

Matt Prather

R.A.D.I.C.A.L.

A Model for Developing Voice Interfaces

By Matt Prather, 2009

R.A.D.I.C.A.L.: A model for developing voice interfaces

Executive summary	3
Understanding voice applications	4
Deconstructing a voice application	4
Putting it together: a simplified diagram	5
The RADICAL process	6
RADICAL components and phases	7
Methodology inputs and deliverables	12
Glossary	13

R.A.D.I.C.A.L.: A model for developing voice interfaces

Executive summary

Introduction

Despite the advent of ever more impressively powerful technologies, our first means of human communication remains one of our most powerful, as well: spoken conversation is still one of the most natural and effective ways for people to access information and conduct business. Increasingly, technology itself enables this “birthright” means of human communication; with the introduction of affordable mobile services, the growth of wireless handsets is estimated to have exceeded one billion—with a “b!”—some time in 2003. This makes telephones by far the most ubiquitous network device. Since voice recognition servers are accessible from any telephone, they become the perfect devices for offering services to the mass market and capitalizing on the built-in communications medium that comes as “standard equipment” on every human.

With the rapid advances in computer processing and language/speech algorithms, telephony-based voice recognition systems can now “understand” natural spoken conversation (or very closely approximate such understanding), and act upon it to realize all kinds of user goals and activities—things that would have been unimaginable just a few short years ago. Since the mid-1990s, increasing numbers of prominent blue-chip companies have operated VCommerce (voice-driven e-commerce) systems to deliver services ranging from flight information and real-time stock trading to account creation or cancellation, using both voice recognition and voice authentication. The horizons for effective and even pleasurable voice-interface deployments are as boundless as the human imagination.

The purpose of this paper, then, is to help lead the reader through the development paths and procedures that lead to those boundless horizons, by:

- Explaining what makes a speech application, including the differences between “speech” and “language” applications, VUI components and supporting infrastructure.
- Providing a clear and complete methodology (the “RADICAL” system) and protocol for creating voice-enabled applications.
- Describe application enablers and development tools that vendors offer for rapid deployment of speech applications.
- Describe the suite of speech application and voice interface services that most vendors and/or their partners can provide.

Understanding voice applications

Overview

Your company or organization may be investigating the possibility of using a speech recognition or voice authentication system to

- reduce costs while generating revenue, and
- improve customer service.

First, though, it's important to understand what steps are involved in making this a reality and a success. Introducing speech applications into a company calls for thorough planning in a consistent fashion and with a thorough understanding of both the strategic (your goals, what you hope and plan to accomplish) and tactical (how you plan to reach those goals) components.

Voice User Interfaces (VUIs) are different from other interface projects, however; there are certain aspects that are unique to voice-enabled applications. VUIs operate differently than their GUI counterparts and are used differently by their target audiences, so it becomes even more important to carefully design a voice application so that the VUI precisely meets the needs and expectations of the target audience.

Despite the differences between Graphical and Voice user interfaces, the proper combination of tools (technology), technique/process and team, designing a VUI can be straightforward and rewarding.

Deconstructing a voice application

Voice-enabled solutions, as you might expect, require a number of key or typical components, regardless of the particular vendor or company providing the voice application. These include

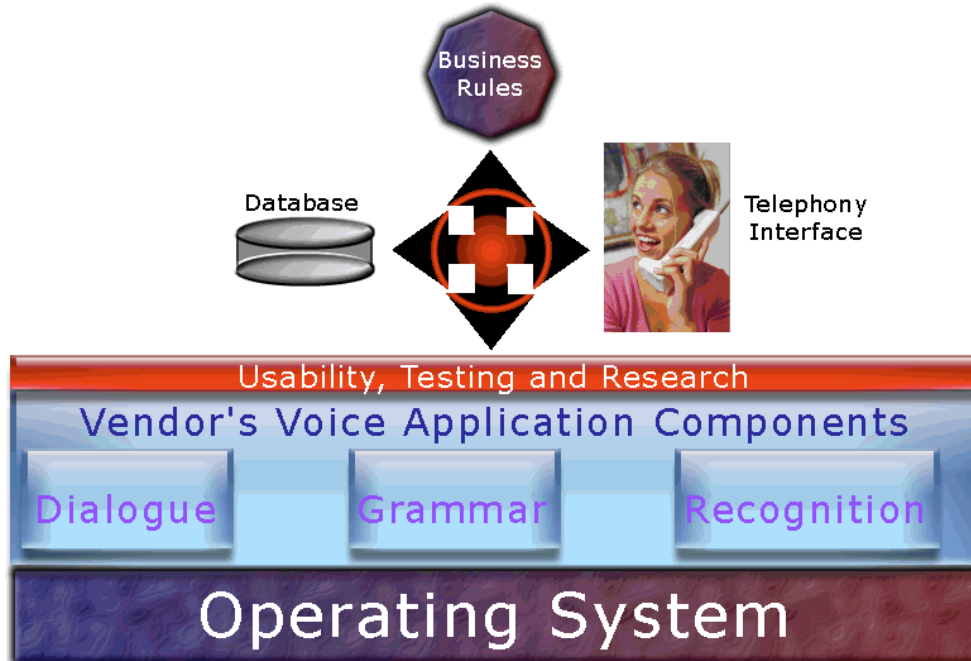
- the speech application platform (for example, will the speech application be hosted on Windows, UNIX, or Linux machines?)
- hardware and software
- speech software and servers (typically unique to the voice-application vendor—for example, the servers and software necessary to interface with the General Magic magicTalk platform or the Nuance or ScanSoft platforms)
- the telephony interface (this enables the speech application to interact with users by telephone, as well as to respond to telephony conventions such as TouchTone signals)
- application servers
- the host functionality and database (this is where the particular information about your application is stored—for example, what is or is not “in grammar” or considered a recognizable and actionable thing for users to say)
- the system integration of all of these components.

Most vendors now offer their core speech software on all leading operating systems and there is increasing availability and support on the popular Linux operating system. Consider maintenance and support costs of your chosen operating platform, not just the initial acquisition costs. It could be that you and your company save money in the long run by investing a bit more upfront to build your voice application on a stable and scalable platform with readily available support options.

Similarly, vendors typically support all market leading IVR and telephony platforms such as Lucent and Octel. Figure 1 below is a simplified illustration of the key components of a speech application.

Putting it together: a simplified diagram

Figure 1: Typical construction of a voice application (simplified)



- The choice of **operating system or “platform”** can affect the scalability of your voice application, as well as long-term operating costs. Even if you’re not serving lots of customers now, consider whether you’ll need your voice application to be able to grow quickly and smoothly as your customer base grows.
- **Dialogue** — The vendor and designer of your voice application supplies this component, which corresponds roughly to the idea of a “conversation” (or at least, one end of a conversation) and contains recorded voice prompts, call flows (the “logic” that helps determine what prompt to play at what time) and responses.
- **Grammar** — The grammar determines what is or is not valid and considered for recognition, what the application will attempt to recognize and act upon. For example, a grammar might include “Yes” and many other synonyms such as, “Uh huh,” “Mmm hmm,” “Yeah,” “Sure,” and “Okay,” but might not include such things as “You betcha!” The application will attempt to recognize something that sounds close to the elements in grammar, but will not try to process something like “You betcha!” that is not part of the grammar.
- **Recognition** — This portion does exactly what the name implies: it attempts to match user utterances to known words and phrases in the grammar.
- **Usability, Testing, and Research** — This critical component can’t be touched or held, but it is vital to developing and fine-tuning a voice interface experience that satisfies you, your company, and your customers. Only by testing and refining the interface can you know if you have truly met your customers’ needs.
- **Databases** — The backend databases operate in conjunction with the business rules, voice application, and telephony interface to inject customer data or other information as needed, to assemble a smooth and informative voice user experience.
- **Business Rules and Telephony Interface** — The business rules determine the fundamental logic of the application and customer needs, while the telephony interface provides a means of delivering that information via the customer’s telephone (through landline, wireless, VOIP, or other emerging technologies).

The RADICAL process

A mnemonic, a protocol, a sequence of procedures

As you have probably guessed by now, RADICAL represents a mnemonic “memory jogger” acronym, as well as a protocol and sequence of procedures designed to help ensure that you get the most from your voice-application development process. The RADICAL process breaks down as follows:

- **Requirements Analysis.** While this step represents the “RA” portion of RADICAL, it’s important to note you must of course first gather requirements before analyzing them. Remember, too, that this applies to multiple kinds of requirements in multiple areas:
 - Business requirements. What do you and your organization want and need for your voice application to achieve or accomplish?
 - User requirements. What do your customers, subscribers, and users want and need for your voice application to achieve or accomplish?
 - Application requirements. What does your application need to do in order to satisfy the business and user requirements?
- **Design.** Armed with a thorough understanding of the requirements, this phase includes:
 - Prototyping, from low-fidelity and “Wizard of Oz” prototypes (in which you simulate the system and have users “ignore the man behind the curtain”) to high-fidelity simulations.
 - Dialogue and call flow design.
 - Character design. What will be the “personality” of your voice application? Will it be a sassy 20-something from Brooklyn or comforting 60-something Aunt Evelyn from Omaha? What will be most appropriate and most successful for your audience?
 - Audio design. This is the “signature sound” of your voice application, the distinctive sound that sets it apart from its competitors or imitators.
 - Application design. How is this different from Audio design? In this component, the designers are primarily concerned with how the application will function, how it will incorporate all of the preceding elements.
- **Implementation.**
 - Implementation involves actually “building” out the designs that have been scoped, prototyped, tested, and refined. At this point, the design of the character(s), dialogues, and application appear solid, stable, and reliable enough to warrant full-scale implementation and testing.
- **C**heck, **A**nalyze, and **L**earn
 - Post and deploy a pilot program. No matter how good your iterative process, it will still only provide research results within a controlled environment. A pilot program, in which real users in real environments with real issues needing real solutions use your voice application, will provide an opportunity to check your efforts against the demands of an “in the field” deployment.
 - Monitor and research the user experience “in the field” and commit to concrete steps for putting the results to good use in the next version of this voice application.
 - Analyze the data you receive, early and often. Identify areas in which you might be able to tune and refine your voice application against real-world demands placed on it.
 - Learn from every user experience with your voice application. Don’t just rely on success stories. While they’re great for the ego and the team’s résumés, remember that the not-so-successful customer stories can sometimes be even more instructive. Plan to use everything you learn from this time around to build an even better voice application next time around!

RADICAL components and phases

Requirements Analysis

The success of any project depends on a thorough and shared understanding of the project goals, objectives, and metrics. After all, if you don't know where you're going, how will you know when you get there? (Or whether you've ended up somewhere else?) A complete analysis of requirements includes not only the desired goals and objectives but the risks involved in realizing those goals, the scope and complexity of the project and its constituent components, a definition of the purpose and audience of the project, and the intended rewards for having realized the project goals and objectives.

- **Business requirements.** This phase precedes all others and attempts to answer such questions as why the application is being built, what it intends to achieve, how it will pay for itself, how it will generate revenue over and above its cost, how it will distinguish itself from (and, ideally, improve upon) its competitors, and how successful implementation will be measured.
- **User requirements.** Once the business requirements have been identified, it's time to look at the requirements of your users. Why will users call or interact with this application? How will they use it? What will their usage patterns be? Who will be using it? What do they need it to do? What will they need it not to do? What will they want it to sound like, to say, to do, to anticipate, to listen for? What type of "character" will they want to interact with? What sort of environment will they be in while they interact with this application?

Ultimately, answering—or at least, asking, even if the questions can't be fully answered—these questions will help realize the goal not only of satisfying user expectations, but of exceeding them, and that's what will make both your voice application and your bottom line stand out from the crowd.

- **Application requirements.** What will the application itself need to do in order to realize the business and user requirements? How complex will speech recognition need to be? Will this application need a high level of naturalness or can it be more structured and perhaps "artificial"? Will it need large grammars, databases, database integration schemes, application platforms? Will it need additional supporting technologies such as text-to-speech? All of these questions must be addressed to determine the scope of the voice application development project.

These high-level requirements are then collated into a Project Requirements Document (PRD), the first step toward returning an accurate quote for pricing and scheduling of the voice application project. This quote, along with a description of the services to be rendered, becomes part of a detailed Statement of Work (SOW). Once a signed SOW is received, Colla Voce can begin the project and initiate the design phase of RADICAL.

Design

A key focus of design is not only to construct the "scaffolding" of the application based on the identified requirements, but to then build out the overall application framework, including

- the introductory behavior of the application
- menu states (more on this in a moment)
- universal functionality of help and error responses (how will the application handle the user's need for help or correction, across all aspects of the interface?)

In some cases, the vendor may be able to offer a pre-packaged design for a specific market, that can dramatically streamline the design phase.

The design phase involves three distinct, though often simultaneous, components:

- **Dialogue Design**—This step involves first producing high-level call flows, the logic that determines “then what happens?” at various points within the user’s call to the voice application. It then moves into detailed dialog design.

The detailed design process includes writing a description of every call state and how each state transitions to another state. It requires

- a framework for specifying call flow structure
 - state-specific prompts and natural language results
 - sample phrases for each state’s grammar
 - help and error recovery
 - universal behaviors (applicable in every state, throughout the application)
 - sample dialogs
- **Character Design**—Extensive research indicates that any time a caller interacts with a speech recognition system, the caller perceives the system to have a personality or “character”. This occurs even when the system was not specifically designed to have a character, and appears even in the people who develop the application itself. It is, therefore, important to design a character that is both appropriate and agreeable to the target audience (if your target audience is pre-teen girls, for example, and you are a near-retirement grandparent, you may not share your audience’s perspective on what is appropriate or agreeable). Use of an appropriate character helps create comprehension, ease of use, and a positive user experience.

Relating dialogue and character design

If you’re thinking, at this point, that character design and dialog design seem similar and closely linked—you’re right. In order to create a consistent, predictable, and enjoyable user experience, the character and personality of the voice application must infuse the entire dialog design process; it must be clearly detectable in every piece of the voice application’s interaction. Early in this stage, the character designer can create one or more fictional “autobiographies” for potential characters of the system.

Most vendors can also provide a number of off-the-shelf, standard personas to meet a variety of application needs in a cost-effective manner. Some vendors, in fact, have based their entire platform design on their ability to provide characters and “personality” in their voice applications, tailored to the particular needs of specific vertical market implementations.

Next, based on the initial call flow provided by the dialogue designer, the character designer creates a sample interaction between a caller and the application—almost as if s/he were writing a one-act play. The interaction is scripted and may then be recorded using a variety of voices playing a variety of characters. The resulting audio samples are shared with the customer and potential users, in order to determine an ideal match for the “voice of the system”.

Once the customer selects a character, a tightly focused character design effort begins, in which all baseline prompts are edited to evoke that character and ensure clarity and comfort in the interaction. These prompts are then incorporated into the dialog design specification, ready for the customer to approve.

Audio branding and other non-verbal audio—This stage is also when other non-speech and non-verbal audio is considered, including:

- audio logos for branding (example: the distinctive “bing, bing, BONG” chimes of AT&T, or the “bor-del-DEEEP” signal that immediately tells you “the number you have dialed is not a working number...”)
- sounds to cover latency (what will the user hear while the application retrieves information or acts on a user request?)
- “markers” to instantly identify place and functionality by way of effects or music (example: the sound effects of TiVo, used to identify actions, unavailable actions, and “TiVo Central”. These audio cues work the same way within a VUI.)

- **Application Design**—The emphasis in this component of the Design phase is integration: determining, finalizing, and documenting exactly how the various components of the application will be seamlessly integrated and how it will achieve the goals and requirements laid out in the Requirements Analysis phase. This includes
 - planning for the installation of the recognition software
 - storage of the application (this could involve substantial investment not only in server space but in the physical space to house the servers, as well)
 - access to the recorded prompts and other audio files
 - connecting the application to the platform and any required DTMF (popularly known as TouchTone) backup
 - providing secure and reliable read/write access to back-end databases or data feeds.

Any required business rules that go beyond the dialog design are also defined (such as capturing the caller's number, either to greet the caller by name or perhaps to detect a first time caller for different prompting), and the estimated size/complexity of the necessary application grammars are scoped.

Throughout the design phase, iterative usability and design procedures and rapid repeated prototyping create a “test and refine” loop to optimize the interaction of dialog design and character design prior to a fullscale implementation. Early, incremental, and repeated usability testing is recommended on every project; it speeds time to market and ensures a design that creates a compelling user experience. It's important to remember, though, that the findings from your usability testing must be used, in order to be of value to you and to your users. The design phase provides the perfect time and opportunity to make low-cost adjustments that will reap big financial and customer-satisfaction benefits.

Implementation

The implementation phase involves several parallel or concurrent tasks or components: the application is built, prompts are recorded and the recognition package is completed. Because the dependencies and interfaces between these tasks (such as naming of the states, prompts and grammars) were addressed and documented in the design phase, these tasks can occur concurrently. The three key components of implementation are:

- **Production**—A team of specialists in voice talent coaching, prompt recording, and audio environment development ensures that the project delivers on goals for the character and overall audio experience, and that prompts are recorded in a fashion that facilitates clarity of communication, user comfort and user control. Specific tasks include:
 - Identify and hire specific voice talent that fits the character
 - Approve and certify studios for prompt recording to ensure the highest standards of voice recording for the specific application (some voice-application platform vendors either have their own studios or have cost-effective solutions available)
 - Produce the final prompts on-site or in certified facilities around the world
 - Produce and combine non-speech audio with prompts
- **Grammar Development**—The goal of grammar development is to anticipate what a user might say. The grammar, in conjunction with the recorded prompts, is developed in a manner that encourages the caller to say something that will be understood. Usually, the grammar developer will begin by leveraging a library of existing grammars associated with different types of applications, to define acceptable user utterances and their associated natural-language interpretation. Then, it's time to add to or edit those grammars, or create custom grammars, as required for the application. Don't forget to plan for and include frequent updates for information that needs to be constantly maintained such as names of US equities, mutual funds, and international airport/airline information. telephone area codes, or other information that changes or expands frequently.
- **Application Development**—This step involves coding the core speech application using any language supported by the vendor APIs (VoiceXML is becoming one of the most popular languages and now appears in many vendor-specific “flavours”, but other languages include C++, Java with SpeechObjects, and magicTalk). Additional development may

be necessary if external data sources are required by the application (e.g. stock quote feeds, audio feeds, customer databases or a host).

Before beginning an in-depth testing stage, the team performs unit tests on each separate component of the application. This involves the testing of each component in the design (i.e., dialog, persona, and application), grammar testing, and testing of each application module. The creator of each component (Colla Voce, a Colla Voce partner, the application platform vendor, or the customer) is responsible for conducting the respective unit test. Upon completion of unit testing, the application components are integrated and prepared for full application testing in the next phase.

Check, Analyze, and Learn

“Checking”, in this context, means thoroughly testing the application as a whole and “checking” its performance against defined goals and objectives as outlined and documented in the Requirements Analysis phase. Once the application is developed and all individual components have been tested, it’s time to adequately test the full application leading up to a pilot program to check its performance in the “real world”. A quality assurance test plan helps manage these efforts.

In testing the application as an integrated deployment, the team conducts three primary types of test:

- **Application Test**—Verify that the application is functional, that it works as designed, and that all hardware and databases can be accessed Application tests include:
 - DialoguePathTest—A series of test cases that cover all possible paths through the dialogue. This test determines whether the right prompts are played (at the right time), whether each state in the call flow is reached under all of the proper conditions, and whether all universal, error, and help behaviors function as designed.
 - SystemQATest—Rigorous, system-wide validation of the overall integration of the application with the platform hardware and APIs, as well as external interfaces such as databases, live data feeds and telephony.
 - SystemLoadTest—Simulation of a high in-bound call volume to the new system. This test ensures that the system can handle expected call volumes (ideally, that the system can handle more than the expected call volume), and that the load on the system can be properly balanced for optimal performance.
- **Recognition Test**—After the dialog path test is completed, and any required modifications are made to the application to ensure compliance with the dialog specification, the next step is to test the recognizer to determine that it has been correctly integrated and is performing well. This typically involves test scripts for callers to generate utterances by talking to the application. These utterances are then transcribed and scored to determine the accuracy of the recognition results when compared with the transcript.
- **Usability Test**—While usability tests are also conducted early in the design process, it is often helpful at this stage to validate the performance of an application against the metrics laid out by the customer in the requirements analysis phase. Think of usability testing in this phase as “course correction”, to ensure that the implementation is still aligned with the requirements, designs, and specifications.

Another integral part of the “Check, Analyze, Learn” phase of the RADICAL process is tuning the application as a result of checking (testing) the application’s performance and analyzing the results. As always, it is what is done with the results of the analysis that truly matters most.

Tuning is critical to ensuring that the system functions effectively and remains optimized for the user group as the service rolls out geographically and as users become more experienced. It is only through capturing, transcribing and analyzing live caller utterances that the application can be tuned for optimal performance. Because it occurs as a result of repeated periodic usability testing, tuning should be performed in conjunction with such testing during the pilot(s) and, optimally, during post-deployment.

Regardless of when they are performed, tuning efforts focus on three principal areas: grammar, recognition, and dialog tuning.

- **Grammar tuning** analyzes and corrects the grammar to increase the grammar coverage while removing unnecessary words, thereby resulting in overall improved system response and accuracy.
- **Recognition tuning** focuses on conducting experiments to determine the optimal search, confidence, and acoustic parameters of the software to reach the ideal balance of accuracy and speed.
- **Dialog tuning** ensures that the prompts correspond appropriately to the grammars and that the dialog flow meets user expectations, consistent with the designed character, the design, and the documented requirements.

User experience and live caller research contribute valuable feedback from real users during the tuning process, and this is where the **L**earn phase of RADICAL comes into play. This phase is typically conducted in three steps:

- **Pilot Analysis and Tuning**—In order to optimize the performance of the system before a full deployment, the full application is made available through a pilot to a limited number of users. Then, transcriptions are analyzed, looking for such performance metrics as
 - in-grammar accuracy
 - out-of-grammar rates
 - error distribution

A performance report, submitted to the customer or partner, details the results of the pilot analysis. The design and development team then conducts experiments with modified grammars and/or parameters until recognition performance is optimized for the data before releasing a revised grammar and/or parameter recommendations for use in updating the system.

To be effective and serve as a valuable learning opportunity, a true pilot program must be an iterative process. Achieving optimum application performance generally requires at least three cycles of data-collection, reporting and modification for each pilot. For some applications, more than one pilot program may be in order: the first with internal or “friendly” users and the second with a larger set of callers selected from the target end- user population.

Once pilot analysis and tuning is complete, and the customer is comfortable with the application performance, it can be moved into production and deployed to the full user base. This deployment may be coupled with a targeted communication to new users, or a wider media campaign, to introduce the new service.

It's also advisable to use this phase to reevaluate the application hardware requirements, which are heavily dependent on the size of the grammars and complexity of the dialog. Although this is estimated at the very beginning of the project, periodically sizing the hardware based on real call data ensures the application is provisioned correctly, and will be ready to accept the expected volume of calls.

- **Post-Deployment Analysis and Tuning**—The same iterative process used during the pilots can be used to
 - identify additional optimizations based on data collected from the full caller population
 - track changes in usage due to increased caller familiarity or changes in the caller population.
- **Performance Monitoring and User Experience Research**—Schedule performance monitoring at regular intervals to check that the system is still maintaining peak performance. Adding additional callers often introduces new—and unpredictable— variables that may also need to be factored into a system's design. Colla Voce recommends two research techniques for gaining immediate insight from actual system users.
 - Interview callers immediately following their interaction with a particular application to collect usability data while it's still “fresh in the mind”. This ensures that callers remember enough of their interaction to provide detailed feedback on the application.

- User Experience Research also allows interaction with live users, but is designed to get feedback over time. This research technique is valuable for frequently used applications, where the design should accommodate the differences between frequent and infrequent callers, and potentially encourage particular behavior (for example, moving to natural-language usage with ability to “barge in” and interrupt the voice application).

Methodology inputs and deliverables

Phase	Inputs (to IDG)	Deliverables (from IDG)	Criteria for completion
Requirements Analysis	<ul style="list-style-type: none"> • Documented requirements (See earlier discussion) 	<ul style="list-style-type: none"> • Proposal of services rendered • Statement of Work 	<ul style="list-style-type: none"> • Signature approval of SOW • Completed purchase order from client
Design	<ul style="list-style-type: none"> • Proposal of services rendered • Completed and approved SOW 	<ul style="list-style-type: none"> • Dialogue Design Specification (including any supporting documentation such as use-case scenarios) • Character definition(s) • Voice talent selected and retained • System architecture • Audio environment defined 	<ul style="list-style-type: none"> • Client signoff approval of design specifications • Client signoff approval of character definitions • Client signoff approval of voice talent
Implementation	<ul style="list-style-type: none"> • Design-phase outputs 	<ul style="list-style-type: none"> • Production (audio, recorded prompts, etc.) • Code the application • Recognizer (and supporting modules or components) 	<ul style="list-style-type: none"> • Successful unit testing of each application module
Check, Analyze, Learn	<ul style="list-style-type: none"> • Integrated application • QA test plan • Application in “ready” status for pilot programs and tuning 	<ul style="list-style-type: none"> • Dialogue-path test results • Integration results • Load test results • Recognition test results • Performance reports • Optimized recognition packages • Tuned grammars • Research results from live, real-world users 	<ul style="list-style-type: none"> • Successfully meeting all test standards • Application meets or exceeds all metrics, standards, and client expectations

Glossary

- **Barge-in**—The ability of a caller or user to interrupt a recorded prompt or message to give a voice command.
- **Character**—The “personality” of the voice application, usually assigned a name and a biography and reflected in the application through voice, inflection, use of language, and audio environment. A character is typically designed for each application (sometimes more than one character per application) and is based on the needs of the target audience for that application.
- **Dialogue**—The “conversation” or interaction between the voice application and the caller or user.
- **DTMF**—Dual-Tone Multi-Frequency, more widely known as TouchTone or telephone keypad entries.
- **Grammar**—The set of caller or user utterances considered valid at any given point or state within the voice application.
- **IVR unit**—Interactive Voice Response unit, a computer system that evokes and responds to a user’s DTMF or voice commands.
- **NLP**—Natural Language Processor, the ability of a language-recognition system to translate many different utterances with the same meaning into a single, actionable result.
- **Prompt**—A speech system’s audio output, measured as a single recorded or generated segment. Prompts are used to ask questions, soliciting caller input, giving directions, or conveying information.
- **Speech Recognition System**—A system that interacts with a caller to elicit speech, recognizes the callers’ responses, understands the speaker’s meaning, and performs some predefined action based on that understanding.
- **Usability**—Fundamentally, Usability is concerned with making systems easy to learn and easy to use. The term is used to describe the quality of a user’s experience when interacting with a system – whether a Web site, a software application, mobile technology, voice application, or any other human operated device. A usable system is one which enables users to perform their job effectively and efficiently.
- **Voice authentication**—Also known as speaker verification. Voice authentication compares a speaker’s voice to a recorded and stored voiceprint associated with that speaker, to provide secure access to restricted areas within a voice application.
- **Voiceprint**—A unique audio pattern created by the sound of a person’s voice. Like a thumbprint, this can be used to uniquely identify and authenticate individuals. It can replace the need for a user ID or PIN when trying to access secure areas of a voice application.
- **VoiceXML**—An emerging development standard for creation of voice-enabled applications, based on the eXtensible Markup Language. VXML is especially well suited for developing the Voice Web.
- **VUI**— (Typically pronounced “voo-ey”) Voice User Interface, a broad term for the experience a caller is presented with when using the voice-enabled application or system. It is analogous to a GUI (graphical user interface) for a Web site. A VUI includes the dialogue components as well as the character, and other audio files used to build the application.
- **WOZ**—Wizard of Oz, a form of usability testing that simulates the flow of a voice application, but during which the caller is speaking with a human (i.e. – “the man behind the curtain”) rather than an implemented system.

Matt Prather

Copyright © 2007 Matt Prather. All Rights Reserved.

Logos, and trademarks or registered trademarks of Colla Voce Consulting or its subsidiaries in the United States and other countries, copyright ©2007 Matt Prather. All Rights Reserved.

Other names and brands may be claimed as the property of others.

Information regarding third party products is provided solely for educational purposes. Matt Prather and/or IDG are not responsible for the performance or support of third party products and do not make any representations or warranties whatsoever regarding quality, reliability, functionality, or compatibility of these devices or products.