Published in International Journal of Multiagent and Grid Systems, Vol. 4, No. 4, IOS Press, 2008

Integrating Embodied Conversational Agent Components with a Generic Framework

*Hung-Hsuan Huang[1], Aleksandra Cerekovic[2], Kateryna Tarasenko[1], Vjekoslav Levacic[2], Goranka Zoric[2], Igor S. Pandzic[2], Yukiko Nakano[3], and Toyoaki Nishida[1]

[1]Graduate School of Informatics, Kyoto University, Japan, [2]Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia, [3]Department of Computer, Information and Communication Sciences, Tokyo University of Agriculture & Technology, Japan

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 sophisticated and require numbers of building assemblies. They are thus difficult for an individual research group to develop. To address this problem, we are developing an approach to connect those components with a Generic ECA (GECA) framework. GECA is composed with a blackboard-model based platform, a high-level protocol and a set of APIs which are meant for easing component wrapper development. As an expectation, with such a generic ECA framework, rapid ECA system prototyping is possible while research result sharing and the collaboration between ECA researchers can be facilitated. This paper describes the basic concepts of this framework, an initial implementation and evaluations of actually using it to build a realistic ECA application.

Keywords: embodied conversational agent, distributed system, blackboard, user interface, verbal / non-verbal interaction

* Corresponding author:
huang@ii.ist.i.kyoto-u.ac.jp, Tel: +81-75-753-5867, Fax: +81-753-4961, Room 131, Engineering Faculty Bld. 10, Kyoto University, Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501, Japan

1. Introduction

Embodied Conversational Agents (ECAs) are computer generated life-like (not necessarily human-like) characters that interact with human users in face-to-face conversation and possess the following abilities:

- Recognize and respond to verbal and nonverbal input from the human user
- Generate verbal and nonverbal output
- Perform conversational functions (e.g. utterance turn taking, feedback and repair mechanisms)
- Give signals that indicate the state of conversations as well as to contribute new propositions

To achieve the functions described above and to compose a non-trivial ECA system, many assemblies or components may be required. According to their functionalities in the information flow of the interaction with human users, they can be divided into four categories:

ECA Assemblies in the Input Phase. Since ECA systems communicate with their human users in a natural way like human-human communications, by the observation on the natural face-to-face communications between humans, we can find non-verbal behaviors are the indispensable counterpart of verbal information. For example, the pronoun “this” with a pointing gesture can be a very typical example. Without the pointing gesture, which object this “this” is mentioning becomes ambiguous. At the same time, the same words spoken in different voice tones with different facial expressions can be interpreted into totally different meanings. Therefore, embodied agents have to accommodate the capabilities to acquire various types of inputs from human users include verbal and non-verbal communicational modalities.

Verbal channel: natural language speech capturing and recognition.

Non-verbal channels: the following modalities can be included in the consideration of an ECA system design.

- Head movements such as nods or head shakes, they can be detected by a head tracker or another sensors like an acceleration / motion sensor
- Gaze direction that can be detected by eye tracking devices
- Body postures and hand gestures, they can be captured by using magnetic, optical motion capturing devices or computer vision methods
- Finger movements, which can be acquired by data gloves
- Emotion recognition, it can be estimated by facial expressions extracted from visual data, acoustic analysis, or bio sensors those measure heart rate, blood pressure, etc.

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, it is also true in the case of a conversational agent. In the deliberate phase of an ECA, the following functional components can be included.

- Input understanding from both verbal and non-verbal modalities. For example, natural language processing and understanding, gesture recognition, etc.
- Mind issues: a non-trivial ECA should have its own conversational goal to achieve, for example, to sell an airplane ticket to a human user, or guide a human visitor to the place where he wants to go. Therefore, an inference engine with a background knowledge base and a dialogue manager are required for conducting a discourse plan to achieve the agent's conversational goal. Besides, the plan should be conducted according to the agent's background knowledge, user input, and its own internal mind state.
- Personalization: talking to a conversational agent without emotion and facial expression is weird and will be easily satiated, like a human in the real world, personality, emotion, culture, and social role models should be incorporated into an ECA system to improve its believability in communicating with human users.

ECA Assemblies in the Output Phase. Symmetrical to the input phase, in the output phase of the communication between a conversational agent and its human users, a body with some degree of expressiveness in both verbal and non-verbal channel of it is required. Rather than robots, software animated characters are preferred by researchers because of their freedom of expressiveness. The output phase can include the following components.

- Verbal output or natural language synthesis is generally done by a Text-To-Speech (TTS) engine to speak out the text output from a dialogue manager. To increase the believability and naturalness of the voice, it is desirable that the TTS engine's output parameters such as prosody properties, speed, and volume can be adjusted to represent the voice expressions of the characters.
- Spontaneous non-verbal outputs such as facial expressions, eye blinks, spontaneous hand gestures, and body vibrations are often generated randomly or according to the syntactical information of accompanied utterance by using the result of statistical analysis.
- A 2D/3D character animation player that renders the virtual character animation and possibly the virtual environment where the character resides on the screen.

Platform for Integrating ECA Components. To integrate all the various assemblies of an ECA system described above, a platform or framework that seamless integrate them is also a critical part. This platform transports all the sensor data streams, decisions, and command messages between all the components.

There are four essential requirements in the ECA component integration problem. First, the platform has to keep all output modalities to be consistent with the agent's internal mental state, for example, if the agent is in a happy mood, it is not supposed to speak in a voice tone that sounds angry or with a sad facial expression. Second, all the verbal and non-verbal outputs are required to be synchronized. For example, the pointing gesture must be synchronized with the utterance, "this", or what the "this" is specifying becomes unclear. Third, ECAs have to response to their human users in real-time. Fourth, the information flow includes the two ways, "pull" and "push" are required in ECAs. Therefore, previous integrating solutions like traditional remote method invocation (RMI) APIs or distributed object / service model like CORBA [18], Web Services [25] those provide only "pull" direction information flows, or the solutions those do not support explicit temporal model and real-time applications like KQML [9] are not appropriate for the ECA components integration problem.

Furthermore, ECA systems are very sophisticated and the functions described above actually involve works from multiple research disciplines like artificial intelligence, computer graphics, cognitive science, sociology, linguistics, psychology, etc. They are in so broad range of research disciplines such that virtually no single research group can cover all aspects of a fully operating ECA system. Besides, the software developed from individual research result is usually not meant to cooperate with each other and is designed for different purpose. While at the same time, there are numbers of outstanding ECA system like Rea [3] and Mack [5] have been proposed previously, however, their integration architectures of the components are usually ad hoc designs 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 modulized ECA components those communicate with each other in a well-defined and common protocol, the rapid building and prototyping of ECA systems will become possible, and the redundant efforts and resource uses of ECA researches can be prevented. This work is going to propose such a platform that eases the development of ECA systems for general purposes.

2. Related Works and Activities

Researchers have realized the demand for a solution of ECA component integration problem; many interesting topics and issues are discussed in a workshop [7] and a number of related works have already been initiated.

MPEG-4 [12] FBA is a standard for specifying humanoid facial and body animation with model definition parameters and animation parameters via a narrow communication channel to achieve low-weight face-to-face communication. H-Anim [8] is a VRML97 extended standard for specifying a hierarchical geometry of humanoid figures for Web applications.

Some high-level conversational agent or virtual human description markup languages have been proposed or are being developed such as AML [10], VHML [24], and CML [2]. AML is a high-level script language specifying avatar animations; the AML processor reads AML scripts containing high-level descriptions of avatar facial expressions, body animation, and utterance text of the avatar, or references to MPEG-4 FAP / BAP files, and then it generates the corresponding MPEG-4 bit stream for Web based applications. However, their agent architecture is deterministic and thus has no flexibility; the script language does not consider the input part from the human user, either. VHML is an high-level markup language that describes a virtual human for general purpose, it is composed with a set of sublanguage includes descriptions on emotion, facial expressions, and gestures, etc. However, the specification of VHML is distinct and thus has little flexibility to include supplement FAP / BAP files like AML does. Many parts of it are still undefined, especially the gesture or body animation parts. CML is another under-development high-level virtual character description markup language which is similar to AML. It differs to AML with the specification of emotion and personality model while its predefined base set of movements can not be extended dynamically.

MPML-VR [13] and TVML [22] are very high-level script languages designed for easy making of presentation or TV-program like contents. With their user friendly interfaces, these contents can be created by writing a simple script to describe a limited predefined set of virtual word, objects, characters, and character behaviors.

OpenAIR [19], which is proposed by mindmakers.org [11] is a specification of an XML based message format for low-level routing and message passing between stand-alone software modules using subscribe-publication mechanism. OpenAIR specifies the characteristic features like timestamp, priority fields, and a way to transmit binary data streams. It is simple enough to be considered to be suitable and effective for real-time interactive A.I. applications. The same organization has developed a relatively simple reference ECA based on it, Mirage [21]. In our work, we also adopted OpenAIR in our work as the low-level routing protocol and want to further develop a generic solution.

3. The Concepts of Generic Embodied Conversational Agent Framework

To address the ECA component integration problems mentioned in section 1, our research group is developing a generic ECA framework which is meant to be suitable for easy and rapid prototyping of ECA system development. We are designing it to provide the following functions while its basic concepts are shown in Fig. 1:

– A general purpose platform (GECA platform) and a set of managing programs for mediating and transporting data stream and command messages among stand-alone ECA software modules those compose an ECA system interacting with human users in multiple modalities.
– A specification of a high-level protocol (GECAML) based on XML messages that are used in the communication between a standardized set of ECA components such as sensor inputs from the human users, inference engine, emotion model, personality model, dialogue manager, face and body animation, etc.
– An application programming interface (API) available on main-stream operating systems and programming languages for easily adapting ECA software modules to hook to the platform.

The backbone architecture we chose for the integration framework for ECA systems is blackboard model using subscribe/publish mechanism to share information between the components. Blackboard model is a well-known methodology often used in distributed and large scaled expert systems; it works like a shared memory where independent knowledge sources read and write information. Each knowledge source subscribes to the blackboard manager their interested information and is informed to read the blackboard whenever the other knowledge sources publish that kind of information. The knowledge source then updates its output to the blackboard according to the new information which it has just read. The problem is then incrementally solved by weak cooperation of a number of knowledge sources. Also, we use the very simple OpenAIR protocol as the low-level routing and message passing protocol. We decided to use OpenAIR and blackboard (or "whiteboard" in OpenAIR's terminology) combination because of the following reasons:

– Distributed architecture based on network and XML to absorb the differences of working operating systems and programming languages of components and to distribute the computation complexity.
– Unlike previously proposed dedicated ECA architectures those are usually in a

pipeline [4], the single-layer topology of blackboard model 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 directly integrated into the ECA system as long as they understand and generate the same message types. For example, for a research group focusing on speech synthesis aspects of virtual humans, they can equip their agent with a simple pattern matching program instead of a complicated inference engine in the development stage. Various configurations can be achieved flexibly with different combinations of ECA components.
– Multiple blackboards those are logically isolated with each other can distribute information traffic that is concentrated on only one blackboard in traditional systems.
– OpenAIR's priority field allows the blackboard manager to transmit system messages and reflexive action messages in higher priorities.
– OpenAIR's timestamp field makes the management of timing and synchronization problem possible.

Based on this low-level communication framework, we are specifying an XML based high-level protocol for the communications between ECA components. In this framework, every XML message has 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 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. Considering the information flow from the human user's input to the response of an embodied agent, we 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. A component does not necessarily write its output one layer above or under its input, it can write its outputs arbitrarily in other layers, and thus this framework allows components with completely different level of sophistication. This basic idea is depicted in Fig. 2 and described below.

Low-level Parameter Layer in Input Phase: As described in section 1, the same input modalities can be captured by multiple types of sensing devices with a wide range of precision and prices. Besides, new hardware devices can be developed in the future because of the advance of technology. Furthermore, raw data streams from the sensors are often very large in sizes; it is usually inefficient to propagate raw data across the platform. Therefore, to absorb the differences of the lowest-level raw sensors data, the sensor data handling components read messages in this layer, interpret them into low-level parameterized representation, and then write them into higher layer of 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 speech recognition component, rather than the absolute current positions and angles of a sensor of the motion capture system, the numbers are transformed into 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 the movements of human users includes the angles of body joints, eye gaze direction, facial expression primitives, speech in text, physiological parameters and so on.

These parameters are then propagated among the platform like text streams, no mater the descriptions come from what kind of sensors or even just be typed by the user from a console. 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 base on appropriate and popular existing standards. For example, we are currently specifying this part like a simplified MPEG-4 FAP / BAP for the face and body representation that can be recognized by motion capture devices.

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)"*, from the movement of eyebrows, mouth and mouth corner movement to recognize the facial expression, *"smiling."*

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 while allowing user-definable extensions at the same time. That is, the message format should be flexible enough to allow new primitive actions to be included in the messages and the action will be interpreted as long as the specified component that deals with messages in this layer can recognize and understand it. A recognized primitive action is then propagated through the platform in a uniformed way 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 these parts are left as a black box. We assume the inputs of this black box are text streams recognized from the human users' utterances which are annotated with semantic event or primitive action markups. The outputs are then the 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 of the animated character.

Output Phase: Message flow of the semantic interpretation, primitive action and low-level parameter layers of output phase are processed in a reversed order compared to those in the input phase, 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* to the user, this semantic command then may be interpreted by an action catalogue component to 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 follows the framework of a process for an agent to greet to response to a human user's greeting behavior is shown in Fig. 3.

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 multi-modality 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 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 [17] and to utilize the explicit timestamp field in each message as well as incorporating synchronization specifiers such as "*after the next action*", "*begin at the same time with the next action*" for primitive actions.

4. Prototype Implementations

This work is still in its very early stage of development. At present, we have implemented a prototype of a Java OpenAIR blackboard server that routes and transports the communication between the ECA components those have registered on it. In addition to our customized version of Java AIR Plug modified from the reference implementation from mindmakers.org, we have also developed C# and C++ version AIR Plug libraries that ease the task to adopt ECA assemblies to the platform. Besides, based on the backbone platform, we are defining the ECA specialized high-level protocol (GECAML) which is currently focused on multi-modality inputs and CG character animation outputs.

On the other hand, we have developed two experimental ECA systems for preliminary system evaluation. One is a university campus guide agent and one is an application for experiencing cross-culture differences of gestures.

The campus guide agent has a configuration contains seven components and is shown in Fig. 4.

- A 3D motion sensor [16] module that recognizes head shakes and nods by data from a sensor which is attached to the human user's head
- A magnetic motion capture device module that recognizes the user's pointing gestures
- A speech recognition module that recognizes a set of questions like "What's that?", "Ok, I see."
- An input integrator that maps the positions that the user is pointing to an area and combines the three input modalities into one single user query event by the message timestamps
- A wrapped AIML [1] interpreter [20] that answers the user's questions
- A wrapped CAST [14] engine which selects spontaneous gestures according to the agent's verbal outputs
- An animation player which is implemented with a MPEG-4 compatible CG character SDK [23] and has a built-in MPEG-4 BAP database of the gestures that may be selected by the gesture selector component

In this system, there is an embodied agent who stands in front of a background image; say a photo of the entrance of a university. The human user can ask the agent to explain what an object is in the background image with speech, pointing a location on the screen with right hand and head movements.

The application for experiencing cross-culture gesture difference is a virtual environment contains one user avatar and multiple embodied agents. The avatar reproduces the user's hand gestures such as beckoning while the embodied agents react to those gestures pretending they are belong different countries, for example, Japanese and British. The user's actions are captured by a magnetic motion capturing device and translated to low-level joint angle parameters to drive the avatar character in real-time. The ten embodied agents are driven by ten reflexive controlling units individually with a common BAP catalogue component and ten individual figures those are driven by low-level MPEG-4 BAPs sent from the blackboard. This system is shown in Fig. 5.

To concentrate on the development of input phase and output phase, in current prototype systems, a component running an AIML scenario script controls the behaviors of the conversational agent. AIML is originally designed as a script language defining a question-answer database for a text based chatting software robot and thus lacks the capabilities to describe the non-verbal inputs from the user and the non-verbal responses for the conversational agent. In AIML, one interaction between the agent and its user is defined inside a "`category`" element while the utterance of the user is defined in a "`pattern`" element and the response utterance of the agent in a "`template`" element, AIML interpreters then do pattern matching from the input utterance to the output utterance according to the AIML script with some degree of context state expressiveness. We extended the original AIML by adding a set of GECAML tags that specify non-verbal inputs, non-verbal outputs, and system control such as changing the scene. Since these tags are not defined by AIML and can not be processed correctly, the controlling tags are denoted by square bracket "[" and "]" marks instead of standard tag marks.

Each ECA software component makes its contribution to the whole system by annotating its input message and then sends the result back to the shared blackboard to be read by the other components. A virtual message annotating pipeline is built dynamically according to the message types in input phase and output phase.

In addition to directly triggered reflexive agent behaviors such as looking at the direction where the user is pointing, one agent's response behavior is triggered by certain inputs from the user defined in the scenario script. The triggering inputs may be either verbal or non-verbal events, that is, results from speech recognition or sensor devices. In the case of input phase, the pipeline starts with a message in type "`input.speech.text`", which is typically generated by a speech recognition engine from the user's utterance, or non-verbal input components that generate the same message type. For example, the head movement sensing component generates "yes" when the user nods and "no" when the user shakes his (her) head. The message is then annotated by the other components in input phase with GECAML tags and ends in the message type "`input.integrated`" while the intermediate message types can be inserted according to the system configuration. The final message is matched with the patterns described in the AIML scenario script and triggers a message annotating pipeline in the output phase.

In the output phase, the message annotating pipeline begins with the message type "`output.scenario`" generated by the AIML processing component and ends with type "`output.animation`" which will be executed by the animation player. As the input phase, intermediate message types can be inserted according to system configuration. To prevent the conflicts occur among different output action deciding component, output phase components are assigned priorities, such as scenario actions > spontaneous gestures > reflexive actions, the action execution time is examined by each component and lower priority components do not overwrite actions on the outputs from the higher priority components if there is some overlap in execution time.

Below is a simple example of an AIML category with non-verbal input/output descriptions:

```
<category>
<pattern>What is this
[PointingAt Object="monastery"]
</pattern><template>This is the big
Onofrio's Fountain [Action
Type="pointing" SubType="Null"
Duration="2300" Intensity="0" X="0" Y="0"
Z="0" Direction="rightUp"
ActivationFunction="sinusoidal"/] built
in 15th century. The Fountain is a part of
the town's water supply system which
Onofrio managed to create by bringing the
water from the spring located 20 km away
from town.</template></category>
```

Here, the non-verbal action "`pointing`" of the agent character is described. Its duration is specified by opening tag and closing tags that enclose a segment of an utterance and thus the actual value depends on the TTS (Text-To-Speech) synthesizer if it supports prior phoneme timing output or absolute values in milliseconds. The attribute `SubType` has the value of "Null", as there are no possible subtypes defined for it. The "Intensity" attribute is to have integer values, with "0" value meaning that the intensity is not specified for the action in question. Other actions, for which the attribute "`intensity`" has sense, do have an intensity scale specified. For example, to distinguish between a slight bow used while greeting in European countries from the deep Japanese salutation bow, we introduce a scale of values for the "bow" action.

Further, we envisage the use the coordinates (`"X"`, `"Y"`, `"Z"`) integer-valued attributes in the future. The meaning of these coordinates will be dependent on the action. For example, for the "`pointing`" action such this triad would mean the position on the background screen where the agent is supposed to point. At the moment the alternate attribute "`Direction`" is used.

The "`ActivationFunction`" attribute stands for the dynamics of the action. Possible values are "`linear`", which uses a linear function to activate the corresponding MPEG4 Face and Body Animation parameters (FAPs), "`sinusoidal`", which uses trigonometric functions to activate the FAPs, and "`oscillation`" function , which is used for the repeated actions, such as "`Nodding`" or "`HeadShaking`". In addition to these attributes, the attribute "`sync`" with possible values "`PauseSpeak`", "`BeforeNext`", "`WithNext`" specifies the synchronization between non-verbal actions and speech synthesizer.

The action "`pointing`" is an invocation of one character action with the name "`pointing`" which is stored in a high-level action database. The database is currently implemented as one part of the visage animator and stores low-level MPEG4 FBA parameter specifications as well as run-time configurable parameters for high-level action invocations. Table 1 shows some examples of such high-level character animation actions.

In addition to the character action specification tags, animator controlling tags such as "`Scene`" are also defined. This tag informs the animator to switch scene settings while the scenario advances. Besides, the "`PointingAt`" tag is generated by a component which maps raw coordinate data to object names according to the information provided by motion capture device and scene changing messages.

5. Evaluation

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 real-time 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 de-signed 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.

As the next step, we tested the prototype platform in our proposed project, "An Agent Based Multicultural User Interface in a Customer Service Application" in the eNTERFACE'06 [6] summer workshop on multimodal interfaces, which was held in the summer of 2006. The nature of this workshop is to invite volunteer participants to jointly develop proposed application projects in a four-week period. Although only one participant does not belong to our research group, only two of the six student participants were the core developers of the GECA project and thus most of the team members were supposed not to be familiar with the GECA platform.

The objectives of the proposed project were listed as the follows while its system configuration is shown as Fig. 6:

– To use GECA framework to build a simple but working tour guide of the venue, Dubrovnik city of Croatia
– This tour guide system answers queries from its human users with verbal and non-verbal interactions
– This tour guide in Croatian way or Japanese way according to where the user comes from
– Most of the required software components are going to be built and the protocols used in their communication are going to be specified during the workshop

Due to the lack of Croatian speech recognition and synthesis engines, the system was implemented to use English speech recognition engine to recognize Croatian by using a grammar definition which approximates Croatian words with English pronunciations and prerecorded voice tracks while lip synchronization animations are generated automatically by visage | SDK.

About the cultural difference issue, we observed and analyzed the non-verbal behaviors of Japanese guides in Japan and European tour guides in Dubrovnik. As a result, we tried to implement the features to distinguish their differences in our agent. Also, some very typical Japanese emblem gestures, that are not inherent to the European culture, were implemented. For example, the so-called "HandsCrossed" gesture shown in Fig. 7. In Japanese culture, to show that something is prohibited, people tend to cross their hands before the chest. Sometimes, this action is accompanied with head shaking. Similarly, our agent, uses this gesture when prohibiting in the Japanese mode, in contrast to the Croatian mode, where only head shaking is envisaged.

About the non-verbal input components, because of the transportation issue of equipments, we adopted a small and portable infrared ray camera [15] to capture the user's hand position instead of a magnetic motion capture system that we usually used. In the eNTERFACE system, non-verbal inputs from the user include nodding, head shaking, pointing, waving and raise the hand to stop the agent‘s utterance or get its attention are implemented by using the combined information from motion sensor, infrared ray camera, and data glove.

At last, a Dubrovnik guide agent runs in Japanese and English mode with 5 scenes and 16 AIML QA categories is completed. It is driven by speech recognizer and provides scene changing feature. Although the ambitious planned system could not be completed before the end of the very short four-week period, nearly all of the individual components except the component to distinguish where the user comes from are implemented but just were not put together and tested. After starting the project, we realized that it is difficult to distinguish where the user comes from non-verbal inputs with the very limited hardware configuration which was not a problem of GECA framework itself.

6. Conclusion and Future Works

We do not expect the platform to be fully satisfying in the current preliminary release;

we are also going to complete the definition of GECAML and improve the system base on the new requirements, problem reports and other suggestion gathered in developing the two experimental systems and in the eNTERFACE'06 workshop. As near improvement goals: at first, we would like to move the animation synchronization part from the animator to real-time driving message passing and thus can allow wider range of components. Second, develop the deliberate phase with internal context state instead of the current very simple AIML script executor. As further goals, we are planning to extend the system to support simultaneous multiple sessions and runs even on the Web environment.

References

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

[2] Arafa, Y. and Mamdana, A.: Scripting Embodied Agents Behaviour with CML: Character Markup Language, in *The Proceedings of IUI'03*, 2003.

[3] Cassell, J., Bickmore, T., Billinghurst, M., Campbell, L., Chang, K., Vilhjalmsson, H., and Yan, H.: Embodiment in Conversational Interfaces: Rea, in the *Proceedings of the CHI 99 Conference on Human Factors in Computing Systems,* 1999.

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

[5] Cassell, J., Stocky, T., Bickmore, T., Gao, Y., Nakano, Y., Ryokai, K., Tversky, D., Vaucelle, C., and Vilhjlmsson, H.: MACK: Media lab autonomous conversational kiosk, in the *Proceedings of Imagina 02: Intelligent Autonomous Agents*, 2002.

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

[7] 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.

[8] Humanoid Animation, http://www.hanim.org

[9] Knowledge Query and Manipulation Language, UMBC, http://www.csee.umbc.edu/kqml/

[10] Kshirsagar, S., Vuilleme, A., and Kamyab, K.: Avatar Markup Language. *The Proceedings of 8th Eurographics Workshop on Virtual Environments*, 2002.

[11] MindMakers.org http://www.mindmakers.org

[12] ISO/IEC JTC1/SC29/WG11, ISO/IEC 14496:1999, Coding of Audio, Picture, Multimedia and Hypermedia Information, N3056, 1999.

[13] MPML-VR 2.0. http://www.miv.t.u-tokyo.ac.jp/mpmlvr/

[14] 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.

[15] NaturalPoint OptiTrack Flex 3, http://www.naturalpoint.com/optitrack/

[16] NEC Tokin Corp 3D Motion Sensor, http://www.nec-tokin.com/english/product/3d/index.html

[17] RFC 1305 Network Time Protocol, http://www.ietf.org/rfc/rfc1305.txt

[18] Object Management Group CORBA/IIOP Specification, http://www.omg.org/technology/documents/formal/corba_iiop.htm

[19] OpenAIR 1.0, http://www.mindmakers.org/openair/airPage.jsp

[20] Program D, aitools.org, http://aitools.org/

[21] Thorisson, K., Benko, H., Abramov, D., Arnold, A., Maskey, S., and Vaseekaran, A.: Constructionist Design Methodology for Interactive Intelligences, AAAI *A.I. Magazine*, pp.77-90, Winter 2004.

[22] TVML 2.0. http://www.nhk.or.jp/strl/tvml/index.html

[23] visage|SDK, visage technologies, http://www.visagetechnologies.com/index.html
[24] Virtual Human Markup Language 0.3, http://www.vhml.org/
[25] Web Services Activity, http://www.w3.org/2002/ws/

Table 1. Defined high-level actions

Type	Sub-Type	Inten-sity	Direction
Pointing			Righ-tUp/…
Bow		1-3	
Inviting	HR/JP		
Hands Crossed			
Nodding			
ShakeHead			
Extend-Hand			
Wave			
Expression	Smile/...		
Walk			Left/…
Beat	a-e		
Contrast	a-c		
Warning	HR/JP		

* For all actions, the duration attribute is required

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

Fig. 2. The hierarchy of the high-level protocol, UECAML, notes that reflex action links are shown in blue arrows

Fig. 3. Example messages for a greeting response of the conversational to a human user's greeting behavior

Fig. 4. The campus guide agent's configuration

Fig. 5. An application for experiencing the cross-culture differences of hand gestures

Fig. 6. The dataflow and the system configuration of the Dubrovnik tour guide agent

Fig. 7. An example of the "Negation" gesture in the Japanese mode: waving with a hand while the arm is extended