Toward a Culture Adaptive Conversational Agent with a Modularized Approach

Hung-Hsuan Huang, Toyoaki Nishida
Graduate School of Informatics, Kyoto University, Japan
huang@i.i.i.kyoto-u.ac.jp, nishida@i.kyoto-u.ac.jp

Aleksandra Ćereković, Igor S. Pandžić
Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia
{aleksandra.cerekovic, Igor.Pandzic}@fer.hr

Yukiko Nakano
Department of Computer, Information and Communication Science, Tokyo University of Agriculture & Technology, Japan
nakano@cc.tuat.ac.jp

ABSTRACT
Embodied Conversational Agents (ECAs) are computer generated human-like characters that interact with human users in face-to-face conversations. ECAs are powerful tools in representing differences in cultural aspect and interacting with human users for training and edutainment applications. In this paper we present the preliminary results of our modularized approach by using our Generic ECA Framework to build a culture adaptive virtual tour guide agent for serving Japanese, Croatian and general western users with appropriate display of appropriate verbal and non-verbal behaviors.

ACM Classification: H.1.2 [User/Machine Systems]: Human factors

General terms: Design, Human Factors, Standardization.

Keywords: Embodied Conversational Agent, Blackboard.

INTRODUCTION
The advancement of traffic and the Internet changed the world to be more and more globalized [17] and increase the importance of the communication with people who come from other cultures in modern daily life. The differences among cultures appear not only in languages and their uses, but also in the display from internal emotion to facial expressions, gestures and the range of movements, interpersonal distance and so on [12]. Embodied conversational agents (or computer generated life-like characters) that can provide rich facial expressions and large degree of freedom in their bodies are therefore ideal as the media of the interfaces of the systems involving cultural aspects. There are already some research works applying virtual characters in edutainment applications addressing individual cultural issues, for example, the language training system developed by University of Southern California for US soldiers to communicate with Iraqi citizens to smooth the execution of their missions [13], or an attempt to use virtual peers to encourage African American children to switch their language coding to increase school-based literacy [10]. However, very few of them address the issues about an interface agent which is adaptive to the users come from multiple cultures.

From research reports like [15], people feel more comfortable, more trustworthy and prefer interface agents with the same ethnicities as themselves. Consider an interface agent for users who may come from many cultural areas like a tour guide agent of a sightseeing spot or an explanation guide of a museum, if the agent can show homogeneous behaviors as the user, the knowledge conveyance efficiency should be facilitated and better human-agent communications without cultural disturbance can be expected. Our ongoing work is an attempt on building such an virtual agent based interface. It is started in the period of the eNTERFACE’06 workshop. This international workshop focuses on the topic of multi-modal human computer interface and was held in Dubrovnik, Croatia in the summer of 2006. Unlike usual workshops that researches only present their research results without actual work, the nature of this workshop is to invite volunteer student participants to jointly develop proposed application projects in a relatively short four-week period and then present their research results.

The title of our proposed project was ”An Agent Based Multicultural User Interface in a Customer Service Application [9].” After the announcement of project proposal over sponsored universities, we got six student members in our team where three of them were in our research group. Based on the discussion done bye the team members prior to the workshop, the target application is decided to be a tour guide agent of Dubrovnik city where is specified as a UNESCO World Heritage. It is a famous sightseeing spot and attract thousands of tourists from all over the world especially in summer because the summer festival. Since most of the team members come from Japan and Croatia, it is most convenient to gather first-hand Japanese and Croatian cultural information where the differences are supposed to be fairly obvious.

The general objective of this project is to build an agent who mediates the site seeing information of Dubrovnik to its vis-
itors via verbal and non-verbal interactions. An example scenario is: when a visitor comes to the system, the system recognizes the visitor as a Japanese or Croatian from the combination of the speech recognizer’s result and the non-verbal behaviors of the visitor such as bowing in greeting in Japanese culture. The agent then switches to its Japanese mode, that is, speaks Japanese and behaves in a Japanese way to accept and answer the queries from the visitor. At the same time, the visitor can interact with the agent not only by natural language speaking but also by non-verbal behaviors such as pointing to an object on the background image or raising his (her) hand to indicate that he (she) wants to ask a question.

Although it was impossible to fully achieve the ambitious objective during the limited project period, we still continue to develop it after the workshop. To save save efforts and produce as many results as possible, the system is implemented in our modularized Generic Embodied Conversational Agent Framework (GECA Framework) and all of the system functions are divided into individual functional components.

**GENERAL EMBODIED CONVERSATIONAL AGENT FRAMEWORK**

To realize a believable ECA capable to take out natural face-to-face and multi-modal conversations with humans is not so easy. Knowledge and techniques on signal processing, natural language processing, gesture recognition, artificial intelligence, dialog management, personality and emotion modeling, natural language generation, gesture generation, CG character animation and so on are required.

ECA involves so many research disciplines that it is difficult for individual research teams to develop from the scratch. The usual way to build ECA systems is thus by utilizing software tools developed by other research groups. However, because of software tools developed by different institutes are neither meant to cooperate with each other nor designed for the same application domain, usually it is laborious or even impossible to make them work with each other. More than that, redundant efforts and similar approaches are repeated by the researchers due to their common needs.

To relief these problems, if there was a common framework that absorbs the heterogeneities to connect diverse ECA software tools and drives the connected components as an integral ECA system, redundant efforts and resource uses can be saved. Furthermore, the sharing of research results can be facilitated and the development of ECA systems can become easier.

Like typical modeling of autonomous agent systems, ECAs need to posses the following capabilities:

- Acquire verbal and nonverbal inputs from the human user and the environment
- Interpret the meaning of the inputs and deliberate the responding verbal and nonverbal behaviors
- Output those behaviors with computer graphics characters

To realize these capabilities, various functions like sensor data acquiring, speech recognition, gesture recognition, natural language understanding, believe-desire-intention planning, speech synthesizing, CG animator and so on are required. ECA developers then need to implement these functions by their own or by utilizing available software tools and then integrate them to work as an ECA with certain architecture. Here, we call the modules that handle each individual function as Components of the system. ECA component integration involves various issues while some of them are already mentioned in [7].

The Generic Embodied Conversational Agent (GECA) Framework is being developed by us to address this problem. It has three parts, the integration backbone GECA Platform, communication libraries GECA Plugs, and a high level protocol GECA Protocol. Figure 1 shows the diagram of the GECA Framework’s basic concepts.

**The Basic Architecture of the GECA Framework**

Many excellent individual ECA systems like REA [5] have been proposed, but they are usually designed for specific applications, and their architectures typically feature fixed processing pipelines of functional components and thus can not be easily adapted to other applications. On the other hand, blackboard model is a methodology widely used in distributed and large-scale expert systems. The interdependency among the knowledge sources can be minimized, and thus it is considered suitable for integrating heterogeneous knowledge sources. Considering blackboard’s convenience and generality in integrating various heterogeneous system components, we adopted it as the basic architecture of GECA Platform.

In GECA, multiple shared blackboards are allowed. Components connecting to those blackboards share data with subscribe-publish message passing mechanism, that is, every message has a message type, when a message is published to a blackboard, it is then forwarded to the components which subscribed the corresponding message type. A component process the message it receives and publishes its contribution to the blackboards. To reduce the overhead of message forwarding, direct communication between components is allowed, too. Every blackboard has its own manager, and there is a server that provides message subscription and naming services for the whole system.

There are various ways available for implementing distributed systems, but most of them suffers some drawbacks and are not appropriate in the ECA context. For example, KQML does not provide explicit temporal control, CORBA, Web Service, and remote procedure invocation technologies do not support two-way data passing. Therefore, a simple and light traffic weight protocol, OpenAIR [3] is adopted as the low-level routing protocol for the communication among components, GECA server and blackboards.

OpenAIR is a specification of XML message passing for distributed systems in a TCP/IP network. We considered that is suitable for real-time interactive system because its message format is very simple and it has some features like explicit timestamps. A reference Java implementation of its library called Plug is freely published, too. The second part provided in the GECA Framework is so called GECA Plug libraries. They are extended OpenAIR Plug with GECA origi-
inal classes and functions. Currently C#, C++ versions have been developed while the Java version is modified from the reference implementation. The purpose of the GECA Plugs is to absorb the differences caused by operation systems and programming languages and to make system development easier. By utilizing GECA Plugs, an ECA developer only needs to implement a small wrapper for an existing software tool; then it can be plugged into the framework and cooperates with the other components.

**GECA Protocol**

Based on the low-level communication platform of GECA Framework, GECA Protocol (GECAP) is an XML based high-level communication protocol for the components. In GECAP, every message has a type, for example, "input.action.speech" for a speech recognition result, "output.action.speech" for a text to be synthesized by a Text-To-Speech (TTS) engine, etc. Each message type has a specified set of elements and attributes, for example, "Intensity", "Duration", "Delay", etc. All data is represented as text and transferred by OpenAIR on the GECA platform. All data is sent in the form of plain text via OpenAIR’s content slot. GECAPlug is a specification of message format style and a set of core message types, the syntax is not fixed and can be easily extended to meet the demands of individual applications.

Considering the information flow from the human user’s inputs to the agent’s responses and the system needs, GECAP message types can be divided into three categories: input phase, output phase, and system messages. Input and output messages can be further categorized into three layers, raw parameter, primitive action, and semantic interpretation in the sense of abstractness.

**Input Phase Messages**

The task of the components which generate input message types is to acquire and to interpret human users’ inputs from verbal and non-verbal channels. The followings are some examples of defined input message types where "input.action." types transfer primitive actions and "input.raw." types transfer raw parameters. Speech recognition result "input.action.speech", head movements such as nodding and shaking that can be detected by an acceleration sensor "input.action.head", gaze direction that can be approximated by a head tracker "input.action.gaze", hand shapes acquired by data glove devices "input.raw.hand", the angles of the arm joints that can be approximated by three motion capture sensors attached on each arm "input.raw.arm", predefined hand gestures which is recognized by motion capturing devices "input.action.gesture", convenient pointing gesture which can be detected by a motion capturer or even a mouse "input.action.point".

The following is an example of an input.action.speech type message. This message type also utilizes the language attribute of content slot of OpenAIR to store the recognized natural language with values like "English."

```
<Action Begin="117508369171" Duration="500" Weight="1.0">
  <Hypothesis Confidence="0.9">
    <Speech>what is this</Speech>
  </Hypothesis>
  <Hypothesis Confidence="0.1">
    <Speech>what is these</Speech>
  </Hypothesis>
</Action>
```

The recognized result is stored in the Speech element. Programs like speech recognizer or gesture recognizer usually have ambiguity in recognizing the data from real world sensors. The Hypothesis elements are used to present a list of hypotheses of the recognition result on a single input event with confidence ratings in values from 0 to 1. Begin attribute stores when this input event begins with the absolute time represented in milliseconds while Duration attribute stores how long the input event lasted.

**Output Phase Messages**

The only actuator of software based ECAs is the character animation player. This player plays plain text with TTS and drive the CG character to move in the virtual environment when a command message arrives in real-time. Although current prototype GECA player is implemented by using commercial software, visage|SDK [4], the design of GECA’s output message format is not dedicated to Visage and should be able to be ported to other animation systems. All parts of the full 3D anthropomorphic character...
like the limbs, fingers, eyes, mouth and so on can be animated to perform arbitrary actions that are possible for a real human. The animation player also provides the support of MS SAPI compatible TTS engines for the character’s speech. To simplify the problem and also because a picture usually looks more realistic than a full 3D environment which lacks enough details, the virtual environment for the agent’s activities is represented by switching 2D background pictures.

**System Message Types** There are system controlling message types such as `system.status.player` or `system.control.player` to query the status of the ECA character (e.g. whether the character is speaking something) or make the character to stop speaking and playing any animation, etc.

**GECA Scenario Mark-up Language**
To achieve really natural conversation between the ECA and a human user, many factors need to be considered in the deliberate process of an ECA: natural language understanding, inference engine, knowledge representation, dialogue management, personality and emotion model, social role model, natural language generation and so on are required. Considering the complexity and the fact that the present level of technology is still impossible to drive an ECA to behave like a human in an indistinguishable level, instead of a block of complex deliberate process, we have defined a script language, GECA Scenario Mark-up Language (GSML) that defines the interactions between the user and the agent. A script definable ECA is less general than a deliberative process, but it will be much easier to create contents and should be useful enough for simpler ECA interface applications.

The GECA Scenario Markup Language (GSML) shares the most basic concept of AIML [1] which is a widely used script language for defining text based chatbot agents on the Web. An AIML script represents an agent’s knowledge that is composed by a set of `Category` elements. One `Category` contains a pair of `Pattern` and `Template` that describes one of the possible conversations between a chatbot and its human user. When there is a user's utterance comes into the interpreter, that utterance is matched with all of the defined patterns. The agent then responds with the utterance described in the corresponding `Template` element. However, AIML can not be applied to the ECA context due to the following reasons: supports English only, unexpected template may be triggered because the same patterns can not be distinguished in different circumstances, can not describe non-verbal behaviours of neither human user nor agent, no way to specify objects in the virtual world, agent behaviors need to be triggered from the human side.

GSML extends AIML's syntaxes to cover more complex situations in ECA-human conversations. Extend to AIML's one-layer categories; GSML represents the human-ECA conversations as states and the transitions among them. In GSML, one `Scenario` defines an interactive scenario between the ECA and the human user. A scenario can contain one or more `Scene` elements while each `Scene` means a physical location in the virtual world and is coupled with a background image. In an individual, there may be one or more conversational `State` elements. Each `State` contains one or more `Category` elements. The conversational states are linked by `Transition` specifications described in `Template` elements. Further, templates can be triggered right away when conversational state transition occurs without user inputs. The Scenario-Scene-State-Category hierarchy narrows the range of possible categories into a conversational state and prevents the problem that templates may be triggered unexpectedly in AIML agent which practically has only one conversational state. Besides, the Language attribute in states allows a multi-lingual ECA to be defined in a single GSML script.

GSML's patterns and templates do not only present verbal utterance of the agent but are also extended to describe non-verbal behaviors of the agent and the human user. `Action` tags that specify face or body animations can be inserted into the utterances of the agent, the timing information is specified by the position of the `Action` tags in the utterance texts. The action tags (`Speech`, `Point`, etc) can be inserted inside the `Pattern` tags then the corresponding template will be triggered if the user does that non-verbal behavior. Further, areas of the background image can be named by `Object` elements and can be referenced (e.g. pointed at or gazed at) by the user during the multi-modal conversation.

By observing usual face-to-face communications between humans, we can find non-verbal behaviors are the indispensable counterpart of verbal utterances. For example, the verbal utterance “What is this?” with a pointing gesture is a very typical example. Without the pointing gesture, which object that this “this” is mentioning becomes ambiguous. On the other hand, a pointing gesture can not fully convey the user’s intention, either. Generally, the order, combination, and occurrence of multi-modal perceptions and their relationship are difficult to be described and identified. Like the discussion in the specification of W3C’s multi-modal interface description language for Web browsing, EMMA [2], it is not appropriate to propose a general algorithm for multimodality fusion. In GSML and its interpreter (the scenario component), we adopted a simplified description for multi-modal perception of the ECA and a relatively simple mechanism to solve reference ambiguities.

Set `element` means a non-ordered set of multiple verbal or non-verbal perceptions and every one of them must be fulfilled. OneOf element means at least one of the multi-modal perceptions needs to be fulfilled. `Sequence` means the multi-modal perceptions need to be performed by the human in the specified order. The three specifiers can be further nested with each other. Whether two multi-modal perceptions occur concurrently is judged by the period coverage of involved perceptions according to the `Begin` and `Duration` attributes in the message sent from the sensor data acquiring components. The `Scenario` component keeps a current status of the multi-modal perceptions and triggers the corresponding `Template` if any one of the available patterns defined in the current conversational state can be exactly matched. This matching is calculated every time when a new input message arrives. The combination which has highest value of the sum of the product of confidence and component weight is chosen in the matching. The following is an example code segment...
describing the interaction between the human user and a tour guide agent at the entrance of the Dubrovnik old town.

```xml
<Scene ID="Entrance" InitialState="Greet" X="1250" Y="937"/>
<Objects><Object ID="Fountain" X="900" Y="0" Width="350" Height="937"/>
<Object ID="Monastery" X="0" Y="0" Width="377" Height="937"/>
</Objects>
<State ID="Greet" Language="English">
<Category><Pattern>
  <Speech>Hello</Speech></Pattern>
  <Transition Scene="Fountain">
    <OneOf>
      <Action Type="pointing" Duration="1000" Direction="right">
        The fountain
      </Action>
      <Action Type="pointing" Duration="1000" Direction="left">
        the monastery?
      </Action>
    </OneOf>
  </Transition>
</Category>
<Category>
  <Pattern></Pattern>
  <OneOf>
    <Speech>fountain</Speech>
    <Set><Speech>I want to go there
      </Speech>
      <Point Object="Fountain"/>
    </Set>
  </OneOf>
  <Template>Please follow me here.
  </Template>
</Category......
```

The fore part of this code specifies the scene with a background image that can be identified by the scene id, "Entrance." The Object elements specify two areas of the background image, "Fountain" and "Monastery." These areas are used to in the matching of the coordinates sent from some pointing component with the Object specifiers in second Category. According to the description of perception specifiers, when either one of the two circumstances is fulfilled, a conversational state transition to the initial state of the scene, "Fountain" will be triggered. When the human user says "fountain," or when the user says, "I want to go there" while performing a pointing gesture on the screen where the position is recognized as an X value from 0.72 to 1.0 and a Y value from 0 to 1.0 at the same time.

The Action elements are the specifiers of non-verbal animations of the ECA character. The timing to start to play the specified animation is determined by the position of the opening tag relative to the verbal utterance. In the case where the agent will not say anything, a Delay attribute is used to specify when the animation will be played relative to the beginning of the template. The playing of this animation will end when the agent speaks to the closing tag of Action element or meets the time specified by the Duration attribute. Subtype specifies another action in the same category if available. Intensity specifies the strength if specifiable. X, Y, and Z specify a position in the virtual world if the action has a destination, e.g. walking, pointing, gazing actions. Direction specifies a direction of the action if available. Trajectory specifies the temporal function to change parameter values in playing the animation. Linear, Sinusoidal, and Oscillation are currently available values. Sync attribute specifies the temporal relationship between the actions in an utterance. There are three possible values: "WithNext," "BeforeNext," and "PauseSpeaking" stands for do not wait for this action, to wait for this action to end and to pause TTS while executing this action respectively. A template is transferred as an Utterance element in GECAP, the contents of it is broken into phrases and sentences as described in section.

A special action type created is the PlayTrack action, this action plays a background music track, voice track, or a pre-defined animation track. It can be used to implement an ECA system in a language which has no available TTS engines. For example, an agent speaking Croatian can be implemented with pre-recorded human voice tracks and lip actions. The Delay attribute can be utilized in this case to synchronize the tracks with each other. GSML (and output phase of GECAP) provides the distinguishing features include word-level precisely aligned non-verbal behaviors to speech channel and multi-language support.

BUILDING THE DUBROVNIK GUIDE AGENT
Handling cultural issues is very relevant to emotional part, or the deliberate phase of the agent [6]. However, in a four-week project, it was infeasible to explore those issues in enough depth, the implementation needed to be done as simple as possible. Also because the nature of a tour guide system, large freedom of user-agent interactions not required. The decision was made quickly that to build the system with as fewer new component as possible and reuse more existing component as possible in a modularized way.

Software Components
The component and hardware configuration of the Dubrovnik tour guide system is shown in Figure 2 and Figure 3 respectively. Sensor Data Acquiring Components. The non-verbal behaviors of the users are recognized by using the data from data gloves, motion capture, head tracker and acceleration sensor. These components acquire the raw data from the sensor devices, interpret them with text strings and send the results to other components for further processing. The configuration of these hardware devices is shown in figure figure:dubrovnik. Speech Recognition Components. These component are wrapped MS SAPI compliant recognition engines which recognizes Japanese or English spoken by the visitors by matching predefined grammar rules. Because the lack of a good enough speech recognizer for Croatian, it is recognized by an English speech recognizer with the grammar rules which will be explained later. Sc-
The multimodal tour guide agent, as shown in Figure 2, is equipped with various sensors and devices to capture and track user actions and behaviors. The components include IR Camera, Data Glove, Microphone, Acceleration Sensor, IR Reflexive Strap, Head Tracker, and Magnetic Motion Capturer. This agent is designed to identify and interpret gestures, head movements, and speech inputs.

**Nonverbal User Inputs**

In the non-verbal input recognition issue, to simplify the problem, we are not recognizing culture specific non-verbal behaviors from the user but merely the following general ones at this moment, point to the interested objects shown on the display or shake head and nod to express positive and negative answers. These behaviors are recognized by combining the data from the sensor devices. For example, pointing gesture is recognized by a pointing shape from the data glove and the pointed positions on the display from the coordinate values of motion capture.

**The Action Animations**

To achieve more natural communication, culture specific non-verbal behaviors are very essential. Since our target is a tour guide agent who serves with the users come from Japan, Croatian or somewhere in the western cultural area, the first thing is to gather culture specific behaviors in the tour guiding context, especially the culturally “coded” emblem gestures. The material is obtained by taking video data of Japanese tour guides at several famous sightseeing spots in Kyoto and those of one of European tour guides in Dubrovnik. Appropriate non-verbal behaviors of the agent are determined by observing the video data. While selecting styles and gestures for the character, we aimed to express diversities of each culture; for example we insisted on the so-called "handsCrossed” gesture in Japanese culture. This gesture seems to be pretty unique, and normally draws attention of Western people who first come to Japan and are used on the head shaking to express prohibition. In Japanese culture people tend to cross their hands before the chest (Figure 4). Another example is the “Negation” gesture (Figure 4) in the Japanese mode: waving with a hand while the arm is extended. In Japan, the negation is expressed by shaking one’s upright hand near one’s mouth with two thumbs closer to one’s face. Sometimes shaking head sideways is also added. When asking to wait, Japanese people usually show the palm of one hand to another person. At times, both hands could be used. Despite the Japanese gestures are significantly different to the western ones, we could not find obvious differences among the western tour guides even they come from different countries of Europe.

**Croatian Speech Input/Output**

Although Croatian is spoken by around 5 million people, commercial speech and language community has not yet produced general-purpose recognizers, synthesizers and translation engines for it. When observing assemblies of ECA systems we found 2 essential components are missing for Croatian; component that can gather and recognize Croatian speech in input and component to generate speech of an agent as output. The following are alternative solutions we propose to replace this non-existence.
**Table 1: The difference of gesture displays in each culture.**

<table>
<thead>
<tr>
<th>Action</th>
<th>Culture Dependency</th>
<th>Croatian</th>
<th>Japanese</th>
<th>Western</th>
</tr>
</thead>
<tbody>
<tr>
<td>bow</td>
<td>In this gesture, we present three types of bowing: shallow bow, using only head; deeper bow (Japanese style) and polite bow that shows respect to the listener</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>invite</td>
<td>Croatian gesture presents waving upwards and backwards with one hand</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>handsCrossed</td>
<td>This is an emblem Japanese gesture, meaning that something is not allowed. The hands are crossed in front of the lower part of the chest</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>extend</td>
<td>This gesture means right arm extended with the palm open and oriented upwards. In the Japanese culture means &quot;wait please&quot;</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>wave</td>
<td>This gesture presents oscillating right hand waving. Used in combination with the &quot;extend” action as part of the Japanese gesture meaning &quot;No&quot;.</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

**Croatian Speech Recognition**  On the field of speech recognition for Croatian, some research has been done, but neither of it has produced general-purpose recognizers. Ipsic et al. [11][16] developed a bilingual database of a Slovenian and Croatian weather forecasts and perform speech recognition experiments. Their recognition results for the two languages are very similar and in future they plan to perform bilingual speech recognition system simulation. Still, Croatian speech recognition engine is currently not available neither to research community neither to industry. Therefore it has been decided to modify an English speech recognition software component to recognize Croatian speech. Within the system, classification of user’s utterance is done with a limited domain of specific keywords that user speaks and that trigger scenario component. The pronunciation of Croatian keywords in scenario is approximated by using English alphabet. Since some Croatian words from scenario were impossible to write with an English alphabet and scenario was not that strict, we avoided them and used some other words instead. During the approximation we performed experiments with different Croatian speakers. For trained speaker’s profile, word recognition results were satisfactory. However, if grammar contained similar words, those words were sometimes mixed by the recognizer, so it was better to choose words that are not so similar. For example, pronunciation of a Croatian word “da” (eng. “yes”) is approximated with an English alphabet as “ddhaa”. Although speech recognizer works well with recognition of word “da” in Croatian, it often mixes that word with words that contain syllable “da”, like “slobodan” (eng. “free”). Therefore, it is better not to use short words e.g. “da” or “dan” (eng. day) that can appear in longer words. In the end, we had to make changes of the original English scenario by defining new keywords that have following conditions:

- Can be approximated by using English alphabet
- Keywords are not short and have no cross-similarity (means keywords do not contain syllable of other keywords)

Because only 5 scenes exist in our current system, transitions between the scenes and between the states in each scene do not require many key words from an input. In English and Japanese scenario we used 8 words for transition and in Croatian 7 of them. In one scene, we use automatic scene transition. Later, while improving the system, we intend to increase the number of scenes and therefore it would be necessary to perform new experiments and define new Croatian words. To define it we can use grammar rules we created or create new one, depending on the syllables that words contain. Also, we are interested in how Croatian speech recognition by using engine for English will behave when number of words increases. We assume that percentage of recognition for each word will fall when we define new words, but we are interested in empirical results. Also, because of limited number of the words that can be defined for recognition, we are considering to research alternative solutions of this problem, e.g. instead of speech, to use classical mouse, keyboard or touch-screen as the system input.

**Croatian Speech Output**  In our system implementation, English and Japanese words that ECA speaks are generated by English and Japanese Text-To-Speech (TTS) engines. Concerning Croatian, we could not find a Croatian TTS with satisfying quality. Our goal is to have conceivable Croatian tour guide agent and since we have an automatic Lip Sync system [18] at our disposal, we chose to use a real person’s recorded voice instead. Our automatic lip sync system Lip Sync introduces a method for mapping a natural speech to the lip shape animation in the real time. The speech signal is derived from a type of cepstral representation of the audio clip, MFCC vectors, and classified into viseme classes using neural networks. The topology of neural networks is manually configured after series of experiments. When viseme is being calculated, it is send to CG Character animation player, Visage player that uses MPEG-4 FBA International Standard for animation. Once we had Croatian scenario, a native Croatian speaker has recorded speech the agent was supposed to say in certain situation in the noise free room. By applying a lip sync application we have created animations from the prepared speech files and made a database containing pairs of speech-animation files for every situation according to the scenario. Integration included a modification of a script language GSML to describe action animations of non MS SAPI5 TTS engines and adoption of a CG Character player for playing those actions. Although solution with lip-sync animations is not flexible because database extension depends on the native Croatian speaker, the advantage is...
natural approach to human-computer interaction because of the quality of a human speech is far better than any synthetic voice.

DISCUSSION AND CONCLUSION

ECAs are very useful tools in representing cultural differences in training and edutainment applications. In this paper, we presented the preliminary results of the development of our culture adaptive tour guide agent system that is implemented in a very modularized way with our GECA Framework. Although both the tour guide agent and GECA itself are still in their relatively early stage of development, this very loosely modularized approach can have two major benefits in handling cultural issues. First, research group distributed to several countries can develop their own cultural modules and combine the final system easier. Second, once built components can be reused in other systems easier.

On the other hand, this work focuses on the rapid building of ECAs and only features the surface traits of cultures, that is, languages and emblem gestures. A more principle research based on the theories of inter-culture communications is necessary in the future. For example, in the case of an interface ECA serving Japanese and western users, the high-context / low-context issue [8] should cause obvious differences in the utterances of the agent and can not be simply done by one-one mapping in the current system.

By scripting methodology, the intractability will be very limited and the quality of the whole system heavily depends on the knowledge and skill of the system developers. At this moment, we only showed the feasibility of our modularized approach. Apparently, this is not yet a sound solution, but we would like to further develop the mental part of the agent and culture modules to affect its outputs with culturally specific differences and to explore the high level aspects of culture issues like the use of silence, intonation, or the choice of words and so on in the future.

REFERENCES


