PRONEXUS

Selecting an Interactive Voice Response Rapid Application Development Tool

pronexus.com | 2934 Baseline Road, Suite 301, Ottawa, ON Canada K2H 1B2 | Tel +1.877.766.3987 | Fax +1.613.271.8388

VBVoice

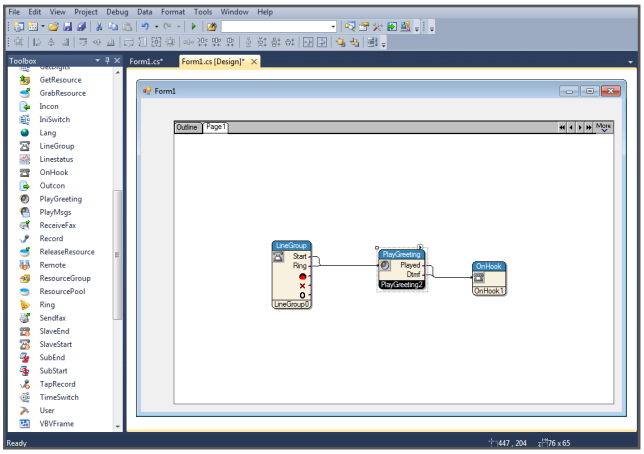
Companies around the world use VBVoice® in numerous industries from banking to government,healthcare, insurance, utilities and many more.

Service providers and in-house developers can reduce IVR application development time and bring their IVR solutions to market faster with VBVoice. Other IVR benefits include streamlining internal processes, improving customer care and increasing the bottom line through efficiencies and reduced overhead costs. The benefit of VBVoice is rapid application development, easier, less costly maintenance, support of industry leading voice technologies, and support for industry leading telephony technology.

What Should you Consider when Selecting an IVR Tool?

Developers looking to enter the voice business market face a number of hardware and software choices. Chief among these decisions is the choice of development environment – it can determine success or failure of a development initiative and make the difference between a profitable project rollout and a costly failure. This document highlights some of the critical issues that should be considered:

- Ease of use
- Industry-standard language
- Debugging
- Modularity
- Architecture
- Rapid application development tools
- Speech processing
- Automatic speech recognition
- Text-to-speech
- Deployment considerations
- Support for standards



Why Choose VBVoice?

- Create sophisticated IVR applications using familiar programming skills and industry-standard programming languages (C#, .NET)
- Leverage leading speech and telephony technologies including ASR, HMP, MRCP, TTS and VoIP
- Cut development time with:
  - Visual call flow environment
  - Fully customizable voice controls
  - Event-driven framework
  - Source-level debugging
- Give your project a head start with:
  - Prompt library (English (UK & US), French, German, Italian, Japanese, Spanish (Castilian & South American))
  - Tons of sample applications; e.g. help desk, speech attendant, fax service, predictive dialer, etc.

Best of all, **VBVoice is available free of charge!**
Download your free copy of VBVoice 10 from pronexus.com or vbvoice.com today!

Ease of Use

The ideal programming environment should accomplish a number of goals, the most important being ease of use.

To maximize ease of use, most telephony and speech toolkits offer a degree of visual programming through a 'drag-and-drop' style interface.

However, while visual programming has the potential to greatly enhance ease of use, it can also become a limiting factor. A short learning curve sometimes comes at the expense of other developer productivity aspects that may only become obvious after the tool is in use. Extensibility is key among these. The developer's ability to extend the visual programming environment by using modern programming languages to incorporate other components not provided by the toolkit or to take full advantage of new hardware and software is critical for most real-world telephony applications. For example, the ability to integrate with .NET environments and applications is an increasingly significant decision criterion.

Industry-standard Language

To further shorten the learning curve for developers new to telephony and speech, the programming language should, ideally, be industry-standard (such as C#, VB.NET) and not proprietary to the selected tool.

While most development environments for computer telephony applications allow for the addition of external functionality, some require knowledge of proprietary scripting languages or handling of complex, low-level programming. Preferably the application development environment will enable developers to leverage their programming knowledge and expand on the telephony controls included in the toolkit through custom code, as well as make use of third-party components.

Debugging

The right development and debugging tools can save your project. Attempting to uncover call flow or recognition problems in an application without a rich set of tools sharply reduces developer productivity and increases error rates of the final application.

Ideally, the chosen programming environment should:

- Tie into an industry-standard debugging platform that you or your developers are already familiar with
- Have the ability to generate sophisticated call log files for further analysis.

Together these characteristics allow developers to create sophisticated and powerful applications without a long learning curve and ongoing trial-and-error during application development. If you're building an application on Windows®, don't miss out on the benefits of the next generation technologies from Microsoft® - your tool must support .NET!

Modularity

The ability to break your speech application into cooperating modules is a must. Not only does it improve scalability, reliability and performance of your system, but it also saves you money in both development and production.

A modular system is cheaper to build and maintain. In development, programmers benefit from working in parallel on well-defined modules. In production, independent module provisioning and software hot-swaps eliminate costly system downtimes. At the same time, separating application logic from telephony and speech processing allows resource sharing, which in turn leads to more efficient utilization. Finally, distributing your modules across a Local Area Network (LAN) enables load balancing and effortless scalability - again resulting in savings on system maintenance.

The biggest benefit, however, comes from increased reliability of a modular system. Nothing is more frustrating to callers than a system that crashes into "dead silence" in the middle of a transaction. An unreliable system will be soon pulled out of production, which always means significant financial losses.

A monolithic executable is only as reliable as its weakest component, while a modular system can stay operational even after losing one of its modules. Therefore, it is very important that application modules execute properly separated from each other and from the system processes, so that a fatal error in one doesn't bring down the whole system. The modules should run out-of-process, or even better, distributed across a LAN. Ideally,

modules should be compiled directly into stand-alone executables, not into intermediate scripts or p-code. Not only does this speed up program execution, it also removes the dependency on a shared runtime engine as a single point of failure.

Architecture

The architecture of your platform should offer proven scalability and the ability to hot-swap applications. It is important that applications can execute independently from each other and from the system processes, can be hot-swapped, easily provisioned and configured.

In hosting environments or in situations where multiple applications are being deployed, the platform should enable the sharing of telephony resources (i.e. speech licenses and telephony hardware) across multiple independent applications. Such a distributed architecture also allows individual applications to be interrupted for upgrades or other maintenance without interrupting other applications on the same server.

While some environments allow the sharing of telephony hardware and hot-swapping of applications, developers should ensure that the platform does not create a monolithic executable for multiple applications. In such a system, the failure of one module could stop the entire system, as the monolithic executable is only as reliable as its weakest module.

While your code may be bulletproof, can you guarantee the same for all the components and libraries you have to use?

Rapid Application Development Tools

The world of telephony and speech applications is a complex one, with multiple hardware and programming interfaces and a plethora of standards. The resulting learning curve for new developers tends to be very steep. This is where rapid application development tools really shine!

Their controls encapsulate and abstract common call processes to simplify application development and shield the developer from hardware specific programming. In evaluating your application development tool, you should look not only at the raw number of these controls, but rather at their depth, customizability and extensibility. Otherwise, you may find that the feature you are looking for is simply not doable in the environment of your choice.

The tool should also support advanced protocols such as ISDN and VoIP and offer a comprehensive lineup of call control features for your particular application. For example, call queuing, agent monitoring, recording and conferencing capabilities are significant in call center applications, while other solutions may require broad fax support, web integration, switch integration via TAPI, etc.

Automatic Speech Recognition

The implementation of an effective automatic speech recognition (ASR) solution can reduce the number of agents, supervisors, trainers and quality assurance specialists that are needed by your business. If a consumer is provided the option of gathering the information he or she needs without accessing an agent more agents are free to handle calls that cannot be resolved with self-service.

The market for speech recognition engines is continually evolving. Consequently, the development platform should support multiple ASR engines, letting you select the appropriate engine for each individual application development effort.

Additional speech capabilities to look out for:

- Does the tool support speaker verification capabilities of one or more speech engine vendors?
- What other areas of speech deployment are being supported? Dynamic grammars, for example, can be used to simplify complex application development.

Most of the top ASR vendors are now exposing their ASR engines through Media Resource Control Protocol (MRCP), a standard which allows using the same client with different engines.

Text-to-speech

Text-to-speech (TTS) is a technology that allows you to create a real-time link between text-based content in your database and a customer awaiting an immediate reply. TTS can read any text out loud without using pre-recorded prompts. This technology is mature; it has been validated by market deployments and is already largely used in telephony services provided by carriers and enterprises alike.

The development platform should support multiple TTS engines. Most of the top TTS vendors are now exposing their ASR engines through MRCP, a standard which allows using the same client with different engines.

The flexibility to mix and match TTS engines through a MRCP connector is important, because it offers the choice of the desired engines and voices.

Deployment Considerations

Another important consideration for developers is the ease of product licensing and deployment. This is particularly true for commercial application developers, such as system integrators and Independent Software Vendors (ISVs). Ideally, deployment of a finished application should be as painless as possible, yet assure the developer of the licensing integrity of the finished product.

Consider whether the platform requires the use of “dongles” for commercial application deployment, or whether the process is only software-enabled. The latter approach allows for easy upgrades and also has the potential for the re-licensing of developed applications to other customers.

Support for Standards

Depending on your organization, standards such as .NET or VoiceXML can also play an important role in your tool selection. In the author’s opinion, however, other considerations as outlined in this document will likely be more important in the day-to-day development work. In addition, standards may impose restrictions on your development effort, since certain new developments and/or specific capabilities that you are looking for have not yet been reflected in a generally slower evolving public standard.

VoiceXML, for example, has an acknowledged weakness in the area of call control. For the foreseeable future, other platforms will continue to exist, and some of them are beginning to extend to support standards environments.

Programming in .NET and C#, which are industry standard programming languages, eliminates the need to learn proprietary languages and shortens the learning curve for developers new to the Interactive Voice Response (IVR) & telephony landscape without the need to worry about the telephony hardware APIs or media processing layers. Thereby allowing you to rapidly create powerful IVR and voice-enabled communication solutions while significantly reducing your time to market.

Go to **pronexus.com** to download the VBVoice 10.1 IVR toolkit or contact **sales@pronexus.com** to get your application up and running.

Pronexus

Pronexus is the creative force behind VBVoice, recognized as one of the most seasoned and powerful IVR development toolkits available today. After 20 years of consistent innovation and technological advancement in the field of IVR development, Pronexus has expanded its product range to include the comprehensive IVR monitoring tool IVRGuard™, and Pronexus’ new offering, VisualConnect™. Pronexus’ commitment to innovative, future-proof solutions is demonstrated by our dedication to offering this new functionality as well as our status as an established Microsoft Silver Partner. VBVoice integrates with the latest versions of Microsoft® Visual Studio, enabling use of familiar programming skills and industry-standard programming languages. The intuitive visual call flow environment and programmable controls as well as the time-saving features of VBVoice, such as a multi-lingual prompt library and many sample applications that help new users to learn and understand toolkit functionality, make complex IVR applications built on VBVoice easy to develop and quick to deploy.

pronexus.com | 2934 Baseline Road, Suite 301, Ottawa, ON K2H 1B2 | Tel +1.877.766.3987 | Fax +1.613.271.8388