A Unifying Architecture for Easy Development, Deployment and Management of Voice-Driven Mobile Applications

Jakub Dolezal, Lukas Kencl
Department of Telecommunication Engineering, Faculty of Electrical Engineering
Czech Technical University in Prague, Czech Republic
{jakub.dolezal, lukas.kencl}@rdc.cz

***Abstract*—With the advances in voice recognition and synthesis, voice-interactive applications are reaching the mass mobile market. However, creating, deploying or managing context-rich voice-driven applications is complicated due to the scattered and highly diverse nature of specific pieces of software, tools and skills required for a complete system solution. To overcome this, we propose a novel unifying architecture for easy development, deployment and management of voice driven applications which seamlessly integrates mobility and context-awareness, user- and dialog-management, multi-modal interaction and continuous performance measurement and evaluation. Finally, we validate the architecture by stress tests examining possible bottlenecks.**

I. Introduction

Over the last 20 years, the World Wide Web (WWW) has established itself as a major platform for global information spreading and sharing. However, WWW relies heavily on text and visual representation of information, excluding the visually impaired users and limiting others from engaging in multiple parallel activities.

The challenge we focus on is to enrich the Internet experience by voice modality. This will lead to more natural interaction emphasizing strengths of each modality, e.g. voice as a user input and visual output for presenting results. In addition to WWW, service-oriented approach is emerging as a way for delivering information, offering the possibility to process data in the cloud environment.

Voice modality differs from the well-known text or graphical applications. Current approaches build on top of long-term research of automated speech recognition (ASR) and text-to-speech synthesis (TTS). In order to make both ASR and TTS widely available in distributed networks, VoiceXML [1] markup language and MRCP [2] protocol have been created and adopted by many companies. However, special software and skills are required for both development and deployment. Similar to conventional development, it is necessary to handle code reusability and modularity. Latest progress in virtualization, service-oriented approach and cloud computing advantageously allows accessing and utilizing third party infrastructure for compute-intensive technologies, if needed.

This paper introduces a novel unifying architecture for feasible development and deployment of voice-driven applications on large scale, targeting the mobile environment and context-awareness. The architecture relies on the Web Services paradigm, concealing all the specifics of voice modality from the developer. Hence the voice-application developer is required only to implement a few Web-Services, according to the interface definition, and register them. The service registry supports automatic discovery, simple semantic classification and multiple access points for each service, hence services can be switched in case of failure. To validate readiness for mass deployment and scalability, performance evaluation of possible bottlenecks has been conducted.

II. Related Work

Our work has been inspired by several ideas. Context-awareness is a subset of pervasive computing, addressing user centered design by changing the behavior according to user's context (e.g. location). This design principle is deeply analyzed in a survey of new opportunities to increase usability in various deployments scenarios [3]. Various configurations of mobile services architecture are proposed in [4]. The architecture includes REST/HTTP based access management for both recognizer and synthesizer, targeting primary developers familiar with voice driven applications. The authors of the World Wide Telecom Web [5] envision an ecosystem for large scale deployment of mutually-linked voice-driven applications, as a parallel to the widespread WWW. The motivation is to support emerging economies with low Internet penetration.

Design of speech interface for information retrieval is investigated in [6]. VoiceXML is used for dialog implementation. The work also proposes handling large result sets, aiming to document oriented information retrieval. Design of complex dialogs in VoiceXML is considered to have drawbacks in [7]. The work proposes more productive and flexible approaches of the design by introducing DialogXML, a declarative language for dialog transitions, but it is still required to learn another XML markup language. VoiceXML has been adopted for information retrieval interfaces exploiting Wikipedia.org [8] or virtually any WWW page by employing semantic partitioning [9], however these works introduce a one-purpose solution. Our long-term research focuses on voice interaction within a network infrastructure. Earlier, we have proposed and proved useful the Voice2Web platform for convergence of VoiceXML

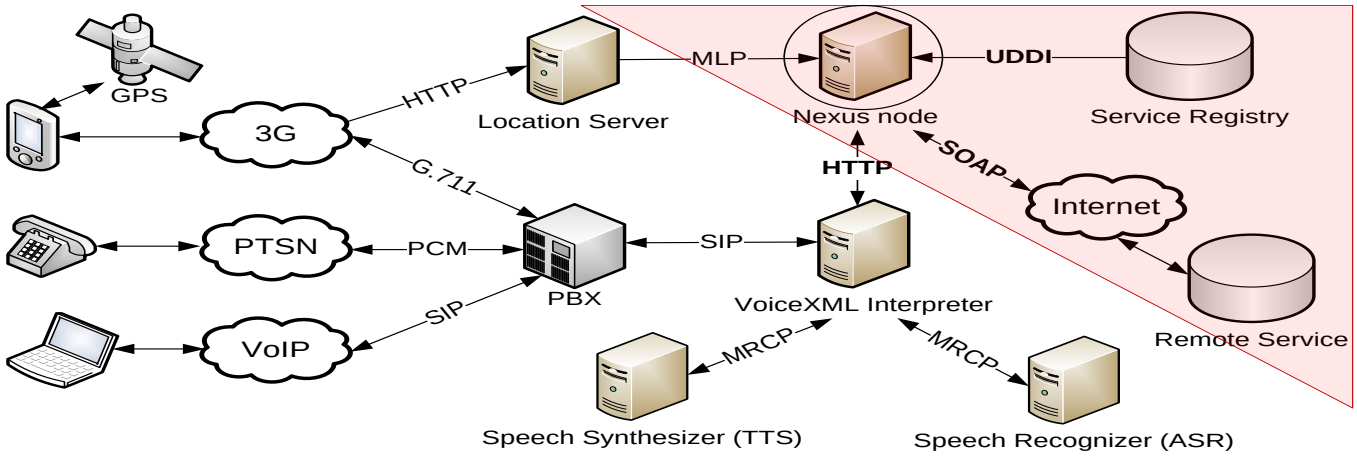


Fig. 1: Various client devices can reach the architecture, which is bounded by the red triangle, surrounded by telecommunication infrastructure. The architecture utilizes user's location and discovers available services. Based on service response, a VoiceXML document is rendered and forwarded to the interpreter that consequently employs ASR and TTS. Bold formatted protocols interconnections (UDDI, SOAP and HTTP) are evaluated in this paper.

services and WWW [10], letting developers create their own applications effectively.

A system for large-scale development and deployment of voice-driven applications is described in [11]. The system allows non-technical people to create simple applications employing voice. Service oriented platform for location aware services is introduced in [12], however the platforms omits dynamic service discovery and user interaction. Future advance of voice toward multimodal interaction is emphasized in [13] as a integration of complementary modalities in order to maximize strengths of voice, DTMF and haptic interaction.

III. Architecture

The proposed distributed service-oriented architecture, consisting of a Nexus node, remote services and a register, is depicted in Figure 1. The Nexus node is the heart of the platform, maintaining context and connecting services with the registry that contains name, description, access URLs and taxonomy for every deployed application and service. Based on information retrieved from the services the Nexus node renders a VoiceXML document that comprises both the dialog structure and content and that is next interpreted by a dedicated piece of software. During the interpretation, both the recognizer and synthesizer are employed. The Nexus node requests each service according to an interaction sequence, identical for every application:

Initialization takes place at startup only, just after the connection with the user is established. Record of the application is found in the registry and its taxonomy is used as a key to discover matching services that will later provide appropriate information for dialog creation. *1. – 3. Settings*, *Location* and *Authorization* services are requested to get access policy, user history, preferences and current location respectively. *4. – Task selection* service provides information to render selection dialogs, consisting of one or more steps, each offering cascade menus with tasks to select. Hence a selection may handle even complicated user inputs. *5. – Task invocation* service is requested, user selection is passed as an input parameter, so a service will return desired information according to the selection. Invocation can be interrupted anytime to make another selection.

Each of these five kinds of remote services is required to comply to a given SOAP interface to enforce appropriate behavior but to claim no requirements to implementation. This approach facilitates modularity, service reusability and robustness [14]. The registry offers both publishing and inquiry, relying on the UDDI [15] protocol. Every application and service is required to be registered via a web interface to become visible to the Nexus node. A record per each service can hold multiple access URLs, which are used sequentially in case of runtime failure. The taxonomy consists of four sets that describe application's requirements and service's abilities: supported languages, places and locations, supported functions and objects, manipulated by functions. Hence, appropriate services can be discovered and switched at runtime.

A. Layer Model

The architecture introduces a four layer model similar to ISO/OSI as shown in Figure 2. Each layer provides functionality for the layer above and encapsulates the used algorithms or protocols. Along cloud support discussed above the layers implements four further functions:

Mobility and Context-Awareness: The architecture is linked with user's environment and offers appropriate information to the user. This is achieved by supporting GPS and Mobile Location Protocol (MLP) [16] for interoperability with various location services. Moreover, advanced user management enables to monitor the user to suggest help, provide history of previously selected tasks and remember preferred languages. A list of known user's locations is also stored at runtime, so that more complex mobility analysis is feasible.

Dialog Management and Multimodality: The architecture is capable to render advanced dialogs. This incorporates authorization, multi-language support, speech rate adjustment, his-

tory navigation, intelligent confirmation based on recognition results and many others. Rendering of these dialogs is fully automated, however a developer can override the default ones via service. Since speech recognition may be inaccurate under noisy circumstances, the platform also supports synchronous multimodal interaction, that in addition to voice takes also advantage of DTMF.

Experimentation and Measurement: The architecture has built-in support for tracking grammars and recognition results in order to manage large scale experiments and monitor QoS. A record for each result consists of list of candidates and confidence scores obtained from a recognizer, language, GPS location, timestamp and ID of the engaged operator. Recognition errors such as nomatch or noinput are also tracked. Huge sets of records are exploited to analyze critical grammars and to pinpoint noisy places. Such automated feedback is a great hint for developers in order to deliver next generation of smart voice driven applications.

Rapid Application Development: The architecture goes beyond traditional development of voice driven application. It offers declarative approach of describing dialog content. The dialogs are dynamically rendered according to available choices to recognize or structured information to synthesize, both received from a registered service. E. g. a developer registers via WWW and implements a service to send sequence of choices; the Nexus node then discovers the service in registry, receives choices, renders grammar, handles result confirmation and recognition errors. Hence there is no need for a developer to learn VoiceXML.

IV. Performance Evaluation

To discover possible bottlenecks and performance bounds, we conducted parallel access evaluation for protocols UDDI, SOAP, HTTP and HTTP Cached employed by the architecture as shown in Figure 1. The MLP protocol is not included since its behavior depends on origin of location (GPS, mobile network). Each protocol was evaluated by one scenario:
(i) *UDDI* is used to query the registry, that returns a list of services found according to search criteria. The registry uses MySQL database as a storage for services records.
(ii) *SOAP*, used by Task Service, provides a menu with 50 constant choices to select. This service has been created for evaluation purposes only.
(iii) *HTTP* carries the VoiceXML document from the Task Servlet to the VoiceXML interpreter. The servlet is a part of the Nexus node, which is responsible for maintaining user-context and rendering the VoiceXML document according to information retrieved from the Task Service. The servlet also tracks grammars and stores them into MySQL database.
(iv) *HTTP Cached* is a complementary scenario to HTTP, providing low consumption of resources. Resulting VoiceXML document generated by the Task Servlet is GZIP-compressed and cached in memory upon first request and uncompressed and returned upon following requests. Real life usage enables fine-grained lifespan adjusting.

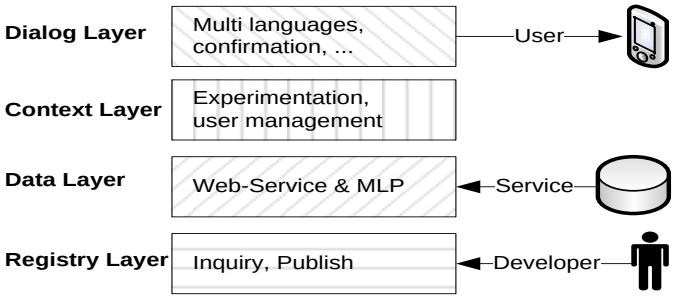


Fig. 2: The architecture comprises of four layers that discover available services, contact them and retrieve information, maintain user's context and finally render voice dialogs.

We state a hypothesis to verify: *HTTP* communication will be a bottleneck with the greatest overhead due to context management and dialog rendering. Next, *UDDI* will also bear a great burden, due to the massive join operations on underlaying database. Conversely, times of *SOAP* will be low, since only XML parsing and serialization is involved. Finally, *HTTP Cached* will be lightweight, employing memory accesses only.

A. Measurement Setup

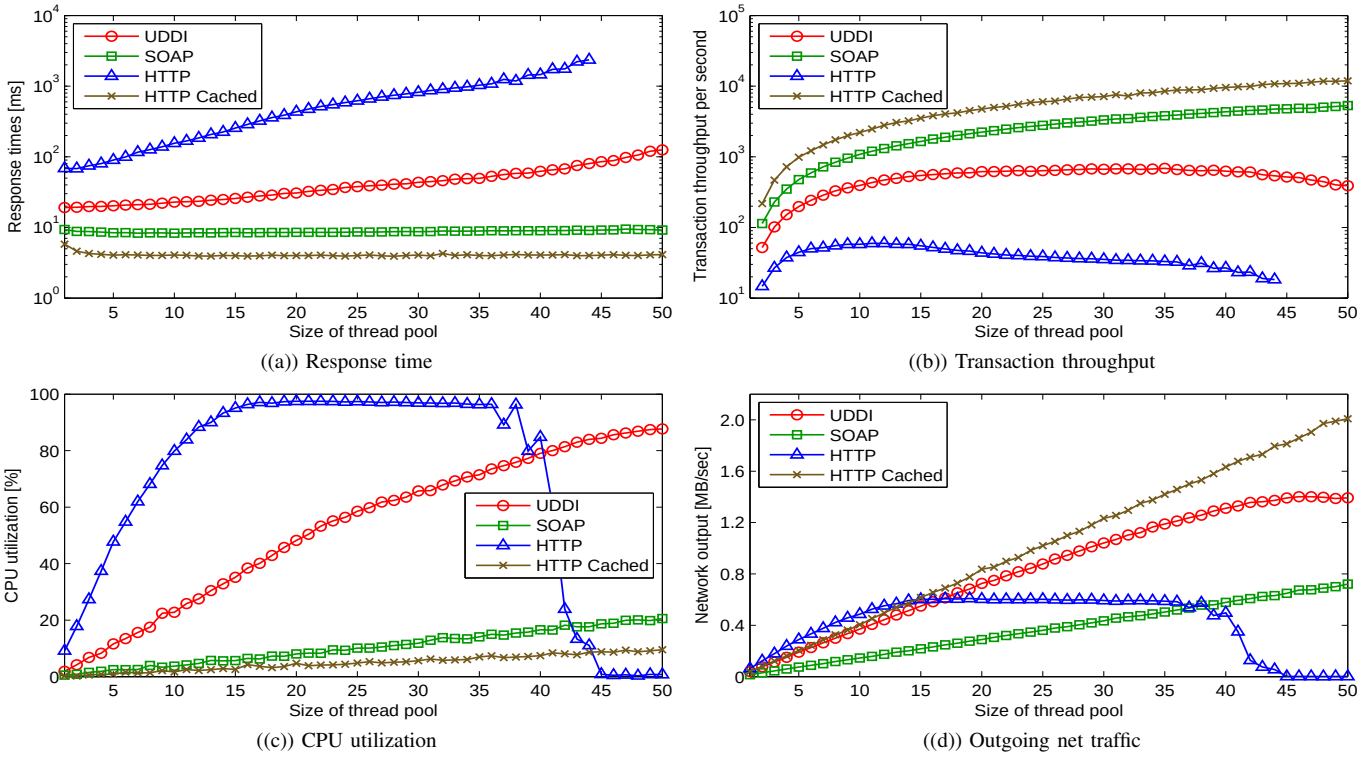
Each scenario was performed separately five times during the measurements. A stepping thread pool was employed for simulation of parallel access. Every 24 seconds a new thread was started, up to the total of 50 threads after 20 minutes of experiment duration. After receiving a response, each thread slept according to a Gaussian random timer, configured with mean delay 300ms and variance 100ms. The Nexus node, registry and service were all deployed on single machine (dual core Pentium 4 CPU at 3.60 GHz and 2GB RAM memory) with installed Tomcat servlet container [17], hence networks effects were not considered. Tomcat was configured with maximum heap size 1280 MB, 40 maximum threads. To eliminate an impact of server start-up, each scenario was performed first off-record to let Tomcat allocate enough threads. The request-generating client was deployed on a separate machine, with 100Mb network connectivity.

B. Results

Response times were measured for an increasing size of request thread pool. The results for each scenario were averaged. Next, transaction throughput T was computed by: $T = \frac{S}{R}$, where S is the size of the pool and R is the average response time of 1 thread. The throughput is an estimate of maximum transaction count per time for various size of thread pool.

TABLE I: Summary of response times for various scenarios

	UDDI	SOAP	HTTP	HTTP Cached
Requests [-]	85194	98536	22588	101266
Average [ms]	45.76	8.75	670.83	4.08
Min [ms]	14.40	6.00	44.66	3.00
Max [ms]	715.60	662.00	10976.66	697.00
Std. Dev [ms]	26.86	0.32	581.25	0.27
Percentile 95% [ms]	145.60	11.80	1455.66	5.20



((a)) Response time

((b)) Transaction throughput

((c)) CPU utilization

((d)) Outgoing net traffic

Fig. 3: Overall results outlying average response times (a) and transaction throughput (b) are augmented by performance analysis containing CPU utilization (c) and outgoing network traffic (d) for various size of the thread pool.

Figure 3 contains response times and throughput analysis along with CPU utilization and amount of outgoing data. The HTTP scenario, performed by the pool of more than 33 threads, caused exponential memory consumption repeatedly growing into out-of-memory failure of Tomcat. All other requests received a correct answer, having zero error rate. Table I contains statistic insight into results, both average and std. deviation were computed from averaged response times over all sizes of the thread pool.

C. Discussion

Each new request is served by a thread, slower responses for the larger thread pool are caused by parallelism overhead and database access, as proven by the *UDDI* and *HTTP* scenarios, where response times and CPU utilization are high. Conversely, *SOAP* and *HTTP Cached* scenarios are lightweight and easily scalable. Results of *SOAP* depends however on the amount of payload transfered. To verify this, a configuration of the scenario, carrying 600 choices to select instead of original 50, was investigated. The results indicate a maximum throughput value at approximately 38 threads. The *UDDI* scenario reaches maximum at 35, *HTTP* at 12 threads. This drawback can be overcome by employing caching functionality. Our hypothesis of the possible bottlenecks is thus verified.

Scalability can be viewed in two aspects: number of applications deployable and number of concurrent users. Application scalability is achieved by combining distributed resources, allowing for easy composition and execution of many parallel services. The key scaling benefit is shown in the *SOAP* scenario, which proves the architecture's ability to maintain cooperation with many web services simultaneously. In real-life, the ASR would most likely become the bottleneck, however, it can be scaled individually, using standard techniques.

V. Conclusion and Future Work

This paper proposes a novel architecture to facilitate large scale development and deployment of voice driven applications. The architecture greatly reduces today claimed requirements and simplifies necessary effort to create such application by employing a Web Service access to dialog creation. The key benefit of this approach is the feasibility of creating voice driven applications without knowledge of either VoiceXML or dialog-design patterns. Web services are registered so they can be dynamically discovered and switched in runtime. Therefore the architecture is fully ready for cloud computing.

The architecture, currently deployed at our university laboratory, serves for purposes of conducting experiments and prototyping applications: a tourist guide navigator and an assistant to visually impaired people. We also intend to open the architecture to academic community in order to open up an avenue for new ideas and solutions.

Acknowledgment

This research was supported by CTU grant SGS11/123/OHK3/2T/13 and FRVS grant 2163/2011. We wish to thank IBM Research and Vodafone Foundation Czech Republic for their generous support.

REFERENCES

[1] W3C. (2007) Voice Extensible Markup Language (VoiceXML) 2.1. [Online]. Available: http://www.w3.org/TR/voicexml21/

[2] IETF. (2006) Media Resource Control Protocol (MRCP). [Online]. Available: http://tools.ietf.org/html/rfc4463

[3] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," *Int. J. Ad Hoc Ubiquitous Comput.*, 2007.

[4] G. Di Fabbrizio, T. Okken, and J. G. Wilpon, "A speech mashup framework for multimodal mobile services," in *Proceedings of the 2009 international conference on Multimodal interfaces*, 2009.

[5] A. Kumar, N. Rajput, D. Chakraborty, S. K. Agarwal, and A. A. Nanavati, "WWTW: the world wide telecom web," in *Proceedings of the 2007 workshop on Networked systems for developing regions*, 2007.

[6] J. E. Gilbert and Y. Zhong, "Speech user interfaces for information retrieval," in *Proceedings of the twelfth international conference on Information and knowledge management*, 2003.

[7] E. Nyberg, T. Mitamura, and N. Hataoka, "DialogXML: extending VoiceXML for dynamic dialog management," in *Proceedings of the second international conference on Human Language Technology Research*, 2002.

[8] C. Kolias, V. Kolias, I. Anagnostopoulos, G. Kambourakis, and E. Kayafas, "Design and implementation of a VoiceXML-driven wiki application for assistive environments on the web," *Personal Ubiquitous Comput.*, 2010.

[9] I. V. Ramakrishnan, A. Stent, and G. Yang, "Hearsay: enabling audio browsing on hypertext content," in *Proceedings of the 13th international conference on World Wide Web*, 2004.

[10] J. Rudinsky, T. Mikula, L. Kencl, J. Dolezal, and X. Garcia, "Voice2Web: Architecture for Managing Voice-Application Access to Web Resources," in *Proceedings of the 12th IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services*, 2009.

[11] A. Kumar, S. K. Agarwal, and P. Manwani, "The spoken web application framework: user generated content and service creation through low-end mobiles," in *Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility (W4A)*, 2010.

[12] Y. H. Ho, Y. C. Wu, M. C. Chen, and A. Sinica, "PLASH: A platform for Location Aware Services with Human Computation," *IEEE Communications Magazine*, vol. 48, pp. 44–51, dec 2010.

[13] Z. Trabelsi, S.-H. Cha, D. Desai, and C. Tappert, "A voice and ink XML multimodal architecture for mobile e-commerce systems," in *Proceedings of the 2nd international workshop on Mobile commerce*, 2002.

[14] V. Agarwal, K. Dasgupta, N. Karnik, A. Kumar, A. Kundu, S. Mittal, and B. Srivastava, "A service creation environment based on end to end composition of Web services," in *Proceedings of the 14th international conference on World Wide Web*, 2005.

[15] OASIS. (2011) UDDI Version 3 Specification. [Online]. Available: http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm

[16] P. Vlacil and R. Bestak, "Implementing Mobile Location Protocol," in *32nd International Conference on Telecommunications and Signal Processing*, 2009.

[17] The Apache Software Foundation. (2011) Tomcat Servlet Container. [Online]. Available: http://tomcat.apache.org/