Toward a Universal Platform for Integrating Embodied Conversational Agent Components

Hung-Hsuan Huang[1], Tsuyoshi Masuda[1], Aleksandra Cerekovic[2],
Kateryna Tarasenko[1], Igor S. Pandzic[2], Yukiko Nakano[3], and Toyoaki Nishida[1]

[1] Department of Intelligence Science and Technology, Graduate School of Informatics,
Kyoto University, Japan
{huang, masuda, ktarasenko, nishida}@ii.ist.i.kyoto-u.ac.jp
[2] Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia
{aleksandra.cerekovic, Igor.Pandzic}@fer.hr
[3] Department of Computer, Information and Communication Sciences,
Tokyo University of Agriculture & Technology, Japan
nakano@cc.tuat.ac.jp

Abstract. Embodied Conversational Agents (ECAs) are computer generated life-like characters that interact with human users in face-to-face conversations. To achieve natural multi-modal conversations, ECA systems are very sophisticated and require many building assemblies and thus are difficult for individual research groups to develop. This paper proposes a generic architecture, the Universal ECA Framework, which is currently under development and includes a blackboard-based platform, a high-level protocol to integrate general purpose ECA components and ease ECA system prototyping.

1 The Essential Components of Embodied Conversational Agents and the Issues to Integrate Them

Embodied Conversational Agents (ECAs) are computer generated life-like characters that interact with human users in face-to-face conversations. To achieve natural communications with human users, many software or hardware assemblies are required in an ECA system. By their functionalities in the information flow of the interactions with human users, they can be divided into four categories:

ECA Assemblies in the Input Phase. Non-verbal behaviors are the indispensable counterpart of verbal information in human conversations and thus embodied agents have to possess the capabilities of both of them. In addition to capturing natural language speech, non-verbal behaviors such as head movements, gaze directions, hand gestures, facial expressions, and emotional conditions are acquired by various types of sensors or visual methods in ECA researches. Further, input understanding tasks such as speech and gesture recognition are also required to be done in this phase.

ECA Assemblies in the Deliberate Phase. This is the central part of an intelligent agent to determine its behaviors in responding to the inputs from the outside environment. An inference engine with a background knowledge base and a dialogue

B. Gabrys, R.J. Howlett, and L.C. Jain (Eds.): KES 2006, Part II, LNAI 4252, pp. 220 – 226, 2006.
© Springer-Verlag Berlin Heidelberg 2006

manager are required for conducting a discourse plan to achieve the ECA's conversational goal according to the agent's internal mental state. Talking to a conversational agent without emotions and facial expressions is weird and will be easily satiated while being like a human in the real world, personality, emotion, culture, and social role models are incorporated into ECAs to improve their believability.

ECA Assemblies in the Output Phase. Verbal output or natural language synthesis is generally done by a Text-To-Speech (TTS) engine to speak out the text output from the dialogue manager. Spontaneous non-verbal behavior outputs such as facial expressions, eye blinks, spontaneous hand gestures, and body vibrations are generated randomly or depending on the syntactical information of accompanied utterance by using the result of statistical analysis like CAST [5]. At last, a 2D/3D character animation player that renders the virtual character body and probably the virtual environment where the character resides on the screen is necessary.

A Platform for Integrating ECA Components. To integrate all the various assemblies of an ECA system described above, a platform or framework that seamlessly integrates them is a critical part. This platform has to transport all the sensor data streams, decisions, and command messages between all the components. It has been proposed that there are four essential requirements in the ECA component integration issue [4, 6]. First, the platform has to keep all of output modalities to be consistent with the agent's internal mental state. Second, all the verbal and non-verbal outputs are required to be synchronized. Third, ECAs have to be able to response to their human users in real-time. Fourth, the support for two ways of the information flow, "pull data from a component" and "push data to a component" are required in ECAs.

2 Universal Embodied Conversational Agent Framework

ECA systems are so sophisticated and their functions actually involve multiple research disciplines in very broad range such that virtually no single research group can cover all aspects of a full ECA system. Moreover, the software developed from individual research result is usually not meant to cooperate with others. There is a number of outstanding ECA systems that have been proposed previously, however, their architectures are ad hoc designed [2] and are not for a general purpose use.

Therefore, if there is a common and generic backbone framework that connects a set of general-purpose reusable and modulized ECA components which communicate with each other in a well-defined and common protocol, the rapid building and prototyping of ECA systems become possible, and the redundant efforts and resource uses of ECA researches can be prevented. This work proposes such an architecture that eases the development of ECA systems for general purposes. In our current design, it contains the following three parts, a general purpose platform (Universal ECA Platform) which is composed by a set of server programs for mediating and transporting data stream and command messages among stand-alone ECA software modules, a specification of a high-level protocol based on XML messages (UECAML) that are used in the communication between a standardized set of ECA components, and an application programming interface (UECA API) for easy development of the wrappers for the ECA software modules. These basic concepts are shown in Fig. 1.

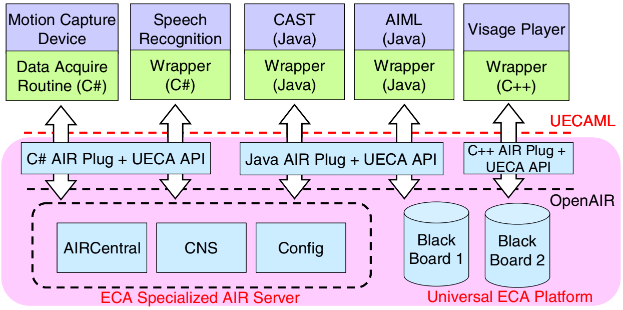


Fig. 1. The conceptual diagram of our Universal ECA Framework that includes the Universal ECA Platform server, UECA API, and a high-level protocol, UECAML

We use blackboard model as the backbone platform and OpenAIR [6] as the low-level routing and message passing protocol for the following reasons:

- Distributed architecture and XML absorb the differences of operating systems and programming languages of components and distribute the computation complexity.
- Unlike a long pipelined architecture, the single-layer topology provides the possibility to support reflexive behaviors that bypass the deliberation of the agent.
- The weak inter-connectivity of the components allows the online switching of components and thus makes online upgrading and maintaining of components.
- Components with different levels of complexity can be integrated into the ECA system as long as they understand and generate the same message types and the system can still work even some components are absent.
- Logically isolated multiple blackboards can distribute information traffic that is concentrated on only one blackboard in traditional systems.

Based on this framework, we are specifying an XML based high-level protocol for the communications between ECA components. Every XML message belongs to a message type, for example, "input.speech.text", "output.body.gesture", etc. Each message type has a specified set of elements and attributes, for example, "intensity", "time_interval", "start_time", etc. Each component *subscribes* its interested message type(s), read them from the blackboard when they are *published* by another component, generates its output and publishes messages in other types to the blackboard. In the current stage, we are focusing on the specification on input and output phases and categorized the message types in the procedure of the I/O phases into an abstract hierarchy having three layers in the blackboard according to their abstractness. This basic idea is depicted in Fig. 2(a) and described below.

Low-level Parameter Layer in Input Phase: To absorb the possible variance even for the same modality in the lowest-level raw sensors' data, the sensor data handling components interpret raw data into low-level parameterized representations, and then write them into the blackboard. For example, rather than the raw wave data from the voice capture subsystem, the user's voice is interpreted into a recognized text stream by a speech recognition component, rather than the absolute current positions and angles of a sensor of the motion capture system, the numbers are transformed into

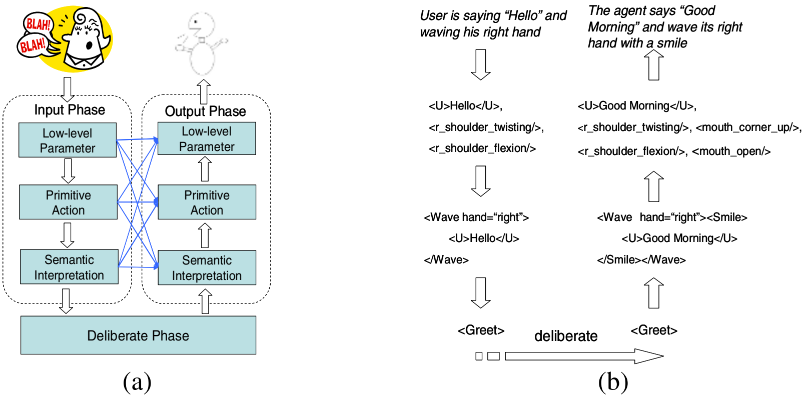

BLAH!
BLAH!
Input Phase
Low-level Parameter
Primitive Action
Semantic Interpretation
Output Phase
Low-level Parameter
Primitive Action
Semantic Interpretation
Deliberate Phase
(a)
User is saying "Hello" and waving his right hand
The agent says "Good Morning" and wave its right hand with a smile
<U>Hello</U>, <r_shoulder_twisting/>, <r_shoulder_flexion/>
<U>Good Morning</U>, <r_shoulder_twisting/>, <mouth_corner_up/>, <r_shoulder_flexion/>, <mouth_open/>
<Wave hand="right"> <U>Hello</U> </Wave>
<Wave hand="right"><Smile> <U>Good Morning</U> </Smile></Wave>
<Greet>
deliberate
<Greet>
(b)

Fig. 2. (a)The hierarchy of the high-level protocol, UECAML, notes that reflex action links are shown in blue arrows (b) Example messages for a greeting response of the conversational to a human user's greeting behavior

angles of the joints of a human body. As a result, the total output of the components in this stage is a parameterized text representation of movements of human users includes the angles of body joints, eye gaze directions, facial expression primitives, speech in text, physiological parameters and so on. Because the parameters in this stage must be specified in great detail with specific expert knowledge, we are going to specify the protocol of this layer based on appropriate and popular existing standards such as MPEG-4 FBA.

Primitive Action Layer in Input Phase: The task of this layer is to read the low-level parameters from the last layer and to interpret them into messages expressing abstract primitive actions. For example, from the change of the head orientation in horizontal and vertical directions to higher level primitive actions like *"head-shaking"* or *"head-nodding"*, from the change of bending angles of the joints of shoulder, elbow, and wrist to recognize an user action like *"waving-right-hand"* or *"raising-left-hand"*, from the angles of the joints of arms and fingers to recognize the action, *"pointing (X, Y, Z)."* Because there is virtually no limit in the range of body actions, we are going to specify just a generally useful set for specifying primitive body actions. At the same time, the message format should be flexible enough to allow new primitive actions to be included in the messages and that action will be interpreted as long as the specified component dealing with messages in this layer can recognize and understand it. A recognized primitive action is then propagated through the platform as an event in the instant when it is recognized with a timestamp and probably additional attributes such as intensity and time interval.

Semantic Interpretation Layer: The messages belong to this layer are semantic meaningful events and are interpreted by components from the primitive actions, for example, a user behavior recognizing component may interpret the primitive actions *"smiling"* and *"waving-right-hand"* done by the users to a *"greeting"* semantic explanation.

Deliberate Phase Layer: We plan to specify the messages in this layer to include the inference, knowledge representation, dialogue management, personality model, emotion model, and social role model functionalities as future works. Currently, we assume that the inputs of this black box are text streams recognized from human users' utterances which are annotated with semantic events or primitive action markups. The outputs are then utterances of the conversational agents that are going to be spoken by a TTS engine and annotated with markups to specify facial expressions, body movements, gestures, and other non-verbal behaviors.

Output Phase: In output phase, message flows are processed in a reversed order comparing to input phase, where messages from the deliberate phase are decomposed to more concrete concepts with lower abstractness by responding components. For example, when the deliberate phase decided that the agent should *greet* the user, this semantic command then may be interpreted by an action catalogue component into the *"utterance ("Good Morning")"*, *"smile"* and *"wave-right-hand"* primitive actions. These two primitive actions are then further interpreted into low-level facial animation parameters and body animation parameters by a FAP / BAP database component to drive the CG character of a MPEG-4 FBA compatible player to smile and wave its right hand. A sample message flow that follows the framework of a process for an agent to greet in response to a human user's greeting behavior is shown in Fig. 2 (b).

The shared blackboard(s) mechanism allows the components to exchange information easier between different logical layers; a component can write its outputs arbitrarily into other layers and thus components with different level of sophistication can work together. Further, reflex action controlling components that bridge input phase messages directly to output phase messages are also allowed in this architecture.

Generally, blackboard architecture suffers from two major disadvantages. First, due to the distributed problem-solving methodology, it usually lacks a mechanism to centrally direct how a problem is going to be solved. This problem as well as the multimodality consistency issue can be remedied by introducing a centralizing component to issue action confirming messages in the deliberate phase, that is, the actions sent to all output modalities will not be executed without confirmation except the reflexive behaviors. Second, the additional information traffics involving the shared blackboard cause inefficiency. The performance deterioration can be reduced by the direct information exchanges between the components while the message traffic load centralized on a single blackboard can be reduced by using multiple logically isolated blackboards at the same time. Besides, we plan to address the ECA component synchronization issue by the following ways, to require all the machines composing the system to be synchronized with each other to absolute standard time by NTP and to utilize the explicit timestamp field in each message as well as incorporating "*after the next action*", "*begin at the same time as the next action*" specifiers for primitive actions.

3 Prototype of the Universal ECA Platform

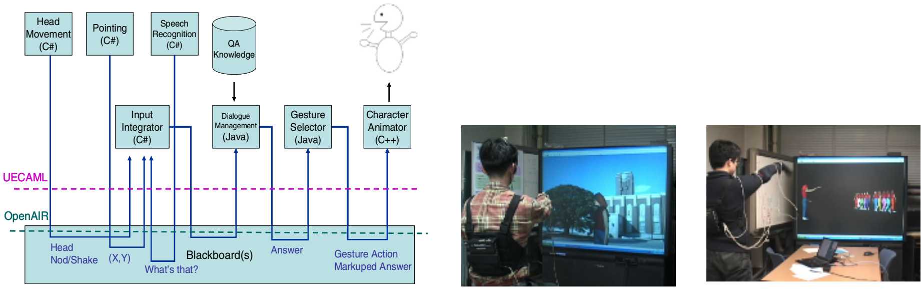
We have implemented a Java prototype of the platform that routes and transports the communication between the ECA components those have registered in it. In addition to the reference Java AIR Plug implementation from mindmakers.org, we have developed a C# version and are developing a C++ AIR Plug library. Based on the backbone platform, we are defining UECAML, which is currently focused on multimodality inputs and CG character animation outputs.

As a premier evaluation, we developed two experimental ECA systems. One is a campus guide agent, it stands in front of a photo of somewhere in a campus while human users can ask it what an object in that photo is with natural language, hand pointing and head movements. As shown in Fig. 3(a), 3(b), the campus guide agent is composed with seven modules, head movement module utilizes an acceleration sensor to detect head shakes and nods, pointing module uses data from a magnetic motion capture device to judge which area the user is pointing at with his (her) right hand, a wrapped SAPI compatible Japanese recognition engine, a wrapped AIML [1] interpreter for dialogue management, gesture selector module is a wrapped CAST engine, input integrator module integrates all the tree modalities into individual input events, and a character animator player developed with visage|SDK [7]. The other one experimental system is an application for experiencing cross-culture differences of gestures and is shown in Fig. 3 (c). In this virtual environment, an avatar replays the user's hand gestures such as beckoning, and there are multiple computer controlled agents that react to those gestures. Their reaction differs depending on which country they are supposed to come from for example, Japan or Britain.

The two experimental ECA systems themselves are relatively simple; however, this work is not emphasizing on how strong the built ECAs are but is trying to ease the development and provide sufficient capabilities for general ECA researches. In the preliminary evaluation, the campus guide agent proves the platform's capability to seamlessly deal with multimodal inputs and sufficient performance for smooth realtime conversation. In the gesture experiencing application, our three-machine configuration showed satisfying performance to drive an avatar with motion capture device and ten computer controlled agents in real-time. Besides, both of these two systems can be built by incorporating software tools which are not specifically designed for these systems with little efforts, just by wrapping those tools according to the specification of universal ECA platform, and then an ECA system works. For example, the campus guide agent was built in three hours by writing two scenarios in AIML and the input integrator in addition to the other general purpose modules and pre-defined gestures. Further, in these two experimental systems, it usually requires only several dozen lines of code to wrap a software tool.

4 Future Works and Evaluation

This work is yet far from reaching its objectives. We are going to complete the definition of the standard high-level protocol to allow the integration of common ECA assemblies, improve the infrastructure to support the necessary features for the protocol, a set of client-side libraries supporting easy integration of ECA assemblies developed in various programming languages on various operating systems as well as a set of wrapped common ECA tools. The ultimate objective of this work is to pack all of these as an ECA development toolkit including a workable skeleton ECA.



(a) (b) (c)

Fig. 3. (a)The campus guide agent's configuration (b) snapshot of the campus guide agent while it is performing a pointing gesture (c) An application for experiencing the cross-culture differences of hand gestures

We plan to produce a preliminary release for a field test in our proposed project at the eNTERFACE'06 [3] summer workshop on multimodal interfaces. We expect that several participants will join our team during the workshop; they will be provided with an initial release of the platform and jointly develop an ECA application during the relatively short four-week workshop period. During the practical field use of the platform, we expect to evaluate the platform in the following aspects, expressiveness of the high-level protocol, the ease of use, and the performance of the platform in the sense of responsiveness and the consistency of all modalities. We do not expect the platform to be fully satisfying the requirements in the preliminary release but are going to update the requirements, gather problem reports and other suggestion during the workshop. We will then improve the platform based on these experiments and make a public release.

References

[1] Artificial Intelligence Markup Language (AIML), http://www.alicebot.org/

[2] Cassell, J., Vilhjalmsson, H., Bickmore, T.: BEAT: the Behavior Expression Animation Toolkit, in *The Proceedings of SIGGRAPH '01*, pp.477-486, 2001.

[3] The eNTERFACE'06 workshop on multimodal interfaces, http://enterface.tel.fer.hr

[4] Gratch, J., Rickel, J., Andre, E., Cassell, J., Petajan, E., and Badler, N.: Creating Interactive Virtual Humans: Some Assembly Required. *IEEE Intelligent Systems*, pp.54-63, 2002.

[5] Nakano, Y., Okamoto, M., Kawahara, D., Li Q., Nishida, T.: Converting Text into Agent Animations: Assigning Gestures to Text, in *The Proceedings of The Human Language Technology Conference* (HLT-NAACL04), 2004.

[6] Thorisson, K., List, T., Pennock, C., and DiPirro, J.: Whiteboards: Scheduling Blackboards for Semantic Routing of Messages & Streams, AAAI-05 Workshop on Modular Construction of Human-Like Intelligence, 2005.

[7] visage|SDK, visage technologies, http://www.visagetechnologies.com/index.html