

Creating Successful Speech Applications in an Open Standards Environment



Paul V Mellor
CTO Vicorp Group PLC

CONTENTS

Audience and Scope	3
The Impact of Open Standards	3
The Standards Hourglass.....	5
<i>VoiceXML Features</i>	6
Enterprise control point.....	8
<i>Differentiators using VoiceXML</i>	8
<i>Proprietary Extensions</i>	10
<i>Vendor Specific Features</i>	10
The Service Execution Platform	12
The Service Creation Platform	12
Summary	13

Audience and Scope

This white paper is written for organisations that wish to provide speech-oriented services to their customers, based on an open standards environment such as VoiceXML. In this category we include not only speech services themselves, but also touch-tone IVR (Interactive Voice Response), and also services using different media types but which utilise a similar interaction paradigm: such as video, mobile self-service and the like.

How can organisations benefit from an open standards world? Well, the ability to provide interoperability between different elements in a solutions stack means that service providers can increase the type and functionality of the services offered to the end user, unencumbered by the limitations or even presence of proprietary application interfaces (APIs).

This is the theory anyway; the flip side of this coin is that providing an open standards framework also provides a level playing field for application development, and with many vendors in the IVR chain offering enhanced functionality (often with proprietary add-ons) that will differentiate their particular products, application developers face pressure to trade off the benefits of open standards in favour of proprietary differentiation. To adopt the right choice, buyers need to stand back and view the IVR chain or “stack” as a whole – taking up seemingly tempting elements at one point in the stack may see them being left with little or no choice elsewhere.

The question then is how to benefit from open standards, yet still be differentiated and avoid ending up on the commodity pile. The Service Creation Environment (SCE) is key to solving this problem: unlocking the potential of open standards platforms, while providing for service innovation. This white paper focuses on the relationship between the SCE and the underlying platform: and the capabilities that an open standards SCE needs to provide in order to maximise the potential of the overall IVR solution.

Support for open tooling is vital – without this the opportunity to build applications that are totally separated from the platform vendor’s environment will be lost. We have seen how standards based tools have transformed the development of HTML on the Internet – now the same wave of change is happening with VoiceXML and disrupting the traditional landscape of IVR.

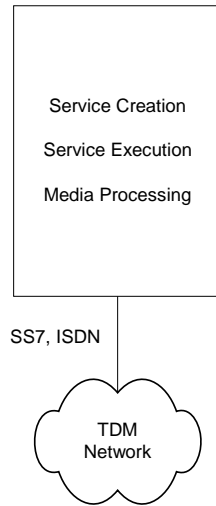
Many platform vendors will go to great lengths to persuade clients to use their own proprietary tools – usually offering them for free – so that clients remain locked into operating on their proprietary platforms. Of course the actual costs associated with the SCE are simply absorbed into the per-port costs for the media server solution as a whole. Clients who go down this path inevitably end up paying the price later in high application costs and support, as well as losing the ability to move away from proprietary platforms because the cost of future migration becomes prohibitive. This wouldn’t seem so bad on the face of things, after all who really cares which media server does the heavy lifting? The hidden problem is that the voice applications created on proprietary platforms cannot subsequently be moved anywhere else and these account for the lions share of IVR investment.

The Impact of Open Standards

Legacy proprietary speech and IVR solutions were typically monolithic in nature, as in Figure 1, providing the media processing and the service creation and execution environment as a single product suite provided by a single vendor. They were tied into standards only at the telephony layer: these typically being SS7, ISDN, or other TDM related protocols, handled by hardware cards included in the solution bundle.

The advent of VoiceXML has opened up the potential for independent SCE’s that sit above and separated from the VoiceXML layer and which are no longer tied to a specific media server. It has also separated out the media processing from the application domain, introducing a strong degree of competition between media server vendors, and enabling far greater creativity at the application level, since applications have now been freed up from the constraints of their supporting platforms.

Proprietary Solution



Open Standards Solution

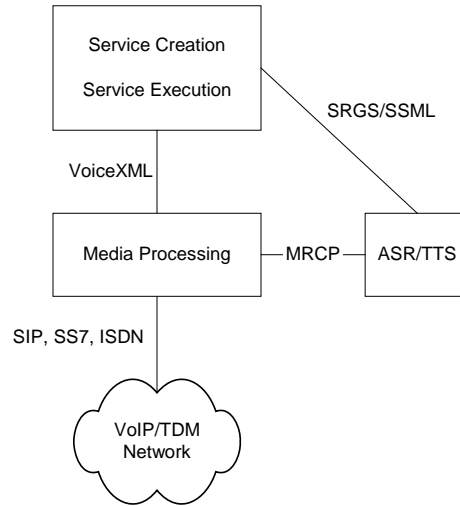


Figure 1 Proprietary and Open Standards Solutions

With this change has also come a migration over from TDM to VoIP, which has allowed vendors to move away from hardware based solutions to software only solutions, creating an environment where a media server can now be simply downloaded, installed and run on a standard, low-cost off the shelf server. Also in this picture is the adoption of open standards for interconnecting the speech recognition (ASR) and text to speech (TTS) parts of the solution. Here we see MRCP being used to interact with the ASR/TTS engines, and SRGS/SSML standards being used to specify what they are to do. These standards allow free selection of ASR/TTS engines, rather than have them bundled with the media server, creating yet more flexibility for clients.

Figure 2 Solution Focus

These technology shifts have also manifested themselves in how solutions are sold, as in Figure 2. The sales focus for legacy solutions was driven predominantly by the hardware requirements and availability of built-in middleware and applications, with the emphasis being on selling from the

technology layer up. Now, businesses place the emphasis on the solutions required to drive their businesses forwards, with the emphasis being on the applications first and foremost, and the underlying layers becoming commoditised down through the stack.

The Standards Hourglass

As can be seen in Figure 3, standards such as VoiceXML provide a vital role in abstractly connecting the application domain and the media-processing domain.

The top half of this diagram represents the application domain and the SCE; whereas the bottom half represents the media processing domain and the media server. The two are interconnected by the VoiceXML standard. It is via the VoiceXML standards that most of the interactions occur. However, on the left we see some interactions that are transparent to the technology, and on the right some that are not and which create lock-in. These are explored in subsequent sections of this paper.

Figure 3 The Standards Hourglass

Although the applications layer is constrained at the base by the VoiceXML standard, it is open at the top, and this allows SCE vendors to innovate heavily in terms of the features they provide to their customers. Of course, all these features must ultimately be supported through the media server, so there is an underlying constraint, but nevertheless there remains greater freedom here than elsewhere in the system.

Unlike the application domain, media servers are constrained by standards at both ends: they take instructions in via VoiceXML and interact with callers via SIP, SS7, or other telephony protocol. This can make it difficult for vendors to achieve significant product differentiation. However, the goal of open standards is not to stifle creativity, but to provide a common base from which it can be launched. Whilst supporting open standards, media sever vendors will also provide proprietary extensions and additions in order to differentiate their products by their features. Customers can benefit from this by selecting a media server vendor whose strengths and differentiators tie in to the customer's own strategic goals.

Features can be split into three segments, relating to vertical columns on the hourglass diagram:

- The central column contains those features that are direct representations of VoiceXML capabilities: such as the ability to let the user choose between a menu of options, or to make a recording. The SCE can directly reflect these to service creators, or can provide some degree of abstraction to isolate users from the details of the standard.
- The left hand column contains differentiating features that can be implemented using VoiceXML, such as support for video. Here access to media related features can be made transparently in the SCE, since they either do not concern the standard, or can be built out of standard interactions
- The right hand column contains vendor specific features that require interactions outside of the VoiceXML standard: such as CTI integration. Here there is potential lock-in if the SCE ties itself into vendor specific features - although this possibility is constantly being eroded by the advent of new standards.

We next examine the issues relating to each of these areas.

VoiceXML Features

When combining an open standards SCE and media server into a solution, there are several factors that must be considered that relate to the basic VoiceXML features provided by both.

Compliance

The most obvious VoiceXML feature for the media server is compliance to the standard itself. Here the main issues are: which standards are supported; how fully are they supported; and what is required in order to obtain this support? Most media servers now support VoiceXML 2.1, but not all have this level of compliance yet. Even those that support a specific version of the standard may have restrictions in terms of individual tags or features they do not support – especially if these are optional: one major player does not support the <transfer> tag for example. Then again, even if a feature is supported, extra equipment may need to be ordered in order to obtain this capability: in some cases additional investment is needed even for such a basic capability as accessing a WAV file from a URL.

Certification

The flip-side of compliance for the media server is certification of the SCE. Open standards SCE's need to be able to work with *any* open standards compliant media servers. Despite standardisation, each media server inevitably has its own restrictions and interpretation of the standards. Whilst these differences were significant in VoiceXML 1.0 implementations, with VoiceXML 2.0/2.1 they are now much less severe – but nevertheless it falls to the SCE to accommodate them and ensure that all generated applications will work regardless of the media server in use. This is typically achieved through some sort of adaptation: where the generated VoiceXML is adapted to meet the nuances of the particular media server in use. It is important therefore to chose an SCE vendor with a strong certification programme.

Multi-Vendor Operation

It is increasingly recognized that single-source platform strategies involve considerable risk, both from a technical and a commercial perspective. To obviate this risk, customer organizations can ensure that the media server platform options available to them support open standards such as VoiceXML and MRCP, enabling the use of any ASR (automated speech recognition) engine.

Vicorp Group Plc

In the past, it has not been easy to adopt a dual or multi-platform policy because IVR-based applications were coded specifically for the IVR platform on which they ran. However, thanks to open standards, in today's marketplace voice applications can be completely abstracted from the platforms that they run on, and thus a multi-vendor strategy for media servers becomes feasible, and conveys several advantages:

- More than one type of media server can be deployed in the distributed IVR network, and traffic can be divided across media servers at will. The resilience that arises from this approach also means that low cost servers can be deployed at a fraction of the cost of some proprietary servers.
- Rack built media servers mean that CPU upgrades are not very easy to carry out and normally come disguised as expensive product upgrades, whereas standalone server technology can be upgraded at will.
- A multi-platform approach ensures that media server vendors must compete with each other. They must maintain competitive pricing at the same time as they seek to provide competitive advantages.
- A multi-vendor approach means that different business needs can be addressed with products that are specific to requirements. For example, simple DTMF services can be supported by media server vendors focusing on low-end cost-efficient solutions. High-end services (e.g. video) can be supported on media servers that focus on higher-end installations and have video support as a key differentiator.

Sometimes, enterprises will simply not have a choice: through acquisitions or other organizational changes, they find themselves running applications over a range of platforms from a range of vendors. Migrating to an open standard solution allows this diversity to be managed effectively. Perhaps this migration will not happen overnight, but having a migration strategy is key if harmonization is to be achieved over time.

Working in a multi-vendor environment creates its own challenges for an SCE. Not only must the SCE work with any given media server, in a multi-vendor environment it must also commonly work with several different types of media server simultaneously. If we assume that even in today's environment some level of adaptation of the generated VoiceXML is required for compatibility, it is important then that this adaptation must be done *dynamically*. That is: the SCE should not require any details of the deployment platform as the service is created.

This is required since a single application may be working across multiple media servers, and it may not be possible to restrict access to the application to be from a single type of media server without severely restricting flexibility and efficient use of resources.

Sizing

Each VoiceXML installation needs to be sized to meet the needs of the business, and then re-sized as business needs and customer take-up change. The VoiceXML standard defines the concept of a VoiceXML interpreter, the function of which is to take VoiceXML documents and process them. It is industry-standard practice to size VoiceXML solutions according to the number of parallel VoiceXML interpreter sessions (usually referred to as VoiceXML ports) that can be supported.

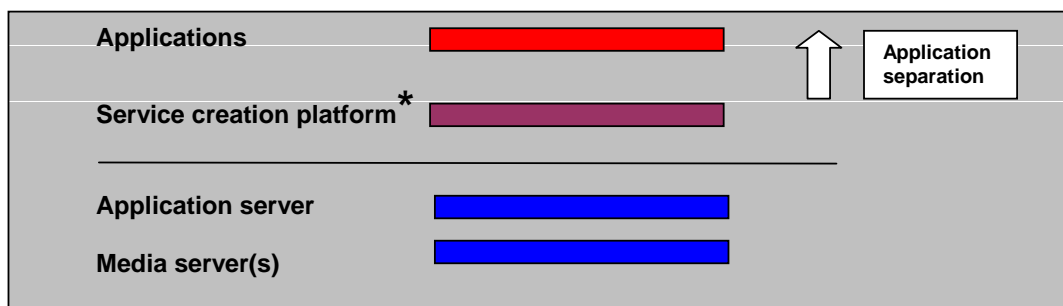
The support of VoiceXML ports by actual server machines is an implementation issue in both the application domain and the media-processing domain, and both media server and SCE vendors need to be able to size their solutions accordingly.

SCE vendors must be able to calculate how many application servers are needed to support the requisite number of VoiceXML ports. Of course this depends somewhat upon the nature of the applications themselves, and can vary depending on such factors as how much back-end processing is taking place, how much content or other media needs to be retrieved, and the level of dynamic generation. Nevertheless, the SCE vendor should undertake whatever load and performance testing is required in order to be able to help its customers accurately calculate the application server requirement to support the number of ports they have bought.

Media server vendors must be able to perform the equivalent task for their own products, and should be able to advise their customers on the processing resources required to support the required number of VoiceXML ports. This, again, is application dependent and can vary depending on the specific VoiceXML capabilities being used by the application: the number of sub dialog calls, the amount of speech recognition being undertaken, and so on. Nevertheless, the media server vendor should undertake whatever load and performance testing is required in order to be able to help its customers to accurately calculate the server requirements to support the number of ports they have bought. There can be wide disparities in performance among media server vendors based on whether the latest chip technology is being used and on the way in which some operations such as caching are designed.

Enterprise control point

The SCE can also be used to control the subsequent call routing. This can enable calls to be routed across several IVR stacks - or across business units.



Differentiators using VoiceXML

Enabling Performance

There are many ways an application can generate VoiceXML in order to provide a specific service to a caller. Although these may be functionally equivalent, the style of VoiceXML generated by the application can sometimes affect the performance of the media server.

Although each type of media server has its own performance characteristics, there are certain VoiceXML operations that are implicitly more expensive to perform than others. This fact allows SCE developers to produce applications that avoid expensive operations and are inherently efficient to run. Two examples follow:

Sub-Dialogs and Re-Use

SCEs should demonstrate the ability for either the enterprise itself or its' application partners to build applications that can be broken down into components that can be used again in similar services elsewhere. For enterprises that are embarking upon upgrade and or migration of existing voice applications, the prospect of being able to reuse large amounts of common application elements or features is a big leap forward in reducing TCO. This kind of re-use is supported in VoiceXML via the sub-dialog feature, which lets one application call another.

The VoiceXML sub-dialog is an implicitly expensive operation, since it requires the media server to store its entire context and create a new one. Despite this, the sub-dialog is a useful feature since it allows applications to call other applications, and therefore obtain application level re-use. However, the SCE can provide equivalent (or even more powerful) features at the application level – without the need to use the media server's sub-dialog based context handling capabilities. Since this takes place on the application server, the media server is left to focus on media processing and is not encumbered with re-use requirements and processing, greatly enhancing its performance.

It must be noted however, that no standard exists for application level re-use, so solutions today are proprietary, meaning that re-usable components from one SCE will not work in another SCE.

Caching and Scripting

Some media server vendors will promote their products performance characteristics by claiming that they can be “tuned” to render higher density VXML ports and so forth. These claims are almost certainly incorrect and should be carefully benchmarked. In particular media servers that claim higher throughput because of caching data are actually likely to be among the lowest performing vendors.

Whilst caching of prompts and grammars can lead to significant performance gains, caching of VoiceXML documents is more complex. In order to cache the VoiceXML document, it needs to be static in nature and therefore not contain any information explicitly related to the specific call in progress. Typically, static VoiceXML like this can be used only for very small, simple applications. Most applications would require that all call-related processing take place on the media server itself in order to have static VoiceXML. This would place a significant additional processing load on the media server, and would significantly reduce its port capacity. Consequently, most systems today use dynamic VoiceXML.

In systems that generate dynamic VoiceXML, this processing takes place as part of the application, on the application server – freeing the capacity of the media server to focus on its main job: processing the media. Application servers are designed to perform exactly this type of processing, and have a large number of facilities for back-end integration and so on, which are not present on the media server.

By keeping application processing off the media server and on the application server, applications can avail themselves of much better capabilities for CDR (call detail records) generation, logging, alarm generation, monitoring, management, reporting and so on, as well as significantly increasing the performance (in terms of number of ports per CPU) of the media server.

Non-functional differentiators

Media servers typically have differentiators in the following non-functional areas:

- **Hardware vs. Software:** Modern designed-for-purpose media server solutions are typically software based, with the customer being able to choose the type and quantity of COTS computing server platforms required to meet their capacity needs. This allows customer to benefit from Moore’s law of increasing processing power (and its associated price reductions), and to maintain a degree of uniformity across their organisation. Legacy platforms, and solutions based on them, are typically hardware based, and do not enjoy these benefits. Similar considerations apply for hardware vs. software Digital Signal Processing (DSP) functions.
- **Performance & Scalability:** Media servers with fully distributed architectures can more readily support different scaling options in order to meet the customer’s current (and future) performance needs.
- **Cost:** As the media server becomes commoditized, prices will inevitably fall, and price will become an increasingly important differentiator.

SCE’s also differentiate in ways that build on the underlying VoiceXML capabilities:

- CDR – the ability to generate dynamic call details that can directly bear upon the pricing of the service being handled.
- Reporting – the ability to capture call performance statistics at all levels in order to analyse and improve services
- Audit Trails – to record what changes have been made when and by whom

Proprietary Extensions

Media servers typically have differentiators in the following functional areas:

- **Video:** By and large VoiceXML is just as useful for video as it is for speech. Most of the basic features to play, record, navigate video menus, and control the system through spoken commands or key presses can be used equally as well for video applications as for voice (or with a mix of the two). Several media server vendors differentiate their products through their explicit support for video.
- **Multi-tenancy:** Hosting and Managed Service Providers need to operate in a multi-tenant environment, where each of their clients is a tenant on a shared infrastructure. Indeed this is often the case for enterprise clients, who may have several independent internal clients on a shared platform, each perhaps having their own outwardly facing brand.

In a multi-tenant environment it is critical to be able to manage resources, such that they can be partitioned between tenants. This can allow one tenant to run applications that use ASR for example, whilst another cannot; or to allow one tenant to use 100 ports of TTS, whereas another may use up to 300. Resources must also be monitored in terms of their grouping and partitions, so that appropriate tenant-specific reports can be created.

- **IMS:** Media servers focussed on the Telco core network environment may support the IMS architecture. This requires a different set of protocols to those encountered in an enterprise environment, with MSCML (Media Server Control Markup Language) being used with SIP for example, as opposed to VoiceXML and HTTP. These solutions will typically also be Telco grade, in the sense that they may be rack mounted, 48 volt, hot swappable, highly available, and so on.
- **Call Control:** With VoiceXML only standardising basic call control features (disconnect and transfer), many vendors opt to supply proprietary capabilities such as conferencing and multi-party connection management. These features open up a range of new service opportunities, such as find me, follow me, pre-paid calling card, and the like. Of course ccXML is increasingly viewed as an attractive open standards alternative to proprietary call control extensions, though the standard is not yet fully mature.

Each customer will have a different set of strategic needs and can therefore choose a media server vendor whose product is most closely aligned with those needs. To a Managed Service Provider, Telco grade reliability and multi-tenancing may be the most critical features to support; whereas to a Mobile Operator video support may be critical.

Vendor Specific Features

The SCE must of course generate standards compliant code, but it must also allow proprietary extensions to be accessed when needed. Use of these extensions must be undertaken with some care, since they then lock the application into the specific media server and this in turn negates some of the benefits of a standards based solution. However, the SCE should support this, but make it clear that use of such features is occurring.

Sometimes the restrictions of the VoiceXML standard do shine through. Here are some examples:

- **Classical Debugging:** Some proprietary SCE's contained a debugger that resembled those commonly in use to debug programming languages such as Java; and contained comparable features such as breakpoints and single step execution. This is difficult if not impossible to achieve using standard VoiceXML, since there is no simple way to cause the media server to halt in the middle of processing a page. Of course the generated VoiceXML can be modified to include specific "breakpoint" code, but typically this then results in a large disturbance to the structure of the application, and to such a degree that the debugging effort itself may well be invalidated.

- **Logging:** All media servers store log files: indeed some vendor's products provide a bewildering array of log files that can be difficult even to track down, let alone analyse. Each vendor has to use a proprietary format, as there are as yet no standards, so it is difficult for a SCE to obtain and present these in a uniform way to service creators.
- **Grammars:** Although MRCP standardisation affords customers the flexibility to select a speech recognition vendor of their choice, ASR standardisation today lags that of VoiceXML. Perhaps the biggest gap is in terms of interpretation, which means that different versions of a grammar need to be produced for different ASR engines.
- **Tuning:** Tuning generally requires some access to underlying logging data, such as that provided by the ASR engine. Unfortunately there is as yet no standardisation in this area, so proprietary tools must be used.
- **Management:** Management features have not been standardised either, resulting in proprietary solutions in this area also. This can hamper important features such as the resource management that goes hand in hand with a multi-tenancy solution for example. It is not possible for an open standards SCE to identify, allocate, and police the management of resources (such as ASR ports etc) in the underlying media servers without recourse to media server specific mechanisms.

In general it is very important for the information available at these underlying levels to percolate back up to the service creator – and it must be presented back using the same model as that used to create the service in the first place. With this approach, service creators can make the most sense of the information they have at hand, and can drill down into details secure in the knowledge that this activity is being undertaken in the context of an application level investigation.

As can be seen in Figure 4, new activities and standards are bridging these gaps, and extending the coverage of the hourglass.

Figure 4 Bridging the standards gap

- **Classical Debugging:** Given the nature of speech applications, it is generally perfectly acceptable to debug through appropriate analysis of the logging data, coupled with suitable tooling for presenting information about how the call proceeded and what all the internal data values etc. were.
- **Logging:** The VoiceXML forum is addressing this topic, with a working group aiming to standardise logging, but this work has not yet reached fruition. Until it does, SCE's either avoid the area, or have to provide media server specific adaptors.
- **ASR:** A new standard called SISR (Semantic Interpretation for Speech Recognition) is addressing this area.
- **Tuning:** ASR specific tools such as mentioned above can be brought under the umbrella of the SCE in general – such that they can be driven from a higher-level view of the service via a “drill down” approach. Often such tools are common in essence across vendors: e.g. the ability to compile grammars, to generate sample utterances, etc.

The Service Execution Platform

There is (as yet) no standardisation between the service creation environment and the service execution environment, so the two must go together from a single vendor. Despite this, standards can still play a significant part: not in terms of functional interfaces and interoperability, but in terms of platforms and manageability.

Service execution environments based on open standards platforms – most typically J2EE application servers – offer numerous advantages over those based on proprietary solutions:

- **Integration:** Open solutions integrate into existing corporate infrastructure, such as corporate server farms, existing web infrastructure, IMS, and SOA solution components.
- **Ongoing Development:** Open solutions are continuously being developed by their vendors, and users of these solutions can benefit from new features as they are rolled out.
- **Skills:** Operations departments can more readily find staff with appropriate skills to manage and maintain open standards based solutions, than they can for proprietary solutions. Also, such staff can often be shared across the whole corporate infrastructure, leveraging economies of scale.

Services themselves can also exhibit differing degrees of openness. Some SCE's produce services that are essentially proprietary in nature, and that therefore require an interpreter at run time in order to generate the VoiceXML required. Other SCE's produce native applications (often using de-facto standard web frameworks such as Struts or Spring) that run directly in the application server environment. Such applications are more easily management in the longer term. As native applications, if the SCE is abandoned, the application itself can still be used, maintained, and even developer further to include new features. This gives a great deal of confidence that application level assets developed in the SCE can be protected and are not irrevocably tied to the SCE initially used to create them.

The Service Creation Platform

No standards exist for the SCE itself, but again, standards can play a part in other ways. There are many advantages for example for using a SCE built in an open framework tool such as Eclipse. Eclipse is an open source project managed by the Eclipse Foundation (backed by IBM). Eclipse provides a core Integrated Development Environment (IDE) which can be extended with the use of plug-ins. This provides the service creation organisation with a number of benefits:

- **Development Community:** Open IDE's like Eclipse have large communities of experienced users, providing a skilled resource pool from which client organizations can draw. Such resources will already be familiar with the basic features of the IDE, and as such are well up the learning curve involved in taking on the speech related aspects of the SCE.
- **Ongoing Development:** These platforms also have a large development community behind them and are constantly rolling out new enhanced versions with additional features that can be leveraged by the SCE.
- **Supporting Tools:** Eclipse, for example, has a large number of openly available (and free) plug-ins such as configuration and management tools that might otherwise not be available or be provided free of charge.
- **Third Party Tools:** Not all tools in the SCE need to come from a single vendor. Open IDE's will often allow plug-ins or tools from multiple speech vendors to work together in a single environment, for example the SRGS editors available as Eclipse plug-ins. Also, new plug-ins can be developed in-house.

- **Multi-Role Support:** Speech service creation is typically a multi-disciplinary activity, with contributions from team members with different skill sets. Open IDE's typically have full tool support for many of the non speech related aspects of service development – most notably for back end integration and the development of any requisite business logic, e.g. via Java coding.

Summary

Chosen IVR platforms should provide seamless and non-restrictive integration with open tooling environments to take advantage of the various features opened up by VoiceXML and similarly the tooling environments should work seamlessly with any Media Server/ASR/Application Server environment. Eclipse based tooling environments provide greater pools of development resources capable of creating enhanced speech applications. Clients should avoid tying themselves into a single Professional Services provider (or at least be able to switch easily if they choose to do so) and should also ensure that they can bring development and support in-house - if required in the future. This will avoid the risk of experiencing long lead-times or inflated prices for future applications.

Lastly the speech application development cycle is considerably shorter using open standards tools – buyers can often obtain Proof of Concept services from vendors quicker than they can specify them for a traditional build approach. This greatly changes attitudes to risk and obtaining value for money.

For further information on anything contained in this paper, please contact the author at Vicorp.
+44 1753 660500, email to info@vicorp.com or visit www.vicorp.com

