Conversational Agents and Natural Language Interaction:

Techniques and Effective Practices

Diana Perez-Marin
Universidad Rey Juan Carlos, Spain

Ismael Pascual-Nieto
Universidad Autónoma de Madrid, Spain

Information Science
REFERENCE

Senior Editorial Director: Kristin Klinger
Director of Book Publications: Julia Mosemann
Editorial Director: Lindsay Johnston
Acquisitions Editor: Erika Carter
Development Editor: Hannah Abelbeck
Production Editor: Sean Woznicki
Typesetters: Mike Brehm, Natalie Pronio, Milan Vracarich Jr., Deanna Jo Zombro
Print Coordinator: Jamie Snavely
Cover Design: Nick Newcomer

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

Library of Congress Cataloging-in-Publication Data
Conversational agents and natural language interaction: techniques and effective practices / Diana Perez-Marin and Ismael Pascual-Nieto, editors.
p. cm.
Includes bibliographical references and index.
Summary: “This book is a reference guide for researchers entering the promising field of conversational agents, providing an introduction to fundamental concepts in the field, collecting experiences of researchers working on conversational agents, and reviewing techniques for the design and application of conversational agents”-- Provided by publisher.
ISBN 978-1-60960-617-6 (hardcover) -- ISBN 978-1-60960-618-3 (ebook) 1. Natural language processing (Computer science) 2. Computer-assisted instruction. 3. Discourse analysis--Data processing. 4. Speech therapy--Data processing. 5. Intelligent agents (Computer software) I. Perez-Marin, Diana, 1980- II. Pascual-Nieto, Ismael, 1981-
QA76.9.N38C674 2011
006.3--dc22
2011001310

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter 15
Design and Development of an Automated Voice Agent:
Theory and Practice Brought Together

Pepi Stavropoulou
University of Athens, Greece

Dimitris Spiliotopoulos
University of Athens, Greece

Georgios Kouroupetroglou
University of Athens, Greece

ABSTRACT

Sophisticated, commercially deployed spoken dialogue systems capable of engaging in more natural human-machine conversation have increased in number over the past years. Besides employing advanced interpretation and dialogue management technologies, the success of such systems greatly depends on effective design and development methodology. There is, actually, a widely acknowledged, fundamentally reciprocal relationship between technologies used and design choices. In this line of thought, this chapter constitutes a more practical approach to spoken dialogue system development, comparing design methods and implementation tools highly suited for industry oriented spoken dialogue systems, and commenting on their interdependencies, in order to facilitate the developer's choice of the optimal tools and methodologies. The latter are presented and assessed in the light of AVA, a real-life Automated Voice Agent that performs call routing and customer service tasks, employing advanced stochastic techniques for interpretation and allowing for free form user input and less rigid dialogue structure.

INTRODUCTION

Automated Voice Agents are systems capable of communicating with users by both understanding and producing speech within a specific domain. They engage in humanlike spoken dialogues, in order to route telephone calls, give traffic information, book flights, solve technical problems and provide access to educational material among others.

DOI: 10.4018/978-1-60960-617-6.ch015

Depending on their design, the speech understanding and dialogue management technology involved, they may be of two basic types:

- Directed Dialog Systems: ranging from finite state-based to frame-based systems (McTear, 2004). The former systems are very simple and inflexible menu-driven interfaces, where the dialogue flow is static, specified in advance, no deviations from that flow are allowed, and only a limited number of words and phrases provided by the user can be understood. The latter systems are more advanced interfaces, where the interaction is not completely predetermined and a more elaborate vocabulary can be handled. While both types of systems are primarily system-directed, frame-based systems allow for a modest level of mixed-initiative by handling over-specification in user's input; that is the user can provide more items of information than those requested by the system at each dialogue turn.
- Open-ended natural language conversational systems: mixed-initiative systems, where both system and user can take control of the dialogue introducing topics, changing goals, requesting clarifications, establishing common ground. Equipped with sophisticated speech and language processing modules, they can handle long, complex and variable user input in an attempt to approximate natural human-human interaction as close as possible.

The two types of systems to a significant extent reflect the differences in trends and directions followed by the spoken dialogue industry compared to spoken dialogue research during the last decades. As commercial dialogue systems aim primarily at usability and task completion (Pieraccini & Huerta, 2008), focus was placed on ways to restrict users' input, in order to amend for speech technology limitations and reach industrial standards for useful applications. As a result, industry opted for more directed dialogue systems, which are the most commonly used on the market today.

Furthermore, the need for cost reduction, ease of development and maintenance has led to the development of reusable dialogue components and integration platforms promoting modularity and interoperability. Accordingly, VoiceXML (McGlashan et al., 2004; Larson, 2002) has become an industry standard for building voice applications, which exploits the existing and universally accepted web infrastructures eliminating the need for specific application protocol interfaces (APIs) designated to speech technology integration. Based on the Form Interpretation Algorithm it incorporates a frame-based architecture, providing an industry-feasible trade-off between naturalness and robustness.

Research, on the other hand, aims primarily at naturalness and freedom of communication (Pieraccini & Huerta, 2008). In an attempt to handle almost unrestricted user input and allow for a fully mixed initiative, conversational interface, focus has been on dialogue manager architectures exploiting inference and planning as part of a truly conversational agent. Speech act interpretation (Allen, 1995, Chapter 17; Cohen & Perrault, 1979; Core & Allen, 1997; Allen et al., 2007) and conversational games (Kowtko et al., 1993; Pulman, 2002), discourse structure (Grosz & Sidner, 1986; Stent et al., 1999; Fischer et al., 1994) and prosody manipulation (Hirschberg et al., 1995; Noth et al., 2002) are only some of the topics in an ongoing research for building natural language interfaces.

Furthermore, accessibility issues have gained attention, being important not only for the visual impaired (Freitas & Kouroupetroglou, 2008) but also to people with various disabilities (Fellbaum & Kouroupetroglou, 2008). In particular, spoken dialogue systems are considered as key technological factors for the universal accessibility strategies

of – for example – public terminals, information kiosks and Automated Teller Machines (ATMs) (Kouroupetroglou, 2009).

Nevertheless, the emerging need to support complex, demanding domain applications such as education, help desk or customer care, along with the evolution and level of maturity accomplished by the current speech and natural language understanding technology have led to a significant number of commercially developed and deployed mixed initiative systems and the introduction of more free style automated voice agents, indicating some level of convergence between the two fields, research and industry.

Building on practices and experiences from developing such a system, this chapter comprises a more pragmatic approach to Automated Voice Agents, focusing on practical spoken dialogue systems, presenting techniques, tools and resources for effective design and implementation, assessing them in the light of a real life customer care and call routing application, commenting on best practices and suggesting ways to best utilize these practices.

We follow the typical lifecycle of an automated voice agent and focus on the requirements analysis and design phase, as well as the development of the Automatic Speech Recognition (ASR) and Natural Language Understanding (NLU) modules.

In the following sections we first give a brief overview of previous related work. Next we present the main features and architecture of an automated voice agent, before going on to describe AVA, the real-life application at hand. Illustration of the main steps in an agent's lifecycle follows, and design and implementation phases are subsequently discussed in detail. Final section summarizes key concepts throughout the process.

RELATED WORK

The field of spoken dialogue systems is one of the fastest growing areas over the last decades. With regards to field textbooks, Cohen et al. (2004) is a thorough, well organized presentation of the complete spoken dialogue interface development process and methodology based on extensive real word experience. A sample application is presented as means to observe how development and design principles can be applied in practice. Harris (2005) is another comprehensive guide to spoken dialogue system development grounded on an in depth grasp of relevant literature, and focusing particularly on development process and design. Hempel (2008) is a collection of articles on system quality and usability issues with reference to multimodal systems as well.

Weinschenk & Barker (2000) and Balentine & Morgan (1999) provide a set of practical design principles and guidelines for building a Voice User Interface (VUI). Pitt & Edwards (2003) is another practical approach focusing on menu and prompt construction applying the proposed principles on real life applications involving road traffic information and a voice mail system.

McTear (2004) is an introduction to technical (among other) aspects of spoken dialogue systems, illustrating development of applications with particular software and toolkits. Huang et al. (2001) is a standard guide to spoken language system technology (including signal processing, speech recognition and synthesis techniques as well as Natural Language Understanding (NLU) algorithms and dialogue management strategies). Finally, Jurafsky & Martin (2000, Chapter 19) introduce algorithms and architectures for dialogue managers in conversational agents.

There is a significant number of practical (Dahl, 2004) and more advanced (Pellom et al., 2001; Allen et al., 1995, 1996, 2001; Wahlster, 2000; Sidner, 2004, among others) system descriptions. Here reference will be made to call routing and customer care related spoken dialogue applications. Riccardi et al. (1997) and Gorin et al. (1997) describe the Automatic Speech Recognition (ASR) and Understanding components of the HMIHY call routing system, which utilize phrase based

language modeling and a classifier that uses salient text fragments as features in order to associate user utterances to predetermined call types.

Walker et al. (2002) report on automatically predicting problems in human-machine dialogues for improved error recovery also within the HMIHY system. Chu-Carroll & Carpenter (1999), Garfield & Wermter (2002, 2006) and Zitouni et al. (2003) also place attention on advanced NLU techniques – such as vector based classifiers and recurrent neural networks – for such systems. Lee & Chang (2002) describe an operator assisted call router that integrates a generic ASR module with an information retrieval module based on keyword extraction from existing company documentation with descriptions of routing destinations (i.e. departments), thus eliminating the need for collecting and transcribing actual call recordings.

Williams & Witt (2004) compare menu driven directed dialogue strategies to open ended, free form "how may I help you" strategies for use in automated call routing. Finally, Gupta et al. (2006) touch upon subjects such as system scalability and minimization of development effort describing the NLU component of VoiceTone, a system that provides automated customer care services in addition to call routing.

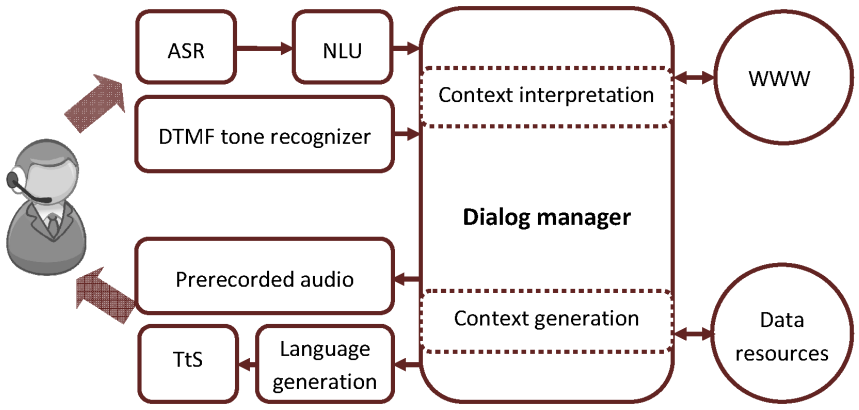
In the vein of theorized practice, this chapter presents a real life call routing and customer care information provision application focusing on design, ASR and NLU implementation and the reciprocal relationship among the three, building on a more general, theoretical perspective.

THE AUTOMATED VOICE AGENT

As mentioned in the introductory section, Automated Voice Agents are programs capable of communicating with users by both understanding and producing speech within a specific domain. Figure 1 illustrates the Automated Voice Agent architecture. The Automatic Speech Recognition (ASR) component converts acoustic user input into text, and passes the text string to the Natural Language Understanding (NLU) component for semantic interpretation.

In addition, a Dual-Tone Multi-Frequency (DTMF) recognizer may be used to allow for DTMF input as well. Next, the Dialogue Manager (DM) evaluates and/or disambiguates the semantic information from the NLU module based on processes such as dialogue history and context interpretation. Depending on input evaluation the DM plans and proceeds to execute certain dialogue actions such as database queries or system prompt formulation. For prompt formulation, the DM output is converted to a well formed written ut-

Figure 1. Automatic voice agent main component layout



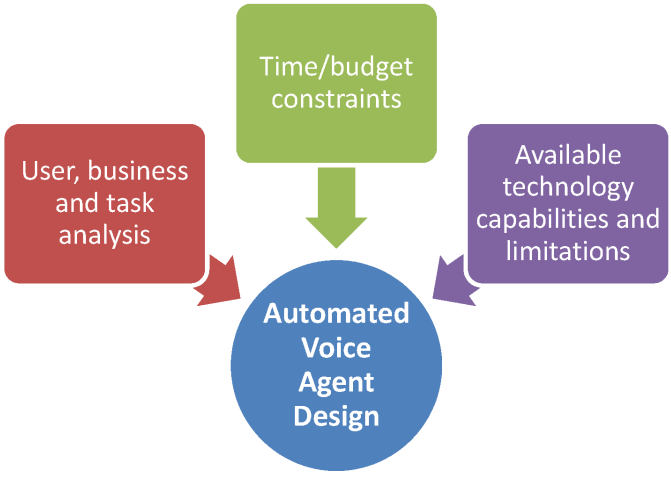
terance by the Natural Language Generation (NLG) module, and then the Text to Speech (TtS) Synthesizer converts the written utterance to speech. In most commercial spoken dialogue systems pre-recorded prompts are used instead.

The main feature of an agent is personification (Harris, 2005). Personification refers to a primitive, inherent human disposition to assign human attributes to non human entities, or in this case, a personality to the automated voice agent. There are certain parameters, design and implementation choices, that affect the personality ascribed to the agent. In particular:

- The kind of language used, namely the vocabulary, syntax, prosody and style, the agent's gender or dialect may cause the agent to appear calm, pleasant, helpful, interesting or encouraging. For example, simple syntax (e.g. use of simple coordination structures rather than subordination) and avoidance of jargon may cause the agent to appear more informal and helpful. Variation in the wording of prompts and tone may make the agent sound more interesting. Use of prosody to convey emotion increases the level of perceived conversation engagement.
- The range of functions, the capabilities and limitations of the agent, the dialogue initiative handling, the interaction style, the choice on grounding and error recovery strategies may cause the agent to appear competent, trustworthy, dependable, credible, co-operative, intelligent, sensible, helpful or knowledgeable. For example, mixed-initiative strategies are usually signs of intelligent behavior. In contrast, an agent that sequentially asks for pieces of information already given just sounds brainless. Changing dialogue strategies (e.g. backing off to a more conservative directed dialogue strategy when the dialogue is problematic) may be considered a sign of co-operative and helpful behavior.

Given the reciprocal relationship between design and available technology, a successful voice agent, one that sounds smart, pleasant and helpful, depends both on effective design methodologies and adequate speech and language technology tools. At the same time business requirements should be met and cost/time constraints should be taken into consideration. Figure 2 illustrates these interactions.

Figure 2. Design considerations and interactions in building an automate voice agent



In the next sections we present AVA, an Automated Voice Agent for a customer care system, with the aim to address the following question: how do our tools and techniques affect design, implementation and evaluation choices, and how can we make the best choice possible?

AVA: AN AUTOMATED VOICE AGENT FOR CUSTOMER CARE

AVA is an automated voice agent built for a Customer Care call centre of a Mobile Telephony company. She performs two major tasks: a) appropriate routing of the client's call to one of 17 dedicated queues, and b) database information retrieval for speech-based automated self-service modules. For both tasks she needs to identify and correctly categorize the caller's request as one of the approximately 100 services and respective thematic categories provided by the Customer Care department.

In addition, AVA displays the following key features (among others): a) recognition and understanding of free style user input. If there is under-specification and ambiguity in the user's input, AVA should formulate an appropriate question, in order to determine how the call should be handled, b) support of mixed dialogue initiative in the following sense: on one hand AVA should be able to handle over-specification; on the other hand, the callers should be able to shift goal at almost any time during the interaction. For example, if the users have already chosen a particular self service, and within the self service sub-dialogue they decide that they want a different service after all, AVA should be able to understand this new request and handle the call accordingly.

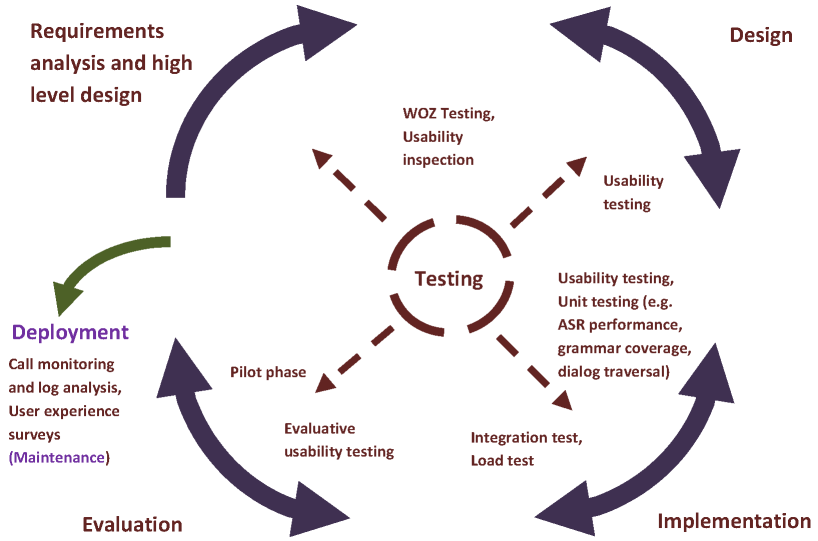
As is often the case with automated voice agents, AVA replaces an existing DTMF system. A DTMF system is static and menu driven and so providing coverage for a complex domain such as customer care eventually results in a large and complicated menu hierarchy. Consequently, the caller is forced to spend precious time navigating through various levels of this hierarchy before finally being transferred to a human agent; efficiency decreases, while user dissatisfaction increases.

Furthermore, even with an overcomplicated menu hierarchy, no exact mapping is guaranteed between the user's request and the menu options offered. The result is an increase in the number of hang ups and misroutings. As AVA replaces the old DTMF system, there is no longer need for dysfunctional, complex menus, a lot more services can be handled and the interaction becomes more natural, efficient and effective.

Spiliotopoulos et al. (2009) compare DTMF systems to spoken language interfaces performing usability testing on a real life paradigm involving both types of systems, showing a great increase in user satisfaction and system efficiency when using a spoken dialogue interface. In particular, the average call duration was 25 and 44 seconds for the spoken dialogue and the DTMF system respectively, while user satisfaction score was 9 percentage points higher for the former compared to the latter.

As is, AVA poses three major challenges with regards to design, implementation and maintenance considerations respectively. The first one involves building on the existing user's mental model and breaking down the customer care domain into a service hierarchy that reflects the user's point of view. A second interdependent challenge involves the automatic recognition of the user input to open-ended questions covering a large range of responses, and mapping it to one or more services. The latter requires, among other things, a large set of typical caller utterances for training the statistical models for speech recognition and interpretation. The third challenge involves minimizing the cost and need for support in a constantly changing domain. These challenges are analysed and addressed in the following sections.

Figure 3. Testing methodology to develop automated voice agents



THE AUTOMATED VOICE AGENT'S LIFECYCLE

Figure 3 shows the basic steps in the process of building an automated voice agent. The adapted phases from Cohen et al. (2004) are the following:

- Design phase, which is further divided into a) the requirements analysis and high level design, where the basic system functionality is analysed and defined and key design decisions are made, and b) detailed design, which results in a complete, detailed specification of the dialogue.
- Implementation phase during which the system components – the Automated Speech Recognition (ASR) Module, the Natural Language Understanding (NLU) Module, the Dialog Manager (DM), the Language Generation Module and the Text to Speech Synthesizer (TtS) – are developed. As an alternative to having a generation module and a TtS synthesizer, static prompts may be pre-recorded and used. Integration with third party software systems is completed. At the end of the implementation phase there should be a fully integrated, working prototype.
- Evaluation and Tuning phase. Final usability tests are performed with the (nearly) finished product aiming primarily at application fine tuning, since important design choices should have already been made and evaluated during the previous steps. Pilot phase is a significant part of the whole tuning process. It is because of the abundance of in service realistic data available for training, testing and tuning purposes, when the market-ready system is released to real users.
- Deployment: the final production system is released to the entire user population.
- Maintenance and quality assurance monitoring.

As Kouroupetroglou & Spiliotopoulos (2009) note, testing is an important, inherent part of all phases rather than the evaluation phase alone (for example it can be in the form of usability testing or usability inspection during the design phase, unit testing and usability testing during the implementation phase (cf. following sections)),

essentially turning the process of building an automated voice agent into an iterative design, (re)test, redesign and (re)implement procedure.

Therefore, even though the steps are depicted in the sequence they principally apply (cf. Figure 3), in practice there is no clear cut line distinguishing among phases, which typically blur into each other. Figure 3 further illustrates the main tests available at each phase, analysed in the following sections. For a more detailed description of how each test applies within the automated voice agent development lifecycle readers may refer to Cohen et al. (2004).

DESIGN PHASE

First in an automated voice agent's lifecycle come the requirements analysis and the design phase. During these phases the developer needs to analyze the *users' characteristics* (demographics, linguistic characteristics, domain and task knowledge, frequency of use, experience with similar systems), the *business goals* (motivation behind the development of such a system, company image, competition, time and cost constraints) and the *application domain* (tasks and features, current and desired workflow, technical environment), in order to make appropriate design choices and proceed with the complete dialogue specification. Based on the analysis the developer has to decide upon high level features, such as initiative and grammar type, down to prompts, dialogue states and slots. At the end of the design phase a complete description of the call flow and all prompts played by the system should be available (Cohen et al., 2004).

A key for effective design is user-centered design (Norman & Draper, 1986; Gould & Lewis, 1985). User expectations, attitudes and behaviour should be accommodated rather than constrained. In this view, an important aspect of analysis feeding directly into design is the understanding of the "natural" mental model that first time users bring to the interaction, their existing – and possibly expected – view of the interaction, a model of how things have worked so far.

The success of an interface greatly depends on the correspondence between this "natural" mental model and the conceptual (Weinschenk & Barker, 2000) or design model (Norman, 1988); that is the proposed model afforded by the design of the interface. Ideally, a system should build on and adapt to the users' prior knowledge and experience, in order to create a more familiar, intuitive, easier to learn and use interface.

For voice agents in particular, the latter is tightly connected to the kind of language – vocabulary-wise and syntax-wise – understood and introduced by the agent. Domain specific spoken dialogue studies are therefore very important and presuppose the existence of appropriate language resources. On a final note, as speech recognition and understanding modules typically require domain specific corpora for training purposes (cf. "Implementation Phase: Speech Recognition and Understanding Modules" section), many of these resources can be shared between design and implementation teams.

The most common techniques for designing voice agents are presented based on the aforementioned considerations and according to the phase of the development cycle in which they are used. The techniques are evaluated with the following parameters: a) ease of application, i.e. the feasibility of these techniques in light of strict industry time frames and cost constraints, b) their contribution to understanding the mental model and interaction patterns, c) their usefulness for determining the vocabulary and other linguistic constructions and d) their appropriateness to serve as ASR and NLU resources.

Gathering information from company personnel and domain experts. Starting early in the agent's lifecycle it comprises a valuable tool throughout the building process, as it provides insights in the business goals and the application in general. Meetings with human agents in

particular can prove to be very informative with regards to identifying typical usage, terminology, confusions and "risky" or complicated steps during the interaction. However, they may lack objectivity, providing invalid information, blurring understanding of actual user behaviour.

Examination of available documentation (e.g. marketing materials, statistics about use) and/or other in-domain applications (e.g. website, DTMF system to be replaced) can provide information on functionality and terminology. However, there are two very important points of caution involved. First of all, company documentation often reflects a business view of the application domain. Migrating this business view into the interface often results in bad performance, as business and user scope do not coincide and thus user expectations are rarely met. Secondly, the audio modality differs from visual or other modalities. The transient, ephemeral nature of speech along with human cognitive limitations place constraints on the speech output and the application structure in general.

Balentine & Morgan (1999), among others, recommend presenting no more than five information units at one time opting for the lowest possible number. Graphic User Interfaces (GUIs) on the other hand exploit vision and space, and can present a large amount of information that can be easily and quickly processed by the user. Therefore, a direct translation of a GUI into a Speech User Interface will most likely result into unfriendly, unusable applications.

Similarly, DTMF modality differs from speech in terms of user psychology, timing, menu structures and selection methods (Balentine & Morgan, 1999). Co-operative, natural conversation is simply not menu navigation. In short, when transferring knowledge and experience from one modality to another, one should be careful to filter out distinct psychological and design principles that refer to or are particularly important to each modality alone.

While both the above mentioned techniques are important for gaining a basic understanding of the business and the application context, they cannot be a substitute for direct contact with users.

User interviews and observation studies allow for such direct user contact, comprising a significant aspect of user-centered design. Interviewing users can provide insights on the how, when and whys of the task. Nevertheless, as users are asked to remember and comment on events and processes "that may normally be performed without a lot of conscious thought" (Weinschenk & Barker, 2000), their input may be inaccurate and imprecise.

Also, care should be taken when forming the interview questions, so as not to bias the interviewees' answers causing them to deviate from their natural language and usage patterns. These concerns do not apply in the case of observation studies, whereas one can directly observe real users performing the task and gain insight into their interaction patterns and language use. On site field studies may be less time – and hence cost – effective, but can provide an opportunity to ask human agents specific questions. For telephone-based applications, on the other hand, there may be a – time and cost saving – abundance of records of calls to live agents immediately available.

Analysis of actual users' calls to human agents is already considered to be an important resource for effective design (McTear, 2004; Cohen et al., 2004), as it provides significant information regarding the vocabulary used, the nature of the interaction and the mental model of the task in general. Most importantly, in contrast to language resources obtained from usability testing, these calls are collected from real users truly engaged in performing realistic tasks.

Alternatively, the next opportunity for collecting utterances from real users is during the pilot phase, which comes later in the development process, and so involves the risk of costly changes due to overlooked early design shortcomings. Furthermore, call records can be transcribed and

used as training corpus for stochastic models for the ASR and NLU modules.

Since the necessary amount of corpus is already available at the beginning of the development lifecycle, there is no need to spend valuable time for the collection of training corpora during implementation. Moreover, having adequate resources for ASR and NLU early in the development process enables the obtainment of more reliable results during evaluative usability testing, as low recognition and interpretation success rate considerably affect the user experience (Kamm & Walker, 1997), and interfere with the evaluation of the dialogue structure per se.

On the other hand, human-human dialogues are intrinsically less restricted in nature compared to human-machine dialogues, and associated with diverse caller behaviour. In a study comparing a corpus collected from human-human dialogues to a corpus collected from human-machine dialogues significant differences were found in the vocabulary used, the length and complexity of the utterances as well as the performance of the statistical language models for ASR (Stavropoulou et al., 2011).

Human-human utterances were approximately three times longer (45% larger corpus vocabulary size), more complicated, and the language model trained on them performed worse (9-14 and 11-14 percentage points increase in word error rate and concept error rate respectively). In conclusion, whilst developers may observe actual users, they cannot observe actual user-system interaction. In fact one can never be absolutely certain what the real users' reaction and perception of the system will be.

At least that is what previous experience has shown us when building a system similar to AVA with regards to complexity, but for – familiar with the domain – telecom shop representatives only. After launch we discovered that the company employees insisted on using specific keywords and phrases essentially reproducing the limited functionality of the DTMF system that was replaced, rendering the use of elaborate vocabulary, grammars and dialogue structure redundant.

All the techniques presented so far aid early design choices and are relatively cost free. In this regard, they form an indispensable part of requirements specifications and high-level design. Next, techniques are presented that require the existence of a basic design skeleton at least, and as such they are used to evaluate high level design choices and guide more detailed design.

Wizard of Oz (WOZ) testing (Fraser & Gilbret, 1991) is one of the most prevalent techniques in the design and early development stages. It is the first time the system is presented to end users, and developers have a chance to observe user – system interaction, obtaining invaluable, hands-on information on attitude peculiarities and problems faced with regards to the specific interface.

In a WOZ study, the system is actually a mock up (prototype simulation), and a human acts as the system. The main advantage of the WOZ method lies in the ability to test early, without a working system. Therefore, updates based on feedback are easier, and early detection of design shortcomings that would be costly to fix later is possible.

Furthermore, the dialogues collected can be used as initial training corpus during the implementation of the ASR and NLU modules. On the other hand, the WOZ method faces the disadvantages of end user testing in general. First of all, test participants are not motivated in the same way as real users are, and are often not representative of the end user population.

Earlier studies (Turunen et al., 2006; Ai et al., 2007) have shown that there are differences between usability testing and actual use conditions; main differences lie in the use of barge-in, explicit help requests, significant silence timeouts, speech recognizer rejection rate, use of touchtone, speech rate, utterance length and dialog duration.

Furthermore, as test participants are asked to perform specific tasks, the language used to describe these tasks inevitably influences the participants' choice of vocabulary and utter-

ance structure, undermining the usefulness and reliability of elicited discourse patterns (Harris, 2005). That is especially true in the case of WOZ testing, which usually takes place before final prompt design and specification, and so users may take cues from prompts that will be replaced in the final system.

The realistic aspect is further compromised, as it is difficult for the wizard to simulate speech recognition and interpretation errors. In addition, setting up a WOZ experiment requires tools that can be costly to develop (Weinschenk & Barker, 2000; Jankelovich interview).

Finally, with regards to the utility of the corpus collected for the development of stochastic recognition and interpretation models, the following should be taken into account: given that a typical test session involves 10-15 participants (Cohen et al., 2004), besides lacking the realistic aspect of actual system use, the number of collected dialogs is usually very limited.

Usability testing with working systems. Usability testing is "a process that employs people as testing participants who are representative of the target audience to evaluate the degree to which a product meets specific usability criteria" (Lauesen, 2005). The term "working systems" does not necessarily mean systems that contain the entire intended functionality of the production system. On the contrary, by initially testing the usability of limited functionality systems and gradually adding more modules and functions, user involvement may take place early in the implementation phase and become an indispensable part of an iterative testing, design and build process, probing and refining design choices at each iteration (Kouroupetroglou & Spiliotopoulos, 2009; Spiliotopoulos & Kouroupetroglou, 2010).

Evaluation usability tests using a close-to-market, fully integrated system can then be performed at the end (or near the end) of the implementation cycle, primarily for fine tuning purposes. Optimally, major problems in design should have already been identified, as addressing them at that point is usually a difficult and costly procedure. It should be noted, though, that under fast paced conditions that are typically the case in industry, often dealing with budget constraints as well, the high cost of conducting usability tests in an iterative manner throughout the product's lifecycle is sometimes prohibitive.

A good compromise is to design and test the riskiest parts early in the process. Regarding the quality of the collected resources, the same shortcomings apply as with WOZ testing, only, in this case, tests benefit from the realistic aspect of actual system use. All in all, usability testing should be an indispensable part of the development process, highly important for conciliating business, developer and user view, validating design decisions, identifying problems early in the process, when it is easier to address them, and preventing the release of embarrassing, unusable systems (Galitz, 2007).

Finally, usability inspection methods, such as heuristic evaluation (usability experts examine whether usability principles are met) or pluralistic walkthroughs (group meetings, where stakeholders go through dialogue scenarios) are an important, cost and time effective tool that can be easily utilized throughout the development lifecycle.

Accordingly, Rubin & Chisnell (2008) note: "In some cases it is more effective both in terms of cost, time, and accuracy to conduct an expert or heuristic evaluation of a product rather than test it. This is especially true in the early stages of a product when gross violations of usability principles abound. It is simply unnecessary to bring in many participants to reveal the obvious".

There are a number of other usability inspection methods (e.g. heuristic estimation, feature and consistency inspection); heuristic evaluation is considered to be the most common and beneficial one (Nielsen, 1995). Nevertheless, even the latter can only be complementary to other user-centric techniques, as it often fails to identify a significant proportion of problems (roughly 50%) that real users encounter (Lauesen, 2005). For a detailed

analysis of usability inspection methods readers may refer to Nielsen & Mack (1994).

DESIGNING AVA

For understanding AVA and deciding upon key design features, first we met with technical personnel, explored the existing touchtone system, the company's website and other available documentation. Customer care personnel were able to further provide us with statistics of use based on a detailed segmentation of the customer care domain; all services offered by the department had already been defined and grouped into higher level services in a hierarchical fashion, essentially providing us with a thorough analysis of the application functionality.

Intuitively, though, this analysis seemed to reflect a business rather than a user view of the domain. Our intuition was corroborated by input on keywords and terminology provided by human agents, which substantially differed from the jargon in the company's documentation.

So, in order to gain a better understanding of users' view and attitude, we organized a field study, where we observed live agents performing the task. Unfortunately, monitoring real time calls proved rather ineffective time wise, as within a call we could not skip tasks that were out of the application domain.

Still, we had a chance to get the "look and feel" of the task and interview live agents. Luckily, the application being telephone based, we were given access to existing call records. Call record analysis proved to be a much more effective and useful technique. During the analysis we observed high variation and complexity in users' input in terms of vocabulary and syntax that suggested using more robust methods for interpretation such as NLU classifiers rather than hand crafted NLU grammars. However, building on previous experience and given the inherent differences between human-human and human-machine dialogues, it was necessary to examine actual user-system interaction.

To achieve that at such an early stage in AVA's lifecycle a mock up was developed and "exposed" to real users. The purpose of the mock up was twofold: to aid design and collect high quality corpus for implementation. Only the first – and riskiest – step of the dialogue was simulated, in which the callers ask for the particular service they are interested in.

Following a short message introducing callers to the automated service, a "How may I help you" prompt was played to them and after responding they were directly routed to the existing DTMF system. No actual speech recognition or interpretation was attempted and only one no-input event was allowed. Upon no input a help prompt was played with example utterances and the initial prompt was then repeated.

In designing the mock up application, it was important to have already formulated a basic idea of how the production system would work in terms of dialogue structure and ASR grammar type at least. Both parameters affect the wording of the prompts, and taking into account the observed correlation between prompt wording and the caller's answer, it was important to use prompts as similar as possible to the prompts used in the production system.

Analysis of the simulated part of the dialogue had already indicated that due to the application's complexity a "How may I help you" open end question was the safest choice. Also, care was taken, so that the examples provided in the case of no input were representative of the most frequently asked for services, and avoided confusing jargon. In short, "proactive", strong emphasis on design of prompts ensured the validity and utility of the collected corpus.

Due to its simplicity the mock-up was easy and fast to develop and the heavy call load of the customer care call centre made it possible to collect the necessary amount of utterances within a week. The collected corpus served as the basis

for user centered design, and helped us analyse the users' view of the domain, elicit users' natural discourse patterns, and observe realistic first time user reaction to the introduction of AVA.

To be more specific, corpus analysis revealed significant user deviations from business language as well as the service domain segmentation depicted in company's documentation. On one hand users tended to be rather vague in their requests actually forming super-categories that required disambiguating. Utterances such as "barring" or "activation" were classified as super categories in need of disambiguation, in order to decide upon a unique routing destination.

Such disambiguation sub-dialogues comprise an important part of AVA that could have not been effectively designed and implemented without early access to user-system dialogues. On the other hand, there were many requests for speaking to agents or being transferred to – non existing sometimes – company departments. Such requests do not typically come up in the interaction between clients and human agents.

Furthermore, with regards to user's discourse, utterance structure was far simpler compared to the human-human dialogues previously analysed, rendering the use of hand crafted robust NLU grammars a viable solution. In fact, a significant number of simple one word utterances came up. Still, for some particular services and in the case of a small number of users, syntax and wording displayed higher variation and complexity. As a result we decided to proceed with our initial choice favouring the use of machine learning techniques for the interpretation part of AVA as well. Given that frequency of use was not particularly high for the application and there would always be walk-up-and-use users, we wanted to accommodate these users too rather than force them to adjust to technology limitations.

Finally, we were able to collect a number of different reactions falling under the "volunteer" or "victim" distinction (Attwater et al., 2000), that is a user expecting an automated agent or a human agent respectively. Based on our observations we managed to design a set of help prompts in response to such "victim" caller's reactions. Our observations further served as arguments corroborating the need for notifying customers of the new application beforehand, for example via Short Messaging Service (SMS).

In conclusion, the mock up application was an indispensable part of AVA's lifecycle, providing among others: a) input on real users' discourse patterns, which were in turn used for designing the prompts and building the grammars, b) insight in users' understanding of the domain and the application, which was in turn used for developing the task list and the disambiguation and help sub-dialogues, as well as deciding upon the dialogue structure and the type of grammars, and c) a realistic, "in-service" corpus, which was used for the development of the statistical language models and the NLU classifier presented in the following section. For other, simpler parts of the application such as the Self Service dialogues, heuristic evaluation and walkthroughs were used.

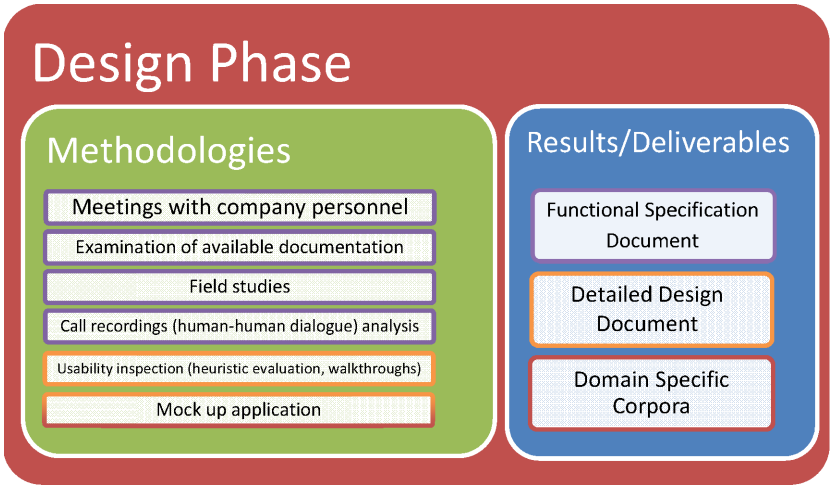
At the end of the design phase, a complete specification of call flows, prompts and back end system communication was prepared and handed for implementation. Figure 4 summarizes the complete design process.

IMPLEMENTATION PHASE: SPEECH RECOGNITION AND UNDERSTANDING MODULES

Broadly speaking, implementation of a voice agent can be broken down into the following processes: a) language modeling and lexicon development for the ASR module, b) grammar development for the NLU component, c) prompt recording, d) dialog coding and e) back end system integration. This chapter focuses on the first two.

A speech recognition model typically used in such applications is comprised of the following:

Figure 4. AVA: design phase



- **Acoustic models**, i.e. models of the language's phones in context. Triphones are usually modeled, that is models of a phone taking into account the effect of the preceding and following phone in its spectrum. Acoustic model sets are in most cases already provided with the speech recognition platform.
- **Dictionary.** The dictionary maps word spellings to pronunciations (i.e. phone sequences). Standard dictionaries are provided with the ASR platform, but most often customized dictionaries also need to be developed to cope with missing pronunciations.
- **Language models.** Language models are grammar networks that constraint the recognizer's search space by specifying permissible word sequences. The type of language model used constitutes a critical feature of an automated voice agent, as "it affects every aspect of VUI design, from the wordings of prompts to dialog strategy, from call flow to the organization of the complete application" (Cohen et al., 2004). Furthermore, the choice of language model used for recognition determines to a great extent the choice of NLU techniques as well.

For the NLU component of commercial spoken dialogue applications, rule based interpretation grammars or robust interpretation grammars are most commonly used, while more advanced stochastic NLU techniques are sometimes also an option.

Following are the most common techniques employed for language modeling in Automatic Speech Recognition (ASR) for practical spoken dialogue systems, along with the NLU techniques that are typically coupled with – interested readers may refer to Huang et al. (2001) for a thorough introduction to language modeling:

- **Rule based context free or non deterministic finite state grammars**, where permissible word sequences are specified by manually written production rules of the form $A \rightarrow \beta$ in which A is a non terminal node and β is a sequence of terminal and/or non terminal nodes. High level concepts such as "Destination" or "City" are typically non terminals, while actual words such as "London" or "Athens" are termi-

nals. Production rules can be augmented with probabilities that allow the recognizer to discriminate among competing recognition hypotheses. Rule based grammars are used both for ASR defining the recognizer's search space, as well as interpretation. In the latter case production rules are augmented with semantic attachments that typically return slot-filling values.

- **Statistical Language Models (SLMs)**, where n-grams are trained on user's utterances to compute the probability of word sequences. N-gram language models compute the probability that a word w will follow given the preceding n-1 words as context. Typically bigram or trigram models are used, where n=2 and 3 respectively. For most applications SLMs are coupled with robust NLU grammars. Instead of parsing the whole string passed on by the recognizer, robust grammars perform word or phrase spotting, searching the string and assigning semantic values to meaningful parts only. Alternatively, for some tasks machine learning techniques can be used for NLU as well. In particular, supervised learning using machine learning algorithms may be used for classification of utterances according to a rich set of features. These features may be word or sentence-related, grammatical, lexical, semantic, such as placement of the words in the utterance, number of words in an utterance, content or functional word flag, part-of-speech, type of phrase, type of utterance (question, verification, disambiguation, explanation, statement, positive/negative, etc.), and so on. All the above can be set for words preceding and following the word that the features are assigned for, if it is deemed necessary.

Rule based grammars are suitable for well defined, simple domains, where users' input is less variable and more predictable. As they are written by hand, there is no need for collecting training data, and they can be easily updated by simply adding more rules to the grammar. SLMs on the other hand are suitable for large, complex domains, where it is hard to predict and manually specify all permissible word combinations in advance. Manually creating such a set of rules can be a hard, time consuming and possibly ineffective endeavor.

The out-of-grammar rate is typically high, and adding more rules often has no improvement in performance. Since the vocabulary size increases, the number of potentially confusable competing recognition hypotheses increases as well. This in turn leads to a decrease in in-grammar recognition accuracy. On top of that, long and complex utterances typically exhibit a higher rate of disfluencies (Shriberg, 1994), such as hesitations, repairs, phrase fragments and filled pauses, which are hard to cope with in a rule based grammar.

In contrast SLMs paired with robust NLU grammars or NLU classifiers lift the need to match the whole utterance string, thus allowing greater flexibility and variation in users' input, and enabling the use of open-ended prompts, less restrictive dialogue strategy and more natural interaction in general. Furthermore, through n-gram smoothing techniques such as probability discounting and backing off strategies, SLMs can accommodate for unknown words and less frequent or unseen sequences.

However, training a SLM requires a large amount of utterances to be collected and transcribed. As an indication, for a typical large scale application of a ~2000 word vocabulary a training set of ~20000 utterances is required. Collecting and transcribing such a corpus can be a cumbersome process that often presupposes the existence of an almost complete or deployed system. Cohen et al. (2004) suggest building an initial smaller corpus, and collecting additional utterances during or after pilot phase. The methods suggested for the collection of the initial corpus

are: a) WOZ testing and b) using a rule based grammar for recognition, which is replaced by a SLM as soon as the required amount of training data is collected. Human-human dialogues could also be used if available. All the above methods face the drawbacks already mentioned, but they can serve as a first, better or worse, approximation to an adequately performing production system.

SLMs can be coupled with robust NLU grammars or NLU classifiers to interpret the recognized string. While robust NLU grammars can more effectively handle disfluencies compared to rule based grammars, there is still problem when it comes to long span grammar rules, as the grammar can only match meaningful parts in a serial, continuous manner. Discontinuous semantic information due to hesitations or scrambled word order may still cause problems. The latter is particularly true for free word order languages, where there is no restriction and therefore greater variation in the order in which syntactic constituents appear within a sentence.

Moreover the developer still needs to write rules by hand. NLU classifiers on the other hand automatically learn these rules from a set of training data. The corpus for training the SLM is also used for training the classifier, so there is no need to collect or transcribe a new corpus. Still, the existing corpus needs to be annotated with appropriate semantic values, which in turn consumes people and time resources. NLU machine learning based classifiers could outperform rule based grammars, as they are:

- more robust to discontinuous semantic information (caused by extraneous, irrelevant input or disfluencies) and "scrambled" word order,
- better at resolving ambiguities, since they are trained as to which interpretation to choose,
- using preprocessing such as stemming to efficiently manage highly inflectional languages, since a stemmer can easily automate the vast amount of rules needed to capture the rich inflectional morphology,
- flexible enough to be allowed to select the best algorithms suited for the specific domain and feature set and even test them in order to decide on the most accurate model to be used.

As an indication of performance, Wang et al. (2002) report an up to 3 percentage points decrease in task classification error rate for support vector classifiers compared to rule based robust semantic parsers. However, NLU classifiers that are trained with recorded single utterance inputs are optimized for returning a single slot in contrast to robust grammars. Thus, that type of classification is more appropriate for applications where complex utterances are mapped to a single concept. A typical example of such applications is call routing. Classification needs to get far more complex in terms of training data and feature annotation in order to be able to predict multiple targets (concepts or concept categories).

On a final note, maintenance and support issues should be addressed and taken into consideration when deciding upon the type of grammar used and when building it as well. In industry fields such as mobile telephony, after a period of time, as new products and services are introduced to the market and others are withdrawn, the recognition and interpretation grammars may no longer achieve high coverage of the caller's input and fail to interpret the caller's request correctly.

In the case of rule based grammars, updates can be more straightforward, as new rules can be more easily, manually added to the existing grammar. In the case of SLMs and NLU classifiers, on the other hand, new utterances need to be collected from scratch, transcribed and annotated, in order to retrain, test and optimize the new, updated models. Nevertheless, as grammar and dialogue/system type are tightly interconnected, updates in rule based grammars may induce significant changes in dialogue structure and prompts.

Since rule-based grammars require properly restricting user's input, in order to be effective, adding or eliminating services (i.e. slots or slot values) typically results in changes in the menu hierarchy and/or the content of prompts. In contrast, statistical models coupled with open ended prompts and less restrictive dialogue strategies enable the sustainment of basic dialogue structure, eliminating the need for redesign and allowing for a smooth transition between old and new system versions. In any case, developers should try to anticipate changes and provide means to easily and quickly cope with these changes.

Finally, with regards to the whole development process, it is important to stress out that implementation is inevitably coupled with testing, including usability testing, often resulting in redesign of initial system parts.

IMPLEMENTING AVA

First, the ASR model was created. In AVA's case the mock up application proved to be the only means for collecting a production quality, "realistic", representative and adequate in size corpus early in the development cycle. Approximately 20,000 utterances were collected and transcribed. 10% of the corpus collected was used as a test set and the rest of the corpus was used for training the SLM.

To achieve greater robustness, classes of words were defined and a class-based language model was trained. Class-based language models constitute an effective way to deal with sparse data, as rarely occurring words of similar semantic function are clustered under the same generalized class; probabilities are then estimated for the generalized class alone and inherited by all words under it. Huang et al. (2001) note that "for limited domain speech recognition, the class-based n-gram is very helpful as the class can efficiently encode semantic information for improved keyword spotting and speech understanding accuracy".

The test corpus was then used to tune the language model and optimize recognition parameters accordingly. The platform default values for most parameters are usually optimal, but developers will still need to optimize pruning, language model scaling and word insertion probability values at least, as well as define language model order (n) and discounting strategy. Finally, domain specific dictionaries were built with pronunciations for words missing from standard dictionaries, mainly involving the domain's jargon.

Next, the NLU components were developed. In accordance with our initial design choices an NLU machine learning classifier was developed for the less restricted, open-ended part of AVA. In order to train the classifier, the existing training corpus was annotated with the correct service tag. Similarly, the test set was annotated and used to optimize the parameters of the classifier.

At the same time, robust sub-grammars were developed, so that particular dialogue states (e.g. confirmation, disambiguation, error-handling and self-service sub-dialogues) could be handled. The latter were tested for interpretation accuracy and coverage and ambiguities to ensure that the test set was completely and correctly interpreted, and ambiguous utterances were appropriately resolved. Furthermore, pronunciation tests were automatically performed to identify missing pronunciations. The results of the latter were fed directly into the development process of the dictionaries for ASR.

In order to cope with the maintenance and upgrade challenges posed by the constant changes and product updates in the fast paced, highly competitive field of mobile telephony, the following solutions were employed:

- Classes for frequently changing products were defined (cf. class based language model), so as to avoid the need for retraining the statistical models every time a product under the predefined class was added or withdrawn. For example, Tariff Plans,

which were subject to frequent changes, formed a typical class for the mobile telephony domain.

- Due to a detailed hierarchical classification of the caller's request, future services were proactively accommodated for. Classification was based on exhaustive service domain ontology rather than available routing destinations. When the service categories falling under the same super-class were routed to the same queue, the model used the super-class for default routing. Maintaining the underlying service breakdown made future possible changes to subcategories easy to handle, while newly added services (new sub-classes) could possibly fall under an existing super-class.
- A system management tool was developed that allowed the customer care department to perform low complexity, yet frequently occurring changes, such as queue reassignments.

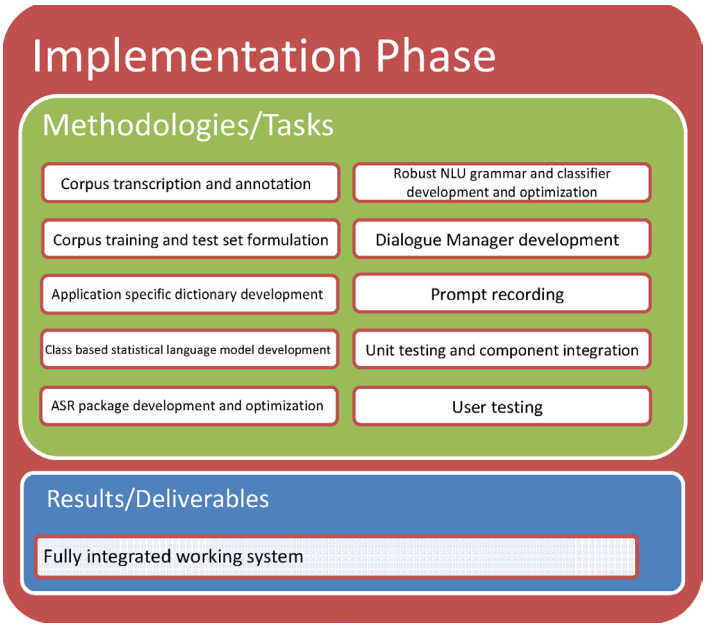
While the ASR and NLU modules were being developed, most of the dialogue manager behaviour had been coded. Once testing for each module was completed, the different components of AVA were integrated. Integration with the back end database and existing CTI (Computer Telephony Integration) software followed. At the end of the phase a fully integrated working system was ready for evaluation, pilot phase, final tuning and full deployment. Figure 5 summarizes AVA's implementation process.

CONCLUSION

The development of a successful automated voice agent depends on both effective underlying technology as well as appropriate design choices. In fact, these two aspects of system development are by no means independent; on one hand, design choices are often restricted or expanded by technology limitations or capabilities, while on the other hand technology effectiveness may be corroborated or undermined by valid or poor design respectively.

In this regard, this chapter focused on both design methodology and implementation techniques,

Figure 5. AVA: implementation phase



analysing the advantages and disadvantages of tools available in the process of building an automated voice agent, illustrating the choice and use of them in light of a real life paradigm. Understanding of the nature, feasibility and effectiveness wise, of each tool is the key in making the best choice possible, as no readily available, fool-proof rules of thumb can always be safely employed. Rather one should focus on the analysis of the specifics of each system separately.

In the case of AVA, the mock up proved to be the optimal technique for both design and implementation, as it provided invaluable resources shared by both phases. In case of other applications such a solution may not even be an option, often for reasons extraneous to the core system engineering perspective such as company policy prohibiting the exposure of an incomplete, mock up system to the entire customer base. In any case, a combination of available techniques should be employed.

Finally, testing was shown to be a key feature embedded in all steps of a voice agent's lifecycle. Usability testing, in particular, constitutes one of the most promising methods for making successful design choices, being part of the user-centered design paradigm. Bringing the user perspective to the design of the voice agent as early in the process as possible, as well as iterating through design, implementation and testing cycles are imperative for the creation of effective and user friendly automated voice agents.

REFERENCES

Ai, H., Raux, A., Bohus, D., Eskenazi, M., & Litman, D. (2007). Comparing spoken dialog corpora collected with recruited subjects versus real users. In *Proc. of the 8th SIGdial workshop on Discourse and Dialogue*.

Allen, J. F. (1995). *Natural language understanding*. Menlo Park, CA: Benjamin Cummings.

Allen, J. F., Dzikovska, M., Manshadi, M., & Swift, M. (2007). Deep linguistic processing for spoken dialogue systems. In *Proceedings of the ACL 2007 Workshop on Deep Linguistic Processing*, Prague, June.

Allen, J. F., Ferguson, G., & Stent, A. (2001). An architecture for more realistic conversational systems. In *Proceedings of Intelligent User Interfaces*.

Allen, J. F., Miller, B. W., Ringger, E., & Sikorski, T. (1996). Robust understanding in a dialogue system. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics.*

Attwater, D., Edgington, M., Durston, P., & Whittaker, S. (2000). Practical issues in the application of speech technology to network and customer services applications. *Speech Communication*, *31*(4), 279–291. doi:10.1016/S0167-6393(99)00062-X

Balentine, B., & Morgan, D. (1999). *How to build a speech recognition application: A style guide for telephony dialogues*. USA: Enterprise Integration Group.

Chu-Carroll, J., & Carpenter, B. (1999). Vector-based natural language call routing. *Computational Linguistics*, *25*(3), 361–388.

Cohen, M., Giancola, J. P., & Balogh, J. (2004). *Voice user interface design*. Addison-Wesley.

Cohen, P. R., & Perrault, C. R. (1979). Elements of a plan-based theory of speech acts. *Cognitive Science*, *3*(3), 177–212. doi:10.1207/s15516709cog0303_1

Core, M. G., & Allen, J. F. (1997). Coding dialogs with the DAMSL annotation scheme. In *Working Notes of AAAI Fall Symposium on Communicative Action in Humans and Machines*, Boston, MA.

Dahl, D. (Ed.). (2004). *Practical spoken dialogue systems*. Kluwer Academic Publishers. doi:10.1007/978-1-4020-2676-8

Fellbaum, K., & Kouroupetroglou, G. (2008). Principles of electronic speech processing with applications for people with disabilities. *Technology and Disability*, *20*(2), 55–85.

Fischer, M., Maier, E., & Stein, A. (1994). Generating cooperative system responses in information retrieval dialogues. In *Proceedings of 7th International Workshop on Natural Language Generation (IWNLG 7),* Kennebunkport, Maine.

Fraser, J., & Gilbret, G. (1991). Simulating speech systems. *Computer Speech & Language*, *5*, 81–99. doi:10.1016/0885-2308(91)90019-M

Freitas, D., & Kouroupetroglou, G. (2008). Speech technologies for blind and low vision persons. *Technology and Disability*, *20*(2), 135–156.

Galitz, W. O. (2007). *The essential guide to user interface design*. Wiley Publishing, Inc.

Garfield, S., & Wermter, S. (2002). *Recurrent neural learning for helpdesk call routing. Lecture Notes in Computer Science 2415/2002, Artificial Neural Networks*. Berlin/Heidelberg, Germany: Springer.

Garfield, S., & Wermter, S. (2006). Call classification using recurrent neural networks, support vector machines and finite state automata. [Springer-Verlag.]. *Knowledge and Information Systems*, 9.

Gorin, L., Riccardi, G., & Wright, J. H. (1997). How may I help you? *Speech Communication*, *23*, 113–127. doi:10.1016/S0167-6393(97)00040-X

Gould, J. D., & Lewis, C. (1985). Designing for usability: Key principles and what designers think. *Communications of the ACM*, *28*(3), 300–311. doi:10.1145/3166.3170

Grosz, B. J., & Sidner, C. L. (1986). Attention, intentions, and the structure of discourse. *Computational Linguistics*, *12*(3), 175–204.

Gupta, N., Tur, G., Hakkani-Tur, D., Bangalore, S., Riccardi, G., & Rahim, M. (2006). The AT&T spoken language understanding system. *IEEE Transactions on Speech and Audio Processing*.

Harris, R. A. (2005). *Voice interaction design: Crafting the new conversational speech systems*. Elsevier.

Hempel, T. (2008). *Usability of speech dialog systems: Listening to the target audience*. Springer-Verlag.

Hirschberg, J., Nakatani, C., & Grosz, B. (1995). Conveying discourse structure through intonation variation. In *Proceedings of the ECSA Workshop on Spoken Dialogue Systems: Theories and Applications*, Visgo, Denmark.

Huang, X., Acero, A., & Hon, H. W. (2001). *Spoken language processing: A guide to theory, algorithm and system development*. Prentice Hall PTR.

Jurafsky, D., & Martin, J. H. (2000). *Speech and language processing. An introduction to natural language processing, computational linguistics, and speech recognition*. Prentice-Hall.

Kamm, C. A., & Walker, M. A. (1997). Design and evaluation of spoken dialogue systems. In *Proc. of the IEEE Workshop on Automatic Speech Recognition and Understanding*, Santa Barbara (CA), 14–17.

Kouroupetroglou, G. (2009). Universal access in public terminals: Information kiosks and ATMs. In Stephanidis, C. (Ed.), *The universal access handbook* (pp. 48.1–48.19). Florida, USA: CRC Press. doi:10.1201/9781420064995-c48

Kouroupetroglou, G., & Spiliotopoulos, D. (2009). Usability methodologies for real-life voice user interfaces. [IJITWE]. *International Journal of Information Technology and Web Engineering*, *4*(4), 78–94. doi:10.4018/jitwe.2009100105

Kowtko, J., Isard, S., & Doherty, G. M. (1993). *Conversational games within dialogue*. Research paper 31, Human Communication Research Centre, University of Edinburgh.

Larson, J. A. (2002). *VoiceXML: Introduction to developing speech applications*. NJ: Prentice Hall.

Lauesen, S. (2005). *User interface design: A software engineering perspective*. Addison-Wesley.

Lee, C., & Chang, J. S. (2002). *Rapid prototyping an operator assisted call routing system*. ISCSLP 2002, Taipei, Taiwan.

McGlashan, S., Burnett, D. C., Carter, J., Danielsen, P., Ferrans, J., & Hunt, A. … Tryphonas, S. (2004). *Voice Extensible Markup Language (VoiceXML) version* 2.0. Retrieved from http://www.w3.org/TR/voicexml20.

McTear, M. F. (2004). *Towards the conversational user interface*. Springer Verlag.

Nielsen, J. (1995). *Technology transfer of heuristic evaluation and usability inspection*. Presented at the IFIP INTERACT'95 International Conference on Human-Computer Interaction, Lillehammer, Norway.

Nielsen, J., & Mack, R. L. (1994). *Usability inspection methods*. New York, NY: John Wiley & Sons.

Norman, D. (1988). *The design of everyday things*. New York, NY: Doubleday/Currency.

Norman, D. A., & Draper, S. W. (1986). *User centered system design: New perspectives on human-computer interaction*. Mahwah, NJ: Lawrence Erlbaum Associates.

Noth, E., Batlinera, A., Warnkea, V., Haasa, J., Borosb, M., & Buckowa, J. (2002). On the use of prosody in automatic dialogue understanding. *Speech Communication*, *36*(1-2), 45–62. doi:10.1016/S0167-6393(01)00025-5

Pellom, B., Ward, W., Hansen, J., Cole, R., Hacioglu, K., & Zhang, J. … Pradhan, S. (2001). *University of Colorado dialog systems for travel and navigation*. HLT-2001, San Diego.

Pieraccini, R., & Huerta, J. M. (2008). Where do we go from here? Research and commercial spoken dialogue systems. In Dybkjaer, L., & Minker, W. (Eds.), *Recent trends in discourse and dialogue*. Springer.

Pitt, I., & Edwards, A. (2003). *Design of speech-based devices: A practical guide*. Springer.

Pulman, S. (2002). Relating dialogue games to information state. *Speech Communication*, *36*, 15–30. doi:10.1016/S0167-6393(01)00023-1

Riccardi, G., Gorin, A. L., Ljolje, A., & Riley, M. (1997). A spoken language system for automated call routing. In. *Proceedings of ICASSP*, *1997*, 1143–1146.

Rubin, J., & Chisnell, D. (2008). *Handbook of usability testing, 2nd edition: How to plan, design, and conduct effective tests*. Wiley Publishing, Inc.

Shriberg, E. (1994). *Preliminaries to a theory of speech disfluencies*. PhD thesis, University of California, Berkeley, CA.

Sidner, C. (2004). Building spoken-language collaborative interface agents. In Dahl, D. (Ed.), *Practical spoken dialogue systems*. Kluwer Academic Publishers. doi:10.1007/978-1-4020-2676-8_10

Spiliotopoulos, D., & Kouroupetroglou, G. (2010). Usability methodologies for spoken dialogue Web interfaces. In Spiliotopoulos, T., Papadopoulou, P., Martakos, D., & Kouroupetroglou, G. (Eds.), *Integrating usability engineering for designing the Web experience: Methodologies and principles*. Hershey, PA: IGI Global. doi:10.4018/978-1-60566-896-3.ch008

Spiliotopoulos, D., Stavropoulou, P., & Kouroupetroglou, G. (2009). Spoken dialogue interfaces: Integrating usability. *Lecture Notes in Computer Science*, *5889*, 484–499. doi:10.1007/978-3-642-10308-7_36

Stavropoulou, P., Spiliotopoulos, D., & Kouroupetroglou, G. (2011). *Resource evaluation for usable spoken dialogue interfaces: Utilizing human – human dialogues*. In preparation.

Stent, A., Dowding, J., Gawron, J. M., Bratt, E. O., & Moore, R. (1999). The CommandTalk spoken dialogue system. In *Proceedings of the Thirty-Seventh Annual Meeting of the ACL*, (pp. 183-190).

Turunen, M., Hakulinen, J., & Kainulainen, A. (2006). Evaluation of a spoken dialogue system with usability tests and long-term pilot studies: Similarities and differences. In *Proceedings of Interspeech.*

Wahlster, W. (2000). *Verbmobil: Foundations of speech-to-speech translation*. Berlin, Germany & New York, NY: Springer.

Walker, M. A. Langkilde-Geary, I., Wright Hastie, H., Wright, J., & Gorin, A. (2002). Automatically training a problematic dialogue predictor for the HMIHY spoken dialogue system. *Journal of Artificial Intelligence Research (JAIR)*.

Wang, Y., Acero, A., Chelba, C., Frey, B., & Wong, L. (2002). Combination of statistical and rule-based approaches for spoken language understanding. In *Proceedings of the International Conference on Spoken Language Processing*, Denver, CO.

Weinschenk, S., & Barker, D. T. (2000). *Designing effective speech interfaces*. John Wiley & Sons, Inc.

Williams, J. D., & Witt, S. M. (2004). A comparison of dialog strategies for call routing. [Springer Netherlands.]. *International Journal of Speech Technology*, 7.

Zitouni, I., Hong-Kwang, J. K., & Chin-Hui, L. (2003). Boosting and combination of classifiers for natural language call routing systems. *Speech Communication*, 14.

KEY TERMS AND DEFINITIONS

Automated Call Routing Application: Interactive Voice Response (IVR) based application that automatically routes incoming phone calls to appropriate destinations. Intelligent routing can be based on parameters such as DTMF or voice input interpretation, caller identification or time of day.

Automated Voice Agent: Program capable of communicating with users by both understanding and producing speech within a specific domain. It is typically comprised of the following basic modules: the Automatic Speech Recognition module that converts acoustic user input into text, the Natural Language Understanding module that semantically interprets it, the Dialogue Manager that handles the conversation flow, the Natural Language Generator that generates system prompts in written form, and the Text to Speech Synthesizer that converts the written prompts to speech.

NLU Machine Learning Based Classifier: It is a system programmed to automatically learn to recognize complex patterns and make intelligent decisions based on data; the difficulty lies in the fact that the set of all possible behaviors given all possible inputs is too large to be covered by the set of observed examples (training data). Trained for natural language understanding, it automatically extracts one or more possible interpretations from a single natural language input.

Robust Natural Language Understanding (NLU) Grammar: Rule based word spotting interpretation grammar. Instead of parsing the whole user utterance, a robust grammar performs word or phrase spotting, searching the utterance and assigning semantic values to meaningful parts only.

Rule Based Grammar: A context free grammar, where permissible word sequences are specified by manually written production rules of the form $A \rightarrow \beta$ in which A is a non terminal node and β is a sequence of terminal and/or non terminal nodes. It is used for speech recognition – restricting the recognizer's search space – as well as speech interpretation – augmented with slot filling semantic rule attachments.

Statistical Language Model (SLM): N-gram model trained on domain specific corpora in order to compute the probability of word sequences. Used for Automatic Speech Recognition, it is essentially a model of what the callers are likely to say when interacting with the system.

Usability: Attribute that refers to various types of interfaces, measured and described in terms of usefulness, effectiveness, efficiency, learnability and user satisfaction. It denotes the extent to which a system can be used to achieve the goals it was designed for with accuracy, completeness and speed in a user-friendly, easy-to-use and learn manner.

Usability Testing: Set of methods and schemes for assessing the user's experience when interacting with a system and evaluating usability attributes such as effectiveness, efficiency and user satisfaction. Typical usability tests for speech based systems include Wizard-of-Oz testing, user testing with limited functionality or fully working systems, usability inspection etc.