Applied Artificial Intelligence, 27:551–574, 2013

ISSN: 0883-9514 print/1087-6545 online
DOI: 10.1080/08839514.2013.813167

Taylor & Francis
Taylor & Francis Group

A NEW DISTRIBUTED PLATFORM FOR CLIENT-SIDE FUSION OF WEB APPLICATIONS AND NATURAL MODALITIES—A MULTIMODAL WEB PLATFORM

Izidor Mlakar[1] and Matej Rojc[2]
[1]*Roboti c.s. d.o.o., Maribor, Slovenia*
[2]*University of Maribor, Faculty of Electrical Engineering and Computer Science, Maribor, Slovenia*

□ *Web-based solutions and interfaces should be easy, more intuitive, and should also adapt to the natural and cognitive information processing and presentation capabilities of humans. Today, human-controlled multimodal systems with multimodal interfaces are possible. They allow for a more natural and more advanced exchange of information between man and machine. The fusion of web-based solutions with natural modalities is therefore an effective solution for users who would like to access services and web content in a more natural way. This article presents a novel multimodal web platform (MWP) that enables flexible migration from traditionally closed and purpose-oriented multimodal systems to the wider scope offered by web applications. The MWP helps to overcome problems of interoperability, compatibility, and integration that usually accompany migrations from standard (task-oriented) applications to web-based solutions and multiservice networks, thus enabling the enrichment of general web-based user interfaces with several advanced natural modalities in order to communicate and exchange information. The MWP is a system in which all modules are embedded within generic network-based architecture. When using it, the fusion of user front ends with new modalities requires as little intervention to the code of the web application as possible. The fusion is implemented within user front ends and retains the web-application code and its functionalities intact.*

INTRODUCTION

Today, it is more and more important that systems allow for transparent and natural human–machine interaction. Systems must be intelligent, more natural, and easier to use. Therefore, a great deal of research is devoted to the development of systems and applications that facilitate a natural interaction between man and machine and systems that allow a user to use media devices evoking natural modalities (e.g., eye, gaze, speech, gesture, etc.), while

Operation financed in part by the European Union, European Social Fund.
Address correspondence to Izidor Mlakar, Tržaška cesta 23, 2000 Maribor, Slovenia. E-mail: izidor.mlakar@uni-mb.si

communicating or exchanging information with the machine. The term "modality" represents the mode of interaction (data input and output) between the user and a machine. Within current computing environments, the human–machine interaction is still limited to the use of, for example, the mouse, touch pad, keyboard, and screen. Multimodality then refers to natural communicative channels that enrich the communication bandwidth between human and machine. Therefore, the invocation of multimodalities, also based on embodied conversational agents, allows for a more flexible interaction between user and machine. Several software frameworks support the development of embodied conversational agents (ECAs), for instance, the Articulated Communicator Engine (ACE; Kopp and Wachsmuth 2004), the Behavior Expression Animation Toolkit (BEAT; Cassell, Vilhjálmsson, and Bickmore 2001), and others. Nevertheless, they are, for the most part, desktop oriented and require many system resources. These systems are also, in general, indirectly compatible with web technologies or with different end-user devices. Because data published on the web is becoming increasingly important, the user accessibility of such data is becoming a crucial factor. Accessing the web data should be easy, more intuitive, and should also adapt to the natural and cognitive information processing and presentation capabilities of humans (Cimiano and Kopp 2010). In addition, more and more end-user devices support web browsers and are able to connect to different networks. By exploiting these devices in combination with the importance of web-based solutions, the use of multimodal human–machine interaction (natural modalities) can be extended even further. Using natural modalities can represent an effective way to accomplish a user's computing tasks. Suitable network-based architecture can also enable the user to have good communication and interaction with the machine on various devices and networks (from anywhere), with the same quality. In light of the above facts, the issue arises of how to integrate/fuse technologies that process/generate natural modalities with web-based solutions. A key problem in web development is that often the client will have difficulty articulating requirements at the commencement of a project, particularly prior to any indication of possible solutions. They are often able to express their requirements only in terms of a solution, and are able to formulate those requirements in any detail only as the solution is developed.

The motivation of this work is to fuse these natural modalities (in general, web-incompatible technologies) and web applications. In other words, to develop a platform for the efficient and flexible integration of natural modalities, including embodied conversational agents, with general web applications and web services, as follows:

a. to present web-based information using different humanlike multimodal channels, simultaneously conveying web-based information as multimodal

output (text, images, speech, gestures, facial expressions, emotions, and body motion) using several natural communication channels;
b. to use an ECA as a mediator of web-based information in order to improve user experience when interacting with the machine and to minimize modality mismatches (e.g., presenting the information in such a modality that the user cannot understand it);
c. to be able to integrate modality-dependent systems into any web-based user interface, using minimal integration effort.

In light of these motivations and the fact that such multimodal user interfaces are, in the Web of Data, yet to be developed (Cimiano and Kopp 2010), we propose a platform that can integrate several modality systems (natural modalities such as speech and ECA) into new or already-existing web applications, thus reducing the effort when developing or redeveloping new modules, plug-ins, and interfaces within web applications. Developing web applications with personalized multimodal interfaces including ECAs is a multidisciplinary effort, joining web technology, animation technologies, artificial intelligence, computer graphic technologies, informatics, and social sciences. All these disciplines together can provide a more humanlike experience to today's web applications. When using ECAs as part of a web application, the minimal required modules are a text-to-speech (TTS) system and an animation engine, with both engines running in real time. The proposed Multimodal Web Platform (MWP) is capable of embedding ECAs and other natural modality engines into web applications, both flexibly and effectively.

This article is structured as follows: first, it addresses the related works, followed by a detailed presentation of the proposed mechanism. Then the integration process, based on an example, is discussed. In "Results," a practical implementation is given for using the proposed MWP with an existing web application. The example in that section also discusses two examples of multimodal web services derived from the fusion of web technologies and natural modalities. This article concludes with a discussion and an outline of our future steps.

RELATED RESEARCH

Modality-dependent technologies, such as TTS synthesis and automatic speech recognition (ASR), have already been integrated into different web-based user interfaces. mTALK (Johnston, Di Fabbrizio, and Urbanek 2011), for instance, is a multimodal browser that enables the development of mobile web-based multimodal interfaces combining natural modalities such as speech, touch, and gesture. The mTALK-based approach to

integration is based on authoring. Instead of integrating the natural modalities into existing web-based applications (e.g., as enhancement of the content), the authors suggest the development of a new web-based browser that would integrate the web content into a multimodal environment. For the purpose of integration, the authors present the mTALK browser, implemented by extending the open-source web browser engine and the custom JavaScript bindings. If we wish a web application to become multimodal, we have to adapt it to mTALK's general architecture and build several resources (Johnston, Di Fabbrizio, and Urbanek 2011), such as the user interface (UI) of an mTALK application, client-based application logic written in JavaScript, and so forth. The speech-centric natural modalities are supported by the AT&T speech mashup (Di Fabbrizio, Okken, and Wilpon 2009). The AT&T speech mashup is a web-based service that implements speech technologies, including both ASR and TTS synthesis, for web applications. This system suggests a three-layered architecture using a speech mashup server (hosting speech-centric natural modalities), a speech mashup client (for relaying/accepting data on the client device), and an application server (hosting the back end, performing data aggregation and processing, hosting application logic). In Zaguia and colleagues (2010), a multimodal fusion is suggested for accessing web services. This approach consists of modules that detect the modalities involved, take into account each modality's parameters, and perform fusion of the modalities in order to obtain the corresponding action that is to be undertaken. Similarly as in Johnston, Di Fabbrizio, and Urbanek (2011), the authors assume a network-based structure for natural modalities' processing and a specifically designed UI. This UI performs modality fusion and interacts with the rest of the system. The system also includes processes for the XML-based modality-description parsing, Extensible MultiModal Annotation markup language (EMMA; Johnston 2009) parsing, and others. Similarly as in mTALK, Zaguia and colleagues (2010) suggest that an application needs to be redeveloped, or developed under certain architecture, if the users are to use natural modalities in their UIs. Polymenakos and Soldatos (2005) present the main design issues associated with multimodal-based web applications and a unifying framework for the end-to-end design and implementation of components that support multimodal-based web applications. The presented solution is also compared with state-of-the-art initiatives such as X + V[1] and speech application language tags (SALT[2]). Embodied conversational agents are also increasingly used in various web-related contexts, for example, teaching, socialization, chats, and branding websites (Diesbach and Midgley 2008). Such agents have already been used in web applications such as question-answering systems (Bickmore, Pfeifer, and Paasche-Orlow 2007; Theune et al. 2007; Goh et al. 2006), learning communicative agents (Xu et al. 2010; Yuan and Chee 2005), or as communicative agents in

different online networks (Repenning and Sullivan 2003; Jan et al. 2009; Paraiso and Tacla 2009). Further examples of such applications are as follows: pedagogical agents (Graesser et al. 2005), health counseling agents (Bickmore, Pfeifer, and Paasche-Orlow 2007), direction-giving agents (Cassell et al. 2002), and so on. In contrast to the web-based speech-centric interfaces (mostly targeting mobile devices), web-based interfaces supporting embodied conversational agents address dialogue generation rather than the integration of these virtual agents into different web-based interfaces. Kimihito (2005) proposes a framework in which end users can instantaneously modify existing web applications by introducing multimodal UIs. Their idea is based on multimodal presentation markup language (MPML; Prendinger et al. 2004) and IntelligentPad architecture (Tanaka 2003). The IntelligentPad architecture is employed for the re-usage of existing web applications' functions. MPML gives users multimodal presentations with character agents. The multimodal UI presents an external application that performs conversion between HTML and MPML. ShockWave Flash and Flash-based agents, such as DTask and Lite Body (Bickmore, Schulman, and Shaw 2009), can also be found on several webpages. As a downside, such agents usually provide only a tiny subset of the ECA's core functionalities today (e.g., the ECA talks to the user only with its head, while displaying a limited range of nonverbal conversational behavior). The main commonality of the related work is to use different (primarily network-based) architectures that host technologies for the generating/processing of natural modalities. Specially developed (nongeneral) web-dependent interfaces then relay data to the UI by using special web browsers, authored web-application servers, or different client-side application programming interfaces (APIs) (or even nonweb-based desktop/mobile applications). The current state of the art, in both mobile and desktop web-based environments also shows the following tendencies:

a. to redevelop (or develop new) web-based applications/services and author them to the selected multimodal architecture. Data between web and nonweb processes is relayed through a newly designed application core and different client-side scripts;
b. to develop rendering applications (e.g., multimodal web browsers) that will render the web content and transmute data between web and nonweb cores;
c. to minimize the integration/redevelopment effort by minifying the modality-dependent services (natural modalities).

The research work presented in this article supports the idea of network-based processing (off-device) regarding natural modalities, and the usage of web-application servers as meditators between web and

nonweb content. However, in our approach, we have integrated natural modalities into the existing web applications as an additional enhancement, rather than the main functionality (e.g., integration of web applications in a multimodal core). The presented approach is also compatible with most web-browsing engines (e.g., Opera, FireFox, Internet Explorer, etc.—no ActiveX or browser specific controllers are used). The work presented in this article can, therefore, be easily adapted for usage on any broadband-supported device. The code and logic of the web application remains unchanged. The fusion of "natural" modalities with web content is implemented through a client interface and a dialogue manager. The "fused" client interface is implemented by using a device-/context-independent Java API (relying on nontext data, e.g., speech transferred to the engines for processing natural modalities), and context-dependent JavaScript APIs. The dialogue manager used can be a simple set of rules or a complex service that automatically models user context and services (Cena 2011; Thang, Dimitrova, and Djemame 2007). Such managers can be flexibly integrated into a cluster of engines for processing/generating natural modalities. The mediation between web and nonweb services, presented in our work, is implemented based on a general web application and can be applied virtually to any web application by fusing any set of natural modalities hosted by a network-based architecture. In order to minimize the integration-process effort, and to allow several modalities to be integrated into any web application, we propose an MWP with the following set of features:

- It enables the usage of task-oriented technologies for processing/generating natural modalities within a broader scope (e.g., usage of the visual speech synthesis of an unknown text can easily be extended to any type/context of human–machine interface (HMI) interaction, even to web applications).
- It inherits distributive principles and, by using cross-domain interaction, extends the scope of traditional web applications with technologies traditionally foreign to the web domain (e.g., TTS synthesis, ECAs, ASR, etc.).
- It provides an efficient process for the client-side integration of advanced HMIs with any existing web application. This platform adds no extra load to the web-application server, and no extensive redesigning of the web application is required.
- It provides client-side integration, based on IFRAME encapsulation, and JavaScript cross-domain communication.
- There is a "universal" web-application server hosting virtual interfaces for client-side integration and mediating between the cluster of engines for processing/generating natural modalities and the web-based UI.

MULTIMODAL WEB PLATFORM

The overall architecture of the proposed MWP for fusing web- and nonweb-based technologies into advanced multimodal-based web interfaces is presented in Figure 1. It consists of the following modules:

- EVA Plug-in module: dedicated to web-based information storage and presentation (e.g., stand-alone existing web page or web application).
- MWP's core: a Tomcat-based[3] web application that serves as an intermediate layer capable of interfacing with both web- and nonweb-based layers and used in mediation between web content and nonweb technologies, configuration, etc.
- DATA-Subsystem: a service-oriented cluster dedicated to the generating/processing of natural modalities such as: TTS synthesis, ASR, and ECA animation engines.

Figure 1 presents the architecture of the MWP. This platform fuses UIs of web-based systems (e.g., web application's front end) with engines for generation/processing of natural modalities (DATA-Subsystem). The presented platform suggests a mediating platform that is able to interact with the web-based UI and at the same time with several engines within the DATA-Subsystem. The web layer of the MWP architecture incorporates an existing (MWP-independent) web application, its UI, and a web-based EVA plug-in. The EVA plug-in is a small API that enables integration on the client side and data exchange between the web-based UI and the MWP. The layer of natural modalities' engines is represented by the cluster of engines running within the DATA-Subsystem. The cluster links a TTS server, an ECA animation server, and possibly other engines for the

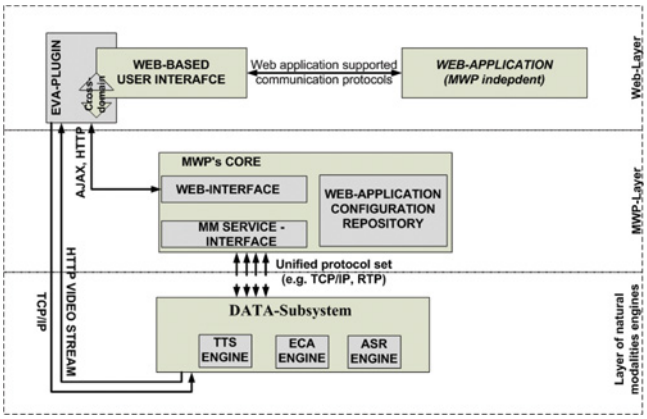

FIGURE 1 The proposed MWP architecture. (Color figure available online.)

generation/processing of natural modalities (e.g., dialogue management, communicative-behavior generation engines, ASR, gesture recognizers, etc.). The MWP layer of the discussed architecture integrates the middleware of the MWP, the MWP's application core. The different layers communicate among each other using standard protocols, such as HTTP and Ajax 2, used within web applications, and TCP/IP and real-time transport protocoal (RTP), used for communication between the MWP and the DATA-Subsystem cluster.

The DATA-Subsystem

The DATA-Subsystem is dedicated to providing natural modalities, such as verbalization of information using ECAs, TTS synthesis, and other possible instances of natural communicative behavior techniques (e.g., communication management, speech recognition and understanding, gesture recognition and understanding, etc.), to general web applications. This cluster represents an abstraction of both input- and output-based modalities. Each server in the cluster is independent but uses common input-/output-based protocols. As presented in Figure 1, the different engines for processing/generating natural modalities are connected locally within a cluster named the DATA-Subsystem. This DATA-Subsystem is based on distributed network architecture, and is implemented using a proprietary Java framework, named the DATA framework (Rojc and Mlakar 2009). This framework is a set of several Java packages that are used for handling TCP/IP connections, the creation and management of common proprietary protocols over TCP/IP connections, interfacing Java with a native code (C/C++), creation, compilation, validation and management of finite-state machine (FSM) engines, interfacing modules with databases, parsing and generating XML documents, and audio/video capturing/transmitting over the internet. This framework uses several powerful Java frameworks: the Java CC compiler framework (Copeland 2007), the Unimod framework (Shalyto 2001; Weyns et al. 2007), and the Java Media Framework (JMF) (Terrazas, Ostuni, and Barlow 2002). Within the MWP, this framework is used for the implementation of Java-based distributed network-based architecture, integrating several autonomous engines for processing/generating natural modalities, using common protocols. Namely, the DATA-Subsystem is complex client/server architecture, composed of one main server (named DATA server), and several module servers, named DATA module servers (Figure 2). Each natural modality engine requires one module server in the cluster. The main server is the core of the cluster and manages connections between clients (DATA clients are light clients on the user's side), and several module servers within the cluster. All

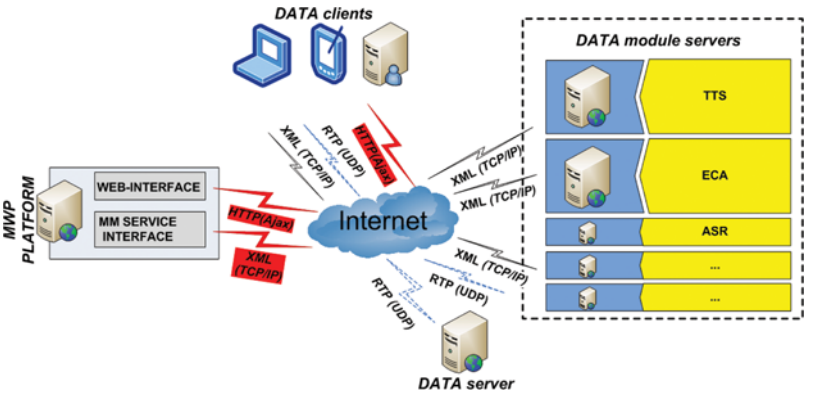

FIGURE 2 Architecture of the DATA-Subsystem. (Color figure available online.)

modules are connected via the Internet using TCP/IP and UDP connections. The main server is able to handle one or several module servers for each client (for each user). Furthermore, the main server can handle several clients and corresponding cluster configurations simultaneously. Module servers are also able to communicate with several clients (e.g., web-based UIs), simultaneously. In general, module servers handle input/output data, in XML-based data format (between the client, main server, and module servers). Module servers that run acoustic-/video-based components and generate audio/video data as output transfer the data via the main server to the clients by using RTP/HTTP protocol.

The integration of new engines into the cluster must be as flexible as possible. The DATA-Subsystem has been developed as an independent subsystem, based on Java, and able to run on Linux and Windows platforms. The implementation of the specific modality driven by a dedicated module server is performed via the XML-based configuration file. In this way, it is unnecessary to change the code, or to develop any additional APIs in order to provide a new modality in the cluster and in the MWP. The DATA-Subsystem's modules are all multithreaded engines. In other words, each module contains a pool of threads that is initiated with a predefined number of threads. These threads are used to serve the requests of different light clients (on the user's side). All requests coming from such clients are redirected via the main server to the corresponding module server(s). When the request is received, the main server and module server(s) involved pick-up an available thread from the pool and establish the connection. The dedicated session then runs for the corresponding client within the assigned thread. If no threads are available in the pool, the main server or module server rejects any further clients' requests until some

connections are closed and some threads are set free again. Two interfaces, based on the DATA-Subsystem, are used within the MWP:

1. A light Java-based client (DATA Client), that provides the ability for audio communication in the direction: *web-based UI* $\longleftrightarrow$ *DATA-Subsystem*. This API is integrated into the EVA plug-in (Figure 1) and automatically runs as client web service. The API is also used for automatic registration of the user's device into the DATA-Subsystem, and also to enable exchange of the device-dependent data streams (e.g., video stream of the verbalized information performed by an ECA, audio stream for ASR, etc.) with the DATA-Subsystem.
2. A multimodal service interface (MM service interface in Figure 1) integrated into the MWP layer, provides the ability to communicate in the direction: *MWP* $\longleftrightarrow$ *DATA-Subsystem*. The MM service interface component serves as a low-level communication interface that enables the web application to benefit from natural modalities within the DATA-Subsystem. Functionally, the interface behaves as a DATA module server and will, for each light Java-based client connected to the DATA-Subsystem, create a dedicated TCP/IP session. The lifecycle of the session ends when either the light Java-based client disconnects from the platform, or the device's web interface is closed.

EVA Plug-In

The major benefit of the proposed MWP is the already-mentioned "client-side" integration. In order to merge different (also incompatible) technologies, we decided to implement an intermediate platform running on external dedicated web servers. Each such intermediate web server can also host one or more web applications. In this way, we prevent additional server load of the existing web-application servers (publishing web content and services).

In order to achieve full web compatibility, the MWP is implemented as a web application. Its front end is represented by the EVA plug-in. This plug-in serves as a user-front-end integration point; it encapsulates the UI of the existing web application. Server-side integration was avoided for the following reasons:

1. Conveying the information simultaneously over multiple channels depends on the user and his/her device (which information channels are desired/supported). The web-application servers are (in the context of our work and fusion of modalities) data and web-service providers and should retain their native (unchanged) form.

2. Depending on the coding style and the architectural properties of the web application, the integration effort may increase exponentially when implementing server-side fusion.
3. The server-load of the web application should not increase, because of the modality system's integration. Because we wish to target any web application, running on any kind of hardware, this assumption seems only natural.

The EVA plug-in (Figure 3) is a user front end with no (or minimal) graphical elements. It implements only:

- a web-based video player (e.g., Flash player),
- an IFRAME that encapsulates the UI of the existing web application,
- JavaScript event-handler, JavaScript API for IFRAME-based cross-domain communication (Li et al. 2011; Chatti et al. 2011),
- a light Java-based client (*DATA-Client*) for direct data exchange between the DATA-Subsystem and the user front end (e.g., for ASR or gesture recognition).

As shown in Figure 3, the idea of the web layer and the EVA plug-in is to fuse the web content from at least two sources: an existing web application, and the MWP's UI. The major issue for such integration is: the traditional *same-origin* web-security model for obtaining third-party data. In order to

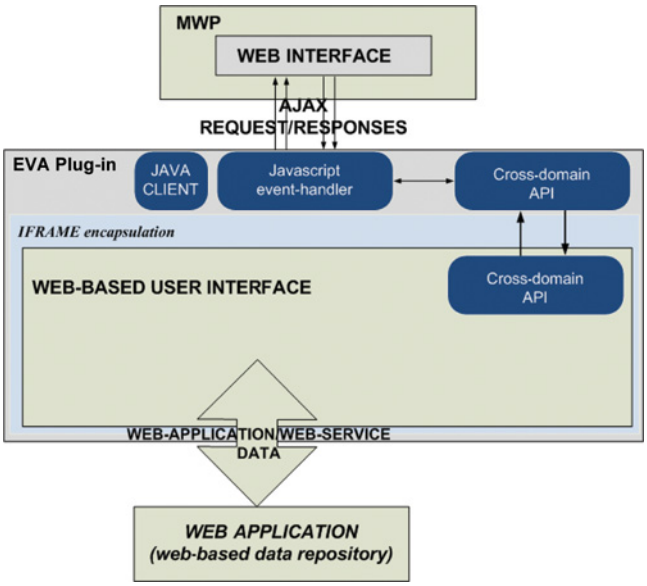

FIGURE 3 The EVA plug-in's architecture. (Color figure available online.)

circumvent this security model, we implement a cross-domain API for JavaScript-based cross-domain communication. The same-origin policy protects those HTML documents to be loaded (accessed) by JavaScript, originated within a foreign domain. However, this policy does not apply to those scripts, which can be loaded from other domains and executed with the privileges of the page that included them. Therefore, the cross-domain API must be integrated into the EVA plug-in, and into the web-based UI. The function of this API is to redirect events from one domain to another. For instance, if the user wishes to perceive the selected text as multimodal output (talking ECA), the text-data to be fed to the TTS and ECA engines will be obtained by the web-based UI's cross-domain API and relayed to the EVA plug-in's cross-domain interface as an appropriate function call (e.g., *Read Selected Text* is relayed to the *EVA Plug-in* as *Process Read Request*).This cross-domain API also captures different JavaScript events performed on the web application's UI (e.g., *page-click, URI change, text-selection,* etc.), and relays them to the cross-domain API contained within the EVA plug-in. The cross-domain API contained within the EVA plug-in serves as a transponder. It receives and relays messages to either a JavaScript-based event handler, or to the cross-domain API, maintained within the web-based UI. The JavaScript event-handler processes and relays different requests/actions toward either the MWP's web interface, or the cross-domain API contained within the EVA plug-in. The event handler is written in jQuery.[4] It contains client-side procedures for Ajax-based communication with the MWP's application core, the Ajax/HTTP event listener, and the event processor that handles both cross-domain and Ajax/HTTP events.

The Multimodal Web Platform's Core

The MWP's core in Figure 1 is a Tomcat-based web-application core that enables interfacing with the DATA-Subsystem. The core facilitates the *web interface, web-application configuration repository*, and the *MM service-interface*. The web interface is further defined by the EVA plug-in, and a back end (administration interface). The back-end interface is used for configuration of the MWP (its users, user groups, and devices), and for setting up the relations between user actions/web services (as performed/provided by the existing web application), and the "multimodal" reactions performed by the MWP. The web-application configuration repository is represented by an MySQL[5] database that stores the following information:

- a user's preferences (user's background information, degree of gesturing present during verbalization, degree of co-articulation and velocity of pronunciation, ECA's gender, and its other customizable preferences, etc.),

- user's multimodal-services (the usage of ASR, the usage of TTS synthesis, and the usage of animating ECAs – visualization services),
- user-device properties (supported web-based video players, supported web browser, device screen resolution, network interfaces, supported multimodal services, full ECA, or talking head, etc.), and
- dialogue scenarios generated for the selected web application (e.g., user action/web service mapped to a dialogue scenario and configuration options).

All the preferences and scenarios are inserted into the web-application configuration repository by using administration interfaces provided by the MWP's back end. The core is designed based on multithreaded web-application architecture, as shown in Figure 4. The MM Service-Interface component (Figure 4) implements software interfaces that allow the web application to benefit from the DATA-Subsystem's distributive architecture. In the context of the DATA-subsystem, the MM Service-Interface represents a DATA module-server, able to communicate with several clients (on the users' side), even simultaneously. The MM Service-Interface component incorporates different protocols and procedures for low-level distributed data transfer. Its main roles are to identify protocols that are executed on the network, and to "translate" and process incoming/outgoing messages between the web-interface and the DATA-Subsystem. The MM Service-Interface component, therefore, recognizes the semantics of the message, as received from the JavaScript's event handler, transforms it into a communication packet (based on the protocol defined by the DATA framework) and relays it to the DATA server for further processing. The

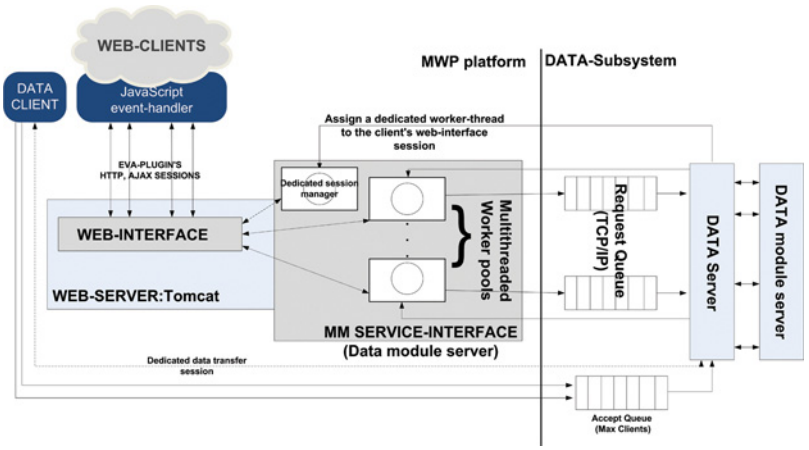

FIGURE 4 Multi-threaded architecture of the MWP's core. (Color figure available online.)

MM Service-Interface component also initiates and relays proper Ajax/HTTP responses to the JavaScript event handler. In this way, the MM Service-Interface component acts as a worker pool used for background processing. The flow of servicing (Figure 4) web clients is two-staged. During the first stage (registration stage), the client device and UI try to register to the DATA-Subsystem simultaneously. The client-device registration is initiated by the Java-client (light Java-based client), whereas the UI tries to register via the EVA Plug-ins by using the JavasScript event handler. Each client-device's registration request enters the TCP/IP accept queue, and each UI registration enters the dedicated session manager's accept queue. In the second stage (if the maximal number of registered client devices has not yet been reached), the DATA-Subsystem initiates a process of dedicated worker-thread generation. In our model, each dedicated worker-thread is represented as a *station*. Each such station is interfaced with those HTTP sessions belonging to the user device, and those TCP/IP sessions connecting it with the DATA-Subsystem. The maximum number of available sessions depends on the current load of the DATA-Subsystem, whereas the service response time depends on the service execution time. When the registration process is concluded, and a proper station is created, the DATA-Subsystem will automatically establish additional *data-transfer sessions*. These extra sessions depend on the natural modalities to be used (e.g., an RTP session for ECA's visualization, a dedicated audio data transfer over the TCP/IP session, as used for ASR, etc.). If, however, a proper *module-server station* cannot be created, the user won't be able to use supported multimodalities within his/her web interface. However, the core functionalities of the existing web application will be available to him/her. The module-server station is automatically destroyed when the user device's HTTP sessions are closed.

INTEGRATION

The previous sections have discussed the concept of the MWP in detail. We have described its architecture and its three key building blocks: the DATA-Subsystem, EVA plug-in, and the MWP's core. The main idea of the MWP is to provide the ability for general web applications to integrate natural modalities that provide additional communication channels. The TTS, ASR, and ECAs are some of those technologies that have already been proven for bringing much more natural experiences when interfaced with computer systems (e.g., web applications). In order to integrate these "multimodal" technologies into the web-application domain as flexibly as possible, we have provided a simple, yet efficient, solution named MWP. It provides subcomponents that can communicate with any web

application, and also subcomponents that can communicate with all systems provided by the DATA-Subsystem. These subcomponents then implement those protocols and algorithms needed for data transfer between the web and the DATA-Subsystem domains. This section focuses on the process of integration and configuration possibilities. The main components used for the integration are the following:

- application-specific cross-domain API,
- application-unspecific web application running the MWP's core, and
- application-specific configuration repository, as part of the web application running the MWP's core.

Client-Side Integration

Let's say we have an operational online store within which we wish to integrate the talking and affective ECA, and speech-driven browsing. The key functionalities of these multimodalities are as follows: the ECA serves as a virtual guide, or as a virtual presenter, and speech-driven browsing enables users to also use speech. First, we have to define the cross-domain APIs. The cross-domain API integrated into the header of the online store implements a "listener" that informs the MWP which subpage the user is currently viewing, or what element has been selected/clicked on the user web interface. In addition, it also implements a method that allows *jQuery*-based selectors, either to obtain web-based data back to the MWP or to populate certain HTML form fields (e.g., checkbox, input field, initiating click event on a button, etc.). The cross-domain API integrated into the EVA plug-in consecutively implements a *request transducer* and *response "listener."* The request-transducer transforms Ajax-based responses (processed by the event handler) into proper client-side actions. Such actions are, for example, "load URI" by changing the IFRAME's uniform resource identifier (URI), insert something into the search field, simulate "*search*" button click, propagate the jQuery selector for selecting product titles, etc. The response "listener," handled by the event handler, listens and processes events initiated by the cross-domain API that is integrated into the online store. These events are, for example, *titles to be read, URI changed*, etc. The event handler is general and can be, as provided with the EVA plug-in, used in any type of web-based application. After the cross-domain APIs have been configured and integrated, we can proceed with the integration process; that is, to configure the EVA plug-in's IFRAME to point to the domain of our online store. With the described tasks, we have prepared the MWP's front end and encapsulated the online store. The MWP can now exchange data with the online store. From now on, the newly generated UI will also contain a web-based video player (for displaying

the ECA), the Java-client API used for audio capturing and transferring data to the DATA-Subsystem, and an unchanged front end of the online store. The second step of the integration anticipates that the MWP's core will be adjusted (configured) to the online store's requirements (context), and possibly for new interaction scenarios to be generated.

MWP's Configuration and Advanced Human–Machine Interaction Scenarios for the Web

The web-application configuration repository and the web interface's back-end UIs are used for the second step of the integration process. A combination of both is used to specify different settings (e.g., devices and device properties, users and application properties, etc.) and to specify different interaction scenarios (*multimodal-based scenarios*). Multimodal-based scenarios define how the enhanced user's web interface will respond to the user's actions (e.g., certain URI loads, user speaks a word, etc.).

Figure 5 presents a general idea of how a multimodal-based interaction scenario would look. We assume that any subpage (URI) of the web application may require a different multimodal-based interaction scenario. Each of such scenarios is then defined by a set of simultaneous/consecutive tasks. Each task is further described by a set of configuration options. In Figure 5 we decided that when loading the selected URI, the following tasks are performed by the whole system:

1. A short introduction is spoken by an ECA. The text for the introduction can be specified within the XML-based scenario description for the DATA-Subsystem (e.g., transformed from meta description, specified as a default introduction or as a user-defined custom greeting).
2. The titles of the articles, with short summaries, are presented to the user. This task has two sets of configuration options. The first set of tasks

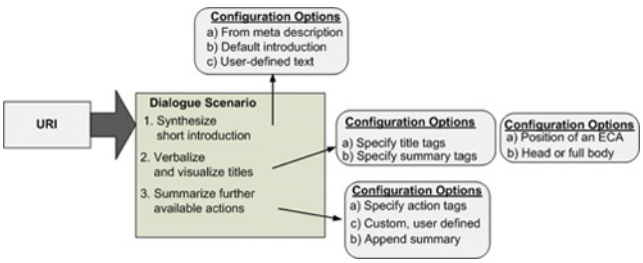

FIGURE 5 Generating a multimodal-based scenario for a selected URI-based location. (Color figure available online.)

defines how the information about titles and summary is obtained. The second set of tasks defines whether the ECA will perform full-body or only head-based visualization, and also the position of the web player (where on the screen the ECA should appear).

3. The scenario concludes by synthesizing further available actions. This synthesis is based on the currently presented web data (actions such as read the selected title in detail, go back to the previous page, present the selected titles again, etc.).

Using the presented concept, several human–machine interaction scenarios can be formed for different subpages of the online store. Each such scenario then specifies how the ECA should respond to different events (e.g., on page load, user click, etc.), propagated directly on the UI. Additionally, such a scenario also defines which keywords (key-phrases) (or key-word encapsulations for dynamical dictionaries) are to be recognized by the MWP. By configuring the MWP according to the specification of the online store, and by defining the human–machine interactive scenarios, we have adjusted the MWP to the context of the web application. The final step of the integration process is to interface the MWP with the DATA-Subsystem. This step is performed automatically, based on the rules and procedures discussed in the following section.

DATA-Subsystem Integration

Complex communications between several distributive machines within the DATA-Subsystem are very flexibly and efficiently described, implemented, and executed in the form of FSM (Mohri 1996). The MM service-interface component is, therefore, also implemented as a finite-state engine. Finite-state engines are constructed by using a UniMod framework (Shalyto 2001; Weyns et al. 2007). This framework already defines objects for the construction and execution of finite-state engines, but needs finite-state engine descriptions in specific XML-based data format. A flexible and efficient tool called the *ProtocolGen* tool has been developed in order to generate XML-protocol scenario files for different clusters' task scenarios (for all DATA-Subsystems' modules). The first step is to draw a graph representation of the cluster's task by considering the supported protocols and desired architecture of the cluster. These graphs are simply FSMs composed of states, transitions, and events on transitions, which trigger graph traversals during certain task executions. Graph representations are then transformed into XML descriptions and stored in the form of XML-protocol scenario files. The UniMod framework's FSM description is too difficult to read or to generate manually/directly from the graph representations. Therefore, a proprietary XML format has been defined. The proprietary

XML format is ultimately just automatically converted into UniMOD XML format. For this step, the corresponding Java parser is used based on JavaCC and JavaC compilers. After the UniMod XML format is generated, the XML-protocol scenario file can be used by all clusters' modules. Nothing has changed, from the user's point of view, after the additional XML-protocol scenario files have been added. The whole MWP is used in the same way; only the DATA-Subsystem can give, in this way, a new human–machine interaction experience to web-application users.

RESULTS

We have used the proposed MWP in order to interface a web application, named BQ-portal, with the distributive DATA-Subsystem. The DATA-Subsystem currently hosts two engines for generating/processing natural modalities: the TTS system PLATTOS (Rojc and Kačič 2007) and the ECA engine EVA (Mlakar and Rojc 2011). The DATA-Subsystem automatically preprocesses (parses the document object model [DOM] structure, removes html tags, and removes images and other nontextual data) any web data and transforms it into *speech* and/or *animated speech*.

The output depends on the multimodal scenario specified in the configuration repository. The animated speech is generated by automatic fusion of TTS and ECA modalities (Rojc and Mlakar 2011). The BQ-portal application is an information kiosk to be used by students and other visitors. In addition to faculty-related information, it supports RSS feed technology, the Google Translator API-based real-time translations, and several interfaces relating to control of the laboratory intelligent ambience (e.g., radio frequency identification (RFID)-based electronic lock, lights, projector, etc.). The BQ-portal web application and DATA-Subsystem, together, present a fusion of web and nonweb modalities into a more natural, multimodal UI. The overall hardware architecture is presented in Figure 6. The architecture currently consists of five computers connected within a 10/100 Mbit local area network. Three machines are used to run the DATA-Subsystem's services: the DATA server and two DATA module servers (for the TTS PLATTOS and ECA EVA). One machine is used to run the MWP and BQ-portal web application, and the last machine for the user's interface and DATA client API. The modality-dependent services within the DATA-Subsystem are simultaneously used also in other proprietary research systems (e.g., auto-attendant, IP-TV system). By using the architecture as presented in Figure 6, the users of the BQ-portal application are able to receive information via affective ECA EVA, instead of, for example, standard reading text data from the web application interface. Text data can be spoken by ECA in three ways: automatically (if specified by the system's scenario), manually (by clicking the "read" button), or by any text selection

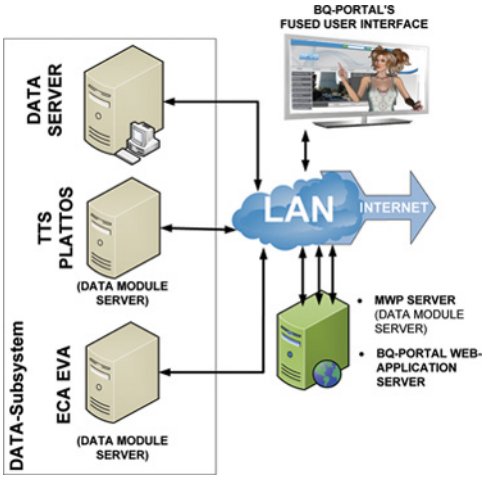

FIGURE 6 Multimodal-enhanced web application: BQ-portal architecture. (Color figure available online.)

within the web application interface. ECA EVA can additionally perform synthesized body motions (e.g., lip-sync, facial expressions, hand gestures, gaze, head and arm motions), resulting in even more humanlike experiences when communicating with the computer. In the following subsections, multimodal-based services are presented, such as BQ-portal's RSS feed reader, and BQ-portal's machine translator.

Multimodal-Based RSS Feed Reader

RSS feeds are usually presented as news, weather services, etc. The BQ-portal implements a news-based RSS service environment, presenting content from different news providers (national broadcast companies). The UI for this service is presented in Figure 7.

When a user selects the RSS news list, the ECA EVA responds, as specified by the service's scenario (Figure 8).

The ECA EVA "reads" an introduction, using the text defined in the scenario. Then ECA EVA also "reads" all the new articles' titles and their short summaries. The text on the titles is stored within HTML *a* tags with attributes *type*= *"title"* and *for*= *"RSS."* The text regarding a short summary is stored within *summary* tags with an attribute *for*= *"RSS."* This text is fed to the DATA-Subsystem, where it is converted into speech and fused with the ECA EVA, played to the user as an audio-video stream. Similarly, when the user selects some specific RSS news, the ECA EVA will "read" the specified content to him. The idea behind the multimodal-based RSS Feed reader is also to handle the visual-based speech synthesis of highly dynamic content, as contained within the predefined HTML encapsulations. The response

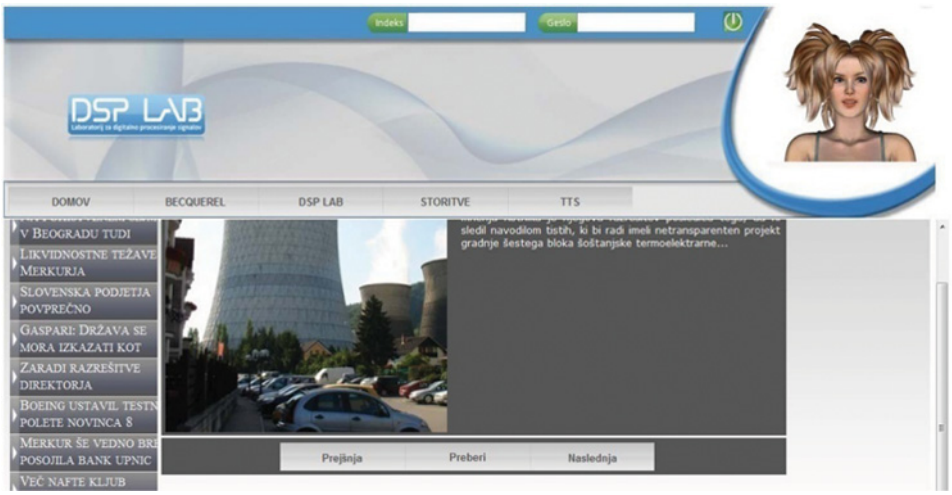

FIGURE 7 Multimodal-based RSS feed reader's web interface. (Color figure available online.)

time (from page load to motion display) of the RSS feed-service is currently about 500ms to 5 s, depending on the lengths of the text sentences (phrases) to be synthesized by the TTS engine.

Multimodal-Based Machine Translation

The BQ-portal web application also enables machine translation on selected text data. The text source can be obtained, either as text from some webpages or as text entered by the user. The Google Translator API has been adopted and implemented within the web application for the machine translation service. In its raw form, the machine's translation service supports language translation for all languages supported by the

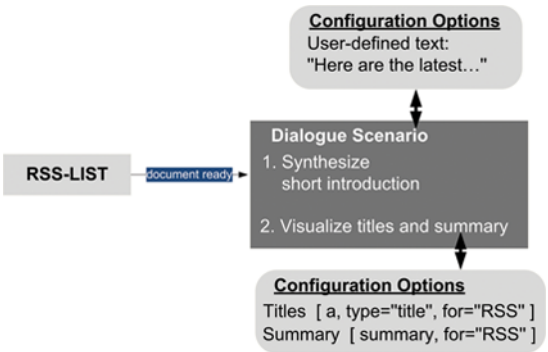

FIGURE 8 RSS feeds service's scenario for human–machine interaction. (Color figure available online.)

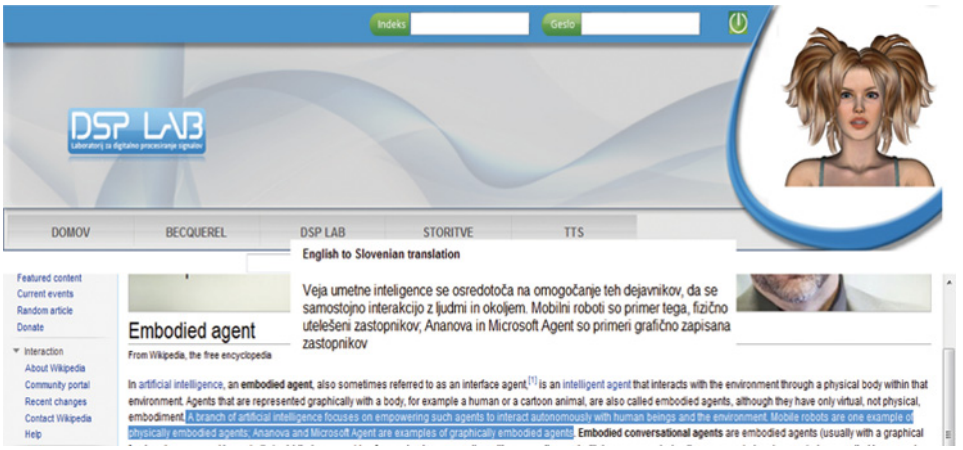

FIGURE 9 BQ-portal's multimodal-based machine translation. (Color figure available online.)

Google Translator API, whereas the multimodal-based version supports *any-language*-to-Slovene machine translations, because the TTS engine currently supports Slovenian language. The machine translation-service is activated simply by selecting/entering text, and clicking the "read" button. Figure 9 presents the machine translation service's UI. The selected/entered text is first preprocessed and transformed into its raw form (e.g., without any HTML elements, etc.). Then the text is passed to the BQ-portal's machine translation API and forwarded to the Google Translator's web-service interface. The response of the Google Translator's web service is then handled as any other text to be synthesized. As such, it is first passed to the DATA-Subsystem as a user's read-request, synthesized, and then fused into the audio/video stream via the ECA EVA engine. Currently, the machine translator service uses completely external service providers. In the case of RSS feeds, the text source for the speech synthesis was maintained within the BQ-portal's domain. However, the text source for the machine translator API was only partiality maintained by the BQ-portal. The actual machine translation service was performed by the Google translator web service. The response time (from request to animated behavior) and the quality of translation depended mainly on the translator web service.

CONCLUSION AND FUTURE RESEARCH

This article presented a novel MWP, which introduces a modular approach for providing multimodal web applications in a distributive manner. The presented approach performs a client-side integration based on encapsulation rather than developing new web applications (Bickmore,

Schulman, and Shaw 2009), multimodal browsers (Johnston, Di Fabbrizio, and Urbanek 2011), or web-operating systems (Lamberti and Sanna 2011). Although the idea of merging different technologies into personal desktop environments based on web technologies is promising and provides a unified access to end users, we believe that a more flexible and end-user device-compatible solution should be based on remote processing networks accessed by different user devices/interfaces. Those applications exploiting different modalities should, however, be enhanced only by those additional modalities and not redesigned/redeveloped. The MWP enables an enhancement of general web applications with several modalities and the further usage of these technologies on devices hosting web browsers. The integration process of fusing web data (web-based UI) with services for the generation/processing of natural modalities, as discussed in this paper, are quite simple. When using the MWP, it requires as little as possible intervention to configure/code web applications or multimodal services. By extending the concepts of IFRAME cross-domain communication and multimodal scenarios, the integration process is implemented automatically. The already-existing *presentation layer* of the web application is simply encapsulated in the MWP's front end, and light JavaScript API is integrated into the HTML header of the web application's front end. The functionality and design of the general web application remains unchanged. The mediating web-based application core, provided by the MWP, is general and can easily be extended to virtually any context within the web domain. As proof of this concept, we also have developed a BQ-portal web application and fused it with TTS and ECA natural modalities. By using the procedures presented in this article, a web-based kiosk was flexibly and efficiently enhanced with affective ECA EVA. By encapsulation of the web-based user front end into the MWP, we have given the ECA the freedom of screen. In other words, our approach enables the ECA to perform head, up-to waist, and full-body animation by freely moving within the region defined by the screen. The concepts presented in this article also enable other natural modalities to be flexibly integrated into the DATA-subsystem, fused with the web data, and subsequently used by different users running different front ends on different devices. A part of our future plans is to integrate three additional engines for generating/processing natural modalities, the ASR engine, gesture recognition engine, and a dialogue processing/generation engine that will enable the fusing of all of the different natural modalities into responsive, humanlike communicative behavior. Because of the fact that the presented approach is browser independent and that no processing is performed on the client's device, we also plan to develop a mobile edition of the BQ-portal application, and fuse it with natural modalities.

NOTES

1. http://en.wikipedia.org/wiki/XHTML%2Bvoice
2. http://msdn.microsoft.com/en-us/library/ms994629.aspx
3. http://tomcat.apache.org/
4. http://jquery.com/
5. http://dev.mysql.com/

REFERENCES

Bickmore, T., L. M. Pfeifer, and M. K. Paasche-Orlow. 2007. Health document explanation by virtual agents. In *Proceedings of the 7th international conference on intelligent virtual agents '07*, 183–196. Paris.

Bickmore, T., D. Schulman, and G. Shaw. 2009. DTask and litebody: Open source, standards-based tools for building web-deployed embodied conversational agents. In *Proceedings of the 9th international conference on intelligent virtual agents '09*, 2009; 425–431. DOI: 10.1007/978-3-642-04380-2_46. Amsterdam.

Cassell, J., H. Vilhjálmsson, and T. Bickmore. 2001. BEAT: The behavior expression animation toolkit. In *Proceedings of SIGGRAPH 2001*, 477–486. DOI: 10.1.1.111.5468. New York.

Cassell, J., T. Stocky, T. Bickmore, Y. Gao, Y. Nakano, K. Ryokai, D. Tversky, C. Vaucelle, and H. Vilhjálmsson. 2002. MACK: Media lab autonomous conversational kiosk. In *The proceedings of Imagina '02*. Monte Carlo, Monaco, January 12–15.

Cena, F. 2011. Integrating web service and semantic dialogue model for user models interoperability on the web. *Journal of Intelligent Information Systems* 36 (2): 131–166. DOI: 10.1007/s10844-010-0126-3.

Chatti, M. A., M. Jarke, M. Specht, U. Schroeder, and D. Dahl. 2011. Model-driven mashup personal learning environments. *International Journal of Technology Enhanced Learning* 3 (1): 21–39. DOI: 10.1504/IJTEL.2011.039062.

Cimiano, P., and S. Kopp. 2010. Accessing the web of data through embodied virtual characters. *Semantic Web Journal* 1 (1–2): 83–88. DOI: 10.3233/SW-2010-0008.

Copeland, T. 2007. *Generating parsers with JavaCC*. Alexandria, VA: Centennial Books.

Diesbach, P., and D. Midgley. 2008. Embodied agents on commercial websites: Modeling their effects through an affective persuasion route. In *Proceedings of the 3rd international conference on persuasive technology PERSUASIVE '08*, 283–286. Berlin, Heidelberg: Springer-Verlag. DOI: 10.1007/978-3-540-68504-3_32.

Di Fabbrizio, G., T. Okken, and J. G. Wilpon. 2009. A speech mashup framework for multimodal mobile services. In *Proceedings of the 11th international conference on multimodal interfaces (ICMI 2009)*, 71–78. Cambridge, MA.

Goh, O. S., C. C. Fung, K. W. Wong, and A. Depickere. 2006. An embodied conversational agent for intelligent web interaction on pandemic crisis communication. In *Proceedings of international conference on web intelligence and intelligent agent technology (WI-IATW '06)*, 397–400. DOI: 10.1109/WI-IATW.2006.37. Washington, DC.

Graesser, A. C., P. Chipman, B. C. Haynes, and A. Olney. 2005. Autotutor: An intelligent tutoring system with mixed-initiative dialogue. *IEEE Transactions on Education* 48 (4): 612–618. DOI: 10.1109/TE.2005.856149.

Jan, D., A. Roque, A. Leuski, J. Morie, and D. R. Traum. 2009. A virtual tour guide for virtual worlds. In *Proceedings of the 9th international conference on intelligent virtual agents*, 372–378. Amsterdam, Netherlands: Springer. DOI: 10.1007/978-3-642-04380-2_40.

Johnston, M. 2009. Building multimodal applications with EMMA. In *Proceedings of the 2009 international conference on multimodal interfaces*, ICMI-MLMI '09, 47–54. Cambridge, MA.

Johnston, M., G. Di Fabbrizio, and S. Urbanek. 2011. mTalk - A multimodal browser for mobile services. Interspeech 2011, in submission.

Kimihito, I. 2005. Introducing multimodal character agents into existing web applications. Poster presented at the 14th International Conference on World Wide Web, 966–967, Chiba, Japan, May 10–14. DOI: 10.1145/1062745.1062821.

Kopp, S., and I. Wachsmuth. 2004. Synthesizing multimodal utterances for conversational agents. *Journal of Computer Animation and Virtual Worlds* 15:39–52. DOI: 10.1002/cav.6.

Lamberti, F., and A. Sanna. 2011. Migration desktop applications to the internet: A novel virtualization paradigm based on web operating systems. *Journal of Web Engineering* 10 (3): 234–272.

Li, Y., D. Shen, T. Nie, G. Yu, J. Shan, and K. Yue. 2011. A self-adaptive cross-domain query approach on the deep web. In *Web-age information management*, Lecture Notes in Computer Science 6897:43–55, ed. H. Wang et al. Berlin, Heidelberg: Springer-Verlag. DOI: 10.1007/978-3-642-23535-1_6.

Mlakar, I., and M. Rojc. 2011. Towards ECA's animation of expressive complex behaviour. In *Analysis of verbal and nonverbal communication and enactment*, Lecture Notes in Computer Science 6800: 185–199, ed. A. Esposito, A. Vinciarelli, K. Vicsi, C. Pelachaud, and A. Nijholt. Berlin, Heidelberg: Springer-Verlag.

Mohri, M. 1996. On some applications of finite-state automata theory to natural language processing. *Natural Language Engineering* 2 (1): 61–80. DOI: 10.1017/S135132499600126X.

Paraiso, E. C., and C. A. Tacla. 2009. Using embodied conversational assistants to interface users with multi-agent based CSCW applications: The webanima agent. *Journal of Universal Computer Science* 15 (9): 1991–2010. DOI: 10.3217/jucs-015-09-1991.

Polymenakos, L. C., and J. K. Soldatos. 2005. Multimodal web applications: Design issues and an implementation framework. *International Journal of Web Engineering and Technology* 2 (1): 97–116. DOI: 10.1504/IJWET.2005.007466.

Prendinger, H., S. Descamps, and M. Ishizuka. 2004. MPML: A markup language for controlling the behavior of life-like characters. *Journal of Visual Languages and Computing* 15 (2): 183–203. DOI: 10.1016/j.jvlc.2004.01.001.

Repenning, A., and J. Sullivan. 2003. The pragmatic web: Agent-based multimodal web interaction with no browser in sight. In *Proceedings of the ninth IFIP TC13 international conference on human-computer interaction*, 212–219. Zurich.

Rojc, M., and Z. Kačič. 2007. Time and space-efficient architecture for a corpus-based text-to-speech synthesis system. *Speech Communication* 49 (3): 230–249. DOI: 10.1016/j.specom.2007.01.007.

Rojc, M., and I. Mlakar. 2009. Finite-state machine based distributed framework DATA for intelligent ambience systems. In *Proceedings of international conference on computational intelligence, man-machine systems and cybernetics (CIMMACS '09)*, 80–85. Stevens Point, WI.

Rojc, M., and I. Mlakar. 2011. Multilingual and multimodal corpus-based text-to-speech system - PLATTOS-. In *Speech technologies* Book 2, ed. Ivo Ipšić, Chapter 7. Rijeka, Croatia: In Tech.

Shalyto, A. A. 2001. Logic control and "reactive" systems: Algorithmization and programming. *Automation and Remote Control* 62 (1): 1–29. (Translated from *Avtomatika i Telemekhanika* 1:3–39.)

Tanaka, Y. 2003. *Meme media and meme market architectures: Knowledge media for editing, distributing, and managing intellectual resources*. IEEE Press, John Wiley & Sons.

Terrazas, A., J. Ostuni, and M. Barlow. 2002. *Java media APIs: Cross-platform imaging, media and visualization*. Sams Publishing.

Thang, M. D., V. Dimitrova, and K. Djemame. 2007. Personalised mashups opportunities and challenges for user modelling. In *User modeling 2007*, Lecture Notes in Computer Science 4511:415–419. DOI: 10.1007/978-3-540-73078-1.

Theune, M., E. Krahmer, B. van Schooten, R. Opden Akker, W. Bosma, D. Hofs, A. Nijholt, E. Krahmer, C. van Hooijdonk, and E. Marsi. 2007. Questions, pictures, answers: Introducing pictures in question-answering systems. In *Proceedings of the tenth international symposium on social communication*, 450–463. Cuba.

Weyns, D., N. Boucke, T. Holvoet, B. Demarsin. 2007. DynCNET: A protocol for flexible transport assignment in AGV transportation systems. Technical Report CW 478, Katholieke Universiteit Leuven, Belgium.

Xu, F., P. Adolphs, H. Uszkoreit, X. Cheng, and H. Li. 2010. Gossip galore: An embodied conversational agent for collecting and sharing pop trivia from the web. *Agents and Artificial Intelligence* 67:164–176. DOI: 10.1007/978-3-642-11819-7_13.

Yuan, X., and S. Chee. 2005. Design and evaluation of Elva: An embodied tour guide in an interactive virtual art gallery. *Computer Animation and Virtual Worlds* 16:109–119. DOI: 10.1002/cav.65.

Zaguia, A., M. D. Hina, C. Tadj, and A. Ramdane-Cherif. 2010. Using multimodal fusion in accessing web services. *Journal of Emerging Trends in Computing and Information Sciences* 1 (2): 121–138.