XSTEP as follows:

```
<action name="walk_forward(Agent,StepLength,N)">
  <repeat action="walk_forward_step0(Agent,StepLength)" times="N"/>
</action>
```

The animations of the walk based on those definitions are just simplified and approximated ones. As analysed in [4], a realistic animation of the walk motions of human figure involve a lot of the computations which rely on a robust simulator where forward and inverse kinematics are combined with automatic collision detection and response. We do not want to use XSTEP to achieve a fully realistic animation of the walk, because they are seldom necessary for most web applications. However, we would like to point out that there does exist the possibility to accommodate some inverse kinematics to improve the realism by using STEP. That is discussed in more detail in [11].

4 XSTEP: Components and Implementation

4.1 Components of XSTEP

A complete XSTEP code consists of these three components: library, head and embedded_code.

- **library**. XSTEP is mainly used to construct gesture and action libraries. The definitions of scripting actions are located in the XSTEP library component. Namely, they are located inside the tag $< library >$ with or without a name of the library. The scripting actions in the libraries are usually formatted as general rules with variables according to the prototypability requirements. They can be re-usable by the calling from other internal/external actions. So-called *internal actions* are located in the same XSTEP files, whereas the *external actions* are ones located in other files.

- **head**. The head component in XSTEP consists of the following elements:

  1. world: states the url of the virtual world/avatar, or whether avatar code is embedded, so that XSTEP can load the virtual world into the web browser;
  2. starting action: states an instantiated action so that XSTEP can start the action for the presentation;
  3. meta-language statement: states the meta-language for the high-level interact operators. DLP is the default meta-language.

4.2 Implementation of XSTEP

We have implemented the scripting language STEP in the distributed logic programming language DLP[10][12]. The scripting actions in STEP can be embedded in DLP code of embodied agents with the STEP kernel. These actions can be called by the interfacing predicates of the STEP kernel for the purpose of the presentation of embodied agents. A STEP testbed has been implemented. Users can use the STEP testbed in web browsers to construct their own STEP scripting actions and test them online without knowledge of DLP and VRML.

We have also implemented an XSTEP editor based on IBM's XML editor Xeena[13]. This XSTEP editor will help the author to edit XSTEP code and translate it to STEP scripts so that they can be run from the STEP tool, testbed, or DLP code. A screenshot of the XSTEP editor on Xeena is shown in Figure 5. We are now working on the development of the tool so that STEP and XSTEP code can be run as a stand-alone file.

5 Conclusions

In this paper we have proposed the markup language XSTEP for embodied agents. Moreover, we have discussed the requirements on markup language design for embodied agents and several aspects of the implementation and application of the markup language XSTEP. In the following, we would like to make a comparison on XSTEP with other XML-based markup languages for humanoids, and discuss future work.

5.1 Comparison

XSTEP can be considered to be one of VHML (Virtual Human Markup Language)-like languages.[14] The language VHML is designed to accommodate the various aspects of human-

[12] http://wasp.cs.vu.nl/step
[13] http://www.alphaworks.ibm.com/tech/xeena
[14] http://www.vhml.org

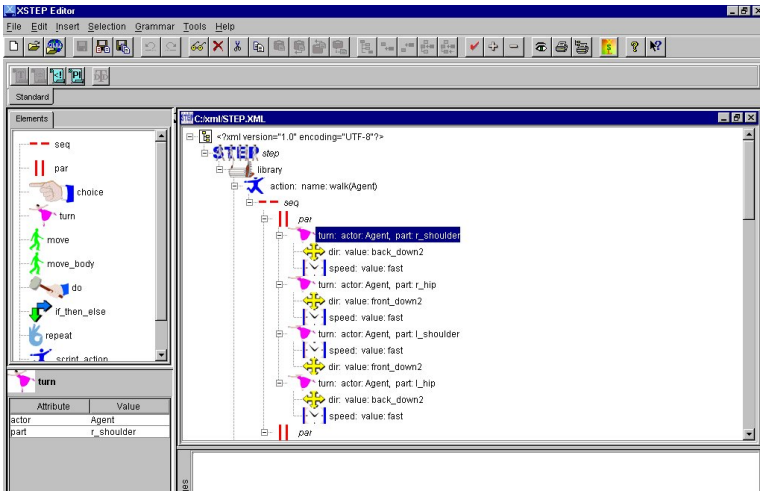

Figure 5: Screenhot of XSTEP editor on Xeena

computer interaction, including facial animation, body animation, speech, emotional representation, and multimedia. XSTEP and VHML share a lot of the common goals.

One of the differences between XSTEP and the VHML is: XSTEP is developed based on the formal semantics of dynamic logic, so that it has a solid semantic foundation, in spite of the rich variants of compositional operators and interaction facilities on worlds. Secondly, Prolog-like parametrization in XSTEP makes it more suitable for the interaction with intelligent embodied agents.

An interesting examples for the animated humanoid avatars is provided by *Signing Avatar*.[15] The scripting language for *Signing Avatar* is based on the H-anim specification and allows for a precise definition of a complex repertoire of gestures, as examplified by the sign language for the deaf. Nevertheless, this scripting language is of a proprietary nature and does not allow for high-order abstractions of semantically meaningful behavior.

More detailed comparisons and related work discussions with respect to STEP and XSTEP can be found in the papers [3, 10].

5.2 Further work

- **ontology of human markup languages**. More human markup languages are expected to be proposed in coming years. These languages may use completely different terminology and semantics models. A good solution to maintenance of the interoperability among multiple reference systems is to make XSTEP ontological-relevant. A so-called *Ontology* is a description of the concepts or bodies of knowledge understood by a particular community and the relationships between those concepts. An ontological investigation for human markup language is needed so that the presentations and their libraries can be interoperable [12].

- **facial expression and emotion models in XSTEP**. We are going to extend XSTEP with facial expressions. These facial expression can be marked as the tags 'anger', 'happy' and 'sad', like those are suggested in VHML. The terminology can be formalized based on

[15] http://www.signingavatar.com

emotion models and further specified by the corresponding ontological claim which is based on the survey in [12].

- **speech and other multimedia modes in XSTEP**. We are also planning to extend XSTEP with speech/voice and other multimedia modes, so that we can enrich embodied agents with the functionality needed to create convincing embodied agents in a meaningful context.

References

[1] Earnshaw, R., Magnenat-Thalmann, N., Terzopoulos, D., and Thalmann, D., Computer Animation for Virtual Humans, *IEEE Computer Graphics and Applications* 18(5), 1998.

[2] Eliëns, A., *DLP, A Language for Distributed Logic Programming*, Wiley, 1992.

[3] Eliëns, A., Huang, Z., and Visser, C., A platform for Embodied Conversational Agents based on Distributed Logic Programming, Proceedings of AAMAS 2002 WORKSHOP: Embodied conversational agents - let's specify and evaluate them, 2002.

[4] Faure, F., Debunne, G., Cani-Gascuel, M., Multon, F., Dynamic analysis of human walking, Proceedings of the 8th Eurographics Workshop on Computer Animation and Simulation, Budapest, September 1997.

[5] Harel, D., Dynamic Logic, *Handbook of Philosophical Logic*, Vol. II, D. Reidel Publishing Company, 1984, 497-604.

[6] Huang, Z., Eliëns, A., van Ballegooij, A., and de Bra, P., A Taxonomy of Web Agents, *Proceedings of the 11th International Workshop on Database and Expert Systems Applications*, IEEE Computer Society, 765–769, 2000.

[7] Huang, Z., Eliëns, A., and de Bra, P., An Architecture for Web Agents, *Proceedings of the Conference EUROMEDIA'2001*, SCS, 2001.

[8] Huang, Z., Eliëns, A., and Visser, C., Programmability of Intelligent Agent Avatars, *Proceedings of the Autonomous Agents'01 Workshop on Embodied Agents*, 2001.

[9] Huang, Z., Eliëns, A., and Visser, C., *3D Agent-based Virtual Communities*, *Proceedings of the 2002 Web 3D Conference*, ACM Press, 2002.

[10] Huang, Z., Eliëns, A., and Visser, C., STEP: a Scripting Language for Embodied Agents, Proceedings of the Workshop of Lifelike Animated Agents, Tokyo, 2002.

[11] Huang, Z., Eliëns, A., and Visser, C., STEP: a Scripting Language for Embodied Agents (full version), WASP Research Report, Vrije University Amsterdam, 2002. `http://wasp.cs.vu.nl/step/paper/script.pdf`.

[12] Huang, Z., Eliëns, A., and Visser, C., An ontological investigation on human markup languages, in preparation, 2002.

[13] Perlin, K., and Goldberg, A., Improv: A System for Scripting Intereactive Actors in Virtual Worlds, *ACM Computer Graphics*, Annual Conference Series, 205-216, 1996.