XSTEP: A Markup Language for Embodied Agents

Zhisheng Huang, Anton Eliëns, and Cees Visser
Vrije Universiteit 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. XSTEP is the XML-based successor of STEP. The scripting language STEP incorporates 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.

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. *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 or other agents in virtual environments, like virtual meeting spaces or virtual theaters, by means of hypermedia tools as part of the user interface[10].

These kind 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 often built in the Virtual Reality Modeling Language (VRML)[1]. These avatars are usually humanoid-like ones. The humanoid animation working group[2] proposes the 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[7].[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.[5]

2 XSTEP design considerations

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 in 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 logic or dynamic logic. The scripting language STEP, and therefore the markup language XSTEP, is based on the

[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

semantics of dynamic logic. Typical temporal operators in STEP are the sequential action seq and the parallel action par, in XSTEP we have the corresponding <par> and <seq> tags.

Agent-orientation Markup languages for embodied agents should be different from markup languages 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 design a markup language with fully-functional computation facilities, like other programming languages as Java, DLP or Prolog, which can be used to construct 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 computations and reasoning. The markup language is designed to be a simplified, user-friendly specification language for the presentation of embodied agents instead of 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 a 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 information exchange over the Web; 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 identified by their own names and can be re-defined for other scripting purposes. XSTEP uses a similar strategy as 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 [9]. 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. This is one of the reasons why STEP introduces a Prolog-like syntax.

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 and most of them are already quite mature. XSTEP does not want to overlap with the existing work. Such languages or specifications can often be embedded in XSTEP. Here are several examples: *2D/3D graphical markup languages*, like the scalable vector graphics (SVG) [6] specification for two-dimensional graphics, and X3D[7], the next generation of VRML, an XML-based language for 3D object specification; *XML-based multimedia markup languages*, like the Synchronized Multimedia Integration Language (SMIL)[8]; *Humanoid markup languages*, like the H-anim specification for humanoids, and the HumanMarkup specification (HumanML)[9]; *Agent specification languages*, like the Intelligent Physical Agents Specification (FIPA)[10].

3 XSTEP: XML-encoded STEP

XSTEP uses the same reference system as STEP. In order to make this paper self-contained, we also present the necessary details of the reference system of XSTEP here.

3.1 A Reference System for STEP and XSTEP

3.1.1 Direction Reference

Based on the standard pose of an humanoid, we can define the direction reference system as sketched in Figure 1. The direction reference system is based on 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. 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 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

[6]http://www.w3.org/Graphics/SVG/
[7]http://www.web3d.org/x3d.html
[8]http://www.w3.org/AudioVideo/
[9]http://www.oasis-open.org/committees/humanmarkup/ index.shtml
[10]http://www.fipa.org/

'left_front' and 'left', as is shown in Figure 2. Both STEP and XSTEP also support the original VRML reference system. Directions can also be specified to be a four-place tuple $\langle X, Y, Z, R \rangle$, for example, $rotation(1, 0, 0, 1.57)$. In XSTEP, the directions are represented either by the tag 'dir', like <dir value="front"/> or the tag 'rotation', like <rotation x="1" y="0" z="0" r="1.57"/>.

3.1.2 Body References

An H-anim specification contains a set of *Joint nodes*, arranged in such a way that they form a hierarchy. Turning body parts of humanoids implies the setting of the relevant joint's rotation. Moving the body 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>
```

3.1.3 Time Reference

The proposed scripting language has the same time reference system as 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, for example the speed 'fast' can be defined as one beat, whereas the speed 'slow' can be defined as three beats. The predefined speed specifications in XSTEP are $\{fast, slow, intermediate, very\_fast, very\_slow\}$.

3.2 Actions Operators

Turn and move are two main primitive actions for body movements. Turn actions specify the change of rotations of body parts or the whole body over time, whereas move actions specify the change of positions of body parts or the whole body over time. For instance, a turn action in XSTEP is expressed as follows:

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

A move action with increment parameters in XSTEP is expressed like this:

```
<move actor="Agent"><increment x="1" y="0" z="0"/>
<speed value="fast"/></move>
```

XSTEP has the same temporal operators/tags as SMIL: sequence action 'seq' and parallel action 'par'. Extended with the action operators as defined in dynamic logic, XSTEP has the following operators:
• **non-deterministic choice operator**: the action <choice> $Action_1, ..., Action_n$ </choice> denotes a composite action in which one of the $Action_1$, ..., or $Action_n$ is executed non-deterministically.
• **repeat operator**: the action <repeat action= "Action" times="T"/> denotes a composite action in which the $Action$ is repeated T times. When using high-level interaction operators, XSTEP can directly interact with internal states of embodied 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.

Examples of several higher-level interaction operators in the XSTEP scripting language are:
• **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 $action_2$. We use lower case Greek letters ϕ, ψ, χ to denote formulas in the meta language.

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 interpolations are shown in Figure 3. The walk action can be described in XSTEP as:

```
<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 as a parallel action which consists of the walking posture and the moving action (i.e., changing position):

```
<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 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, walking forward N steps with a particular $StepLength$ can be defined in XSTEP as:

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

The animations of the walk action based on these definitions are simplified and approximated ones. As analysed in [4], a realistic animation of walk motions of humans involves a lot of computations which rely on a robust simulator where forward and inverse kinematics are combined with automatic collision detection and response. We would like to point out that there does exist the possibility to accommodate some inverse kinematics to improve the realism by using the execution operator 'do' to achieve the computation capabilities in the meta language level. The same approach can be also be used for the interaction with virtual worlds and the communication with other embodied agents. In the next section we will discuss several aspects of the interaction between embodied agents and virtual worlds.

3.4 Example: Interaction with Virtual Worlds

In this example, we want to show how the interaction between embodied agents and virtual worlds can be achieved by using the high-level interaction operators. Consider a situation in which there are several avatars and a ball. The position of the ball is always changing because other avatars may kick the ball. We want to design script actions for embodied agents so that they can always look at the ball or run to the ball no matter where the ball is located.

In the following, we suppose that the meta language of the scripts is DLP. Other languages can be processed by the same strategy. Given the current positions of the embodied agent and the ball, we can always calculate the new rotation of the agent so that it looks at the ball. By using the high level interaction operator '*do*' with the built-in operators in the meta language we can define the script action 'look at ball' as follows:

```
<action name="look_at_ball(Agent,Ball)">
  <seq><do state="getPosition(Agent,X,_,Z)"/>
    <do state="getPosition(Ball, X1,_,Z1)"/>
    <do state="Xdif is X-X1"/>
    <do state="Zdif is Z1-Z"/>
    <do state="R is atan(Zdif/Xdif)-
sign(Xdif)*1.57"/>
    <turn actor="Agent">
      <rotation x="0.0" y="1.0" z="0.0" r="R"/>
<speed value="fast"/></turn></seq></action>
```

Namely, we use the predicates $getPosition$ in the meta language to obtain the positions of the agent and the ball. The new rotation R of the agent can be calculated based on the following formula :

$$R = arctan((Z1-Z)/(X-X1)) - sign(X-X1) \times \pi/2.$$

Using the high-level interaction operators, embodied agents can get and set not only the information of their virtual worlds, but also can change their internal states. They can be used to serve as a communication facility between embodied agents and virtual worlds. Prolog-style parametrization makes XSTEP scripts more flexible for the creation of variants of scripting actions. The support of recursive definitions of scripting actions makes the language more powerful and elegant for programming.

4 XSTEP: Components and Implementation

4.1 Components of XSTEP

Complete XSTEP scripts consist of three components: head, library and embedded_code:
• **head**: the head component in XSTEP consists of the elements *World*, *Starting Action*, and *Meta language statement*. The *World* specifies the URL of the virtual world/avatar and whether avatar code is embedded, so that XSTEP can load the virtual world into the web browser. A *Starting action* states an instantiated action so that XSTEP can start the action for the presentation. The *Meta language statement* defines the meta language for the high-level interaction operators. DLP is the default meta language.
• **library**: XSTEP is mainly used to construct gesture and

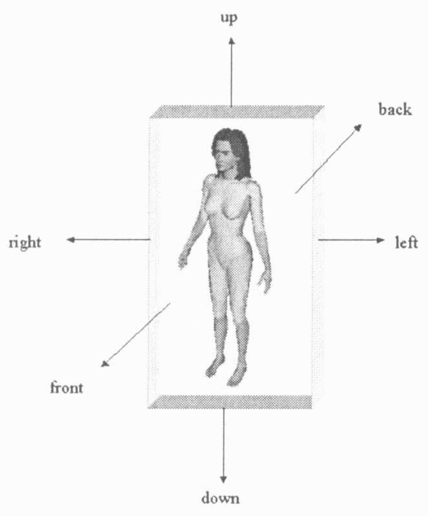

Figure 1. Direction Reference for Humanoid

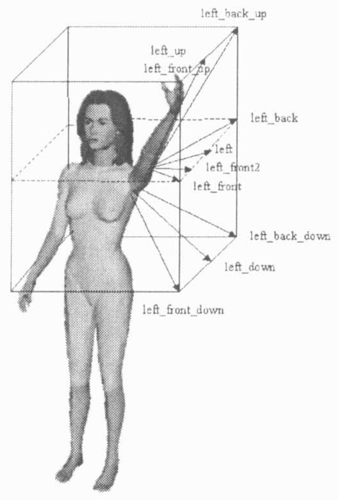

Figure 2. Combination of the Directions for Left Arm

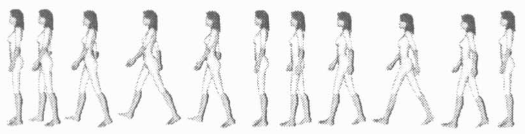

Figure 3. Walk

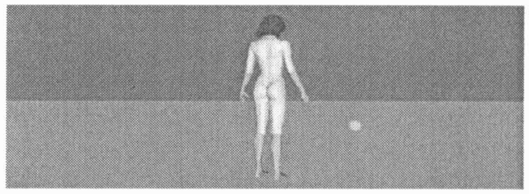

Figure 4. Look at Ball

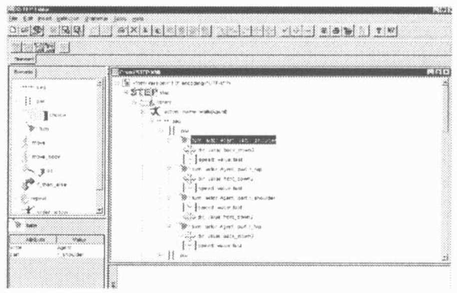

Figure 5. Screenshot of XSTEP editor

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-used by calling them from other internal or external actions. So-called *internal actions* are located in the same XSTEP files, whereas the *external actions* are located in other files.

• **embedded_code**: This component contains additional XML-based code, for example an X3D or SVG script which specifies embedded avatars.

4.2 Implementation of XSTEP

We have implemented the scripting language STEP [8][11] in the distributed logic programming language DLP. 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[12]. 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,

[11] http://wasp.cs.vu.nl/step

[12] http://www.alphaworks.ibm.com/tech/xeena

or DLP code. A screenshot of the XSTEP editor based on Xeena is shown in Figure 5. We are currently working on the development of a tool so that STEP and XSTEP code can be executed as a stand-alone file.

5 Discussion and Conclusion

XSTEP can be considered to be one of the VHML (Virtual Human Markup Language)-like languages.[13] The language VHML is designed to accommodate the various aspects of human-computer interaction, including facial animation, body animation, speech, emotional representation, and multimedia. XSTEP and VHML share many common goals, however, one of the differences between XSTEP and the VHML is that XSTEP is 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. Secondly, Prolog-like parametrization in XSTEP makes it more suitable for the interaction with intelligent embodied agents. Another advantage of parametrization is to introduce variables or parameters in the names of scripting actions, which allows for a similar action with different values. In particular, agent names and their relevant parameters are specified as variables in script libraries, by which the same scripting actions can be re-used for different embodied agents under different situations by different authors. It significantly improves the reusability of scripting actions for the purpose of productivity.

An interesting example of animated humanoid avatars is provided by *Signing Avatar*.[14] 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 [3, 8].

Future STEP and XSTEP work will include:
• **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 semantic models. A good solution to the maintenance of 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 these concepts. An ontological investigation for human markup language is needed so that the presentations and their libraries can be interoperable.
• **Facial expression and emotion models in XSTEP**. We are going to extend XSTEP with facial expressions. These facial expressions can be marked with tags like 'anger', 'happy' and 'sad', as suggested in VHML. The terminology can be formalized based on emotion models and further specified by the corresponding ontological claim.
• **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.

[13] http://www.vhml.org
[14] http://www.signingavatar.com

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 on Embodied Agents, 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, 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 Workshop on Database and Expert Systems Applications*, IEEE Computer Society, 765–769, 2000.

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

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

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

[10] Prendinger,H., Descamps, S., and Ishizuka, M., Scripting affective communication with life-like characters in web-based interaction systems, *Journal of Applied Artificial Intelligence* 16, 519-553, 2002.