ISSN 2409-7284 (Online)

A Robot Motion Control Mechanism using Robot Drama Markup Language

Yung-Hui Li
Department of Computer Science and Information Engineering
National Central University, Taoyuan City, Taiwan
E-mail: yunghui@csie.ncu.edu.tw

*Jim-Min Lin
*Department of Information Engineering and Computer Science
Feng Chia University, Taichung City, Taiwan
*E-mail: jimmy@fcu.edu.tw

Kwang-Yao Lee
Department of Information Engineering and Computer Science
Feng Chia University, Taichung City, Taiwan
E-mail: m9601509@fcu.edu.tw

Che-Wun Chiou
Department of Computer Science & Information Engineering
Chien Hsin University of Science and Technology, Taiwan
E-mail: cwchiou@uch.edu.tw

Received January 2018; Revised April 2018

ABSTRACT. *Recently, there are several markup-languages proposed for rapidly defining the action, expression and conversation of Embodied Conversational Agent (ECA). However, it is difficulty for general users to understand and define markup language unless they are professional programmers or animators. Moreover, general users would also lack of drama flow management capability. Our previous researches have proposed an Interactive Drama Markup Language (IDML) for software agent presentation. Users can produce their own rich and interactive dramas for software agent by using IDML, but there are some limitations in software agent presentation control, for example there are only 2D action control supported. Therefore, in this paper, a humanoid robot is treated as a 3D software agent and integrated with interactive drama to extend the user-agent interaction to a 3D-interaction and thus IDML is then extended to a Robot Drama Markup Language (RDML). Robot control is another adversity for general users. In this study, a middleware proposed to let users conveniently produce rich interactive dramas for robot by editing RDML. Additionally, RDML can also enhance the robot control to get more interests in robot-audience interaction and ease the*

ISSN 2409-7284 (Online)

audience to directly communicate with a drama.
Keywords: *Interactive Drama, Robot Drama Markup Language (RDML), Robot Control Middleware*

1. Introduction

In the field of human-computer interface, more and more studies focus on the effectiveness of the conversation and user's acceptability. Embodied Conversational Agent (ECA) [1] is one of the important topics in the field. ECA is a kind of human-computer interaction mechanism, by simulating human's interaction such as speech, gestures, facial expressions and emotions, it interacts with humans much more realistically and effectively.

In recent researches of ECA, some proposed methods using markup languages such as VHML[2], CML[3] and AML[4] to express ECA animations. These markup languages allow animators to express ECA's animation using xml, then interpret the xml to manipulate animation engines.

Although it is easy for professionals like animators or programmers to use the method proposed in [5] to set up animation parameters and transfer them to different ECA, it is still too complicated for general people like sales person.

Thus, markup languages like MPML[6] and TelMeA[7] were proposed to be a simpler way to manipulate ECA's animation. From the abovementioned research, we can conclude that these researches mainly focus on how to provide a high-level representation that allows users to set up ECA's animation without concerning underlying mechanisms [8].

However, [9] mentioned that people prefer to interact with ECA using natural languages rather than buttons and dialogs. This emphasize the fact that the core value of ECA is not just performing animations, but to create a realistic experience that make you feel like you are talking to real people.

Yet, it is difficult for general users to develop an elaborate ECA which interacts with people according to different situations using artificial intelligence. Therefore, a suitable markup language for general users to describe ECAs animation should also be able to describe desired

ISSN 2409-7284 (Online)

interaction scenarios easily.

An XML based IDML (Interactive Drama Markup Language) was proposed in previous studies [10,11]to help general users to build ECA interaction interface without any knowledge about programming language. IDML follows current principles of ECA animation design, it supports high-level ECA Agent behavior description and contain tags to speed up GUI developing. It also combines the concept of interactive drama, which allow users to describe interactive event by editing tags, and the response of ECA varies based on their interaction with audiences. IDML has been successfully used in ECA applications [12,13].

In recent years, bionic robot is gradually getting into our lives. Generally, real objects tend to impress and attract more reaction from human than virtual objects in screen does. For example, a robotic arm waving in real world attracts much more attention than a virtual robotic arm waving in screen does.

In this paper, a humanoid robot is treated as a 3D software agent and integrated with interactive drama to extend the user-agent interaction into a 3D-interaction. An IDML based Robot Control mechanism, namely Robot Drama Markup Language (RDML), is proposed in this paper. For editors, they can set up a robot to perform as their will in a short time without much knowledge of underlying hardware manipulation. Furthermore, they can attract more attention from audiences and make the interaction more effective using real robot in physical world.

2. Interactive Drama Markup Language (IDML)

IDML was designed with the concept of interactive drama which contains three different layers. Figure 1 demonstrates the structure of interactive drama. An interactive drama can be divided into many acts, each act represents a single stage scene and every act is completed by many characters. Therefore, IDML has three layers in its structure with different kind of blocks in each layer. Blocks can be divided as Drama Block (DB), Act Block (AB), and Scene Block (SB) which form a unified rule to represent how ECA should interact with users.

Volume 5, Number 2, 2018
ISSN 2409-7284 (Online)

Table 1 listed all the properties of different blocks, Drama Block(DB) is the root tag of the whole script, which contains all the Act Block that would appear, and indicates how ABs links each other.

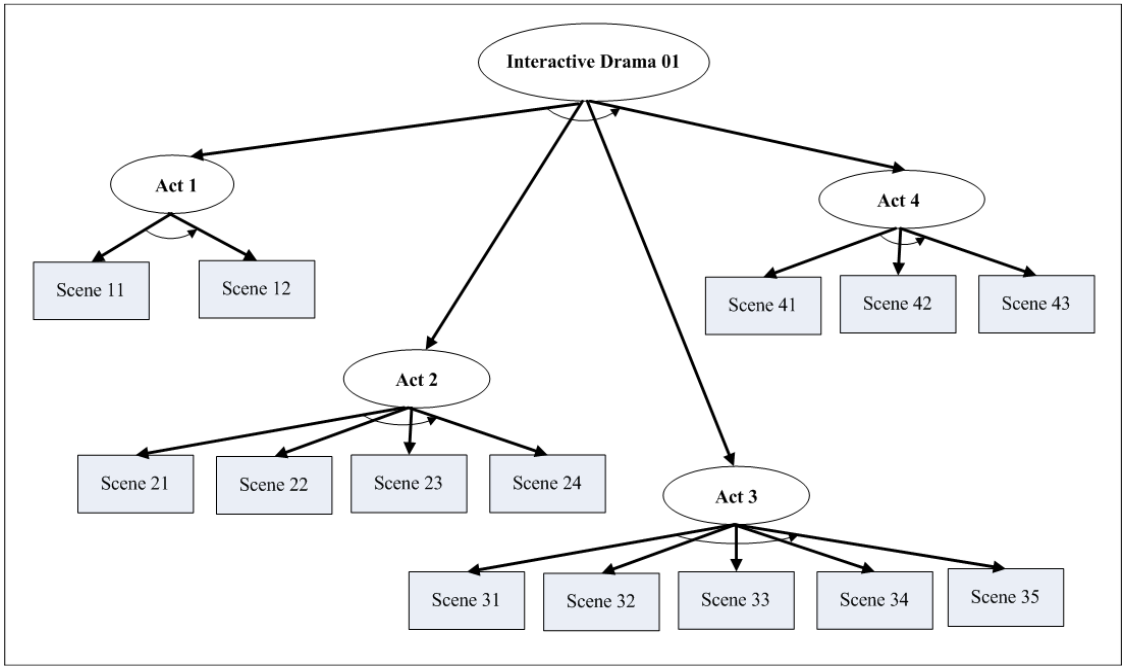

FIGURE 1. IDML three layer architecture (Depicted from [10])

TABLE 1. IDML property table

Block Name	Properties
Drama Block, (DB)	1. Define ECA-audience interaction structure based on different scenario 2. Generate buttons on GUI that links to related Act Block, to develop the whole interaction network with different scenarios.
Act Block, (AB)	1. Describe a single branch plot, namely a sub target. 2. Arrange a series of ECA's basic movement to finish a sub task. 3. Combine all kinds of Scene Blocks to generate the changes during performance.
Scene Block, (SB)	1. Scene Block is the smallest component, used to describe an action of a single character. 2. Controls ECA's behavior, including speaking, emotion and reference 3. Actions described in Scene Block is supported by ECA engine.

3. Robot Drama Markup Language (RDML)

RDML is designed by extending the concept of IDML with robot interactions. RDML is also formed with a three-layer architecture. Figure 3A demonstrates the structure of robot interactive drama. Similar to IDML, a robot interactive drama can be subdivided into many acts, then each act represents a single stage scene and every act is completed by a robot. Therefore, RDML has also three layers, namely Robot Drama Block (RDB), Robot Act Block (BAB), and Robot Scene Block (RSB) respectively, in its structure with different kind of represented blocks in each layer.

RABs formed the second layer, each RAB indicates its own RSB and manage time interval between RSBs.

BSB is the smallest component of the whole script, it is mainly used to indicate perform information like speech/voice, motions, touch sensing, and any other information that performing would need. The following are definitions of RDB, RAB and RSB:

Robot Drama Block(RDB)

Following XML rules, Figure 2 is RDB's Document Type Definition (DTD), the meaning of each tag is as follows:

- RDB: RDB block's root node, with a property named RDB_id indicating the id of this RDB.
- RDB-description: RDB's child element, a brief introduction to this drama from the author, helps viewer to understand the outline of this drama.
- Start_RAB: RDB's child element, the first RAB that would be used upon the script starts.
- Sensor_set: RDB's child element, a group of Sensor tag with a property RAB-id used to indicate which RAB does this Sensor_set belongs to.
- Sensor: child element of Sensor_set, triggering of each Sensor represents a possible divergence in drama. Currently available sensor devices are Infrared (IR) and microphone.
- Text: Sensor's child element, used to indicate the text-to-speech voice on triggering a sensor on robot.

Volume 5, Number 2, 2018
ISSN 2409-7284 (Online)

- Trigger_RAB: Touch's child element, used to indicate the RAB that would be triggered when Touch is being pressed.

Figure 3 is the code of RDB01 that shows an example to greet audiences through Robot Act Block 1, RAB1, and then wait for the object detection from IR sensors, Sensor_IR1 and Sensor_IR2. The corresponding Robot Act Blocks, namely RAB2 and RAB3, will be triggered, respectively then.

```
<!ELEMENT RDB (RDB_description, Start_RAB, Button_set+)>
<!ATTLIST DB
RDB_id CDATA    #REQUIRED>
<!ELEMENT RDB_description (#PCDATA)>
<!ELEMENT Start_RAB (#PCDATA)>
<!ELEMENT Sensor_set (Sensor+)>
<!ATTLIST Sensor_set
RAB_id    CDATA    #REQUIRED>
<!ELEMENT Sensor (Text, Trigger_RAB)>
<!ELEMENT Text (#PCDATA)>
<!EL <!EMENT Trigger_RAB (#PCDATA)>
```

FIGURE 2. Robot Drama Block DTD

```
<RDB RDB_id="RDB01">
<RDB_description>This Robot Drama Block greet audiences through
a robot.</RDB_description>
      <Start_RAB>RAB1</Start_RAB>
      <Sensor_set RAB_id="RAB1">
            <Sensor_IR1>
            <TTS>I am fine.</TTS>
            <Trigger_RAB>RAB2 </Trigger_RAB>
            </Sensor_IR1>
            <Sensor_IR2>
            <TTS>Not Good.</TTS>
            <Trigger_RAB>RAB3</Trigger_RAB>
            </Sensor_IR2>
      </Sensor_set>
            ……..
```

ISSN 2409-7284 (Online)

```
</RDB>
```

FIGURE 3. Robot Drama Block example – RDB01

Robot Act Block(RAB)

Figure 4 is RAB's DTD, the meaning of each tag is as follows:

- RAB: DB's child element and RAB's root element, with a property that indicates the id of this RAB.
- Background: RAB's child element, indicates the name and file name/URL of the background music used in this scene.
- Trigger_RSB: RAB's child element, indicates the RSBs this RAB includes, the order of the RSBs is the performance sequence of RSBs.

Figure 5 is the code of RAB1, showing an example of how a robot to greet audiences with a salute by playing a background music - greeting.mp3 and, at the same time, do a salute motion by triggering a sequence of Robot Scene Blocks, RSB11 and then RSB12.

```
<!ELEMENT RAB (RAB_description, Location?,Trigger_RSB+)>
<!ATTLIST RAB
    RAB_id    CDATA    #REQUIRED>
<!ELEMENT RAB_description (#PCDATA)>
<!ELEMENT Location (#PCDATA)>
<!ELEMENT Trigger_RSB (#PCDATA)>
```

FIGURE 4. Robot Act Block DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE proverb SYSTEM "RAB.dtd">
<RAB RAB_id="RAB1">
    <RAB_description>This Robot Act Block directs a robot to greet
audiences with a salute. </RAB_description>
    < Background >greeting.mp3</Background>
    <Trigger_RSB>RSB11</Trigger_RSB>
    <Trigger_RSB>RSB12</Trigger_RSB>
</RAB>
```

FIGURE 5. Robot Act Block example – RAB1

Volume 5, Number 2, 2018
ISSN 2409-7284 (Online)

Robot Scene Block(RSB)

Figure 6 is RSB's DTD, the meaning of each tag is as follows:

- RSB: RDB's child element and RSB's root node, include a property RSB_id to indicate the id of this RSB.
- RSB_description: RSB's child element, comments of this RSB.
- Actor: RSB's child element, indicate which robot performs this RSB.
- Behaviors: RSB's child element, a group of tags that defines what actions an Actor would perform.
- Motion: Behaviors' child element, indicate a single movement that Actor would perform.
- Speech: Behaviors' child element, indicate the Text-To-Speech (TTS) that Actor would play.
- Reference: Behaviors' child element, indicate the sound effect or music file that would be played from an externally speaker when a robot is performing.
- Stop: Behaviors' child element, indicate how long (in seconds) the Actor stop after it finishes its perform.

Figure 7 is the code of RSB11 showing an example of how a robot, Bioloid, to shake its left hand by specifying Actor's Behaviors, like Motion, Speech, sound effects from an external speaker, and the duration time.

```
<!ELEMENT RSB (RSB_description, Actor, Behaviors+)>
<!ATTLIST RSB
   RSB_id   CDATA   #REQUIRED>
<!ELEMENT Actor (#PCDATA)>
<!ELEMENT Behaviors (Motion?, Speech?, Reference?, Stop?)>
<!ELEMENT Motion (#PCDATA)>
<!ELEMENT Speech (#PCDATA)>
<!ELEMENT Reference (#PCDATA)>
<!ELEMENT Stop (#PCDATA)>
```

FIGURE 6.Robot Scene Block DTD

```
<?xml version="1.0" encoding="UTF-8"?>
```

Volume 5, Number 2, 2018
ISSN 2409-7284 (Online)

```
<!DOCTYPE proverb SYSTEM "RSB.dtd">
<RSB RSB_id="RSB11">
        <RSB_description>The Actor does a salute. </RSB_description>
        <Actor>Bioloid</Actor>
       <Behaviors>
            <motion>Shake Left Hand </motion>
            <Speech>How are you?</Speech>
            <Stop>5s</Stop>
       </Behaviors>
</RSB>
```

FIGURE 7. Robot Scene Block example

Figure 8 is a flow chart of RDML's performance. Upon a performance starts, default Robot Act Block 1 would be executed. Robot Scene Block 1 in Robot Act Block 1 is performed first according to Robot Act Block 1's setting, then Robot Scene Block 2 is performed. After the performance finished, next Robot Act Block is chosen according to the reaction of audiences during performance. If the right one was chosen, then the next RAB that would be performed is Robot Act Block 3, and so on. Using this kind of mechanism, it allows audiences to choose different performance branches as their wish.

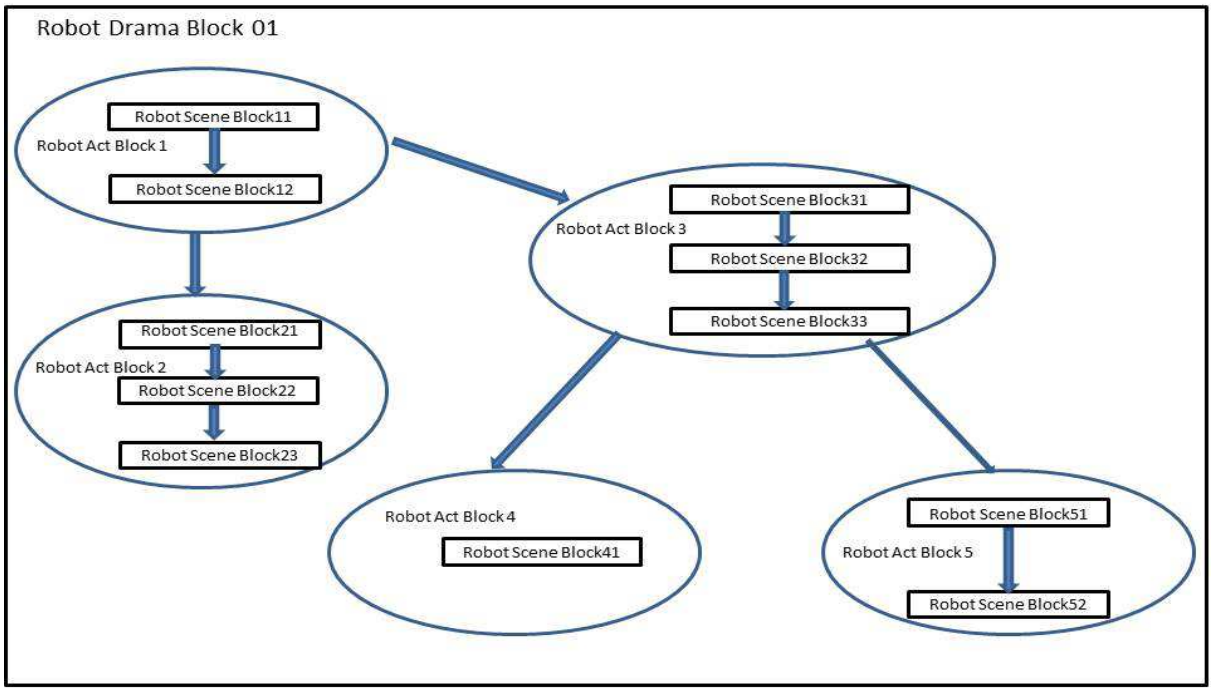

FIGURE 8. RDML working flow example

ISSN 2409-7284 (Online)

3.1 Robot control middleware

The main purpose of a middleware is to connect software components, applications, and hardware components together and help them to interact with each other. A middleware could provide better extensibility to the system and unify the interface between hardware and software. Following the guidance in [8], a robot middleware should follow design rules as below:

- Object-oriented design:

  Object-oriented design provides a way to hide unnecessary information and avoid namespace confusion. The features of polymorphism, abstract classes, and inheritance also speed up the applications development.

- Open architecture:

  There are already tens of thousands of manufacturers developing robot-related hardware and software, each with its own specification. Using open-architecture allows manufacturers to merge their product into current architecture quickly.

- Hardware and operation system abstraction:

  Hardware (robot) abstraction would greatly reduce the dependency between system software/hardware and application software. It also defines a standard interface for programmers and system software/hardware developers, clarify overall system architecture, and makes the inter-development among software application and system software /hardware more consistent and interoperable.

- Multi-platform support:

  Middleware has to support multiple platforms to make it easy for developers to incorporate various robot platforms (various robots) into an existing system.

- Client-Server architecture:

  Middleware should better be implemented based on Client-Server model, this way, it’s much easier to manage large number of robot devices.

Volume 5, Number 2, 2018
ISSN 2409-7284 (Online)

4. System Implementation

This paper reported a robot control mechanism, called RDML, which main purpose is to provide a quick way to build an interactive drama using XML for robot users. The system receives input data from sensors and interact with audiences according to input data. This way, robot's human-computer interaction becomes more realistic and attract more attention from users.

4.1 System architecture

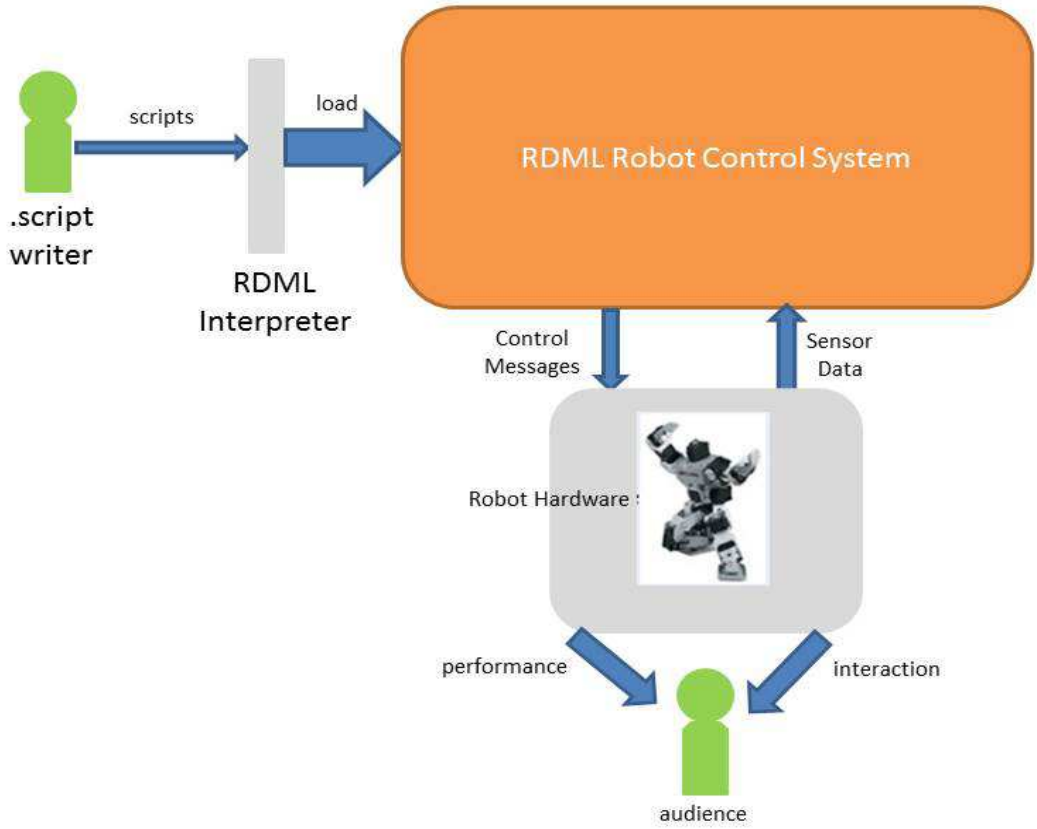

FIGURE 9. User interaction scenario

Figure 9 is the scenario diagram of interactions between RDML Robot Control System, script author and audiences. First, script author writes an RDML script, the script is then loaded and executed by RDML Robot Control System, which would send control messages to robot hardware devices to perform actions according to the interaction scenario described by the script. After the perform finished, robot's sensors collect response data from audiences and send it back to RDML Robot Control System, the system would then manipulate hardware devices to

Volume 5, Number 2, 2018
ISSN 2409-7284 (Online)

perform responsive actions according to the script.

4.2. RDML Authoring Tool

Since it is quite a complicated task to edit RDML tags using an editor for general users, in this paper, Robot Drama Authoring Tool (RDAT) is implemented to help users to generate a RDML script easily. Figure 10 is the layout of RDAT. There are two main panels in RDAT. The right panel is where we set up parameters of scene blocks and action blocks such as RAB's ID, background picture, actors, and actions. The left one is a diagram that visualize interaction drama to help users to understand architecture of their interactive scenario. Users can simply add or edit a scene block by clicking blocks in the diagram.

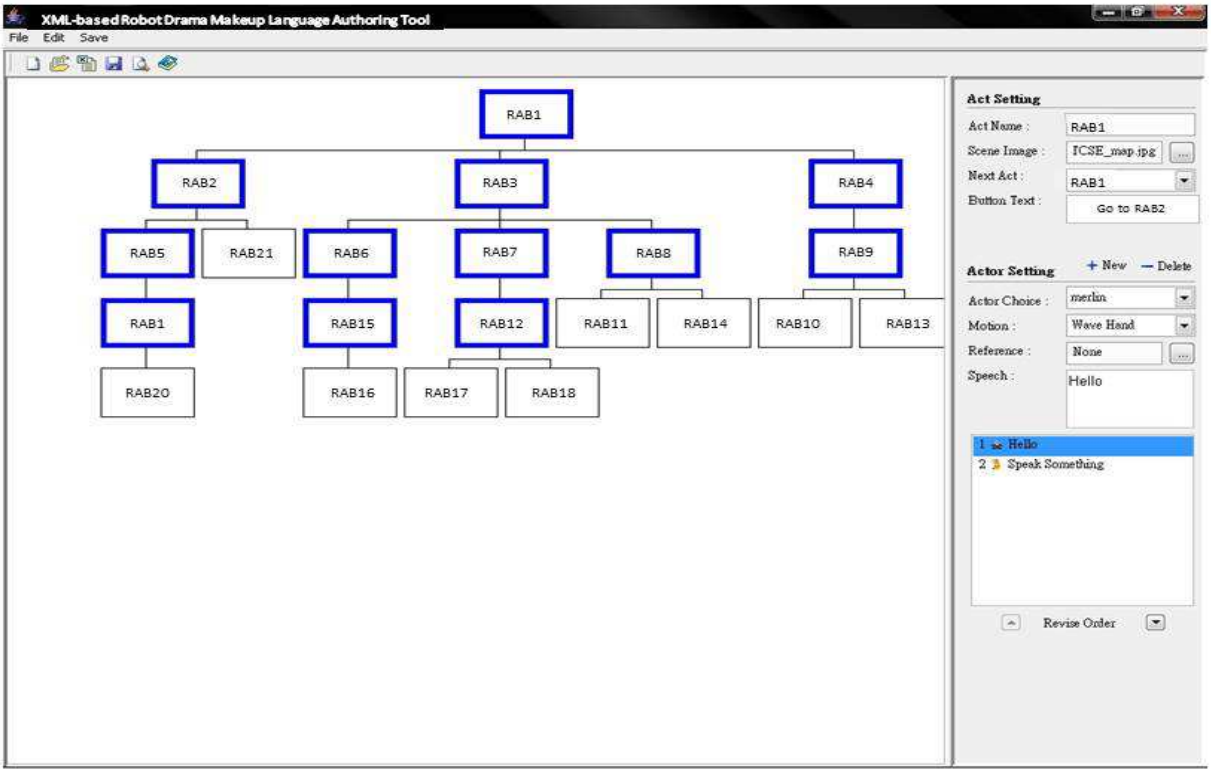

FIGURE 10. RDAT layout

4.3. RDML Robot Control System

After the RDML interactive script is generated, the script then loaded into RDML Robot Control System and execute it. RDML Robot Control System is responsible for parsing the script and control robots to perform actions accordingly.

Volume 5, Number 2, 2018
ISSN 2409-7284 (Online)

4.3.1. Tag mapping

In this research, by using RDML, user can form an interactive drama conveniently. However, RDML was firstly designed to control software ECAs, so there are some differences when we use RDML to control robots. Button tags in IDML Drama Block of ECA are used to be the tag to generate buttons for users to specify their decision when interacting with ECAs, now a robot receive users' decisions by their sensor devices, like Infrared (IR) sensors. In IDML, Act Block was used to control perform flow of Scene Block, so there's basically no difference between cases of software ECAs or RDML robots. Behaviors tag in IDML Scene Block used to be the tag to control ECAs' motion parameters, now it is used to control movements of robots.

Figure 11 shows RDML Robot Control System's architecture diagram in which relationships between different components in the system is shown. RDML Robot Control System contains three main components, namely RDML parser, RDML processor and Robot Control Middleware. These components together finish the task to control robot using RDML scripts.

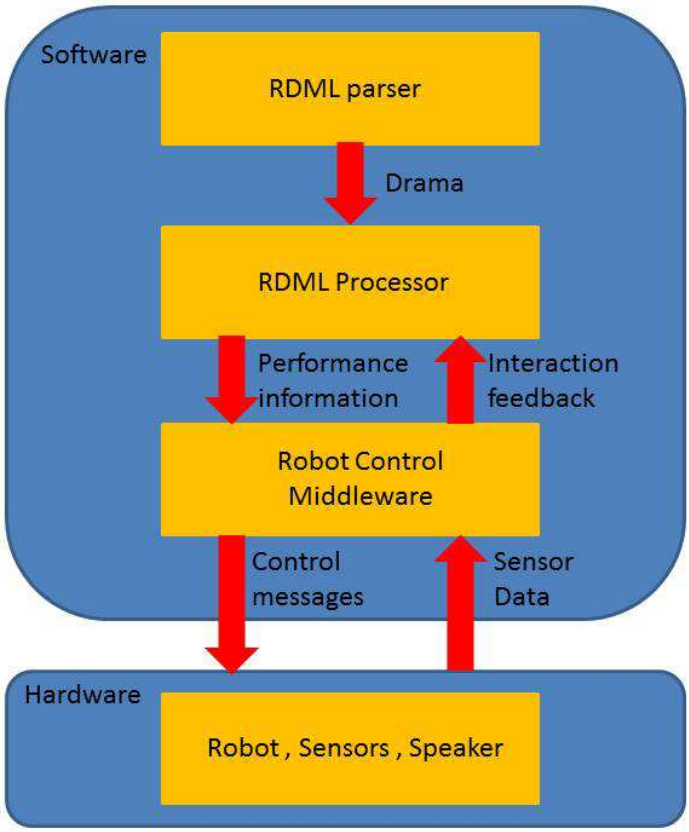

FIGURE 11. RDML Robot Control System architecture diagram

ISSN 2409-7284 (Online)

4.3.2. Programming interface design pattern

A programming interface design pattern can give driver engineers a direction to develop drivers. Thus, in this paper, we proposed a design pattern based Humanoid model and Sensor reception categories as a programming interface to users.

(1) Humanoid model

This model includes raw motion that describe atomic movement of robot parts, basic motion that is pre-defined by combining several raw motion into a logical body action as a human and preset motion. If existing preset motions cannot fulfill user's need, they can combine a series of basic motions to form a new desirable motion, called custom motion. We will demonstrate the abovementioned idea using humanoid robot as an example.

- Raw motion

  Raw motion describes atomic motion of this model, offer a way for users to define robot motions. The following are basic movements defined for humanoid model in this paper. We split humanoid into 6 parts: head, right arm, left arm, body, right leg, and left leg.

  Basic movements of every parts are:

  Head: turn left, turn right;

  Arms: rise upward, rise horizontally, rise rightward, rise leftward;

  Body: tilt forward, tilt backward; and

  Legs: rise forward, rise backward.

  We can add, delete, or edit basic motions according to robot's hardware capability. For example, if a robot is designed to be unable to move its body, basic motions of body can be deleted. For another example, if a robot can blink its eyes, this motion can be added.

- Basic motion

  Basic motions are pre-defined preset motions. This paper proposes following basic motions inspired by human's emotions: glad, angry, worry, thinking, sad, fear, shock. Every robot that belongs to this model must implement basic motions.

ISSN 2409-7284 (Online)

- Custom motion

  Custom motion are user-defined motions consists of a set of pre-defined motions, including raw motions and basic motions. For example, a custom motion that kicks a ball can be defined as: first rise left leg, backward, then rise it forward.

(2) Sensor reception

Sensor data is the main accordance that decides the interaction. As for the part of sensor data reception, this paper categorizes sensor data into three categories: raw reception, basic reception, custom reception.

- Raw reception

  Raw data are directly provided by sensors, such as input bits from microphone.

- Basic reception

  Basic reception provides an easy way for user to use preprocessed sensor data, such as: hands being touched by something, what voice microphone has heard. Users can design how each Robot Action Block is triggered using basic reception. In this paper, we propose the following basic reception inspired by human's senses.

  Head: heard something.

  Arms, body, legs: being touched.

- Custom reception

  Custom reception is designed for more flexible use of sensor data. Users can define Custom reception by combinations of any receptions. For example, a reception that tells whether the hand of the robot is being shaken can be defined as whether the hand is being touched for five seconds.

4.3.3 RDML Parser

RDML parser's main function is to parse RDML files given by script writers, analyze the scripts, and provide interaction information to RDML processor.

4.3.4 RDML Processor

RDML processor is the core of the whole system. It is responsible for controlling robot

according to Robot Scene Blocks written in the script, for example, it tells a robot what action to perform and judge what to do next according to input data received from sensors.

4.3.5 Robot Control Middleware

Robot Control Middleware offers an interface for RDML to control robot and receive sensor data. The abstraction of hardware devices separates underlying mechanisms with upper software implementation. To unify the usage of hardware devices, an interface has to be defined for developers to follow. The following are interfaces we have implemented:

- *Connect()*

  Connect to the robot.

- *DisConnect()*

  Disconnect from the robot.

- *Play()*

  Send control messages to hardware devices of the robot to perform actions.

- *GetMessage()*

  Get sensor data from robot, sensor data acquired from this method can be used to decide next Robot Act Block to perform.

Figure 12 shows a program slice of RDML Robot Control Middleware.

Volume 5, Number 2, 2018
ISSN 2409-7284 (Online)

```
@Override
public void DisConnent() {
    // TODO Auto-generated method stub
    NXTCommand.close();
}

@Override
public String GetMessage() {
    // TODO Auto-generated method stub
    NXTCommand.setVerify(true);
    int ks = 0;

    if(ts.isPressed()){
        System.out.println("You Touch Me");
        return "You Touch Me";
    }else if((ks = ss.getdBA())>60){
        System.out.println("Sound Tail:"+ks+"DBA");
        return "Sound Tail:"+ks+"DB";
    }else if(us.getDistance()<10){
       //System.out.println("I See U:"+us.getDistance()+"Cm");
       //return "I See U:"+us.getDistance()+"Cm";
        System.out.println("I See U");
         return "I See U";
    }

    //NXTCommand.setVerify(false);
    return null;
}

@Override
public void Play(String action) {
    // TODO Auto-generated method stub
    //System.out.print("QQ");
    //Action
```

FIGURE 12. Program code example of Robot Control Middleware

5. Conclusion

Robot technology is getting mature nowadays. Therefore, more and more robot applications will be developed. A user-friendly developing tool for robot applications is needed. In this paper, we proposed an RDML based robot control mechanism to help user to easily manipulate the interaction of a robot with the audiences more directly. With RDML, a robot application developer could write and design a robot application in a top-down manner just like writing a drama which can be further subdivided into several acts and then scenes by following the human drama logic. Thanks to a robot control middleware, various robots with different architecture or from different venders could be installed in an application without changing the RDML dramas in our proposed system.

Currently, the proposed system with RDML is a text-based programming style. However, it is somewhat difficult to image what postures a robot will take and how to translate a posture into the matching robot commands. Therefore, in the future, we will enhance RDML authoring tool

with a more user-friendly style, for example a graphic developing style, that will facilitate and help robot application engineers to coding the robot postures/motions in a program.

Acknowledgment

The authors would like to thank the Ministry of Science and Technology, Taiwan, ROC, for the partly financial support to this research under the contract number NSC100-2221-E-035-038-MY2.

References

Cassell, J., Sullivan, J., Prevost, S., & Churchill, E. (2000). Embodied Conversational Agents. The MIT Press, Cambridge MA, USA.

Marriott, A. & Stallo, J. (2002). VHML-Uncertainties and problems, A discussion, Proceedings AAMAS-02 Workshop on Embodied Conversational Agents- Let's Specify and Evaluate Them!

Arafa, Y. & Mamdani, A. (2003). Scripting Embodied Agents Behaviour with CML: Character Markup Language. Proceedings of the 8th international conference on Intelligent user interfaces, ACM Press, New York, USA, pp. 313-316.

Kshirsagar, S., Magnenat-Thalmann, N., Guye-Vuillème, A., Thalmann, D., Kamyab, K., & Mamdani, E. (2002). Avatar markup language. The 8th Eurographics Workshop on virtual environments, Barcelona, Spain.

Prendinger, H., Mori, J. & Ishizuka, M. (2005). Using human physiology to evaluate subtle expressivity of a virtual quizmaster in a mathematical game. International Journal of Human-Computer Studies, vol. 62, no. 2, pp. 231-245.

Prendinger, H., Descamps, S., & Ishizuka, M. (2004). MPML: a markup language for controlling the behavior of life-like characters, Journal of Visual Languages and Computing, vol. 15, no. 2, pp. 183-203.

Takahashi, T., Bartneck, C., Katagiri, Y., & Arai, N. H. (2005). TelMeA-Expressive avatars in asynchronous communications, International Journal of Human Computer, vol. 62, no. 2, pp. 193-209.

Kim, J. K., Sohn, W. S., Lim, S. B. and Choy, Y. C. (2008). Definition of a layered avatar behavior script language for creating and reusing scenario scripts. Multimedia Tools and Applications, vol. 37, no. 2, pp. 233-259.

Cole, R., Vuuren, S., Pellom, B., Hacioglu, K., Ma, Movellan, J. J. (2003). Perceptive animated interfaces: First steps toward a new paradigm for human computer interaction. Proceedings of the IEEE, vol.91, no. 9, pp. 1391-1405. doi: 10.1109/JPROC.2003.817143.

Chin, K. Y., Lin, J. M., Hong, Z. W., Lin, K. T. & Lee, W. T. (2009). Developing an IDML-Based Embodied Pedagogical Agent System for Multimedia Learning, Ninth International Conference on Hybrid Intelligent Systems. Shenyang China, pp. 37-41. doi: 10.1109/HIS.2009.15.

Kwang-Yao Lee. (2010). An IDML-Based Robot Control Mechanism, Master Thesis, Feng Chia University, Taichung City, TAIWAN.

Chin, K. Y., & Chen, Y. L. (2013). On Designing a Multimedia Mobile Learning System with Animated Pedagogical Agents, ICIC Express Letters, vol. 7, no. 1, pp. 159-164.

Chin, K. Y., Hong, Z. W., Huang, Y. H., Shen, W. W., & Lin, J. M. (2016). Courseware development with animated pedagogical agents in learning system to improve learning motivation, Interactive Learning Environments, vol. 24, no. 3, pp. 360-381. doi: http://dx.doi.org/10.1080/10494820.2013.851089.