FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO
DEPARTAMENTO DE ENGENHARIA INFORMATICA E COMPUTAÇÃO

AMBIENTES GRÁFICOS PARA A CRIAÇÃO DE SERVIÇOS INTERACTIVOS

CÉSAR MANUEL FERREIRA PINTO

LICENCIADO EM ENGENHARIA INFORMÁTICA E COMPUTAÇÃO
PELA FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

DISSERTAÇÃO SUBMETIDA PARA SATISFAÇÃO PARCIAL DOS REQUISITOS DO GRAU DE
MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA E COMPUTAÇÃO

DISSERTAÇÃO REALIZADA SOB A SUPERVISÃO DE
PROFESSOR DOUTOR EURICO CARRAPATOSO,
DO DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES
DA FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

PORTO, ABRIL DE 2008

À minha família, namorada e amigos

Resumo

Vivemos na sociedade de informação. O "estado da arte" da tecnologia actual permite a obtenção e partilha de qualquer tipo de informação quase instantaneamente e a partir de, virtualmente, qualquer lugar através da *Internet*, telefone ou telemóvel. O acesso rápido e actualizado à informação é fulcral para o sucesso dos negócios. Os serviços têm como objectivo suportar eficientemente o acesso ao mais variado tipo de informação através de uma chamada telefónica.

À medida que as exigências do mercado e a complexidade dos serviços interactivos aumentam, são cada vez mais necessários métodos e ferramentas adequadas para acompanhar a sua evolução. Neste contexto, os ambientes de desenvolvimento dos serviços de Sistemas de Resposta Interactiva de Voz têm sofrido bastantes mudanças, procurando satisfazer as necessidades não só dos serviços prestados pelas operadoras de telecomunicações, como também dos seus processos e metodologias de desenvolvimento. Abstraem o designer ou programador das camadas de baixo nível de programação e integram todas as funcionalidades necessárias para a criação de serviços interactivos.

O objectivo da presente Dissertação é desenvolver um **Ambiente Gráfico para a Criação de Serviços Interactivos,** com o recurso a uma ferramenta *open source* para a criação visual de serviços**.** Para tal, foi efectuado um estudo transversal das tecnologias inerentes aos serviços interactivos, principais elementos multimédia, das tecnologias e da interacção com o utilizador, ambientes de desenvolvimento, Sistemas de Resposta Interactiva, linguagens de marcação, arquitecturas e processos de desenvolvimento de serviços. É retratado e avaliado o estado da arte das soluções comerciais actualmente existentes no mercado. Foi feito um estudo a avaliação das diferentes ferramentas para o desenvolvimento de uma Interface Gráfica de raiz. Foi também feita a análise e especificação dos requisitos necessários para que uma interface gráfica disponibilize, de forma completa e consistente, as ferramentas necessárias para a criação de Serviços Interactivos. Foi ainda efectuada uma prova de conceito com o recurso uma solução *open source*, o *openVXML*, através da sua instanciação no ambiente de desenvolvimento empresarial da PT Inovação. Por último, foi feita a comparação e avaliação dos processos de desenvolvimento de serviços com e sem o uso de interfaces gráficas nos ambientes de desenvolvimento.

Palavras-Chave:

Telecomunicações, Serviços Interactivos, GUI (*Graphical User Interface*), IDE (*Integrated Development Environment)*, RAD (*Rapid Application Development*), IP, GSM, UMTS.

Abstract

We live in an Information Society and our “state of the art” technology allows almost instantaneous access to any type of information, anywhere, using the Internet, telephone or mobile. Fast access to the information is a major business success factor. Interactive services using voice, video, fax and email through a phone call have the objective of efficiently supporting the access to the almost varied type of information.

While the market requirements complexity increases, interactive services need to follow up using methods and tools to agilize development process. In this context, the development environments of interactive services on the Interactive Voice Response Systems have suffered a lot of changes. These environments try to provide methods and tools to satisfy the necessities of the services provided by the telecommunications operators, searching for better development methodologies to support the agile development of interactive services.

The mail goal of this dissertation is to create one **Graphical Environment for the Interactive Services Creation** using an open source solution. For such, a transversal study was made, addressing areas like the technologies involved, information presentation, and user interaction, development environments, Interactive Voice Systems, the markup language, architectures and the development processes of interactive services.

The "state of the art" is presented through the study and evaluation of currently existing commercial solutions for graphical interactive services creation. A study and an evaluation of two different tools for the development of a graphical interface were made. A requirements specification for a graphical interface supporting the development of interactive services and an analysis documents were elaborated. As a proof-of-concept, a development environment was constructed using the openVXML open source solution in the PT Inovação development environment. Finally, the comparison and evaluation of the development process with and without the graphical environment were made.

Keywords:

Telecommunications, Interactive Services, GUI (*Graphical User Interface*), IDE (*Integrated Development Environment)*, RAD (*Rapid Application Development*), IP, GSM, UMTS.

Agradecimentos

Gostaria de agradecer a várias pessoas pela ajuda e disponibilidade ao longo destes cinco meses na realização da minha dissertação.

Assim, começo por agradecer ao Prof. Eurico Carrapatoso (FEUP) pelo seu próximo acompanhamento nas últimas fases da minha graduação e pelo valioso contributo na tomada de decisões quanto à definição do meu percurso académico. Queria também prestar o meu sincero agradecimento pela importância e excelência da sua orientação, sempre que foi necessária e sobretudo nos momentos mais cruciais, demonstrando sempre total disponibilidade e ajuda na resolução dos vários problemas que foram surgindo ao longo deste período de elaboração da dissertação.

Agradeço também à PT Inovação, em especial aos colaboradores do PLP3 (Serviços de Plataformas de Voz) do pólo do Porto, onde actualmente me encontro em Estágio Profissional. Queria realçar o especial contributo do Eng. Joaquim Azevedo e do Eng. Nuno Beires pelo apoio na minha decisão de terminar a minha graduação com o grau de Mestrado. Prestar um sentido agradecimento pelo enorme e especial contributo dado pelo Eng. Luís Almeida, orientador do meu Estágio Profissional, demonstrando sempre total disponibilidade e pela ajuda, orientação e resolução dos vários problemas que surgiram ao longo do meu Estágio. Um muito obrigado ao Eng. Alexandre Sapage, Eng. Luís Reis, Eng. Rui Gomes e Eng. Pedro Pinto pelo apoio e ajuda constantes, disponibilizando-me para os meios necessários para por em prática o ambiente de desenvolvimento para a concretização dos objectivos práticos estabelecidos.

Queria realçar a imprescindível presença da minha namorada, pelo seu apoio, força e presença incondicional. Pelo interessado acompanhamento dos meus objectivos pessoais e profissionais. Pelo importante contributo que fez com que as alturas difíceis se tornassem mais acessíveis, simples e claras.

Finalmente, e não por último, gostaria de agradecer à minha família, pelo apoio sem reservas, pela força e energia positiva que sempre me deram, para que esta caminhada fosse concluída com o melhor êxito. Em particular à minha avó, aos meus pais e irmãos, pela forte presença e entreajuda nas minhas decisões e opções ao longo do meu percurso académico e profissional.

A todos referidos anteriormente, os meus sinceros agradecimentos.

Índice

Índice Figuras

Índice Tabelas

Abreviaturas e Acrónimos

API	*Application Programming Interface*
ASP	*Active Server Pages*
ASR	*Automatic Speech Recognizer*
CLR	*Common Language Runtime*
CVS	*Concurrent Version System*
CTI	*Computer Telephony Integration*
DTMF	*Dual Tone Multi-Frequency*
DTD	*Document Type Definition*
EE	*Enterprise Edition*
EMF	*Eclipse Modeling Framework*
GEF	*Graphical Editor Framework*
GSM	*Global System for Mobile Communications*
GMF	*Graphical Modeling Framework*
GUI	*Graphical User Interface*
HTTP	*Hypertext Transfer Protocol*
IDE	*Integrated Development Environment*
IETF	*Internet Engineering Task Force*
IP	*Internet Protocol*
IVR	*Interactive Voice Response*
JDBC	*Java Database Connectivity*
JNDI	*Java Naming and Directory Interface*
JSGF	*Java Speech Grammar Format*
JSP	*JavaServer Pages*
MOV	*Movie file extension*
MNDW	*Map Network Drive for Windows*
MRCP	*Media Resource Control Protocol*
OAM&P	*Operation, Administration, Maintenance and Provisioning*

PBX	*Private Branch Exchange*
PB	*Pronunciation Builder*
PCM	*Pulse-code Modulation*
PSTN	*Public Switched Telephone Network*
RAD	*Rapid Application Development*
RTSP	*Real Time Streaming Protocol*
RTP	*Real-Time Protocol*
SCE	*Service Creation Environment*
SDP	*Session Description Protocol*
SI	*Speech Identification*
SIP	*Session Initiation Protocol*
SV	*Speech Verification*
SDK	*Software Development Kit*
SOA	*Service-oriented architecture*
SOAP	*Simple Object Access Protocol*
SRCL	*Speech Recognition Control Language*
SRGS	*Speech Recognition Grammar Specification*
SR	*Speech Recognition*
TCP	*Transmission Control Protocol*
TTS	*Text-To-Speech*
UDP	*User Datagram Protocol*
UDDI	*Universal Description, Discovery and Integration*
UMTS	*Universal Mobile Telecommunications System*
URL	*Uniform Resource Locator*
VoiceXML (VXML)	*Voice eXtensible Markup Language*
VoIP	*Voice over Internet Protocol*
MRCP	*Media Resouce Control Protocol*
WAR	*Web Application Archive*

WF	*Windows Forms*
WSDL	*Web Services Description Language*
WWF	*Windows Workflow Foundation*
XML	*eXtensible Markup Language*

Glossário

Auto-complete Existência de estruturas de diálogos pré-definidos para terminar a construção de texto.

Barge-In Capacidade de interrupção de um anúncio com o *input* de DTMF ou voz.

Drag and Drop Acção de clicar num objeto virtual e arrastá-lo para uma posição diferente ou sobre um outro objeto virtual.

HTTP HTTP é um protocolo a nível de aplicação utilizado para a transferência de páginas *Web* através da *World Wide Web*. Serve também para a transferência de *scripts* de *VoiceXML*.

Open Source Genericamente refere-se a qualquer programa cujo código fonte é tornado disponível para uso ou modificação por utilizadores ou programadores.

Parsing Análise sintáctica. Verificar se uma determinada frase está de acordo com as regras gramaticais definidas.

MRCP O MRCP é um protocolo proposto pela IETF, este protocolo permite que os servidores de voz fornecam vários serviços de voz aos clientes que a ele estabelecem sessões.

RTP Protocolo do nível da aplicação que define o formato dos pacotes para a transferência de dados (áudio e vídeo) em tempo real. Utiliza para o estabelecimento e controlo da comunicação o protocolo RTSP e utiliza normalmente os protocolos a nível de transporte UDP ou TCP para efectuar a transmissão dos dados.

RTSP Este protocolo do nível da aplicação é utilizado para estabelecimento e controlo da transferência de dados com propriedades de tempo real e utiliza normalmente os protocolos do nível de transporte UDP (*User Datagram Protocol*) ou TCP (*Transmission Control Protocol*) para efectuar a transmissão dos dados.

Softphone Aplicação multimédia para fazer e atender chamadas, que funciona sobre a tecnologia *VoIP*/telefone IP.

Stream Stream pode ser definido com um fluxo de dados em um sistema computacional.

User-Friendly Termo usado para definir a facilidade com que as pessoas podem utilizar uma ferramenta ou objecto a fim de realizar uma tarefa específica.

VoIP	A tecnologia *VoIP* permite a transmissão, em tempo real, de sinais de voz pela *Internet* ou por uma rede privada. Os sinais analógicos de voz são continuamente digitalizados, empacotados e transmitidos como um fluxo de pacotes sobre uma rede de dados. Para estabelecer ligações *VoIP* o utilizador necessita de um telefone SIP ou um telefone *VoIP*.
Voice Browsing	Efectuar a navegação numa determinada árvore de decisão utilizando comandos de voz, é efectuado o reconhecimento da voz e processado a opção respectiva.
Workflow	É a sequência de passos necessários para que se possa atingir um objectivo de acordo com um conjunto de regras definidas.
Wizard	Termo utilizado em interfaces com o utilizador para prover um meio de realizar tarefas complexas através de um esquema passo a passo.

1 Introdução

Actualmente, o mercado das telecomunicações modifica-se e evolui de uma forma extremamente rápida, aparecendo constantemente novas tecnologias que proporcionam a criação e disponibilização de serviços de valor acrescentado. Estes serviços, também designados por serviços interactivos através de CTI (*Computer Telephony Integration*), proporcionam aos utilizadores uma maior interactividade e automatizam as operações que normalmente deveriam ser levadas a cabo pelos utilizadores.

Os serviços interactivos são acedidos através de uma chamada telefónica, por telefone, telemóvel ou *softphone*. Têm como principal objectivo suportar eficientemente o acesso ao mais variado tipo de informação e suportar os mais variados tipos de operações sobre as redes IP (*Internet Protocol*), redes tradicionais PSTN (*Public Switched Telephone Network*) ou as redes móveis de terceira geração UMTS (*Universal Mobile Telecommunications System*).

Podemos afirmar que os serviços interactivos se assemelham a páginas *Web*. A informação contida nos serviços também requer uma constante actualização, o que leva à sua manutenção, recorrendo a consecutivas alterações. Este facto requer um maior esforço e uma maior carga de trabalho, que por vezes se torna difícil e com custos consideráveis para as empresas de telecomunicações.

1.1 Motivação

A necessidade por parte das empresas de telecomunicações de procurar soluções que minimizem os custos na criação e manutenção dos seus serviços interactivos e maximizem a variedade e qualidade desses serviços levou a que recorressem a interfaces gráficas para a criação de serviços interactivos.

As interfaces reúnem num só IDE (*Integrated Development Environment*) a maior parte (idealmente todas) das funcionalidades presentemente disponibilizadas pelos Sistemas Interactivos de Voz (IVR – *Interactive Voice Response*) através dos serviços interactivos. Deste modo recorre-se a interfaces amigáveis, a assistentes de criação de operações e acções de *"drag and drop"* de objectos que representam operações sobre o fluxo da chamada, passando assim, do paradigma de desenvolvimento de serviços interactivos para o paradigma de *designer* de serviços interactivos.

A utilização de interfaces gráficas tem um impacto significativo nos processos de desenvolvimento levados a cabo até então pelas empresas de telecomunicações. Em vez de termos várias fases distintas e elaboradas com ferramentas apropriadas para cada fase de desenvolvimento, temos um IDE que dispõe das ferramentas necessárias para efectuar desde a especificação, o desenvolvimento, os testes e publicação dos serviços interactivos.

Este tipo de interfaces gráficas que estão actualmente em fase de se tornarem aplicações sólidas e comercialmente viáveis para as empresas de telecomunicações, têm ainda que sofrer muitas alterações e melhoramentos de forma a satisfazer todas as necessidades e especificidades que se encontram no desenvolvimento de um serviço interactivo.

1.2 Ambientes Gráficos para a Criação de Serviços Interactivos

O ambiente de desenvolvimento de serviços interactivos, recorrendo a interfaces gráficas, surge da evolução de um considerável número de tentativas, por várias abordagens, de agilizar o processo de criação de serviços. Tradicionalmente, eram desenvolvidos com um *software* específico recorrendo a linguagens de programação específicas de cada Sistema de Resposta Interactiva, com o objectivo de tratamento das chamadas. Contemplam, para além das funcionalidades básicas de tratamento e atendimento das chamadas, também algumas funcionalidades avançadas tais como interacção com o utilizador, reconhecimento de voz e conversão de texto para a fala humana.

As interfaces gráficas agilizam o processo de desenvolvimento de serviços de forma a apresentá-los mais rapidamente aos clientes e paralelamente fornecendo-os a mais baixos custos, com uma utilização baseada em *"drag and drop"* de objectos representativos de operações sobre uma chamada, definindo assim o fluxo do serviço interactivo. As interfaces gráficas são tendencialmente instanciadas em ambientes integrados de desenvolvimento e aglomeram todas as funcionalidades necessárias ao desenvolvimento dos serviços tais como: gestão de ficheiros multimédia, interoperabilidade com sistemas externos, controlo de sessões de utilizadores e publicação dos serviços para as redes tradicionais e redes móveis.

1.3 Objectivos

A presente dissertação tem como objectivo projectar e desenvolver um **Ambiente Gráfico para a Criação de Serviços Interactivos**. Partindo da análise dos ambientes que existem actualmente, propor-se-á um novo ambiente utilizando vários componentes já existentes, de forma a colmatar algumas lacunas, e que se baseia numa ferramenta *open source*. Pretende-se pois, verificar até que ponto é possível criar o ambiente com esta ferramenta.

Para atingir o objectivo proposto, será necessário efectuar uma análise detalhada do actual panorama de desenvolvimento de serviços interactivos, em termos de soluções comerciais e *open source*. Será feita a especificação de uma interface gráfica de tal forma abrangente que ajude a combater os problemas encontrados nas soluções do mercado e proporcionar a utilização do maior número de funcionalidades necessárias para a criação de serviços interactivos. Avaliar-se-á detalhadamente as melhorias e potencialidades obtidas através da utilização das interfaces gráficas, e serão analisadas as vantagens da sua utilização num ambiente de desenvolvimento integrado. Reunindo assim, todas as características e ferramentas de apoio ao desenvolvimento de serviços interactivos, com o objectivo de agilizar este processo comparativamente aos métodos tradicionais. O uso de interfaces gráficas permite a criação de aplicações com uma enorme redução à programação, tornando os processos de Design, Construção, Redefinição do fluxo de uma chamada e todo o ciclo de vida de um serviço numa simples ferramenta de desenvolvimento. Por este motivo, será efectuado o estudo e avaliação dos impactos da utilização das interfaces gráficas nas actuais metodologias de desenvolvimento de serviços interactivos.

1.4 Estrutura do documento

Esta dissertação está organizada em oito capítulos. De uma forma sucinta, pretende-se fazer uma apresentação dos principais conceitos e tecnologias, o estado da arte das soluções comerciais e ferramentas, idealizar uma solução e criar um Ambiente Gráfico de Desenvolvimento para a Criação de Serviços Interactivos.

No primeiro capítulo (1) foi feita uma breve introdução ao conceito, motivação, objectivos deste trabalho e a organização e temas abordados neste documento.

No segundo capítulo (2) faz-se a apresentação dos principais conceitos e tecnologias inerentes ao tema e um pequeno resumo do processo de criação de serviços interactivos recorrendo a ambientes gráficos.

No terceiro capítulo (3) é apresentado o “estado da arte” e avaliação das soluções comerciais para a criação de serviços e o estudo e avaliação de duas ferramentas existentes para o desenvolvimento de uma interface gráfica.

No capítulo quarto (4) é especificada e idealizada uma interface gráfica que reúne idealmente todas as funcionalidades e benefícios encontrados no estudo do “estado da arte” para a criação de serviços interactivos.

Ao longo do capítulo quinto (5) é apresentado um Ambiente Gráfico para a Criação de Serviços. São apresentados os vários componentes que foram utilizados e a demonstração do seu funcionamento num ambiente de desenvolvimento. São também apresentados os vários testes realizados ao ambiente de desenvolvimento criado. Foram realizados testes unitários às principais funcionalidades e testes de usabilidade à interface gráfica.

No capítulo sexto (6) é feita a comparação dos processos de desenvolvimento de serviços interactivos com e sem o recurso a ambientes gráficos de desenvolvimento.

No último capítulo, sétimo (7), são apresentadas as conclusões do autor relativamente ao tema da dissertação, qual a perspectiva seguida, os resultados obtidos e sugestões para a possível prossecução do trabalho já feito.

2 Os principais Conceitos e Tecnologias

Neste capítulo é feita a apresentação dos principais conceitos e tecnologias inerentes aos Ambientes Gráficos para a Criação de Serviços Interactivos. Serão referidos os principais elementos multimédia utilizados e descritas as formas para efectuar o reconhecimento da interacção com o utilizador. Serão descritos os ambientes de desenvolvimento para a criação de serviços interactivos com o recurso a interfaces gráficas, o core dos serviços interactivos, os Sistemas de Resposta Interactiva e a linguagem utilizada para a criação de serviços interactivos baseadas em *Web*. Serão apresentadas as arquitecturas nas quais são disponibilizados os serviços e a forma como estes recorrem ao armazenamento de dados, da mesma forma que é utilizada nas páginas *Web*.

No final, será apresentada uma descrição sucinta da evolução dos Ambientes Gráficos para a Criação de Serviços Interactivos desde as primeiras tentativas até ao actual panorama nas empresas de telecomunicações.

2.1 Prompts

No âmbito dos serviços interactivos uma *prompt* significa a instanciação de um determinado elemento multimédia. Uma *prompt* é a saída de voz, vídeo, email, mensagem ou fax do serviço, ou seja, "aquilo" que a aplicação diz ao utilizador. Também podem ser designadas por "anúncios".

Os serviços interactivos utilizam *prompts* de áudio estáticas (ficheiros pré-gravados) ou dinâmicas (geradas dinamicamente através de *prompts* de texto). Os formatos de áudio frequentemente utilizados nos serviços são o PCM (*Pulse-code Modulation*) nas suas variantes G.711 *a-law* e G.711 μ*-law*.

A evolução das redes tradicionais móveis de GSM (*Global System for Mobile Communications*) para as redes de terceira geração UMTS (*Universal Mobile Telecommunication System*) proporcionou o fornecimento de serviços com débitos mais elevados, tirando partido do aumento da largura de banda. Tendo esta possibilidade, e indo ao encontro da contínua necessidade de disponibilização dos serviços de uma forma *"user-friendly"*, surge a utilização do vídeo como forma de interacção com o utilizador, serviços esses que são comummente designados por vídeo portal. São utilizados *codecs* de compressão *"low-bitrate"*, nos formatos de vídeo H.263 e H.263+.

É também utilizado o email e mensagens escritas para a confirmação das operações feitas pelo utilizador com o objectivo de existir uma confirmação das operações efectuadas. Um exemplo concreto é o envio de uma mensagem ao utilizador num serviço em que este pode alterar o tarifário do seu cartão. O serviço pode enviar a confirmação da alteração do tarifário por mensagem escrita para o telemóvel ou por email para a caixa de correio do utilizador. O recurso ao fax é também uma realidade associada aos serviços interactivos, devido à necessidade de um comprovativo em papel das operações efectuadas pelo utilizador. Um exemplo muito concreto é a compra de um bilhete de avião, em que o utilizador pode seleccionar a opção de receber o bilhete por fax, ficando assim de imediato com o bilhete disponível para o seu uso.

2.2 Tecnologias de voz

Existem várias tecnologias de voz actualmente associadas aos serviços interactivos. Estas tecnologias proporcionam uma maior capacidade de interacção entre as aplicações e o interlocutor, tornando assim, os serviços interactivos mais parecidos com uma conversação entre duas pessoas.

É utilizada a tecnologia ASR (*Automatic Speech Recognition*) para o reconhecimento automático de voz através da interpretação ou reconhecimento da fala humana. Por exemplo, para o reconhecimento do interlocutor ou reconhecimento de um comando por parte do utilizador.

Para a conversão de texto para voz é utilizada a tecnologia TTS (*Text-To-Speech*), que consiste no processo de produção artificial de voz humana. Um sistema texto-voz, realiza a conversão da linguagem textual em linguagem normal, ou seja, de texto para voz. A conversão geralmente divide-se em quatro etapas: análise de texto, pronúncia das palavras, interpretação fonética e geração de sinais de voz parecidos com a voz humana.

É utilizada também a tecnologia SV (*Speech Verification*) em áudio-conferências e vídeo-conferências. Esta tecnologia faz a verificação do utilizador activo numa conversação onde estão presentes vários utilizadores. Tem como objectivo definir qual o canal de voz que deverá ter atribuído uma maior largura de banda no decorrer de uma conversação entre vários utilizadores. Com esta tecnologia consegue-se uma melhor qualidade do sinal transmitido e uma comunicação mais orientada ao interveniente na comunicação, dando prioridade ao utilizador que está mais activo.

Para a identificação do orador, é utilizada a tecnologia SI (*Speech Identification*), que consiste na identificação unívoca do orador através da análise do sinal gerado pela sua voz. Para efectuar a sua identificação, são aplicados algoritmos sobre o sinal da voz do orador, que hoje em dia, já permitem um grau de confiança bastante elevado e um bom desempenho.

2.3 Interacção

A interacção é efectuada com o utilizador através dos elementos multimédia descritos na secção 2.1. O utilizador interage com os serviços interactivos através da sua própria voz ou através de dígitos pressionados DTMF (*Dual Tone Multi Frequency*). Essa informação será reconhecida posteriormente pelo serviço e processada a operação à qual está associada a opção introduzida.

DTMF corresponde ao sinal gerado por um dígito de um telefone quando o utilizador prime uma tecla. Conforme o dígito pressionado, são gerados diferentes pares de frequências (alta/baixa), que são enviados ao IVR a que o cliente se ligou, sendo cada um destes pares único.

De forma similar ao DTMF, é também efectuada a interacção com o utilizador através do reconhecimento de instruções de voz, com recurso à tecnologia ASR descrita na secção 2.2.

Para optimizar o conhecimento dos *inputs* introduzidos por parte dos utilizadores, são utilizadas gramáticas de DTMF e gramáticas de voz. Para a presente dissertação será considerada a norma recomendada pelo W3C – SRGS (*W3C - Speech Recognition Grammar Specification v1.0*) **[W3C-SRGS 07]** para a criação e gestão de gramáticas. Com a utilização

das gramáticas é feito o reconhecimento, através do auxílio de um conjunto de palavras que limitam o universo de possibilidades que são opções válidas introduzidas pelo utilizador. Consistem num conjunto de opções que podem ser extensíveis ao receber qualquer nova entrada, aumentando a funcionalidade da aplicação e a gama de opções válidas. É necessário ter-se no mínimo uma gramática associada a uma operação de reconhecimento.

2.4 Ambiente de desenvolvimento

Para agilizar o processo de desenvolvimento de serviços, recorre-se a ambientes de desenvolvimento em que todas as funcionalidades e aplicações necessárias estão presentes numa só aplicação. Para tornar a utilização mais rápida e intuitiva, são utilizadas interfaces gráficas com acesso "quase" imediato às funcionalidades.

Os ambientes de desenvolvimento integrados, do inglês IDE (*Integrated Development Environment*), são ambientes que reúnem características e ferramentas de apoio ao desenvolvimento de aplicações ou *software* e têm como principal objectivo agilizar o processo de desenvolvimento e a criação de aplicações **[IDE 07]**. Poderão consistir num editor de texto com simples validações intrínsecas à linguagem de programação ou num editor visual com o recurso a *"drag and drop"* de objectos representativos a funcionalidades.

Para o desenvolvimento de um serviço interactivo, deverá existir um componente gráfico para a construção do fluxo da chamada, um componente de compilação para *VoiceXML* (2.6) do fluxo e um componente responsável pela sua exportação para um *Web Server* (2.7), onde ficará publicado e acessível. Poderá existir mais do que um tipo de linguagem de programação, sendo, neste caso, necessárias rotinas de compilação para as várias linguagens. Será também necessária a existência de um módulo para a geração automática de código. Poderá ser a geração de todo o código ou apenas de parte, tornando muito mais rápido o processo de desenvolvimento e deixando a possibilidade ao programador de costumizar essas porções de código.

O componente visual, do inglês GUI *(Graphical User Interface)*, é a designação utilizada para a interface que proporciona aos utilizadores a interacção com uma determinada aplicação recorrendo a atributos gráficos **[GUI 08]**. É um mecanismo de interacção entre o utilizador e computador. Através do rato e teclado o utilizador é capaz de seleccionar objectos gráficos e manipulá-los de forma a obter o resultado pretendido. O recurso a esses objectos permite definir operações ou entidades, e poderão existir ligações entre vários objectos como forma de se definirem acções ou precedências entre acções. O recurso a este tipo de tecnologia torna o uso das aplicações substancialmente mais intuitivo e rápido.

No capítulo 4 é projectada uma interface gráfica para o desenvolvimento de serviços interactivos. Serão definidos quais os objectos visuais que devem existir, as respectivas propriedades e as ligações possíveis entre os vários objectos, para que seja realizada a criação rápida e intuitiva de serviços interactivos.

2.5 Sistemas Interactive Voice Response

Os sistemas IVR são soluções de *hardware* e *software* de atendimento automático de chamadas. O IVR é o componente mais importante do desenvolvimento de serviços interactivos. Sendo o responsável por todo atendimento e gestão das chamadas. Aceitam as chamadas dos utilizadores e realizam a interacção sob a forma de voz – por reconhecimento da fala (ASR) ou tons – gerados através de marcação de teclas DTMF. As respostas aos clientes são, por sua vez, geradas sob a forma de elementos multimédia referidos na secção 2.1. São plataformas de multi-serviços, de arquitectura aberta e flexível de elevado desempenho para a criação de serviços multimédia sobre redes PSTN, que suportam quer as tradicionais tecnologias de comutação de circuitos, através de interfaces de *hardware*, quer as novas tecnologias de encapsulamento de voz em pacotes (*VoIP*) sobre as redes IP. Permitem o desenvolvimento de serviços interactivos, com partilha eficiente dos recursos de telefonia, com acesso a diferentes sistemas de informação e incorporando princípios de gestão centralizada.

2.6 VoiceXML

O *VoiceXML* é uma variante do XML (*Extensible Markup Language*), que foi adaptada para o desenvolvimento de serviços interactivos, sendo também conhecida por VXML. O *VoiceXML*, é uma linguagem modular à semelhança das linguagens de programação por objectos **[Yi-Xuan Li 07]**, que permite criar diálogos de voz, síntese de voz, reconhecimento de voz e interacção com os utilizadores através de DTMF. Veio integrar o mundo de dados com o de voz, através da publicação dos *scripts* em *Web Servers*. Deste modo, permite às empresas de telecomunicações oferecer aos clientes serviços interactivos tendo por base páginas *Web*.

```
1  <?xml version='1.0'?>
2  <vxml version="1.0">
3      <menu id="Simples_Exemplo">
4          <prompt>
5              <audio>
6              Bem-Vindo ao serviço.
7              Diga Notícias para as Notícias da Actualidade.
8              Diga Trânsito para notícias sobre Trânsito.
9              </audio>
10         </prompt>
11         <choice next="noticas.vxml">
12             Notícas
13         </choice>
14         <choice next="transito.vxml">
15             Trânsito
16         </choice>
17         <default>
18             <reprompt/>
19         </default>
20     </menu>
21 </vxml>
```

Figura 1 - Exemplo script VoiceXML

Com a introdução de um interpretador de *VoiceXML* consegue-se tornar independentes os serviços interactivos das APIs (*Application Programming Interface*) dos IVRs, através da utilização dos *scripts VoiceXML*, oferecendo assim, uma maior facilidade de implementação e manutenção de serviços, reduzindo os custos de desenvolvimento.

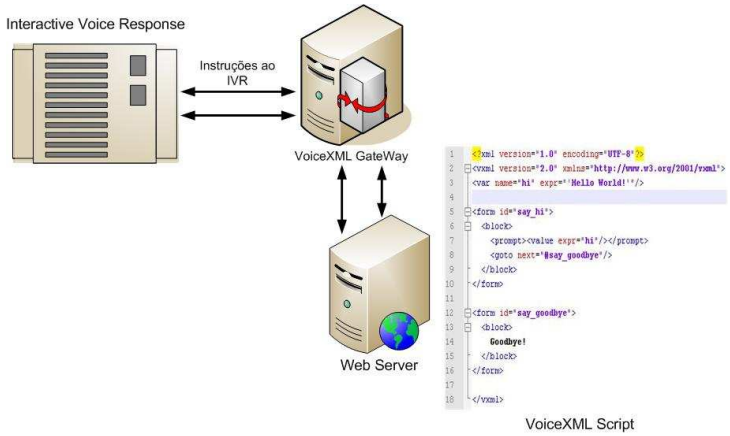

Figura 2 - VoiceXML Gateway ou Voice Browser

O *VoiceXML Gateway* ou *Voice Browser* funciona como um tradutor, acede *scripts* armazenados num servidor HTTP (*Hypertext Transfer Protocol*) comum (local ou remoto) ao IVR e ao *Web Server* e através do interpretador analisa e executa esses *scripts*, utilizando *prompts* de áudio e vídeo, armazenando respostas dos utilizadores e interagindo através de controladores de recursos com os demais serviços e tecnologias. A execução das instruções existentes no *script* fica a cargo do IVR, que é o responsável pela execução das operações identificadas pelo interpretador de *VoiceXML*.

2.7 Arquitecturas dos Serviços Interactivos

Actualmente, os principais sistemas ou aplicações operam de uma forma distribuída, em que todos os componentes prestam serviços entre si para atingir um objectivo comum: o funcionamento global de todo o sistema. Desta forma, proporcionam uma dramática redução dos custos de engenharia e dota-se os sistemas de uma maior agilidade a alterações, e proporciona um maior crescimento sem comprometer a flexibilidade.

Consegue-se colmatar a falta de integração existente na grande variedade de aplicações, fornecedores e plataformas, através de *Web Services* **[WS Activ 07 A]**. Os *Web Services* permitem que a integração de sistemas seja realizada de uma forma fácil, compreensível, reutilizável e normalizada **[Cerami 02]**.

A utilização desta tecnologia torna possível que as aplicações existentes e novas aplicações possam interagir entre si, quer sejam desenvolvidas em plataformas iguais ou diferentes, ficando assim todas compatíveis. Cada aplicação pode ter a sua própria linguagem de programação ou funcionamento. Porém, quando é necessário comunicar com outras aplicações é traduzida para uma linguagem comum entre elas.

Os *Web Services* podem também ser usados para implementar arquitecturas baseadas nos conceitos SOA (*Service Oriented Architecture*), numa arquitectura orientada a serviços. Para a representação e estruturação dos dados nas mensagens enviadas e recebidas é utilizado o XML. As chamadas às operações, incluindo os parâmetros de entrada e saída, são codificadas no protocolo SOAP (*Simple Object Access Protocol*), que também é baseado em XML **[WS Archit 07 B]**. Os serviços (operações, mensagens e parâmetros) são descritos usando a linguagem WSDL (*Web Services Description Language*), e o processo de publicação,

pesquisa e descoberta de *Web Services* utiliza o protocolo UDDI (*Universal Description, Discovery and Integration*).

SOA é um estilo de arquitectura de desenvolvimento de software, cujo princípio fundamental preconiza que as funcionalidades implementadas pelas aplicações devem ser disponibilizadas na forma de serviços **[Open Group 07]**. Frequentemente, estes serviços são organizados através de um barramento, que disponibiliza interfaces (ou contratos) acessíveis através de *Web Services*. A arquitectura SOA **[Oasis 07]** é baseada nos princípios da computação distribuída e programação modular e utiliza o paradigma *request/reply* para estabelecer a comunicação entre os sistemas clientes e os sistemas servidores.

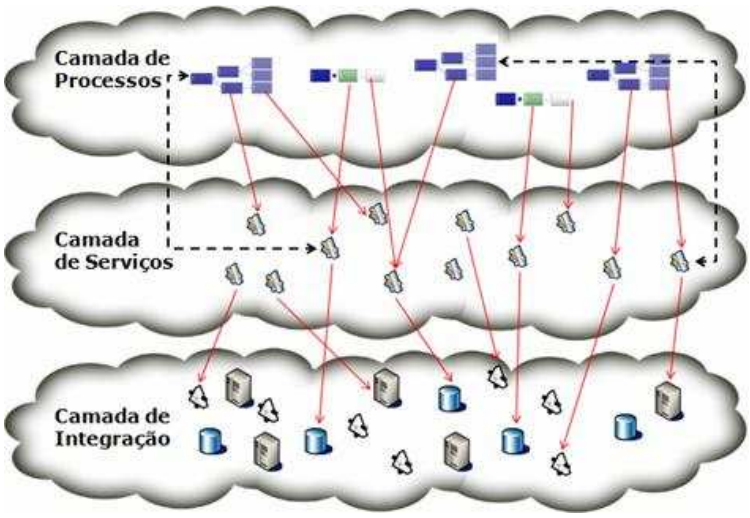

Figura 3 - Service Oriented Architecture

A projecção de um ambiente gráfico para a criação de serviços interactivos enquadra-se numa arquitectura SOA, em que a interface gráfica funciona como um componente prestador de serviços para a arquitectura. A interface gráfica tem como papel na arquitectura proporcionar ao *designer* a criação de fluxos de chamadas, que posteriormente serão utilizados por outros serviços na arquitectura para fazer a sua publicação.

2.8 Armazenamento de informação

Os serviços interactivos têm a necessidade de aceder a base de dados, quer para consultar informação (para a disponibilizar ao utilizador), quer para guardar informação (introduzida pelo utilizador). Por exemplo, num serviço interactivo para a compra de bilhetes de avião, terão de existir consultas à base de dados da operadora aérea, para saber os lugares ainda disponíveis num determinado voo. Poderá também efectuar-se a consulta à base de dados para saber informações sobre o cliente (se é cliente e se tem direito a qualquer tipo de desconto).

Para além desta utilização, as bases de dados são utilizadas para guardar o próprio percurso seguido pelo utilizador, ou seja, o decorrer do fluxo da chamada do serviço. Informações do género, quais os utilizadores que acederam ao serviço, quais as opções que o utilizador tomou ou quais as operações resultantes da utilização do serviço.

2.9 Evolução dos Ambientes Gráficos

Inicialmente, o desenvolvimento de um serviço interactivo era feito sobre um IVR específico e cada IVR tinha a sua linguagem de programação e respectiva API. Este facto, trazia uma enorme dependência entre os IVRs e os serviços. Para colmatar esta dependência e tornar a interacção com os IVRs normalizada surge o *VoiceXML*. O *VoiceXML* é uma linguagem de marcação que uniformiza as instruções a serem executadas pelos IVRs. Este tipo de programação facilita a manutenção dos serviços interactivos, apenas necessitando de um interpretador de *VoiceXML* para fazer a interpretação dos *scripts*.

No entanto, as mudanças do mercado acompanhadas pelos pedidos urgentes de desenvolvimento de novos serviços e de actualização frequente dos já existentes, requerem maior agilidade que aquela proporcionada pela programação utilizando o *VoiceXML*. Assim, o *VoiceXML*, apesar de ser uma mais valia, não é suficiente para dar uma resposta rápida aos pedidos de desenvolvimento de serviços. Desta necessidade, surgem as interfaces gráficas, que proporcionam um ambiente gráfico de desenvolvimento para o programador. As principais funcionalidades do IVR são contempladas e disponíveis de forma intuitiva, fazendo com que o desenvolvimento de um serviço seja ainda mais rápido e expedito, podendo, assim, satisfazer rapidamente o flutuante mercado perante as necessidades de serviços interactivos.

Após o estudo efectuado, podemos dizer que existem três formas para o desenvolvimento rápido de serviços interactivos utilizando o *VoiceXML*. A primeira forma é através de editores de *VoiceXML*, em que a programação é feita com a edição directa do *VoiceXML*. Estes editores foram evoluindo e foram-lhe adicionados *wizards* para adição de *scripts* com porções de *VoiceXML* das rotinas mais comuns. Por último, o recurso a interfaces gráficas ou programação visual, com a utilização de objectos visuais, permite modelar o fluxo da chamada através de *"drag and drop"* dos objectos e no final gerado o *VoiceXML* **[Butler 07]**.

A primeira abordagem referida, o editor de *VoiceXML*, consistia num editor típico de texto, em que o *script* de *VoiceXML* era editado directamente sem qualquer tipo de ajuda ou validação. Este tipo de editores foram evoluindo e ganhando funcionalidades como o *auto-complete*, estruturas de diálogos pré-definidos, verificação e validação do *VoiceXML*. Actualmente, já existem inúmeras soluções no mercado que utilizam esta abordagem. No entanto, não agilizam de forma substancial o processo de criação de serviços interactivos, o que fez com que o seu sucesso fosse condicionado.

A segunda abordagem já teve mais sucesso que a anterior. A utilização de *wizards* para a criação de *scripts VoiceXML* veio diminuir significativamente o tempo de desenvolvimento de serviços interactivos. Utiliza-se um editor de texto e o recurso a *wizards* que permitem adicionar porções de *VoiceXML* com os processos (passos do fluxo da chamada) mais usuais de um serviço. No entanto, a visão global das estruturas é limitada, apenas pequenos passos do fluxo da chamada são atendidos, não sendo, geralmente, o mais abrangente possível, o que torna a flexibilidade na construção de serviços interactivos bastante limitada.

A terceira abordagem referida, o recurso a interfaces gráficas, foi a que obteve mais sucesso até agora. Este facto está a levar a maior parte das empresas de telecomunicações a apostarem em soluções capazes de desenvolver serviços interactivos através de interfaces gráficas. Estas soluções simplificam substancialmente o desenvolvimento de serviços interactivos. Dotadas da capacidade de definir o fluxo da chamada através de *"drag and drop"* de objectos, de vários componentes, de *wizards* de configuração, suportando o recurso a *icons* elucidativos e

de fácil interacção com entidades externas, tornam mais ágil o desenvolvimento. Com as interfaces gráficas, a estrutura global do fluxo da chamada é mais visível, o que proporciona uma maior flexibilidade na definição do fluxo e consequentemente melhor manutenção do serviço face às constantes alterações.

Contudo, ainda continua a existir uma enorme dificuldade em dotar as interfaces gráficas com o suporte de todas as funcionalidades necessárias para a criação de serviços interactivos. É difícil que as interfaces gráficas consigam contemplar todas as especificidades existentes no desenvolvimento destes serviços. No entanto, não impede que uma primeira versão ou uma abordagem simples do serviço seja desenvolvido recorrendo a uma interface gráfica e depois completada por desenvolvimento de código.

3 Estado da Arte dos Ambientes Gráficos para a Criação de Serviços Interactivos

Este capítulo descreve o "estado da arte" dos Ambientes Gráficos para a Criação de Serviços Interactivos.

É feito o enquadramento do tema da dissertação e demonstrada a necessidade das empresas de telecomunicações de ferramentas que agilizem o processo de criação e manutenção dos serviços interactivos. É descrita a arquitectura SOA de desenvolvimento de serviços, onde são apresentados os principais componentes do desenvolvimento de serviços. A interface gráfica tem como função, na arquitectura, proporcionar a criação de fluxos de chamadas para os serviços.

É apresentado o estudo e avaliação do estado da arte em termos de soluções comerciais para o desenvolvimento de serviços interactivos. Para tal, foi elaborado um conjunto de requisitos base para a selecção das soluções. Para cada solução, é feita uma breve descrição das suas funcionalidades e no final é efectuada uma comparação exaustiva entre as várias características de cada uma relativamente aos requisitos definidos.

No final é apresentado o estudo e a avaliação das ferramentas existentes para a criação de raiz de uma interface gráfica para o desenvolvimento e criação de serviços interactivos. Do estudo feito, apenas foram seleccionadas duas ferramentas, as que demonstram mais valias para o desenvolvimento de uma interface.

3.1 Enquadramento

As empresas de telecomunicações desenvolvem serviços com o intuito de satisfazerem as necessidades dos seus clientes, serviços que operam sobre as redes PSTN, IP e UMTS. As necessidades do mercado são de uma extrema variância, o que faz com que o processo de desenvolvimento dos serviços esteja em constante alteração e remodelação **[Yi-Xuan Li 07]**. Toda esta variância implica um trabalho acrescido, uma vez que leva à existência de várias fases de entrega de um serviço e a várias fases de testes internos para verificação dos requisitos pré-estabelecidos. Estes factores tornam mais demorada a finalização do desenvolvimento do serviço e, consequentemente, tornam-no mais dispendioso para as empresas de telecomunicações, o que se reflecte nos preços cobrados aos clientes.

Após a entrega do serviço, já finalizado e pronto a ser utilizado pelo cliente, são feitas alterações e resoluções de possíveis problemas. Praticamente em todos os casos, após alguns dias de utilização dos serviços, estes desejam efectuar alterações e adicionar novas funcionalidades face à evolução das necessidades suscitadas pela sua utilização. Este processo é também moroso e nem sempre possível, devido ao facto de algumas alterações passarem por vezes pela reestruturação total do serviço. Por esta razão é feita normalmente a associação dos serviços interactivos a páginas *Web*, que estão em constante alteração e manutenção, de forma a se encontrarem sempre com informações actualizadas.

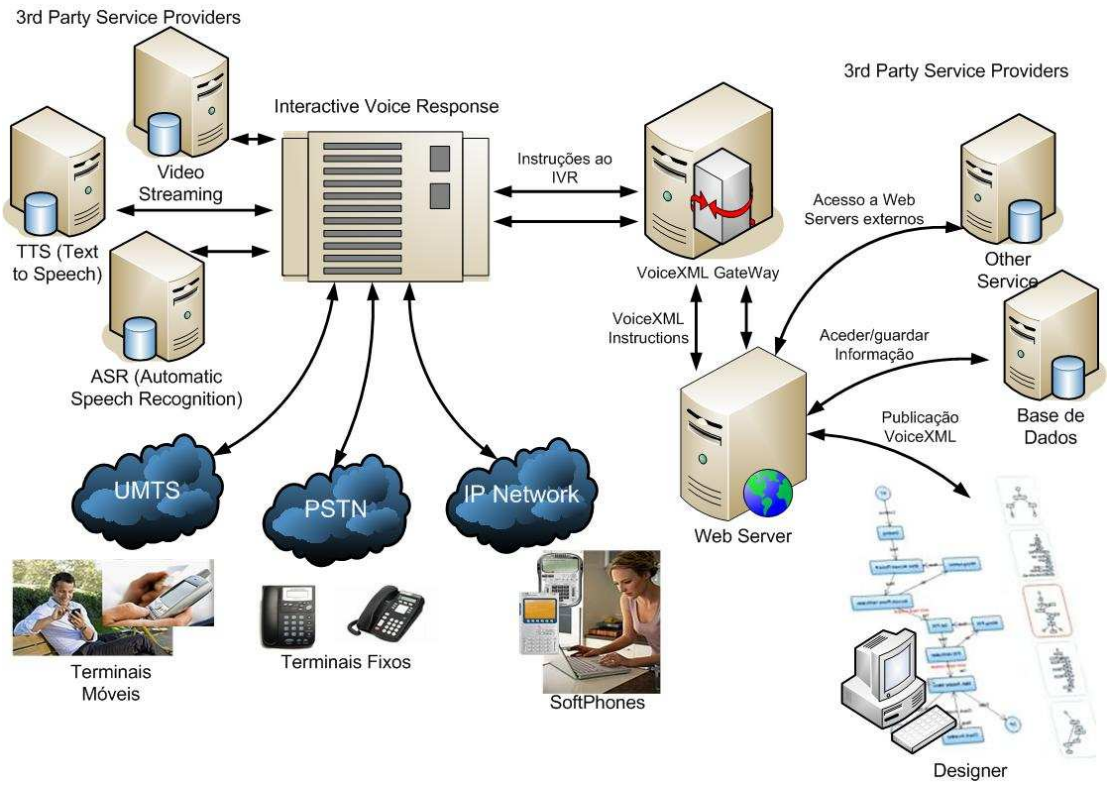

Figura 4 - Enquadramento

Seria óptimo para as empresas de telecomunicações possuírem ferramentas que se enquadrassem na arquitectura de desenvolvimento de serviços e agilizassem o processo de criação e manutenção dos serviços sobre os seus IVRs. A existência de uma interface gráfica de fácil utilização, que abranja todas as funcionalidades dos IVRs, poderia colmatar esta necessidade. Através da interface, o programador ou *designer* pode criar um serviço interactivo de forma intuitiva e rápida, reduzindo, assim, significativamente os tempos de desenvolvimento.

O *designer* irá definir qual o fluxo da chamada (efectuar o desenho do fluxo) num *Canvas* ou folha através de objectos visuais, que ao longo da dissertação irão ser referidos como *Speech Objects* ou Objectos de Voz **[Julian Sinai 07]**. Um *Speech Object* define uma operação sobre o fluxo de uma chamada num serviço interactivo. Estes objectos estarão disponíveis em vários *Stencils* ou *Palettes* de objectos associados à aplicação.

Existem vários tipos de *Stencils*, cada um com grupos de objectos associados a um determinado tema ou área de actuação. O *designer* vai, através de *"drag and drop"* destes mesmos objectos, definir ao longo do tempo o fluxo da chamada, quais as acções a tomar e os respectivos estados da mesma. Este fluxo definido a nível gráfico será utilizado pelo IVR que opera sobre as chamadas atendidas por eles mesmos.

Para além da definição do estado da chamada e quais as operações, o *designer* pode também estabelecer, em qualquer ponto do fluxo da chamada, a ligação com servidores externos. Isto é extremamente útil para que seja possível o desenvolvimento de serviços de forma distribuída, que é, actualmente, a mais utilizada e eficaz. Por exemplo, poderá aceder a uma base de dados, onde estão guardadas determinados dados para mostrar ao utilizador ou até mesmo guardar informação introduzida pelo utilizador.

A Figura 4 ilustra o esquema de desenvolvimento de serviços, que se enquadra numa arquitectura distribuída. Por exemplo, o IVR pode estar num determinado servidor, o *streaming* de áudio, o *streaming* de vídeo, a conversão texto fala (TTS), o ASR e o SR em servidores totalmente independentes e dedicados apenas a fornecer a sua tecnologia ao sistema global **[Butler 07]**. O serviço interactivo deverá estar publicado num *Web Server* independente dos restantes componentes que a arquitectura utiliza. A arquitectura deverá ter interfaces bem definidas entre os vários componentes, para que possam comunicar perfeitamente entre si, tornando assim, a comunicação totalmente transparente para o utilizador **[Inter Gui 05]**. Será da responsabilidade do serviço interactivo, o controlo total das operações, efectuar pedidos de recursos aos respectivos componentes, sempre que o fluxo da chamada assim o necessite.

3.2 Soluções Comerciais

De forma a perceber o "estado da arte" actual em termos de soluções comerciais para o desenvolvimento de serviços interactivos existentes no mercado, foram estabelecidos alguns requisitos (RF – Requisitos Funcionais, RNF – Requisitos Não Funcionais e RS – Requisitos de Sistema) considerados essenciais para seleccionar uma solução. Estes requisitos, foram tomados como necessidades base para a selecção das soluções. No final, será apresentada na Tabela 2 uma comparação transversal destes atributos para as várias soluções.

Tabela 1 - Requisitos genéricos para interfaces gráficas

Código	Requisito	Descrição
RF01	Modelação	Modelação do fluxo da chamada a nível visual com recurso *Speech Objects*.
RF02		Edição directa do *VoiceXML* com o recurso a um editor.
RF03	*Prompts*	Tocar *prompts* de áudio.
RF04		Tocar *prompts* de vídeo.
RF05		Tocar *prompts* de texto.
RF06		Reconhecimento de voz.
RF07		Reconhecimento de DTMF.
RF08		Gravação de voz.
RF09		Gravação de Vídeo.
RF10	Normas	Suporte para o standard *VoiceXML* versões 2.0 e se possível 2.1.
RF11		Ferramentas de construção e edição de gramáticas, segundo a norma W3C – SRGS versão 1.0.
RF12	Operações	Transferência/Encaminhamento da chamada.
RF13		Decisões sobre a informação introduzida pelo utilizador (DTMF ou voz).
RF14	Ferramentas avançadas	*Debugging*.
RF15		*Logging*.
RF16		*Tracing*.
RF17	Interoperabilidade	Capacidade de criação, ligação e interacção com base de dados.
RF18		Capacidade de ligação e interacção com *Web Services*.

Código	Requisito	Descrição
RF19		**Suporte para a publicação da serviço interactivo criado. Localmente ou remotamente para um *Web Server* (ex. Tomcat).**
RS01	**Independência**	**Independência da plataforma IVR**
RS02		**Independência do interpretador de *VoiceXML*.**
RS03		**Solução multiplataforma.**
RNF01	**Interface**	**Interface amigável de utilização rápida, fácil e intuitiva dos *Speech Objects*.**
RNF02		**Utilização e reutilização de vários componentes (componentes para serem adicionados à criação do fluxo da chamada).**
RNF03		**Possibilitar a utilização de *Templates* para definir fluxo da chamada de voz (pequenos diálogos predefinidos).**
RNF04		**Utilização de *wizards* para a criação dos vários passos do fluxo da chamada.**
RNF05	**Licença**	**Se possível uma solução *open source*.**

Após a pesquisa efectuada, foi criada uma lista de soluções existentes no mercado que poderiam satisfazer o desenvolvimento gráfico para a criação de serviços interactivos:

- *IBM WebSphere Voice Toolkit* – **www.ibm.com**;
- *Audium* - **http://www.audiumcorp.com/**;
- *Aspect* - **http://www.aspect.com/**;
- *Avaya* - **http://www.avaya.com/**;
- *OmniView SCE* - **http://www.apexvoice.com/**;
- *GetVocal SDK* - **http://www.getvocal.com/**;
- *Genesys Studio* - **http://www.genesys.com/**;
- *Voice Objects Desktop* - **http://www.voiceobjects.com/**;
- *Vicorp xMP Solutions* - **http://www.vicorp.com/**.

Algumas das soluções encontradas foram descartadas, devido ao seu estado inicial de desenvolvimento, ou por serem soluções prometidas pelas empresas em que apenas se terminaria o desenvolvimento no caso de existirem possíveis compradores para a solução.

Foram também eliminadas as soluções que não cumpriam algum dos requisitos essenciais, com a excepção da *IBM WebSphere Voice Toolkit*, pois esta não tem um editor visual para a criação de serviços interactivos. Optou-se por descrevê-la como forma de demonstrar como o recurso à criação dos *scripts* de *VoiceXML* também é uma solução frequentemente utilizada, e com alguns resultados positivos. Excluindo este pormenor, a solução é extremamente completa e abrange quase todos os recursos necessários para o desenvolvimento de serviços interactivos.

3.2.1 IBM WebSphere Voice Toolkit

IBM WebSphere Voice Toolkit (Figura 5) é uma plataforma completa para desenvolvimento de aplicações de voz baseadas em *VoiceXML* **[IBM 08]**. Não tem um componente gráfico de *"drag and drop"*, no entanto, dá a possibilidade ao *designer* de definir o fluxo das chamadas

através da edição directa recorrendo a *scripts VoiceXML*. Existem outras aplicações com a edição de *VoiceXML* de modo não gráfico, mas optou-se apenas por se descrever a solução da *IBM WebSphere* por se considerar ser das mais completas no mercado.

O editor de *VoiceXML* tem uma funcionalidade muito vantajosa, que é o *auto-complete* das *tags* de *VoiceXML*, isto é, ao digitar uma letra o editor faz o *parsing* dos possíveis campos *VoiceXML* que têm início com a letra pressionada. Para além do *auto-complete*, é feita uma pré-selecção dos campos a apresentar através de uma validação do campo dentro do contexto em que foi digitada a letra, tendo por base a DTD (*Document Type Definition*). Esta potencialidade leva a que o *designer* não cometa erros na estrutura do fluxo da chamada, eliminando à partida possíveis erros que, em outros casos, só poderiam ser detectados após a publicação do serviço.

Baseado na plataforma *Eclipse* IDE, inclui as funcionalidades tradicionais do IDE como o *Project/View Management* e gestão da versão do código CVS (*Concurrent Version System*). Contém um conjunto integrado de ferramentas para *VoiceXML*, incluindo *wizards* de geração de aplicações, ferramentas de desenvolvimento, testes de gramáticas JSGF (*Java Speech Grammar Format*) e SRCL (*Speech Recognition Control Language*). Esta aplicação inclui ferramentas para construir *scripts VoiceXML* estáticos e *scripts VoiceXML* dinâmicos através de JSP (*JavaServer Pages*) e ASP (*Active Server Pages*). Inclui também um *Debugger* de desenvolvimento do conteúdo estático *VoiceXML* e de J2EE (*Java Platform, Enterprise Edition*), que é extremamente útil para o *designer* testar os seus serviços antes de proceder à sua publicação. Contempla também várias funcionalidades para a manipulação de ficheiros de voz, tais como PB (*Pronunciation Builder*), gravador de áudio, motor TTS (IBM TTS) e SR (*Speech Recognition - IBM ViaVoice*).

A solução completa *WebSphere* requer a instalação dos seguintes componentes: *Voice Toolkit*, *WebSphere Voice Server SDK* e *IBM Reusable Dialog Components*. O componente de *Debugger*, o *Voice Application Debugger*, é um componente opcional, mas acrescenta importantes facilidades a nível de depuração de erros, verificando cada passo do fluxo da chamada definido.

A principal vantagem desta solução é a possibilidade de testar o serviço definido sem ter que fazer a sua publicação, possuindo para isso motores TTS e ASR, assim como um interpretador *VoiceXML*. Ou seja, depois de definido o fluxo da chamada, basta seleccionar a opção para fazer a conversão para o *script VoiceXML* e o serviço fica disponível. Uma outra grande vantagem é a funcionalidade validação dos campos de *VoiceXML*, fazendo com que não seja permitido inserir elementos em locais inválidos, forçando desta forma, a satisfazer o *standard VoiceXML 2.0*.

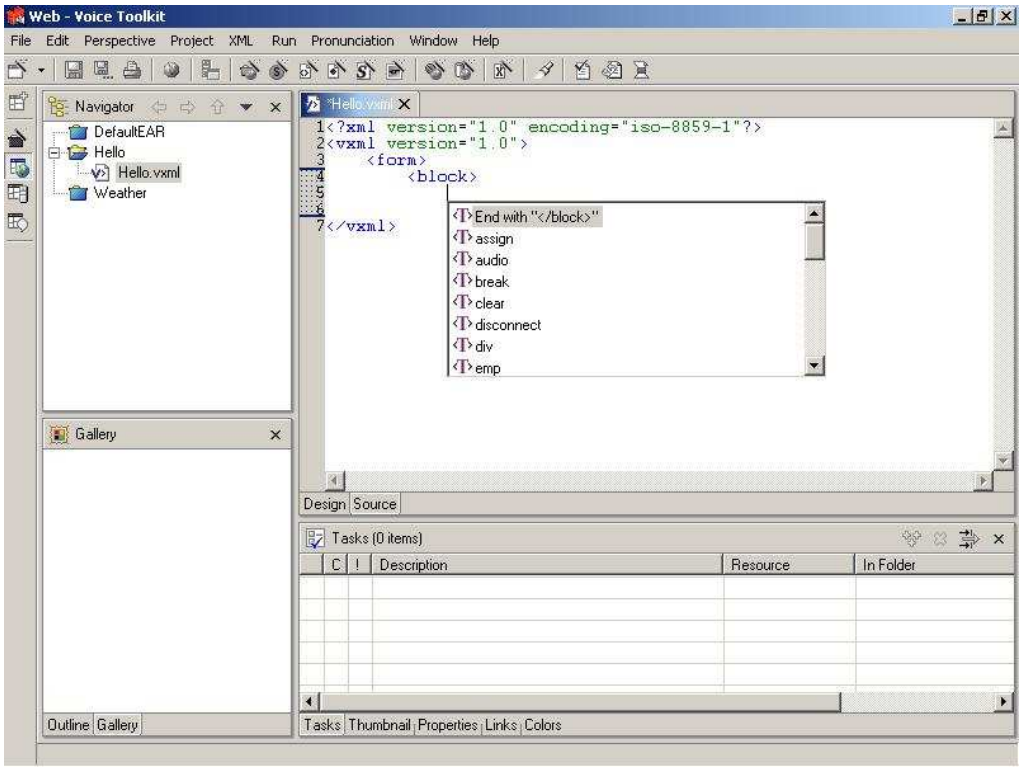

Figura 5 - IBM WebSphere Voice Toolkit

3.2.2 Audium

A empresa *AudiumCorp*, que foi adquirida recentemente pela *Cisco Systems*, apresenta a solução *Audium Studio* (Figura 6). Tem também por base a plataforma *Eclipse* IDE, beneficiando por isso, de todas as suas funcionalidades de ambiente integrado de desenvolvimento, sendo extensível por intermédio do uso de *Plug-Ins* e *Features* **[Audium 08]**.

A solução *Audium Studio* contém um numeroso conjunto de blocos pré-definidos de pequenas porções de *VoiceXML*, em que o *designer* apenas tem de configurar alguns parâmetros para os adicionar ao fluxo da chamada. Para além destes módulos já existentes, a aplicação dá a possibilidade do *designer* definir os seus próprios blocos e de guardá-los para serem utilizados em qualquer outro serviço desenvolvido posteriormente.

A grande desvantagem da *Audium Studio* é a dependência das camadas mais baixas do desenvolvimento de serviços. É necessário que a *Audium* desenvolva uma *Gateway* (com o funcionamento de *Adapter*), para que seja possível interagir com diferentes IVRs. Este processo de ajuste à plataforma, de forma a dotar a compatibilidade com os IVRs, tem um tempo estimado de seis semanas para desenvolvimento e implementação das alterações, execução e testes do adaptador.

Uma outra desvantagem é não contemplar *Speech Objects* que interajam directamente com base de dados. No entanto existem blocos para a interligação e troca de informação com *Web Services*, podendo-se assim, de uma forma não tão prática, contornar esse problema.

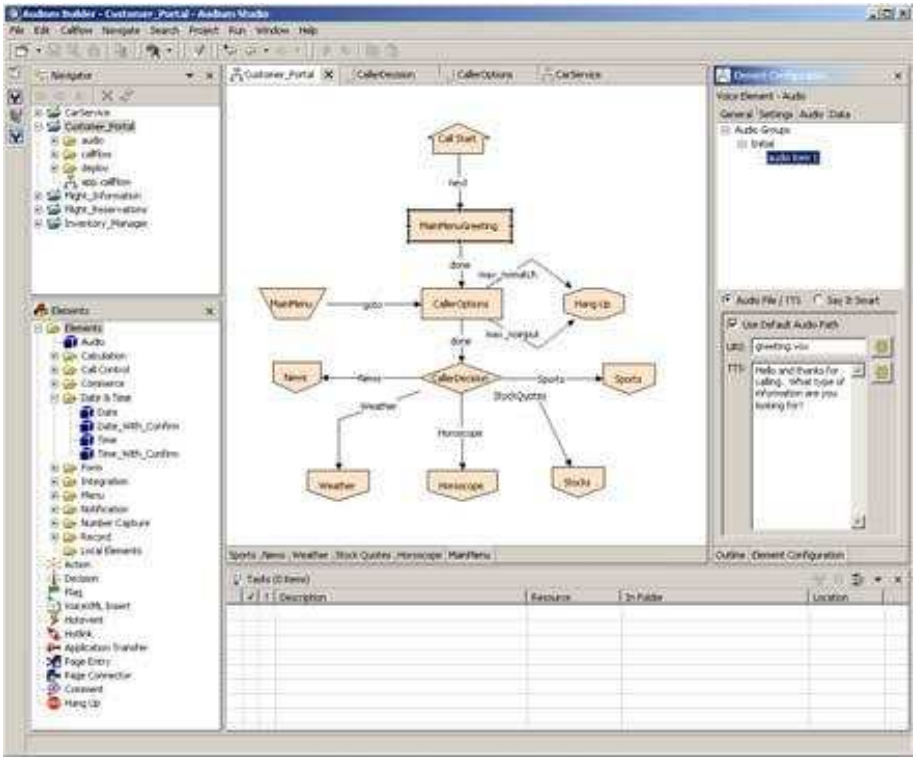

Figura 6 - Audium Studio

3.2.3 Aspect

A solução da *Aspect* (Figura 7) contempla todas as funcionalidades básicas e algumas avançadas para a criação de serviços interactivos, como por exemplo a ligação a base de dados **[Aspect 08]**. Engloba um conjunto de *wizards* para a criação de determinados blocos ou porções de *VoiceXML*, evitando, assim, que o utilizador cometa erros na criação destes. Para além disto, existe um sistema de validação das ligações entre os vários *Speech Objects*. Este sistema selecciona os possíveis atributos ou dados a serem analisados entre os vários *Speech Objects*. Depois de definido o fluxo da chamada, fornece a opção para fazer a publicação do serviço, e automaticamente, será efectuada a conversão do fluxo em *VoiceXML*, podendo, assim, ser publicado directamente para um *Web Server*.

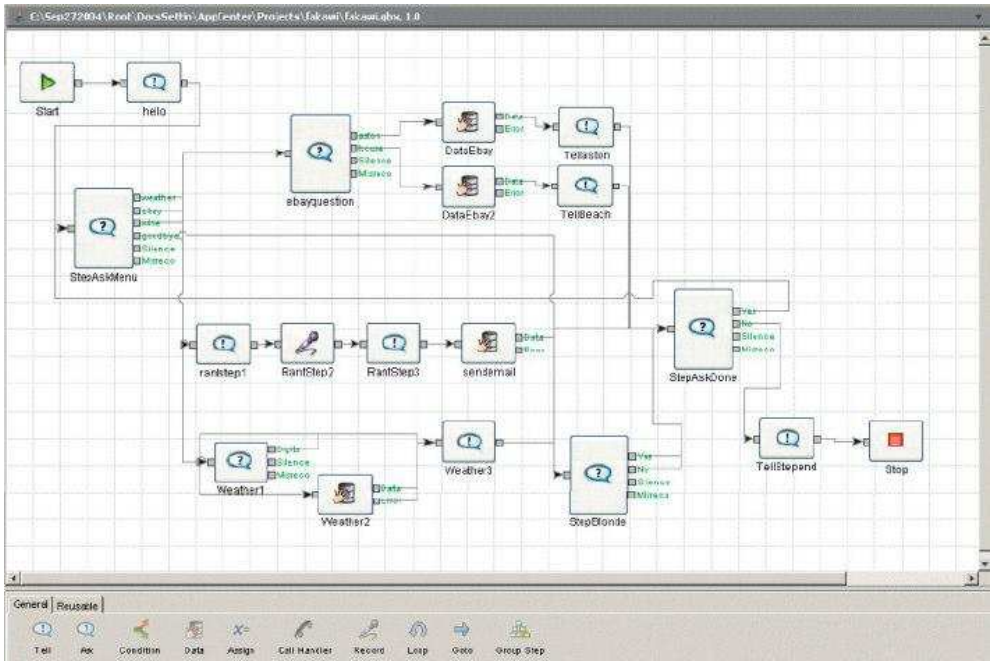

Figura 7 - Aspect

3.2.4 Avaya

A aplicação da *Avaya* (Figura 8) para a definição do fluxo de chamada baseia-se totalmente em funcionalidades recorrendo a *"drag and drop"* de *Speech Objects* **[Avaya 07 A]**. Esta aplicação recorre frequentemente a *wizards* para a criação de vários elementos do fluxo da chamada, tais como gravação, tocar anúncio, definição de gramáticas, reconhecimento da voz, transferência da chamada. A *Avaya* é muito completa e abrange praticamente todas as funcionalidades que actualmente implementam os serviços interactivos. Para além das funcionalidades básicas, contém inúmeras funcionalidades avançadas que podem ser adicionadas através de *wizards* e permite descartar à partida a maior parte dos erros comummente praticados na implementação de serviços.

É importante destacar o *Speech Object* de ligação à base de dados. Este bloco consegue através de *wizards* muito fácies de utilizar, estabelecer a ligação com a maior parte das bases de dados existentes (*PostegresSQL*, *Oracle*, *MySQL*) **[Avaya 07 B]**. Também fornece um conjunto de operações pré-estabelecidas de consulta e inserção na base de dados, que facilmente podem ser editadas de acordo com as necessidades do serviço. A abordagem utilizada para a ligação com base de dados é também utilizada para o estabelecimento e comunicação com servidores externos. Esta é também uma funcionalidade que merece destaque, por ser bastante completa e de fácil configuração. Existe um *Speech Object* que dá possibilidade de definição de ligações a *Web Services*, de forma a obter informações necessárias ao serviço ou até mesmo para enviar *inputs* introduzidos pelo utilizador.

A conversão do fluxo da chamada é efectuada automaticamente em tempo real e garante o funcionamento do serviço interactivo, ou seja, é assegurado que a sua publicação seja compatível com a maior parte dos *Web Servers* existentes actualmente.

A principal desvantagem desta solução é a total dependência dos restantes componentes existentes na criação de serviços interactivos. O funcionamento da interface gráfica requer a instalação de *Application Server Components* da *Avaya* (*Gateway* de *VoiceXML*, TTS e ASR).

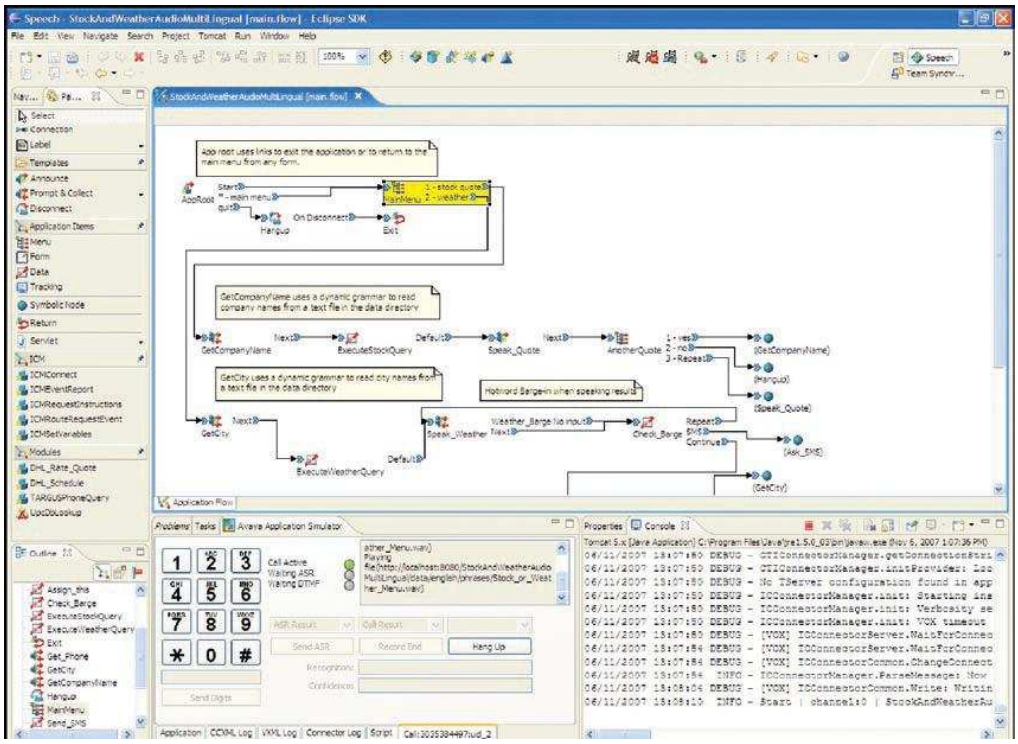

Figura 8 - Avaya

3.2.5 OmniView SCE

A solução *OmniView SCE* (Figura 9) é fornecida pela empresa *ApexVoice*. Esta solução é totalmente baseada no ambiente *Web* e faz parte de um conjunto de soluções de OAM&P (*Operation, Administration, Maintenance and Provisioning*) da *ApexVoice* para os serviços interactivos **[ApexVoice 08]**.

A aplicação não contempla muitas das funcionalidades avançadas, como o fazem a maior parte das restantes soluções apresentadas até então. Não existem *Speech Objects* para a ligação a servidores externos, não existe o conceito de ligação a base de dados ou a *Web Services*. Estão apenas presentes as funcionalidades básicas dos serviços interactivos. No entanto, tem uma enorme vantagem perante todas as restantes soluções: o componente de gestão e administração dos serviços. Esta funcionalidade é considerada como uma mais valia para esta solução e uma falha nas restantes. Toda a monitorização é efectuada através de páginas *Web* com uma actualização constante dos dados. A qualquer momento podemos parar ou disponibilizar um serviço, ou até mesmo analisar exaustivamente os acessos que o serviço teve e quais as interacções que cada utilizador teve com este.

A principal desvantagem da *OmniView SCE* é a total dependência das camadas inferiores, isto é, não consegue ser independente do interpretador de *VoiceXML* da *ApexVoice* (*OmniVoXML Media Gateway*) e do IVR (*OmniVox3D Application Server*).

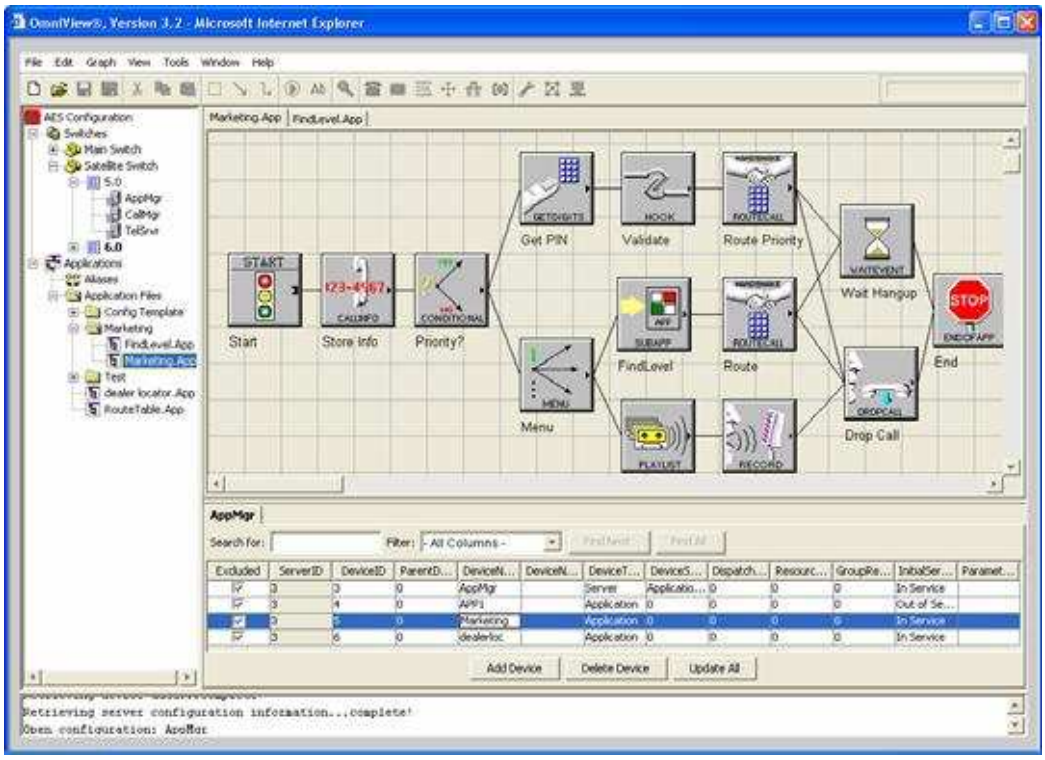

Figura 9 - ApexVoice

3.2.6 GetVocal SDK

A principal vantagem da solução *GetVocal* (Figura 10) é a integração de dois modos de criação de serviços interactivos, utiliza um editor de *VoiceXML* e um editor visual e a edição dos serviços é feita nos dois modos em simultâneo **[GetVocal 08]**. O editor inclui a funcionalidade de *auto-complete* das *tags* e algumas porções de código já pré-gerado. Estas porções de código podem ser introduzidos através de vários *wizards*. Inclui também, um simulador de execução dos serviços, em que ao mesmo tempo que se desenvolve o serviço podemos simular chamadas para este mesmo e assistir à evolução da sua criação.

A *GetVocal* apresenta algumas desvantagens. Por um lado o editor visual é bastante rudimentar, uma vez que apenas contempla as funcionalidades básicas dos serviços interactivos. Por outro lado, no modo gráfico, a adição de novos blocos é um processo moroso, pois os *wizards* nem sempre são apresentadas de forma intuitiva e de fácil utilização e configuração. A *GetVocal* é também totalmente dependente dos motores de TTS e ASR (*Speech SDK 5.1* da *Microsoft*). Apesar de ter a possibilidade de exportação do fluxo da chamada para *VoiceXML*, não suporta a sua exportação para um *Web Server* (terá de ser feita manualmente).

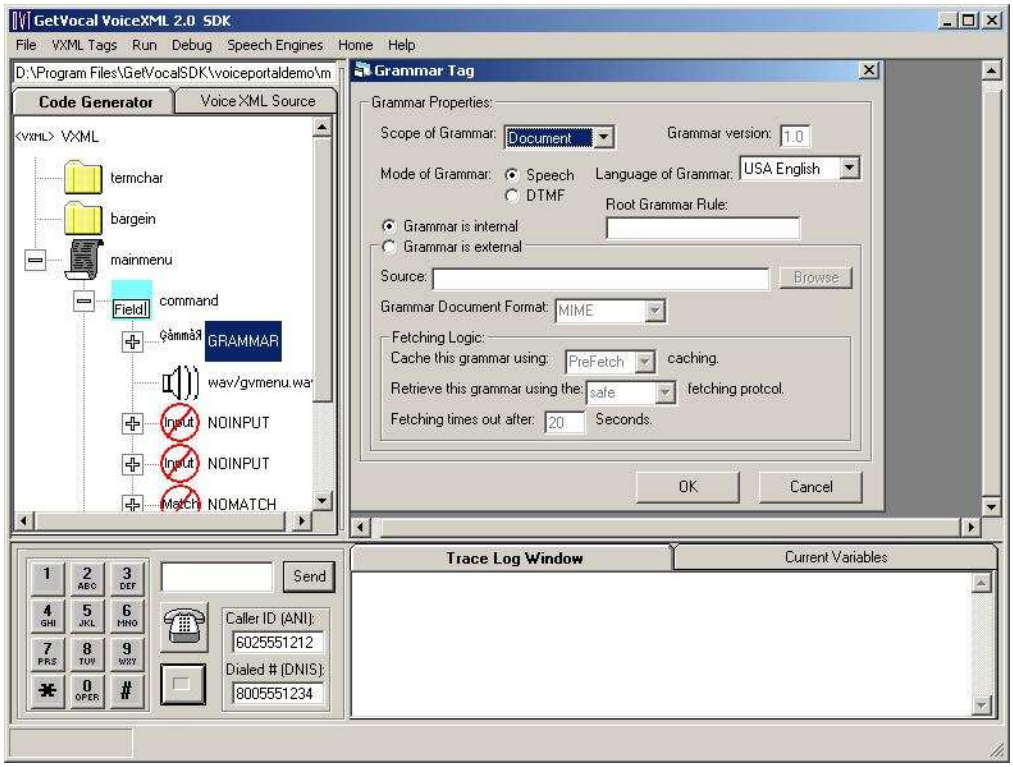

Figura 10 - GetVocal

3.2.7 Genesys Studio

A solução *Genesys Studio* (Figura 11) resulta da compra pela *Genesys*, que pertence ao grupo *Alcatel*, da solução gráfica para criação de serviços interactivos da *Telera DeVXchange* **[Genesys 08]**. Neste momento, a solução *Genesys Studio* apenas é fornecida aos clientes que tenham adquirido plataforma GVP (*Genesys Voice Platform*), o que pressupõe, à partida, uma total dependência da interface gráfica dos restantes componentes da solução *Genesys Voice Platform*.

A interface gráfica é bastante completa, pois contempla a maior parte das funcionalidades necessárias para a implementação de serviços interactivos. Os *wizards* são muito completos e de fácil utilização. Oferece *Speech Objects* para a interligação com servidores externos, tais como base de dados, *Web Services* e JSPs. Os *Speech Objects*, que representam as funcionalidades ou operações no decorrer do fluxo de uma chamada, são bastante parametrizáveis e configuráveis. Permite também ao *designer*, a criação costumizada dos seus próprios blocos *VoiceXML* e a sua utilização no desenvolvimento de outros serviços interactivos.

O desenvolvimento do serviço interactivo é feito com a edição em simultâneo a nível gráfico e a nível do editor de *VoiceXML*. A aplicação está subdividida horizontalmente em duas janelas: na parte superior o modo gráfico de edição visual e na parte inferior o *script VoiceXML* correspondente.

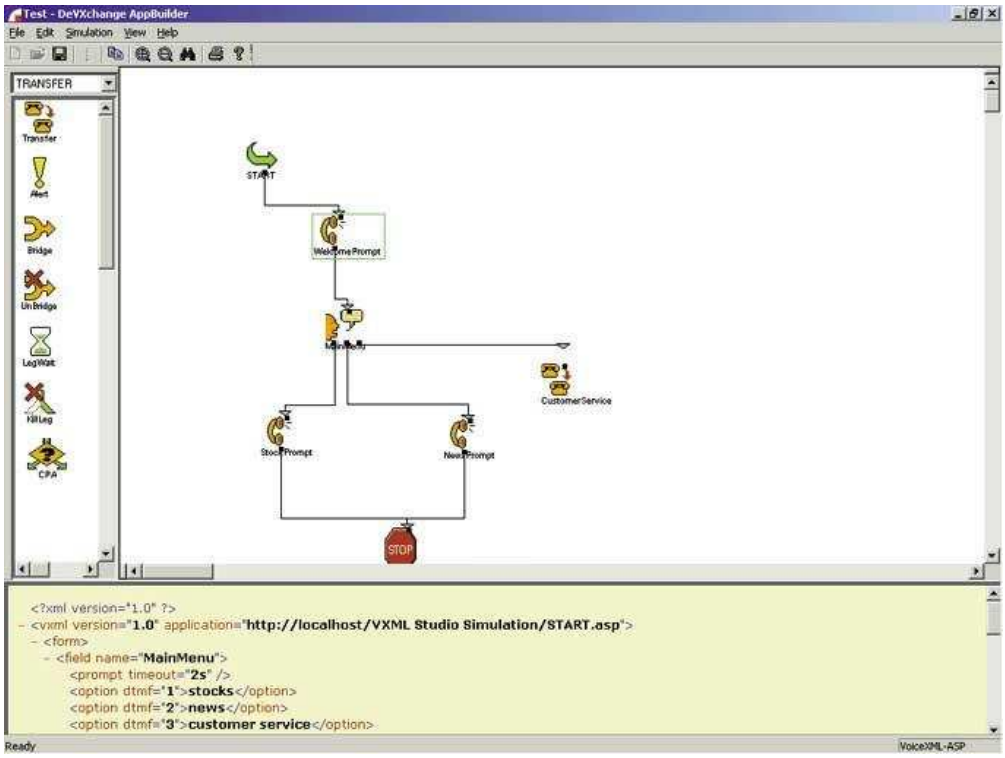

Figura 11 - Genesys Studio

3.2.8 Voice Objects Desktop

A interface gráfica *Voice Objets Desktop* (Figura 12) é um componente que faz parte de uma solução que engloba vários componentes para a criação de serviços interactivos da *Voice Objects* **[Voice Objects 08]**. A solução completa contém o componente *Server* onde serão alojados os serviços. O componente *Analyzer* é responsável pela geração de estatísticas de utilização dos serviços. O componente *Studio* é desenvolvido sobre a plataforma *Eclipse* IDE para a construção visual de serviços através da utilização de vários *Plug-Ins*.

A solução *Voice Objects Desktop* utiliza uma interface *Web* para a criação de serviços interactivos, o que lhe permite uma independência total do sistema operativo. Contempla todas as funcionalidades básicas de criação de fluxos de chamadas, através de uma representação de *Speech Objects* bastante modular e em várias secções de uma *Palette*.

Está dividida em quatro grandes áreas: *Components*, *Resourses*, *Logic* e *Actions*. A secção de *Components* disponibiliza blocos de fluxo da chamada pré-definidos. Tais como, a representação de um menu de opções, *input* de informação por parte do utilizador ou a confirmação de uma opção introduzida pelo utilizador. A secção *Resources* é responsável pelo tratamento de *prompts* e de gramáticas. A secção *Logic* efectua o tratamento de variáveis, expressões, decisões e ciclos. Por último, a secção *Actions* engloba um conjunto de acções a tomar sobre a chamada, tais como a transferência, gravação e a finalização da chamada.

A solução dá a possibilidade da total configuração de cada *Speech Object*, podendo o *designer* definir os seus *Speech Objects*, guardá-los com as suas configurações favoritas e utiliza-los na criação de qualquer outro serviço.

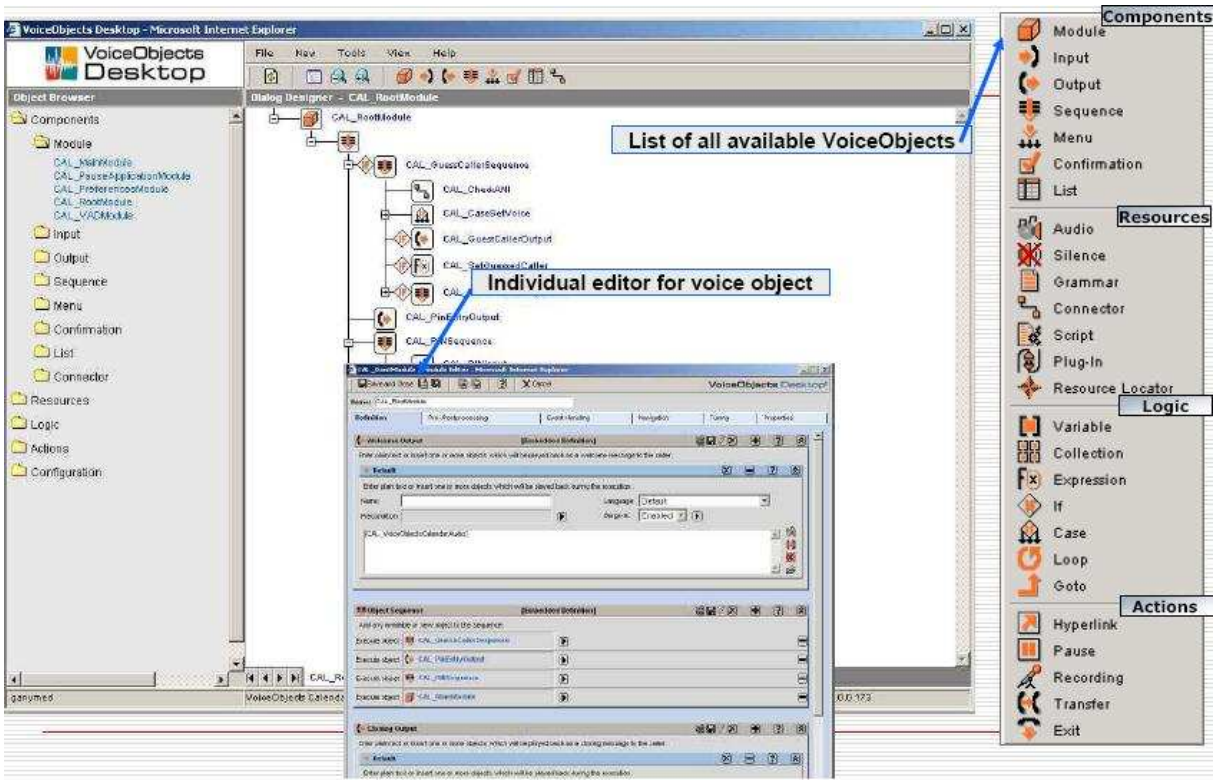

Figura 12 - Voice Objects Desktop

3.2.9 Vicorp xMP Solutions

A *Vicorp* oferece às empresas de telecomunicações um conjunto variado de soluções para a criação e desenvolvimento de serviços interactivos. Uma das suas soluções é a *xMP* (Figura 13), que engloba cinco componentes: *Studio*, *Console*, *Reporter* e *Director* **[Vicorp 07]**.

O componente *Studio* foi desenvolvida sobre a plataforma *Eclipse* IDE, e contém as principais funcionalidades de gestão, criação de *prompts* e criação de gramáticas. O componente *Console* é utilizada através de uma interface *Web*. Este componente contempla a parte de OAM&P dos vários serviços disponibilizados nos *Web Servers*. O componente *Reporter* utiliza também uma interface *Web* e apresenta a geração automática de estatísticas e análises de utilização (sobre o número de chamadas efectuadas e opções tomadas pelos utilizadores) nos serviços interactivos. Por último, o componente *Director* é o *core* da solução *xMP* e é responsável pela criação e desenvolvimento dos fluxos das chamadas referentes aos serviços interactivos.

O componente *Director* opera sobre a plataforma *Eclipse* IDE, o que lhe dá todas as vantagens do desenvolvimento de software baseado nesta plataforma. Para além destas funcionalidades proporcionadas pela plataforma, acrescenta várias *Palettes* de *Speech Objects* que representam as operações sobre o fluxo das chamadas. Contempla uma enorme variedade de blocos representativos de operações, que podem ser adicionados através de *wizards*. São fornecidas, para além das funcionalidades básicas, muitas funcionalidades avançadas, como a ligação a base de dados para consulta ou inserção de informação. Suporta a ligação e configuração a *Web Services* como forma de interacção com serviços externos ao desenvolvimento.

Apesar do componente *Director* fazer parte de um conjunto de componentes da solução *xMP*, esse componente consegue ser totalmente independente das restantes. Isto é possível devido ao facto de todos os componentes se basearem em normas como forma de comunicação entre elas. Desta forma, suporta a publicação dos serviços interactivos para a maior parte dos *Web Servers* existentes, através da norma *VoiceXML 2.0*.

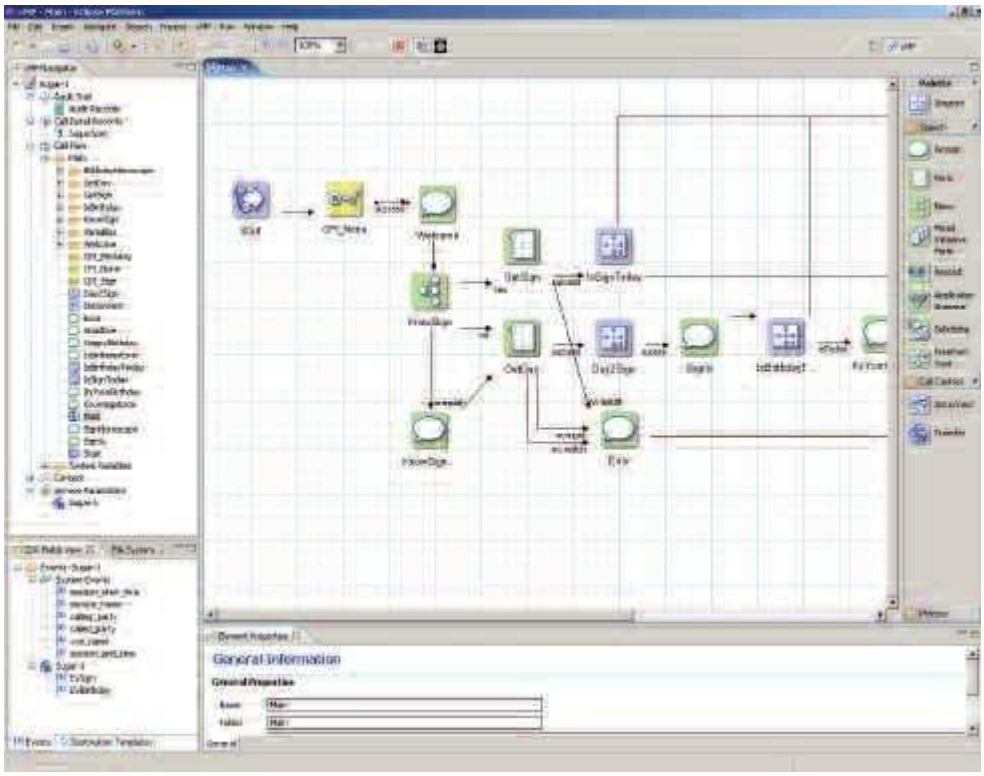

Figura 13 - Vicorp xMP Editor

3.3 Avaliação das Soluções Comerciais

Após o estudo das várias soluções, foi feita uma comparação exaustiva entre as diversas soluções utilizando como termo comparativo os requisitos definidos na secção 3.2, com esta comparação obteve-se a Tabela 2.

Tabela 2 - Comparativo entre as soluções comerciais

Requisito / Fornecedor		*IBM WebSphere*	*Audium*	*Aspect*	*Avaya*	*ApexVoice*	*GetVocal*	*Genesys Studio*	*Voice Objects*	*Vicorp xMP*
Modelação	**RF01**		X	X	X	X	X	X	X	X
	RF02	X					X			
Prompts	**RF03**	X	X	X	X	X	X	X	X	X
	RF04	X				X		X		
	RF05	X	X	X	X	X	X	X	X	X
	RF06	X	X	X	X	X	X	X	X	X
	RF07	X	X	X	X	X	X	X	X	X
	RF08	X	X	X	X	X	X	X	X	X
	RF09	X		X		X		X		
Normas	**RF10**	X	X	X	X	X	X	X	X	X
	RF11	X	X		X	X	X	X	X	X
Operações	**RF12**	X	X	X	X	X	X	X	X	X
	RF13	X	X	X	X	X	X	X	X	X
Ferramentas avançadas	**RF14**					X				X
	RF15					X		X		X
	RF16					X		X		X
Interoperabilidade	**RF17**	X		X	X					
	RF18	X	X	X	X			X	X	X
	RF19	X	X	X	X	X	X	X	X	X
Independência	**RS01**		X	X						X
	RS02		X	X						X
	RS03	X	X	X	X	X		X	X	X
Interface	**RNF01**		X	X	X		X	X	X	X
	RNF02	X	X	X	X		X	X	X	X
	RNF03	X	X	X	X				X	X
	RNF04	X	X	X	X	X		X	X	X
Licença	**RNF05**									

Apesar das soluções apresentadas anteriormente serem capazes de satisfazer as necessidades da maior parte dos serviços interactivos simples, facilmente se encontram problemas quando pretendemos desenvolver serviços mais rebuscados. As soluções estudadas demonstram dificuldades em abranger a maior parte das funcionalidades necessárias para satisfazer as

necessidades na construção de serviços interactivos com funcionalidades avançadas. Para além desta dificuldade, também é difícil encontrar soluções que sejam independentes dos restantes componentes necessários ao desenvolvimento de serviços interactivos.

O problema mais usual é a dependência total de todos os componentes. Isto acontece porque o desenvolvimento, ao utilizar vários componentes numa arquitectura SOA, torna-se muito difícil perante a necessidade de tornar as comunicações de forma normalizada, fazendo com que, muitas vezes, se utilizem protocolos de comunicações específicos entre alguns dos componentes da arquitectura, ao invés de recorrer a protocolos de comunicação normalizados. Torna-se então, mais fácil o desenvolvimento das interfaces gráficas. Muitas vezes, são utilizados protocolos específicos sem que estas possam interagir com outros componentes da arquitectura, fazendo que estas sejam totalmente dependentes das restantes plataformas das empresas (normalmente do interpretador de *VoiceXML, Web Server* e o IVR).

Uma outra grande desvantagem encontrada nas soluções comerciais é a debilidade das funcionalidades prestadas. A maior parte das soluções apenas disponibiliza as funcionalidades básicas para a criação de serviços interactivos (em alguns casos nem as funcionalidades básicas contemplam), fazendo com que não seja uma mais valia, para as empresas de telecomunicações, utilizarem estas soluções no seu desenvolvimento de serviços. Há uma necessidade enorme de que as interfaces tenham a capacidade de comunicação com servidores externos, dado que, actualmente, os serviços interactivos enquadram-se a arquitectura SOA.

O acesso a servidores e serviços externos ao próprio serviço interactivo é, actualmente, a forma mais utilizada na construção de um serviço. Se não forem disponibilizados meios para o fazer aquando da criação do fluxo da chamada, esta não irá satisfazer as necessidades, nem permitir ao *designer* atingir os objectivos pretendidos para o desenvolvimento de serviços.

3.4 Ferramentas

Após o estudo das soluções comerciais retratado na secção 3.2, foi feito um estudo das ferramentas existentes para a criação de raiz de uma interface gráfica para o desenvolvimento e criação de serviços interactivos.

3.4.1 Windows Workflow Foundation

O *Windows Workflow Foundation* (WWF) pode ser aplicado a vários cenários com a existência de um domínio de actividades, no desenho de interfaces gráficas, diagramas de gestão de recursos e até definição e estabelecimento de tarefas para um modelo de negócio **[Chappell 07]**. Uma actividade ou objecto, para o tema desta dissertação, seria a definição de objectos que simbolizavam operações sobre o fluxo de uma chamada e definição dos possíveis relacionamentos entre os vários passos ou actividades de uma chamada.

O WWF é uma linguagem de programação por modelos da *Microsoft* que disponibiliza ferramentas para o desenvolvimento rápido de um fluxo. Tem por base a *framework .Net 3.0* e um motor de desenho de fluxos que vem integrado com o *Microsoft Visual Studio* **[WWF 07]**.

O *Microsoft* WWF é uma estrutura extensível para o desenvolvimento de soluções de fluxo de trabalho na plataforma *Windows* e é um componente do futuro *Microsoft WinFX* (conjunto de

APIs orientadas a objectos). O WWF fornece a API e ferramentas para o desenvolvimento e execução de aplicações baseadas em fluxo de trabalho **[MDN 07]**.

O WWF é uma estrutura de fluxo de trabalho ampla, projectada desde o início para promover a expansibilidade a todos os níveis. As soluções baseadas no WWF são construídas com componentes interligadas e dá a possibilidade de compor as etapas de um fluxo específico através do *design* recorrendo a uma interface gráfica e adiciona código associado aos componentes do fluxo para implementar regras e definir um determinado processo.

Um fluxo é um modelo de um processo ou de um sistema, para tal, é definido um mapa de actividades. Uma actividade é uma etapa de um fluxo que representa uma unidade de execução, reutilização e composição de um fluxo. O mapa de actividades é expressa através de regras, acções, estados e relações entre actividades. O fluxo é criado por meio de um *layout* de actividades, designado por fluxo de trabalho do WWF e posteriormente compilado para um *assembly*.

O WWF está dividido em quarto partes: *Activity Model*, *Workflow Designer*, *Workflow Runtime* e *Rules Engine*.

A *Activity Model* é a parte responsável pela definição e execução das actividades inerentes a cada passo definido. Poderá representar uma porção de código para a execução de uma acção ou a composição de várias actividades, como por exemplo, a definição de uma condição, a interligação com *Web Services*, operação sobre determinados valores com o intuito de obter um atributo necessário para as próximas actividades.

O *Workflow Designer* é a parte visível para o programador ou *designer* e está associada ao *Visual Studio* (Figura 14). Este componente permite a definição visual de *workflows* de actividades. A cada actividade ou objecto visual estão associados bastantes atributos, que poderão ser configuráveis consoante as necessidades específicas de cada actividade representada.

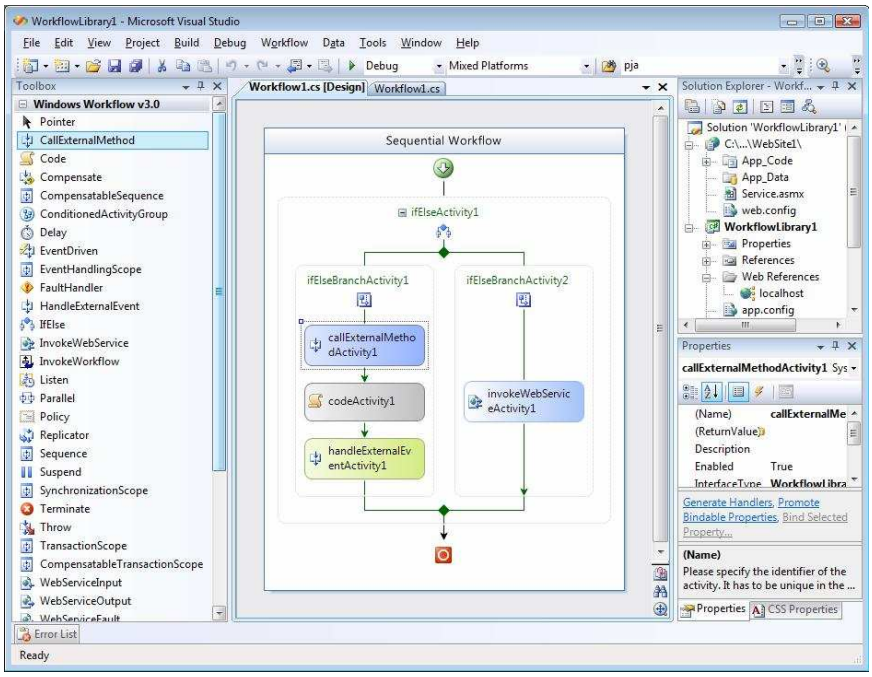

Figura 14 - WWF Workflow Designer

O *Workflow Runtine* é o módulo responsável pela execução das actividades definidas a nível visual para cada modelo de actividade, e as rotinas inerentes à sua execução são levadas a cabo pela *framwork .Net 3.0*. Com este procedimento, e recorrendo a *Windows Forms*, dá-se a possibilidade de serem visíveis os fluxos de actividades definidos em aplicações *Web* através de *ASP.NET*. Também é possível serem visíveis como aplicações ou processos publicados através de serviços no sistema operativo.

O *Rules Engine* (Figura 15) é a parte responsável pela definição de regras sobre uma determinada actividade ou na passagem de fluxo entre actividades. Podemos definir, consoante a verdade ou falsidade de uma regra, a execução de uma operação e definir quais os atributos ou valores que passam para as próximas actividades.

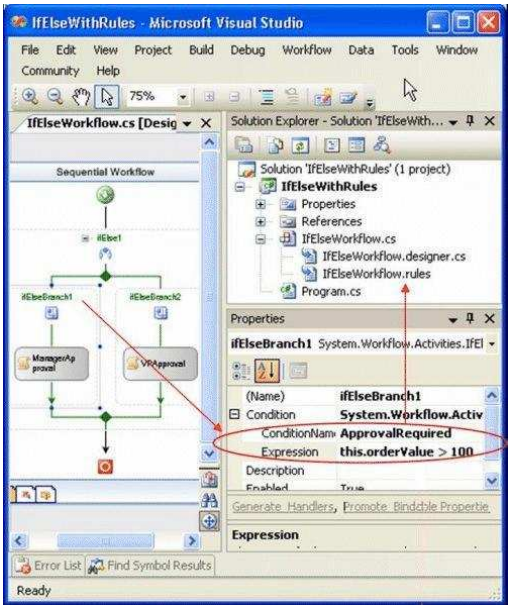

Figura 15 - WWF Rules Engine

As actividades representam a abordagem declarativa à programação de fluxos de trabalho com o WWF. Através do recurso às actividades é possível compor um modelo de fluxo de execução. No decorrer do *design* é também possível a configuração e atribuição de valores às propriedades de cada actividade. Existe para tal um *Stencil* (Figura 16) com um variado conjunto de actividade pré-definidas a que o *designer* pode recorrer para a definição do fluxo.

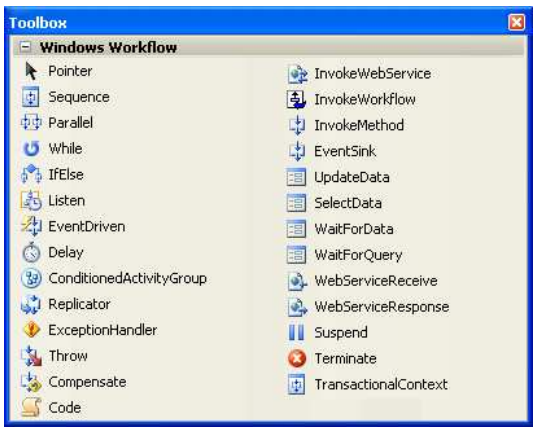

Figura 16 - WWF ToolBox

Para além das actividades pré-definidas, o WWF permite o desenvolvimento de actividades personalizadas, proporcionando assim, expansibilidade do WWF através da definição do conjunto de blocos estruturais que são utilizados para criar modelos de fluxo de trabalho.

3.4.2 Editor visual como *Plug-In* para plataforma *Eclipse*

O *Eclipse* é um IDE com uma plataforma aberta e gerida por uma comunidade aberta, tendo como linguagem base o *Java* e a integração das diversas funcionalidades através de *Plug-Ins* ou *Features*, que são à imagem das bibliotecas para a maior parte das linguagens de programação. Actualmente, existem inúmeros *Plug-Ins* disponíveis em *open source* para a maior variedade de funcionalidades.

No âmbito do desenvolvimento de *Plug-Ins* e para a criação de uma interface gráfica para o desenvolvimento de serviços interactivos, existem três componentes de extrema relevância: *Eclipse Modeling Framework* (EMF), *Graphical Editor Framework* (GEF) e o *Graphical Modeling Framework* (GMF).

A *framework* EMF é utilizada para a modelação e geração de código através de meta-modelos designados por EMFs. Um EMF faz a especificação de uma linguagem visual através de classes que descrevem abstractamente a sua sintaxe, mas não contém nenhuma informação concreta sobre as suas propriedades **[EMF 07]**. Para além desta definição de classes visuais ou objectos, a EMF fornece ferramentas para que em tempo de execução seja possível produzir um conjunto de classes *Java* para o modelo representado. Ou seja, é realizada a instanciação das propriedades das classes abstractas através da edição dos seus atributos, permitindo, assim, a edição do modelo definido inicialmente.

O *Plug-In* GEF permite ao utilizador o desenvolvimento de editores visuais para um modelo de dados especificado num determinado domínio. Um editor construído a partir do GEF requer pelo menos uma plataforma *Eclipse*, pois necessita do seu *Runtime-Workbench*. As acções associadas aos objectos visuais são executadas através de rotinas de atendimento da plataforma *Eclipse* **[GEF 07]**.

O único problema da *framework* EMF é que não consegue interpretar os modelos de dados especificados pelo GEF. A especificação do modelo, a definição dos diagramas para o editor visual e as suas respectivas propriedades, têm de ser feitas através da edição de código **[Ehrig 07 A]**. Como forma de interligar os modelos de dados definidos através de EMFs e os editores baseados em GEF, foi criado GMF. A Figura 17 apresenta o *wizard* (GMF *dashboard*) de criação e configuração de novos editores visuais. O GMF *dashboard* consegue distinguir todos os modelos de domínios de dados já criados e consegue-os combinar para criar um editor visual **[Bierman 07 A]**.

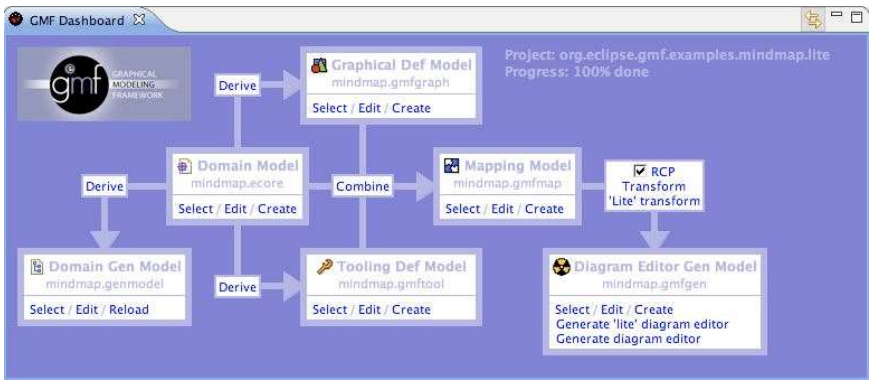

Figura 17 - GMF Dashboard

A definição do diagrama ou fluxo visual será interligado com uma linguagem de definição do domínio EMF e feita a sua representação ou visualização através de um editor GMF. Isto é, o *Plug-In* GMF é a extensão visual dos modelos definidos através de EMF.

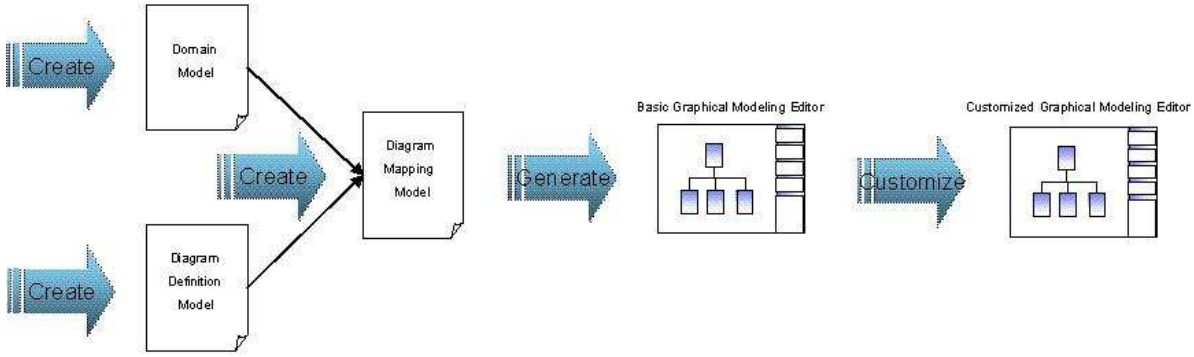

Figura 18 - GMF criação de workflows

Os passos (Figura 18) para a criação do editor visual são:

- ***Domain Model*** – modelo que define a informação não gráfica atendida pelo editor;
- ***Diagram Definition Model*** – modelo que define os elementos gráficos a serem apresentados no editor visual;
- ***Diagram Mapping Model*** – modelo que define o mapeamento entre os elementos do *Domain Model* e os elementos gráficos (definidos pelo *Diagram Definition Model*);
- ***Basic Graphical Modeling Editor*** – primeira abordagem do editor visual criada pela plataforma *Eclipse*;
- ***Costumize Graphical Modeling Editor*** – melhoramento do editor gráfico através da edição do código do *Plug-In* gerado pela plataforma *Ecplise* para o editor visual.

A dependência entre os vários componentes da plataforma *Eclipse* na criação de editores gráficos através do GEF pode ser vista na da Figura 19.

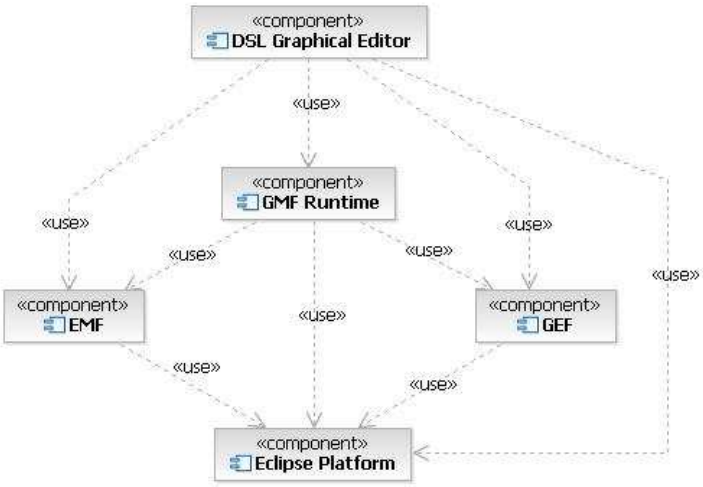

Figura 19 - GMF dependência entre componentes

O editor gráfico depende dos componentes EMF, GEF, GMF *Runtime* e tem por base de funcionamento a plataforma *Eclipse*. Através desta interligação e dependência é possível a definição de um domínio específico para a resolução de um determinado problema. Para tal, é feita a associação entre os objectos gráficos para cada entidade ou operação do domínio dos objectos visuais. Através de um editor gráfico é feita a interligação entre os vários objectos visuais. Posteriormente, será efectuado o mapeamento na informação no domínio especificado.

Após a definição dos modelos de dados e das interfaces gráficas necessárias para o modelo, a plataforma *Eclipse* proporciona a distribuição da própria plataforma como uma solução ou aplicação adicionando os editores e *Plug-Ins* desenvolvidos, tornando-a, assim, uma solução convergente para um IDE e suportando as necessidades para a criação ou desenvolvimento de aplicações específicas **[Ehrig 07 B]**.

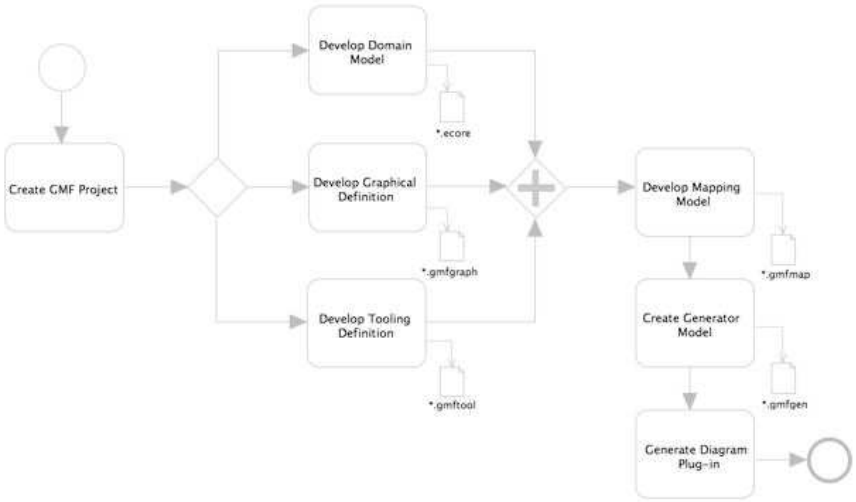

Figura 20 - Publicação como Plug-In de um editor visual

O core do GMF é o conceito do modelo de definição de interfaces gráficas. O modelo contém informação relativa aos elemento gráficos que irão ser apresentados no *GEF-base Runtime*,

sem que haja nenhuma ligação directa com os modelos do *Domain Model*, para os quais o GMF efectua a representação e edição gráfica. É utilizada uma *Tooling Definition* para o desenho do fluxo e dos restantes componentes (*menus*, *toolbars*, etc). É também esperado que a mesma *Tooling Definition* seja aplicada a diferentes modelos de dados (*Domain Model*), atingindo, assim, um dos principais objectos do GMF: a reutilização da definição visual em diferentes domínios através da utilização de diferentes mapeamentos entre modelos (*Mapping Models*).

Depois do estabelecimento do mapeamento o GMF fornece as rotinas necessárias para a geração do novo modelo (*Generator Model*), abrangendo os detalhes de implementação e os detalhes do editor visual. No final será apresentado um novo editor visual sob a forma de *Plug-In* para a plataforma *Eclipse*. A Figura 21 apresenta um exemplo de uma distribuição da plataforma *Eclipse* para um determinado domínio de dados e para os objectos gráficos especificados.

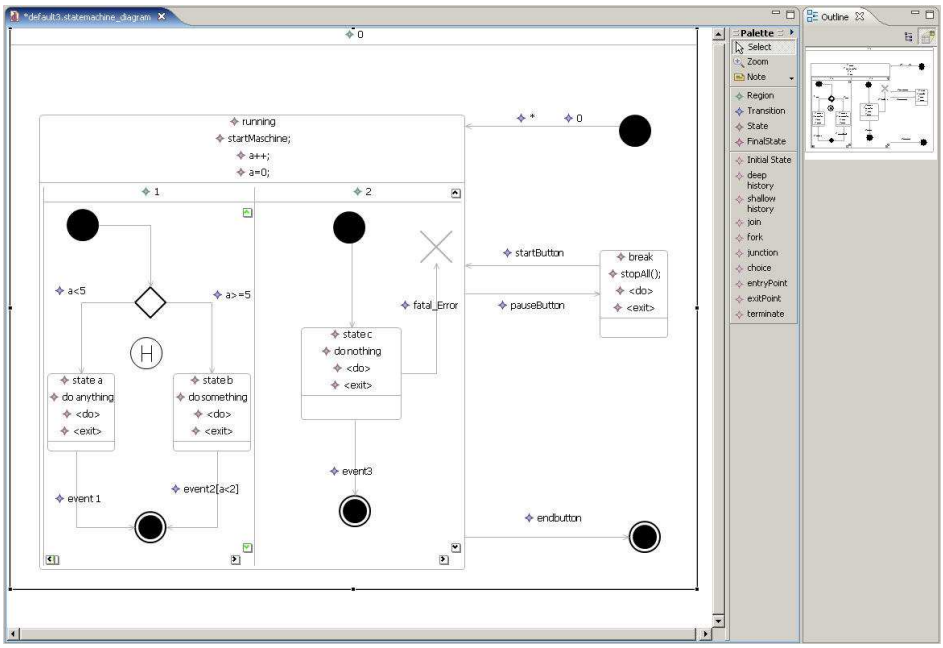

Figura 21 - Exemplo de um editor visual

3.5 Avaliação das Ferramentas

A ferramenta *Microsoft Windows Workflow Foundation* potencia e facilita significativamente o desenvolvimento de interfaces gráficas para qualquer domínio de dados. Para o presente projecto, seria no domínio do desenvolvimento de serviços interactivos. No entanto, desde o início do estudo foi detectado que WWF é totalmente dependente do sistema operativo. Apenas se poderia prosseguir com o desenvolvimento de uma aplicação tendo por base o WWF no sistema operativo *Windows* sobre a plataforma *.Net 3.0*. Apesar desta plataforma ser de acesso e utilização livre tal não acontece com a maior parte das bibliotecas disponibilizadas pelo *Windows*. Este facto, seria um grande impasse perante a necessidade de adicionar novas funcionalidades ao editor gráfico a desenvolver.

Com a criação de um editor gráfico através de *Plug-Ins* para a plataforma *Eclipse*, seriam utilizadas ferramentas e componentes em licenciamento *open source*, ou seja, não teríamos o

problema do licenciamento do WWF. Sendo o recurso a ferramentas *open source* um dos principais objectivos deste trabalho, optamos pela escolha do EMF, GEF, GMF e *Plug-Ins* para a plataforma *Eclipse* na realização da prova de conceito num ambiente empresarial (5). Apesar de não se ter feito o desenvolvimento de raiz de uma aplicação, foi utilizada a plataforma *openVXML* (5.1.4), que consiste na criação de um conjunto de *Plug-Ins* sob licença *open source* para a plataforma *Eclipse*. A plataforma está intimamente ligada a outros projectos no âmbito do *open source*, tais como o EMF, GEF e GMF para a modelação dos fluxos da chamada e edição visual.

A independência total apresentada pelo GMF entre os modelos de dados EMF e o editor visual é uma grande vantagem perante situações de melhoramento ou modificações, quer dos modelos de dados, quer dos editores visuais. É possível efectuar a alteração de ambos os componentes perante a alteração das necessidades, sem ter de existir uma alteração intrínseca entre os dois modelos. Seria apenas necessário redefinir o mapeamento de informação entre o modelo gráfico e o modelo de dados (*Mapping Model*).

No WWF não existe uma visão tão abstracta dos editores gráficos, a criação do editor gráfico incide mais sobre um conjunto de regras aplicado a actividades para um determinado *workflow*. Daí, não haver independência entre os modelos de dados e o editor gráfico como o GMF sob a forma de *Plug-Ins* para a plataforma *Eclipse*. Este facto, perante a alteração de qualquer uma dos componentes, requer um esforço maior para a actualização da informação e por vezes a reestruturação por completo dos componentes.

O ambiente de desenvolvimento de aplicações sobre a plataforma *Eclipse* integra, actualmente, um conjunto enorme de funcionalidades sob a forma de *Plug-Ins* de livre acesso, que estes *Plug-Ins*, podem ser utilizados no desenvolvimento de serviços interactivos. Ao passo que, o número de funcionalidades que podemos adicionar à plataforma *Microsoft Visual Studio* é mais reduzido e faz-nos deparar, novamente, com o problema do licenciamento do software.

4 Análise Funcional

Tendo por base o conhecimento das necessidades para a criação de serviços interactivos e as funcionalidades apresentadas pelas soluções estudadas, foi idealizada e especificada uma interface gráfica que aglomerasse todas as vantagens encontradas em cada uma das soluções e colmatasse as falhas ou funcionalidades inexistentes nas próprias.

Para a construção dos fluxos da chamada de forma visual, foram especificados requisitos de edição visual para a utilização rápida e intuitiva da interface gráfica.

Será usado o termo *Speech Object* para definir uma operação sobre o fluxo de uma chamada num serviço interactivo. Os *Speech Objects* representam acções sobre o fluxo, interacção com o utilizador ou interacção com servidores externos. Será também, usado o termo *Dialog Objects* para representar um conjunto de vários *Speech Objects*. Poderão existir vários *Dialog Objects* interligados entre si com a finalidade de definir um único fluxo de uma chamada. Será utilizado, para que a representação de um serviço interactivo seja feita de forma modular e estruturada, tornando assim possível a reutilização de vários *Sub-Dialog Objects* em diferentes serviços. Todo o desenho do fluxo será feito recorrendo aos vários *Speech Objects* num determinado *Canvas* (folha de *design*).

Deve-se também, seleccionar, configurar e estabelecer a comunicação entre várias aplicações. Por isso, foi especificado quais os requisitos de configuração para o estabelecimento e comunicação com entidades externas. Na Tabela 5 são apresentados os principais atributos de configuração para a interligação com as bases de dados, *Web Services* e servidores externos.

No decorrer da utilização de um serviço interactivo poderão existir vários erros, como, por exemplo, a não interactividade por parte do utilizador, ou quando o utilizador introduz uma opção errada. Deverá existir o tratamento de eventos para as situações adversas ao longo da utilização de um serviço interactivo. Para a resolução dos problemas existentes, deverá existir um sistema de *logs*. O sistema de *logs* serve para que o *tracking* de problemas seja feito de forma mais rápida, tornando possível a detecção de um determinado problema através da análise dos *logs* da execução do serviço.

4.1 Requisitos Funcionais

Tratando-se de uma interface gráfica, foi necessária a definição de vários requisitos para a construção do fluxo de uma chamada de forma visual. Estes requisitos descrevem as funcionalidades que a interface gráfica deve disponibilizar, de uma forma completa e consistente, atendendo aos propósitos a atingir na criação de serviços interactivos.

Como forma de se obter uma melhor percepção das funcionalidades, foram estabelecidos os vários requisitos funcionais em áreas diferenciadas, que foram definidos dentro dos requisitos funcionais para: edição visual (Tabela 3) e no Anexo A são apresentados de forma completa, *Speech Objects* (Tabela 4) e no Anexo B são apresentados de forma completa, comunicação com entidades externas (Tabela 5) e tratamento de eventos e *logs* (Tabela 6).

Tabela 3 - Requisitos para a edição visual

Código	Requisito	Descrição
RFE01	**Inserção de *Speech Objects***	**Ao adicionar um novo *Speech Object* deverá existir a validação da inserção deste, se é possível a sua inserção e quais as ligações possíveis.**
RFE02		**Permitir a configuração das variáveis ou atributos do *Speech Object* tendo em conta pré-validação existente na altura da sua inserção.**
RFE03		**Associação de acções por defeito aos *Speech Objects* (ex. *noinput error, timeout, nomach*).**
RFE04	**Configuração de *Speech Objects***	**Colocar os valores mais usuais nos seus atributos por defeito (ex. *timeout* 5 segs).**
RFE05		**Sempre que possível fornecer uma *drop down list* com a lista de opções possíveis para os atributos.**
RFE06	***Templates* de *Dialog Objects***	**Deverá existir vários *Templates* de *Dialog Objects* com algumas das funcionalidades mais usuais na criação do fluxo de um Serviço Interactivo.**
RFE07	**Gestão dos ficheiros multimédia e gramáticas**	**Existência de vários idiomas (vários ficheiros multimédia).**
RFE08		***Prompts* de vídeo.**
RFE09		***Prompts* de voz.**
RFE10		***Prompts* de texto.**
RFE11		**Gramáticas de DTMF.**
RFE12		**Gramáticas de voz.**
RFE13		**Adição de *prompts* através de *"drag and drop"* para a pasta de ficheiros multimédia.**

Na Tabela 4 são apresentados os principais requisitos para os *Speech Objects* e no Anexo B são apresentados de forma completa.

Tabela 4 - Requisitos para os Speech Objects

Código	Requisito	Descrição	
RFS01	***Prompt***	**A *prompt* pode ser: texto, referencia, número, dígitos, ordinal, quantias monetárias, caracteres, ficheiro de áudio, ficheiro de vídeo.**	
RFS02	**Definição de *User Input***	**DTMF, voz ou ambos.**	
RFS03	***Play Prompt***	***Prompt*.**	
RFS04		***Barg-in* (*True/False*).**	
RFS05	***Desicion***	**Seleccionar o termo de comparação entre variáveis**	
RFS06	***Option Set***	**Opção**	**Adicionar, editar e remover uma opção.**
RFS07			***Prompt* (a tocar na selecção da opção).**
RFS08			**Gramática (se aplicável).**
RFS09		**Definição de *User Input*.**	
RFS10	***Record***	***Maximum recording time*.**	
RFS11		***Type* (áudio/vídeo).**	
RFS12	***Transfer***	***Destination*.**	
RFS13	***Web Service***	**Conteúdo que será passado ao *Web Service*.**	

Código	Requisito	Descrição
RFS14		**Conteúdo que será retornado pelo *Web Service*.**
RFS15	***Data Base Query***	**Base de Dados**
RFS16		**Selecção da base de dados.**
RFS17		**Definição da tabela para efectuar a consulta.**
RFS18		**Selecção e definição dos campos da tabela.**
RFS19		**Selecção da operação (*SELECT, INSERT, UPDATE* e *DELETE*).**

Os serviços interactivos estabelecem frequentemente a interligação com entidades externas. Na Tabela 5 são especificados quais os requisitos de configuração para o estabelecimento e comunicação com entidades externas.

Tabela 5 - Requisitos para a comunicação com entidades externas

Código	Requisito	Descrição		
RFC01	**Base de Dados**	**Adicionar, editar e remover a definição de uma base de dados.**		
RFC02		**Adicionar, editar e remover a definição de tabelas.**		
RFC03		***Database lookup method***	***JNDI Lookup***	***JNDI URI.***
RFC04				***Username.***
RFC05				***Password.***
RFC06			***JDBC Driver***	***JDBC Driver.***
RFC07				***JDBC URL.***
RFC08				***Username.***
RFC09				***Password.***
RFC10	***Web Services***	**Adicionar, editar e remover ligações a *Web Services*.**		
RFC11		**Selecção do ficheiro WSDL com as características do *Web Service*.**		
RFC12	***3rd Party Servers***	**Definição da mensagem de *Output.***		
RFC13		**Definição da mensagem de *Input.***		
RFC14		**Definição do Protocolo de comunicação.**		

Deverá também existir o tratamento de eventos num serviço interactivo e o seu registo num sistema de *logs*, permitindo, assim, a detecção de problemas e a respectiva resolução de forma fácil e rápida. Na Tabela 6 são apresentados os requisitos de tratamento de eventos e de *logs*.

Tabela 6 - Requisitos para o tratamento de eventos e logs

Código	Requisito	Descrição	
RFT01	**Definição e tratamento de eventos**	***Event***	**Definição de tipos de eventos através do elemento *event*.**
RFT02		***Noinput***	**Possibilidade de definir acções a quando não existiu *User Input* esperado pelo utilizador.**

Código	Requisito	Descrição	
RFT03		***Nomatch***	**Possibilidade de definir acções a quando o *User Input* do utilizador não coincidiu com nenhuma opção esperada.**
RFT04		***Catch***	**Através do *Catch*, tornar possível o envio de eventos (*throw event*) no decorrer do fluxo da chamada e posteriormente serem tratados.**
RFT05		***Error***	**Definição dos tipos de erros, para posteriormente serem tratados e guardados nos ficheiros de l*og*.**
RFT06	**Geração de *logs***	**Definição de vários níveis de *log*.**	
RFT07		**Por defeito é criado *logs* referentes a níveis de execução do serviço (normalmente designado como *FINE*).**	
RFT08		**Para cada *Speech Object*, deverá ser guardada informação automaticamente nos ficheiros de *log* (com as características especificas de cada *Speech Object*).**	
RFT09		**Dar a possibilidade ao *designer* definir novas entradas no ficheiro nos ficheiros de *logs* (para além das definidas automaticamente).**	

4.2 Requisitos Não Funcionais

Estes requisitos referem-se aos aspectos não funcionais da interface gráfica, como restrições nas quais o sistema deve operar ou propriedades emergentes. Na Tabela 7 são apresentados os requisitos não funcionais.

Tabela 7 - Requisitos não funcionais

Código	Requisito	Descrição
RNF01	***Interface***	**Interface amigável de utilização rápida, fácil e intuitiva.**
RNF02		**Utilização e reutilização de vários componentes.**
RNF03		**Possibilitar a utilização de *Templates* para definir fluxo da chamada.**
RNF04		**Utilização de *wizards* para a criação dos vários passos do fluxo da chamada.**
RNF05	**Licença**	**Sempre que possível o recurso à utilização de componentes *open source*.**

4.3 Requisitos de Sistema

Na Tabela 8 são apresentados os requisitos de sistema que foram estabelecidos para a interface gráfica.

Tabela 8 - Requisitos de sistema

Código	Requisito	Descrição
RS01	**Independência da plataforma IVR**	**Deverá ser compatível com vários IVRs.**
RS02	**Independência do Interpretador de *VoiceXML***	**O *script* gerado deverá estar formatado em *VoiceXML*, consoante as normas W3C – *VoiceXML 2.1*.**
RS03	**Solução Multiplataforma**	**A Interface Gráfica não deverá estar dependente do sistema operativo em que opera, permitindo que esta seja multiplataforma.**
RS04	**Compatibilidade com múltiplos *Web Servers***	**A exportação para como *Web Service* do serviço interactivo deverá ser compatível com o maior parte dos *Web Servers* existentes actualmente no mercado.**

Nas secções seguintes serão apresentados os principais casos de utilização tendo em conta os requisitos descritos nas secções anteriores. Será referida a forma como é esperada a adição de novos *Speech Objects* ao fluxo da chamada em desenvolvimento e a publicação de um serviço interactivo para um *Web Server*.

4.4 Casos de Utilização

Na subsecção 4.4.1 é apresentado o caso de utilização para adicionar um *Speech Object* ao fluxo da chamada, na subsecção 4.4.2 a definição de opções através do *Option Set* e na subsecção 4.4.3 a publicação de um serviço para um *Web Server*.

4.4.1 Adicionar um *Speech Object* ao fluxo da chamada

A Figura 22 apresenta a inserção de um novo *Speech Object* no fluxo de uma chamada. O *designer* deverá seleccionar o *Speech Object* e, através de *"drag and drop"*, posicioná-lo no local desejado do *Canvas* referente ao desenvolvimento do fluxo da chamada. Ao fazer isto, a própria interface gráfica (motor da aplicação) deverá possuir rotinas que efectuem uma primeira validação do seu posicionamento, devendo existir uma validação prévia do posicionamento do *Speech Object*. Deverá ser feito por forma a não existirem erros na construção do fluxo, optimizando assim os tempos de resolução de problemas.

Após o seu posicionamento, o *designer* deverá definir quais são as ligações entre o *Speech Object* adicionado aos já existentes. Isto é, deverá definir quais os *Speech Objects* anteriores (estados anteriores do fluxo da chamada) e quais os *Speech Objects* posteriores (quais os próximos estados do fluxo da chamada). Ao estabelecer a ligação entre os vários *Speech Objects*, o *designer* irá definir quais as variáveis que serão passadas como *input* vindos dos estados anteriores e o *Output* seguido para os estados posteriores.

Após estas ligações entre os vários *Speech Objects*, o *designer* irá definir a configuração dos atributos específicos aos *Speech Objects*. Deverá definir as características específicas de cada *Speech Object* de forma a executar o pretendido com a sua adição. O que é possível utilizando dois métodos: com o duplo clique no *Speech Object*, e será apresentada uma janela com as características do *Speech Object*, ou num editor de texto com o respectivo *VoiceXML* associado ao *Speech Object*. Deste modo, o *designer* pode definir as características do *Speech Object* das duas formas em simultâneo, e as alterações serão reflectidas em ambos os métodos de configuração do *Speech Object*.

Isto é necessário para que os *wizards* de configuração dos *Speech Objects* não contenham todos os atributos referentes aos *Speech Objects*, mas sim, os mais utilizados na sua configuração. Caso o *designer* necessite de configurações mais específicas (normalmente os *"experts users"*), tem o editor de *VoiceXML* para o fazer. Utilizando esta abordagem, torna-se mais rápida a configuração dos *Speech Objects* e resolve um problema encontrado na maior parte das interfaces gráficas estudadas, a não abrangência dos *wizards* de todos os atributos necessários para a configuração de um *Speech Object*.

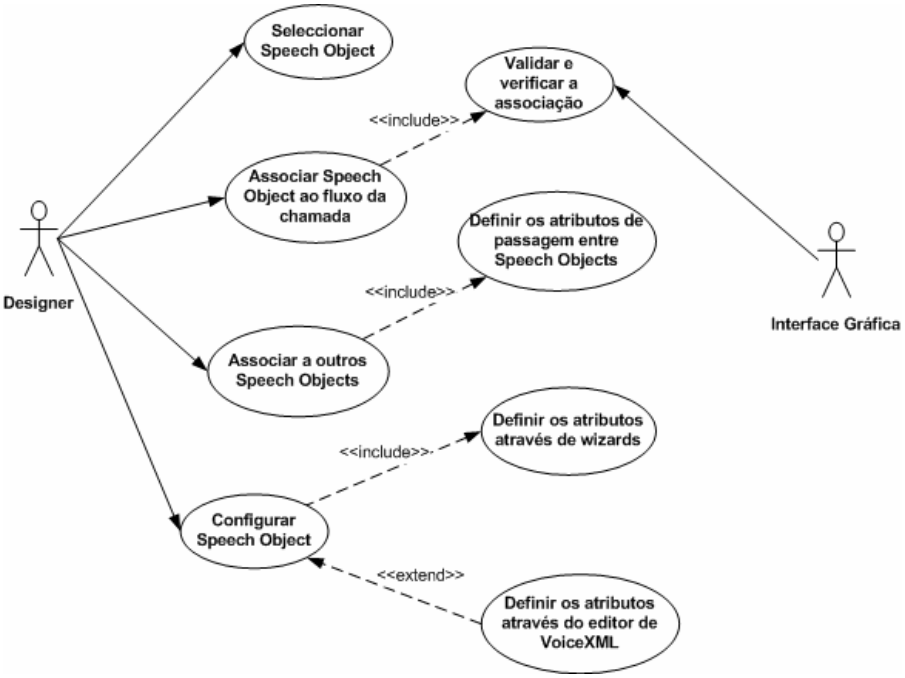

Figura 22 - Adicionar Speech Object ao fluxo da chamada

4.4.2 Definição das opções através do Option Set

Na Figura 23 é apresentada a forma como o *designer* define um conjunto de opções existentes num determinado ponto do fluxo da chamada. Para além da apresentação das opções, a aplicação deverá definir as operações resultantes da selecção de uma dessas opções e deverá definir os passos ou *Speech Objects* seguintes para cada uma das opções e o método de interacção com o utilizador, o *User Input*.

Na selecção de interacção por DTMF, o *designer* deverá definir para além da *prompt*, qual o dígito associado à opção e as definições apresentadas na Tabela 4 e no Anexo B relativa a *User Input* por DTMF. No caso de uma interacção por voz, o *designer* deverá definir qual a gramática a utilizar para o conjunto de opções de voz e são igualmente válidas as definições apresentadas na Tabela 4 relativas a *User Input* por voz.

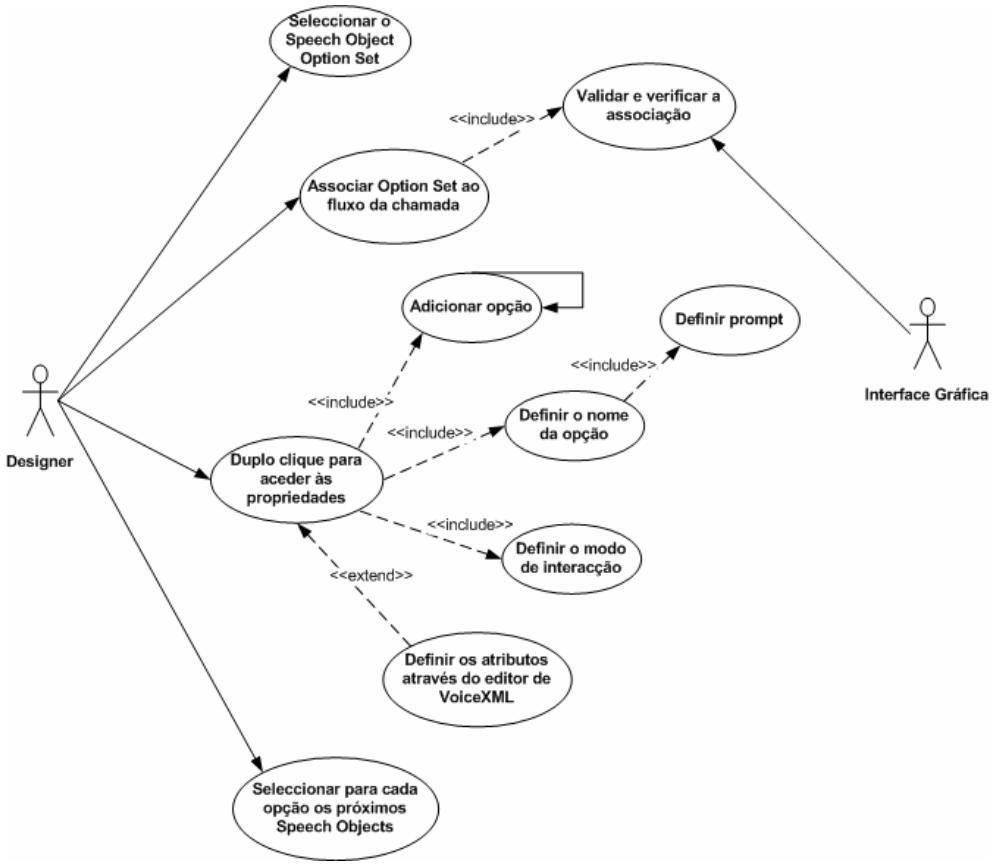

Figura 23 - Conjunto de opções através do Option Set

4.4.3 Publicação de um serviço interactivo

Na Figura 24 são apresentados, de uma forma muito redutora, os passos para a publicação de um serviço interactivo num *Web Server*. Após a definição ou desenho completo do fluxo da chamada com o recurso aos *Speech Objects*, o *designer*, com um simples clique, tem a possibilidade de exportar o serviço para um *Web Server*.

Para tal, deverá seleccionar qual o serviço a publicar, isto é, deverá seleccionar uma das aplicações desenvolvidas num determinado projecto. Ao efectuar esta operação deverá definir qual o nome a atribuir ao serviço interactivo e qual o *Web Server* onde será alojado. A atribuição do nome para o serviço interactivo é importante, porque será o nome usado para fazer o registo do serviço no interpretador de *VoiceXML*.

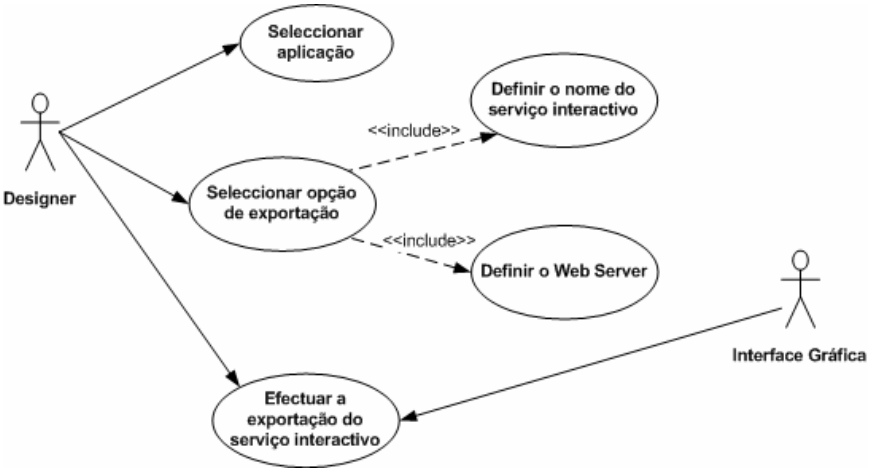

Figura 24 - Publicação de um serviço interactivo

5 Um Ambiente Gráfico para a Criação de Serviços Interactivos

No presente capítulo será apresentado um **Ambiente Gráfico para a Criação de Serviços Interactivos**. Para a demonstração foram utilizados dois componentes proprietários da PT Inovação. O componente para o atendimento das chamadas (Sistema de Resposta Interactiva) e o componente para a interpretação de *VoiceXML* (*VoiceXML Gateway*). Para a criação visual de serviços interactivos foi utilizada a solução *openVXML*. O *openVXML* consiste num conjunto de *Plug-Ins* sob licença *open source* para a plataforma *Eclipse*. O *openVXML* permite criar os mais variados fluxos para uma chamada com o recurso a várias *Palettes* de *Speech Objects* e *Dialog Objects* pré-definidos. Para além da edição visual, o *openVXML* tem rotinas para fazer a exportação dos serviços como aplicações *Web*, onde estão contidos os *sccripts* de *VoiceXML* correspondentes ao fluxo desenvolvido. Para a publicação e teste dos serviços desenvolvidos foi utilizado o *Tomcat Web Server*. Os serviços interactivos estão encapsulados numa aplicação *Web* que tem como objectivo disponibilizar conteúdos estáticos ou dinâmicos através dos *scripts* de *VoiceXML*.

Serão descritos os componentes utilizados para criar o ambiente de desenvolvimento e explicado o papel de cada um na arquitectura. Na secção 5.1 será descrita a arquitectura lógica dos componentes: *openVXML*, *Tomcat*, *InoVXML* e *InoVox-IP*. Na secção 5.2 é descrita a arquitectura física. Na secção 5.3 é apresentada a perspectiva funcional de todos os componentes do ambiente de desenvolvimento, é dado o exemplo do desenvolvimento de um serviço interactivo. Por último, serão apresentados os testes unitários feitos às funcionalidades, configurações dos *Speech Objects*, execução e validação dos *scripts VoiceXML* e testes de usabilidade da interface gráfica utilizada pela *openVXML*.

5.1 Arquitectura Lógica

A arquitectura lógica do ambiente de desenvolvimento (Figura 25), está dividida em quatro principais componentes: *InoVox-IP*, *InoVXML*, *Tomcat 5.5* e o *openVXML*.

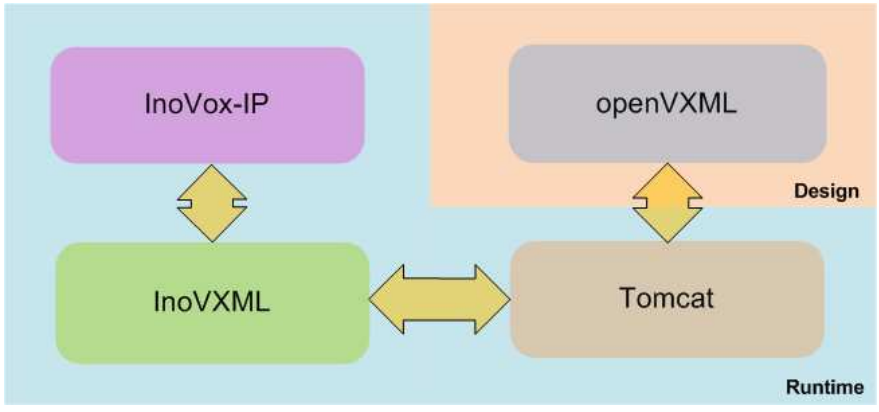

Figura 25 - Arquitectura lógica do Ambiente de Criação de Serviços

O *InoVox-IP* é a solução comercial de Sistema de Resposta Interactiva da PT Inovação. O *InoVox-IP* é o *core* do ambiente de desenvolvimento, é o responsável pela gestão de toda a telefonia. É uma plataforma multi-serviços para serviços multimédia sobre redes PSTN ou IP.

O *InoVXML* é uma solução ainda em fase de desenvolvimento, que funciona como interpretador de *VoiceXML* ou *GateWay* de voz. Tem como função a interpretação de vários *scripts VoiceXML* armazenados num *Web Server* e, consoante o fluxo definido pelos serviços interactivos, realiza operações distintas no IVR.

Foi utilizado o *Tomcat 5.5 Web Server* para a publicação dos serviços interactivos. O *Tomcat* é um servidor de aplicações *Java* para *Web* que é distribuído como software livre e foi desenvolvido em código aberto dentro do projecto *Apache Jakarta*.

O *openVXML*, é o componente *open source* para a criação visual de serviços interactivos. Consiste num conjunto de *Plug-Ins* sob licença *open source* para a plataforma *Eclipse.* Esta solução está intimamente ligado aos EMF, GEF, GMF referidos na secção 3.4.2. Através do *openVXML* o *designer* pode definir o fluxo de uma chamada para um serviço interactivo através de *"drag and drop"* de *Speech Objects*, podendo reutilizar vários componentes já existentes para a construção de fluxos para as chamadas.

5.1.1 InoVox-IP

A Figura 26 apresenta de uma forma simplificada a arquitectura lógica da plataforma *InoVox-IP*, que é uma plataforma multi-serviços, de arquitectura aberta e flexível, escalável e de elevado desempenho para a criação de serviços multimédia avançados sobre redes PSTN, convergentes ou totalmente IP.

Permite o desenvolvimento de novos serviços interactivos de uma forma rápida, com acesso a diferentes sistemas de informação e incorporando princípios de gestão centralizada. A plataforma foi desenvolvida para operar através da rede pública de acesso telefónico, mas, com o objectivo de possibilitar também o acesso a novos e mais sofisticados tipos de serviços, podendo aceder-se a esta plataforma através da rede IP. O core do *InoVox-IP* para a gestão de telefonia é a plataforma *open source Asterisk*. O *Asterisk* é uma plataforma PBX (*Private Branch eXchange*) e IVR inteiramente *open source*, que suporta quer as tradicionais tecnologias de comutação de circuitos, através de interfaces de *hardware*, quer as novas tecnologias de encapsulamento de voz em pacotes (*VoIP*).

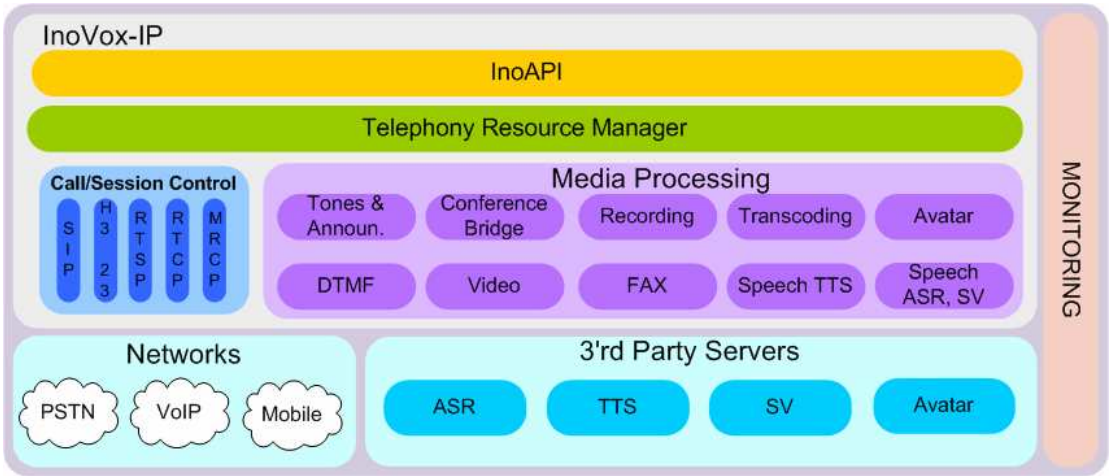

Figura 26 - Arquitectura lógica do InoVox-IP

O *InoVox-IP* disponibiliza APIs (*Application Programming Interface*) de alto nível para ambientes *Windows* e *Unix*, para o processo de desenvolvimento de qualquer serviço de resposta automático, do mais simples ao mais complexo, tornando assim a criação e o desenvolvimento de novos serviços mais rápido e expedito. Nos próximos pontos serão descritos os principais componentes do *InoVox-IP*:

InoAPI

A *InoAPI* disponibiliza uma interface de alto nível para acesso e controlo dos vários recursos de telefonia. Esta interface abstrai a complexidade das operações decorrentes da utilização das APIs de baixo nível, disponibilizadas pelos fornecedores dos vários recursos suportados pelo *InoVox-IP*, e utiliza um modelo cliente/servidor para o desenvolvimento eficiente de aplicações. Permite ainda que as aplicações cliente reservem, configurem e utilizem diferentes recursos de telefonia. Fornece uma camada de abstracção dos detalhes do software/hardware dos diferentes fornecedores. Para mais detalhe ver o Anexo C.

TRM (*Telephony Resource Manager*)

O TRM é o core de todo o sistema e permite a execução de comandos provenientes de múltiplos serviços ou aplicações, o processamento de eventos de várias APIs de diversos recursos tecnológicos e a preparação e execução de respostas a serviços através da API de interface. A gestão dinâmica de recursos permite implementações de serviços em larga escala com significativa redução de custos. Finalmente, o TRM é responsável pela configuração e gestão de todo o sistema.

Call/Session Control

O módulo de *Call/Session Control* tem a seu cargo o tratamento das chamadas e o controlo de sessões suportando diversos protocolos de sinalização e controlo.

Media Processing

Este módulo disponibiliza um vasto conjunto de funcionalidades, básicas e avançadas, de processamento de *streams* multimédia tais como: reprodução e gravação de *prompts*, detecção DTMF, fax e conferência. As funções avançadas de fala são disponibilizadas recorrendo a soluções de terceiros utilizando o protocolo *Media Resource Control Protocol* (MRCP). Soluções de outros fornecedores podem ser facilmente integradas na plataforma. A seguir são apresentados alguns funções que existem no módulo *Media Processing*.

- *Tones & Announcement* – funcionalidades IVR e mensagens de voz;
- *Conference Bridge* – áudio conferência e vídeo-conferência;
- *Recording* – gravação de áudio e de vídeo;
- *Transcoding* – conversão entre vários formatos de áudio e de vídeo;
- DTMF – a detecção/geração DTMF é um elemento fundamental para a interacção do cliente com os diferentes serviços;
- *Video* – vídeo *streaming* permite reproduzir, para a rede IP, vídeos residentes no disco local que estejam num formato proprietário (IVX) ou então reproduzir *streaming* de vídeos obtidos de um *streaming server* remoto;

- FAX – suporta a geração e terminação de sessões fax, permitindo a transmissão em tempo real de fax através da rede IP;
- *Avatar* – vídeo *streaming* de um avatar que permite reproduzir, para a rede IP, vídeos *Avatars* gerados dinamicamente (*prompts* dinâmicas) ou estáticos (vídeo pré-gravados).

Monitoring

Este módulo é responsável pela constante monitorização do desempenho de todos os parâmetros operacionais do *InoVox-IP* e assegura uma interface para operação, administração e gestão.

Networks

Este módulo oferece a capacidade de ligação aos vários tipos de redes actualmente utilizadas para as comunicações, tais como PSTN, IP e UMTS.

3'rd Party Servers

Possibilita a interligação a servidores externos para o fornecimento de novos serviços. Actualmente, os serviços utilizados externamente são ASR, TTS, SV e *Avatar*.

5.1.2 InoVXML

O *InoVXML* (Figura 27) apresenta-se como um módulo independente do *InoVox-IP* e consiste numa aplicação que combina um interpretador de *VoiceXML* com vários componentes adjacentes, estabelecendo uma interface com a *InoAPI*. Está a ser desenvolvido a partir do *software open source OpenVXI*, é um interpretador de *VoiceXML*. O *InoVXML* tem mecanismos internos de controlo de sessões/chamadas, interfaces várias para o *OpenVXI* e para a *InoAPI*, por forma a registar serviços distintos no TRM do *InoVox-IP* e possibilitar a interpretação de vários *scripts VoiceXML* armazenados num *Web Server* e, consoante o fluxo definido pelos serviços interactivos, realizar operações distintas no IVR. Estas operações são efectuadas recorrendo ao controlo de recursos de telefonia, reconhecimento de fala e de dígitos, reprodução de *prompts* e TTS, através de troca de mensagens de pedidos e respostas com a *InoAPI*.

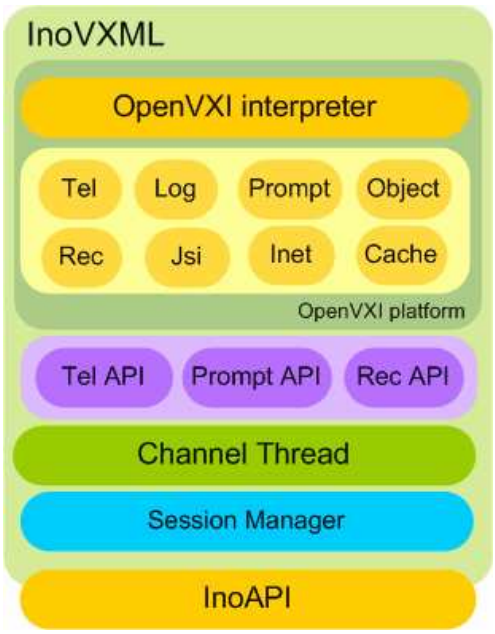

Figura 27 - Arquitectura lógica do InoVXML

O *OpenVXI* é composto pelo interpretador *VoiceXML*, contexto do *VoiceXML* e outros componentes de suporte como, os módulos de *Internet (Inet)*, de *parsing* de documentos de XML *(Xerces)*, de execução de *JavaScript (Jsi)*, *caching* de documentos *(Cache)*, sistema de *logs*, gestão de *prompts*, reconhecimento de interacção *(Rec)* e controlo de vários tipos de objectos *(Object)*.

Assenta em três APIs distintas para o reconhecimento de *Input* do utilizador (ASR, DTMF *Recognition* e *Recording*), para a produção de *Output* (reprodução de *prompts* e TTS) e para os serviços interactivos (serviços de telefonia), sendo elas as interfaces *Rec*, *Prompt* e *Tel* respectivamente.

Do ponto de vista da plataforma de voz, o fluxo de uma chamada para um serviço *VoiceXML* irá passar por uma fase de atribuição de recursos (Telefonia, TTS, ASR), seguindo-se o processamento do *script VoiceXML* que define o serviço, resultando em diferentes chamadas a funções (síncronas e assíncronas) com o objectivo de realizar várias operações no canal. Estas operações que poderão ser a reprodução de uma *prompt*, o *loading* de gramáticas de reconhecimento, o reconhecimento de dígitos e reconhecimento de voz ou uma transferência de uma chamada. A ordem pela qual estas instruções serão executadas depende de vários factores, contudo, será conduzida pela interpretação dos vários campos existentes no *script VoiceXML* que foi carregado.

Do ponto de vista do *OpenVXI*, quando é iniciada uma chamada para um serviço *VoiceXML*, serão invocadas pela plataforma de voz várias funções para a iniciação de recursos da plataforma do *OpenVXI*, seguindo-se uma chamada ao método *Run()* para a interpretação do *script VoiceXML* associado ao serviço.

O *Channel Thread* é o objecto que representa uma sessão. É criado aquando do tratamento de uma nova chamada, tem como objectivo criar os recursos do interpretador e gerir o fluxo de dados entre o *Session Manager* e o interpretador *VoiceXML*.

O *Session Manager* é a entidade que faz a ponte entre o *Channel Thread* (por conseguinte, o *OpenVXI*) e a *InoAPI*. É responsável por criar e gerir os diversos canais (chamadas ou sessões) e afectar os recursos a cada um deles. O *Session Manager* é responsável pela gestão das mensagens e pela sua atribuição aos diversos canais. A estrutura de dados sobre a qual opera, corresponde a uma fila de mensagens. As mensagens representam os eventos ou *replies* (mensagens de resposta) gerados pelo *InoVox-IP* ou pelo *Session Manager*. Existe uma *Reply Queue* (fila de respostas) para cada canal de chamada e tem como objectivo disponibilizar ao interpretador, que está a ser executa no canal, as mensagens de erro ou notificações dos pedidos efectuados. A decisão da existência de uma fila de mensagens para cada canal prende-se com o facto de apenas uma entidade (o *Session Manager*) ter acesso à *stack* de mensagens do *InoVox-IP*, relativas aos serviços e recursos de todas as sessões activas.

No decorrer da criação do ambiente de desenvolvimento foram feitas algumas alterações ao *InoVXML*, já que este não estava conforme com a norma *VoiceXML 2.0*, no que diz respeito ao caminho relativo de prompts remotas. Foram então feitas as alterações necessárias, tornado possível a interpretação de prompts multimédia existentes remotamente, isto é, guardadas no *Web Server* onde estão publicados os serviços interactivos.

O *InoVXML* tinha também um problema na associação e estabelecimento de sessões com o *Web Server*. Procedeu-se à resolução do establecimento de sessões, para que se fizesse a correcta associação entre as várias intruções em *VoiceXML* e a respectiva sessão activa.

5.1.3 Web Server

Foi utilizado o *Tomcat 5.5 Web Server* para a publicação dos serviços interactivos, utilizando a *Java Virtual Machine version 1.5*. Os serviços interactivos foram alojados no *Tomcat* e posteriormente registados no interpretador de *VoiceXML*, no *InoVXML*.

A Figura 28 descreve como é feita a publicação de *scripts* numa aplicação *Web* no *Tomcat*. Os serviços interactivos estão encapsulados numa aplicação *Web* que tem o objectivo disponibilizar conteúdos estáticos ou dinâmicos através dos *scripts* de *VoiceXML* ao IVR. Para tal, os serviços interactivos são exportados pelo *openVXML* com a estrutura hierárquica das aplicações *Web*.

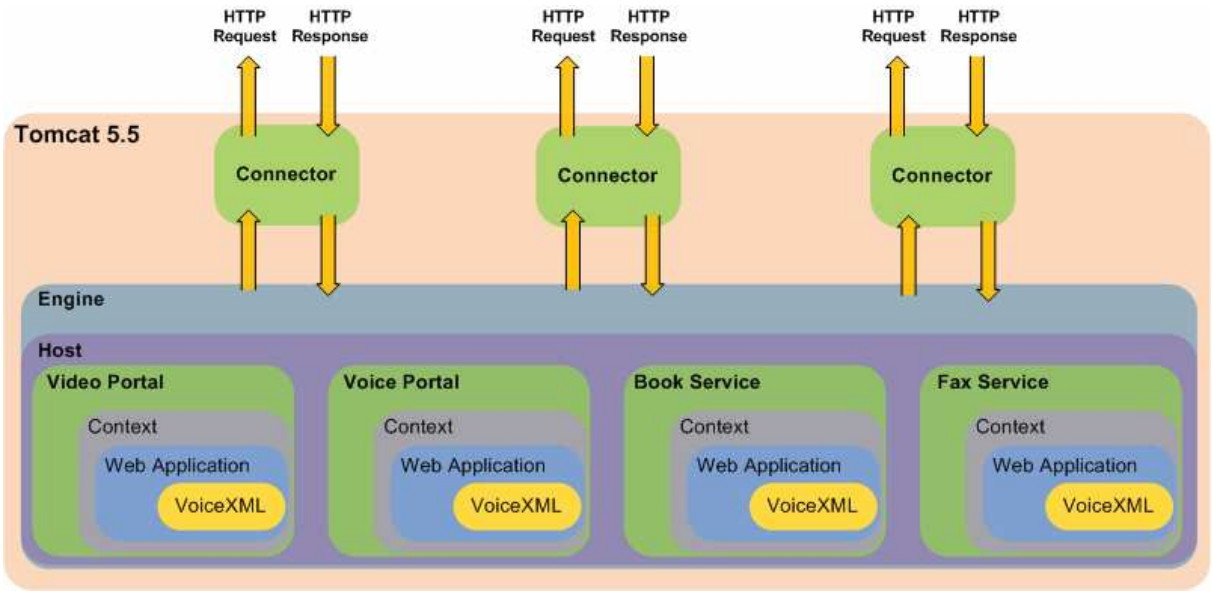

Figura 28 - Arquitectura lógica do Tomcat Web Server

O elemento *Context* representa uma *Web Application* (ou serviço interactivo), a qual está a correr num específico *Virtual Host*. Cada *Web Application* é baseada num ficheiro de *WAR* (*Web Application Archive*) ou no caminho para um directório onde está a hierarquia de ficheiros da *Web Application*, segundo a *Servlet Specification* **[Apache 06]**. O elemento *Host* representa na realidade um *Virtual Host*, o qual tem associado um nome na rede (ex. *www.interactiveservices.com*). Caso não tenha associado nenhum nome na rede, é utilizado o IP da máquina onde está a correr o *Tomcat*. O elemento *Engine* representa todo o processo associado a um pedido HTTP a um determinando serviço interactivo por parte do *InoVXL*. O *Engine*, é o responsável por receber e processar um ou mais pedidos chegados pelos *Connectors* e é também responsável por fazer com que a resposta lhes chegue correctamente e posteriormente pelo envio dos mesmos ao *InoVXML*.

É através de *Java Servlets* que todo o contexto dos serviços interactivos é criado (pedidos em *VoiceXML*). A *Servlet* tem como objectivo dotar o *Tomcat* os recursos necessários para que seja possível aceder através do modelo *request/response* aos *scripts* de *VoiceXML*, utilizando para tal o protocolo HTTP **[Armstrong 05]**. Um *Servlet* necessita de um *Container Web* para ser executado. Os serviços interactivos gerados incluem um directório onde estão presentes todos os *Plug-Ins* necessários para a correcta geração dos *scripts VoiceXML*. Dentro deste directório, são incluídos os vários elementos multimédia, os elementos fiquem alojados juntamente com o serviço. Existe um outro directório onde estão alojados os *Plug-Ins* necessários para a execução em tempo real do serviço.

5.1.4 openVXML

A plataforma *openVXML* surge através da evolução do projecto *Eclipse Voice Tools*, que consiste no desenvolvimento de um conjunto de *Plug-Ins* sob licença *open source* para a plataforma *Eclipse* e tem como principais impulsionadores as empresas IBM, SBC *Communications* e *VoiceGenie* **[VTP 07]**.

A plataforma *Eclipse* é, actualmente, uma das formas mais viáveis para o desenvolvimento de projectos, especialmente para projectos utilizando a linguagem *Java*. No entanto, as últimas evoluções têm dotado a plataforma de funcionalidades para o suporte do desenvolvimento de aplicações *Web* e J2EE. O *Eclipse Voice Tools Project* tem por objectivo disponibilizar bibliotecas à plataforma *Eclipse* sob a forma de *Plug-Ins* para o desenvolvimento de serviços interactivos. Ferramentas como o *openVXML*, têm como objectivo o desenvolvimento de serviços interactivos sobre os IVRs, recorrendo para tal, ao uso da norma *VoiceXML 2.0* e *2.1*, para que seja possível a integração do mundo de dados com o de voz, através da publicação dos *scripts* em *Web Servers*.

O *openVXML* esta intimamente ligado a outros projectos no âmbito do *open source*, tais como o EMF, GEF, GMF referidos na secção 3.4.2. Está também ligado ao projecto WTP (*Eclipse Web Tools Platform)*, para que efectue o uso das ferramentas por ele disponibilizadas para o desenvolvimento *Web* (XML *Standard* e publicação de serviços). O WTP tem por finalidade prover um ambiente de desenvolvimento padronizado para *Web* sobre a plataforma *Eclipse*, é constituído por um conjunto de APIs para JEE, aplicações *Web* e ferramentas para suporte de publicação e testes de aplicações *Web*.

Em 2003 foi criada a empresa *openMethods*, que através da evolução do projecto *Eclipse Voice Tools* fornece a solução *openVXML* (actualmente na versão 3.0). São disponibilizadas duas versões desta solução: uma sob licença *open source* e uma outra (que inclui funcionalidades avançadas) através de uma licença comercial **[OpenMethods 08]**. A versão *open source* veio revolucionar o meio da criação de serviços interactivos baseadas em soluções de livre acesso ao código, tal como afirmou um dos fundadores da *openMethods* na apresentação do *openVXML "This is the first step in openMethods contribution to open source and to speech technology"* **[Tim Barnes 07]**. A solução *openVXML* é a consolidação da contribuição, ao longo de vários anos, de vários elementos que trabalham nas maiores empresas de telecomunicações e paralelamente no *Eclipse Voice Tools Project*, permitindo assim às empresas terem um ponto de partida bastante avançado para a criação das suas próprias soluções gráficas para a criação de serviços interactivos.

Para a prova de conceito, foi utilizada a versão *open source* do *openVXML* no ambiente de desenvolvimento da PT Inovação. O *openVXML* permite criar os mais variados fluxos para uma chamada com o recurso a várias *Palettes* de *Speech Objects* e *Dialog Objects* pré-definidos. Disponibiliza *Speech Objects* para tocar *prompts*, análise de opções introduzidas pelo utilizador, gravação de voz, transferência da chamada, acesso a servidores remotos através de *Web Services*, consulta a base de dados e *wizards* de configuração bastante extensíveis para os *Speech Objects*. Esta versão permite a exportação do fluxo da chamada através de *scripts* de *VoiceXML* para um *Web Server* (a versão *open source* apenas para o *Tomcat 5.5*).

A versão comercial EE (*Enterprise Edition*), disponibiliza um conjunto de funcionalidades avançadas para além das disponíveis na versão *open source*. A versão EE tem funcionalidades de *Logging*, *Deployment* (compatibilidade com a maior parte dos *Web Servers* existentes no mercado), *Reporting* e *Install and Update* e a um componente que permite a alteração em tempo real dos fluxos das chamadas dos serviços existentes num *Web Server*.

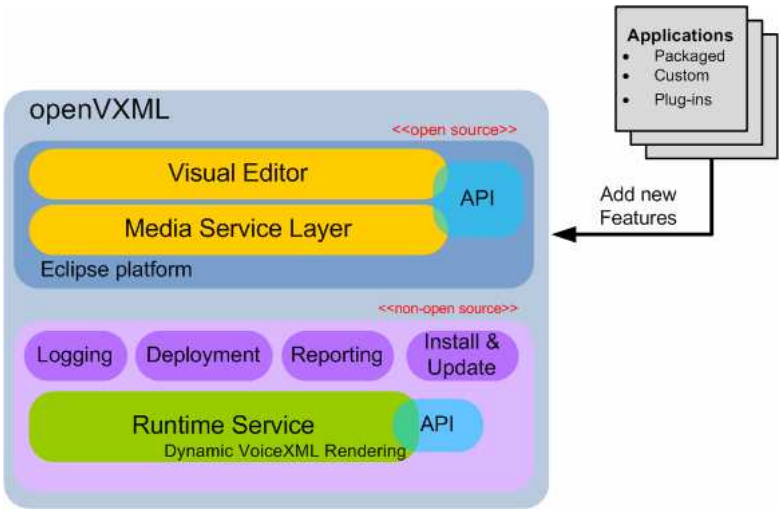

Figura 29 - Arquitectura lógica do openVXML

Na Figura 29 é apresentada a arquitectura lógica da plataforma *openVXML* de forma simplificada. A versão *open source* está dividida em dois principais componentes: *Media Service Layer* (a edição e gestão de *prompts*) e *Visual Editor* (a camada de edição visual).

A *Media Service Layer* tem como objectivo a edição e gestão de *prompts*, sendo, para tal, efectuada a criação de várias *"Personas"*. Uma *Persona* consiste na associação de várias *prompts* e gramáticas, com o intuito de definir uma linguagem específica, que podem estar associadas a diferentes serviços interactivos. Na camada *Visual Editor* é efectuada a criação e representação do fluxo da chamada, sendo através desta interface visual, que o *designer* efectua todo o desenvolvimento dos serviços interactivos. As funcionalidades avançadas de *Logging*, *Deployment*, *Reporting* e *Install and Update* não foram objecto de qualquer tipo de estudo ou desenvolvimento, devido ao facto de estarem sob licença comercial.

Media Service Layer

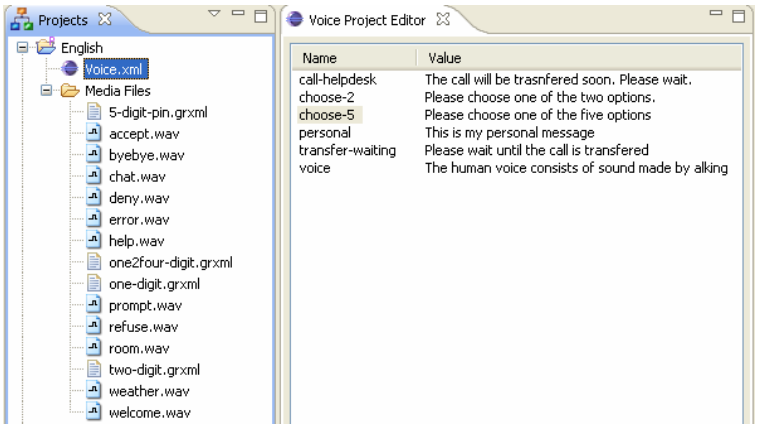

Figura 30 - Janela de edição de ficheiros multimédia

A *Media Resource Layer* é o componente responsável pela gestão e organização dos ficheiros de áudio, vídeo e gramáticas de voz e de DTMF durante todo o ciclo de vida de um serviço interactivo. Consegue efectuar essas funções de uma forma dinâmica, fazendo com que as alterações dos ficheiros e gramáticas sejam reflectidos automaticamente nos serviços interactivos.

Visual Editor

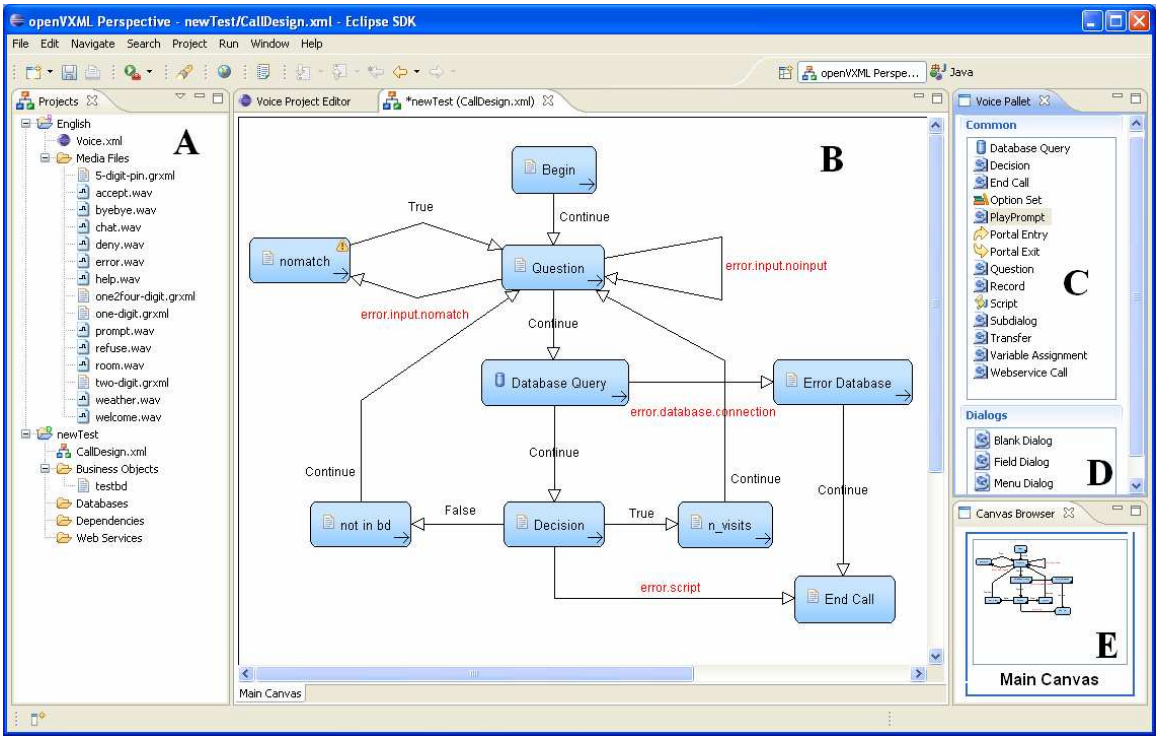

Figura 31 - Edição visual do openVXML

O componente *Visual Editing* é responsável pela criação e geração dos fluxos das chamadas dos serviços interactivos. Reúne em si a informação contida em todos os componentes descritos nesta secção, através de *"drag and drop"* dos *Speech Objects*, configurando as suas propriedades, interligando vários *Speech Objects* e definindo regras de passagem de informação entre os vários estados da chamada. Os pontos seguintes descrevem os componentes da edição visual do *openVXML*.

- Gestão e organização dos ficheiros multimédia, gramáticas e dos vários serviços interactivos (A);
- Desenho estilo *flowchart* através de *"drag and drop"* de *Speech Objects* que descreve o fluxo da chamada (B);
- *Palette* de vários *Speech Objects*, permitindo a sua adição ao fluxo da chamada e a respectiva configuração (C);
- Várias *Dialog Objects* para ajudar o *designer* a criar os serviços interactivos, evitando que erros de desenvolvimento possam ocorrer com muita frequência (D);
- Vista reduzida dos múltiplos *Design Canvas* existentes (E);

À imagem da maior parte das soluções no âmbito do *open source*, foram encontrados alguns problemas na utilização das principais funcionalidades do *openVXML*. Apesar de alguma maturidade, esta ferramenta está em constante alteração e melhoramento, sendo por isso difícil chegar a uma solução completamente estável e funcional. No decorrer da criação do ambiente de desenvolvimento foram resolvidos alguns dos problemas encontrados na fase de experimentação e testes do *openVXML*.

Na primeira fase foi resolvido um *bug* existente na plataforma *openVXML*: esta, após a publicação dos serviços interactivos não conseguia efectuar a associação entre as várias "Personas" e os respectivos elementos multimédia com o fluxo da chamada estabelecido. O problema foi resolvido através do *Script Block*, recorrendo a uma instrução que realizasse essa associação manualmente. No entanto, é necessário, para cada novo serviço, fazer a introdução deste bloco no início do fluxo da chamada. Foi a melhor forma encontrada até ao momento, devido ao facto do não conhecimento aprofundado da plataforma.

A segunda fase foi dotar a *Media Resource Layer* da capacidade para o suporte dos ficheiros de vídeo proprietários à PT Inovação, o formato IVX. Foi definido que este tipo de ficheiros seria tratado exactamente da mesma forma que os ficheiros 3GPP, isto é, a *Media Resource Layer* trata exactamente da mesma forma os vários ficheiros de vídeo independentemente do formato que tenham.

A terceira fase consistiu em efectuar uma análise exaustiva às mais variadas funcionalidades prometidas pelo *openVXML*, incidindo mais sobre a *Pallete* dos *Speech Objects*. Foi detectado que dois *Speech Objects* apresentavam uma enorme debilidade em termos de configurações e funcionamento: a configuração de *Web Services* e a ligação à base de dados. O problema com a ligação com base de dados foi resolvido, a interpretação dos ficheiros de configuração e ligação com *Web Services* foi iniciada, mas não tendo sido concluída devido à limitação do tempo disponível para a criação do ambiente de desenvolvimento.

Para a resolução da ligação à base de dados foi necessário perceber o código referente à ligação com servidores externos, para posteriomente ser possível fazer as alterações necessárias ao módulo, de forma a torná-lo capaz de comunicar com bases de dados e guardar a informação da consulta. Foi dotado o *openVXML* com a capacidade de comunicar com bases de dados *PostgreSQL*, tendo sido obtidos resultados bastantes positivos nos testes realizados (ver a subsecção 5.4.1).

Na resolução do problema da interpretação dos ficheiros WSDL, referentes à configuração de *Web Services*, iniciou-se o desenvolvimento e foram obtidos progressos nesta funcionalidade. No entanto, as alterações não foram terminadas a tempo, não tendo sido possível dotar a plataforma *opneVXML* com a capacidade de comunicar com *Web Services* no decorrer do fluxo de uma chamada de um serviço interactivo.

5.2 Arquitectura Física

A arquitectura física do ambiente de desenvolvimento (Figura 32), mostra o sistema na perspectiva dos seus componentes de *hardware*, explicitando as suas dependências de comunicação entre os vários componentes.

O *openVXML* está a correr numa máquina *Windows* e o *designer* pode definir e criar todo o fluxo da chamada referente aos serviços interactivos. Após a sua criação, é feita a exportação para uma outra máquina *Windows* remota, onde está a correr o *Web Server Tomcat*. Como a plataforma *Eclipse* não tem a possibilidade de exportação para locais remotos, isto foi contornado efectuando o mapeamento de repositórios remotos através do MNDW (*Map Network Drive do Windows*). Deste modo, é efectuada a publicação localmente ou remotamente (através do protocolo *Windows File Sharing*) do serviço. No entanto, não é o suficiente para que este fique disponível para o utilizador. Após a sua publicação é necessário registar o URL de acesso ao serviço na interface do *InoXML*. O *InoVXML* é o componente encarregue de interpretar e descodificar os *scripts* de *VoiceXML* existentes para cada um dos serviços interactivos alojados no *Tomcat*. O *InoVox-IP* corre numa máquina *Linux* e tem o *InoVXML* registado como um serviço (*request/reply* através do protocolo TCP/IP). O *InoVox-IP* quando recebe uma chamada referente ao serviço registado pelo *InoVXML*, passa a atribuir ao *InoVXML* a execução das instruções relativas à chamada.

Para a interacção com base de dados, foi utilizado o *PostgreSQL*, SGBD desenvolvido como projecto software livre. O *Tomcat* comunica através do protocolo JDBC (*Java Database Connectivity*) com a máquina onde estão alojadas as bases de dados a serem consultadas ou usadas para guardar informações pelos serviços interactivos.

A comunicação entre o *InoVox-IP* e o servidor de TTS é feita através do protocolo MRCP. O estabelecimento da comunicação é apenas necessário aquando da existência de *prompts* de texto no serviço interactivo. No caso de serem *prompts* de áudio ou vídeo, estas estão alojadas localmente na máquina onde se encontra o *Tomcat*. A comunicação com o servidor de ASR é feita através do protocolo MRCP. O estabelecimento da comunicação é apenas necessário quando o serviço interactivo necessita de fazer o reconhecimento das instruções de voz do utilizador.

Este ambiente está em funcionamento na rede interna da PT Inovação, sendo o nosso objectivo testar o funcionamento sobre as redes IP, PSTN e UMTS. Neste ambiente é utilizando o protocolo SIP para o estabelecimento das chamadas com o *InoVox-IP*, os utilizadores utilizam como interface qualquer *softphone* (software que funciona sobre a tecnologia *VoIP*, dando possibilidade ao utilizador de fazer e receber chamadas), com capacidade de processamento de vídeo (utilizando os *codecs* de vídeo H263/+ e *codecs* de áudio G.711 *a-law* e G.711 *μ-law*). Nos testes efectuados foram utilizados os *softphones Kapanga* e *X-Lite* para estabelecer as chamadas com o *InoVox-IP*.

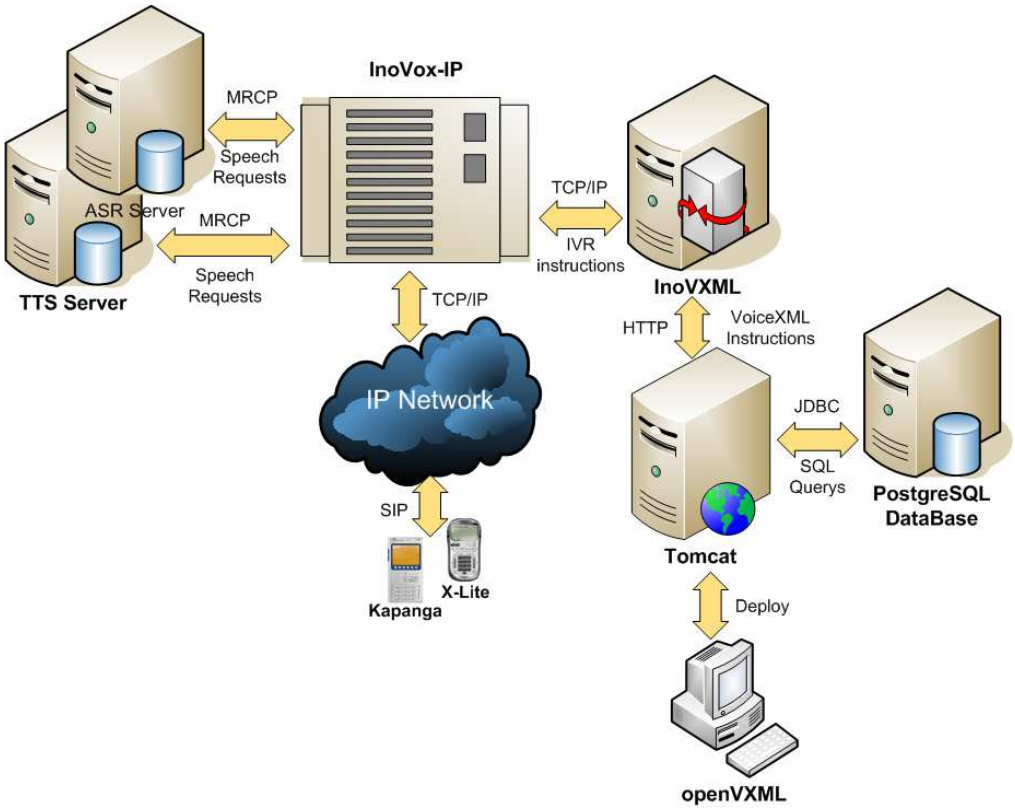

Figura 32 - Arquitectura física

5.3 Perspectiva Funcional

Para demonstrar a perspectiva funcional foi desenvolvido um serviço interactivo exemplo designado "*Find Your Movie*" (no Anexo D estão todas as *prompts* utilizadas). Este serviço, através do número inteiro que identifica univocamente um DVD, apresenta ao utilizador o nome do filme. Para tal, é pedido ao utilizador que insira através das teclas do telefone ou telemóvel o identificador do filme e é feita a pesquisa à base de dados para saber qual o nome do filme correspondente. Se o utilizador desejar pode falar directamente com o assistente e o serviço transfere a chamada para clube de vídeo.

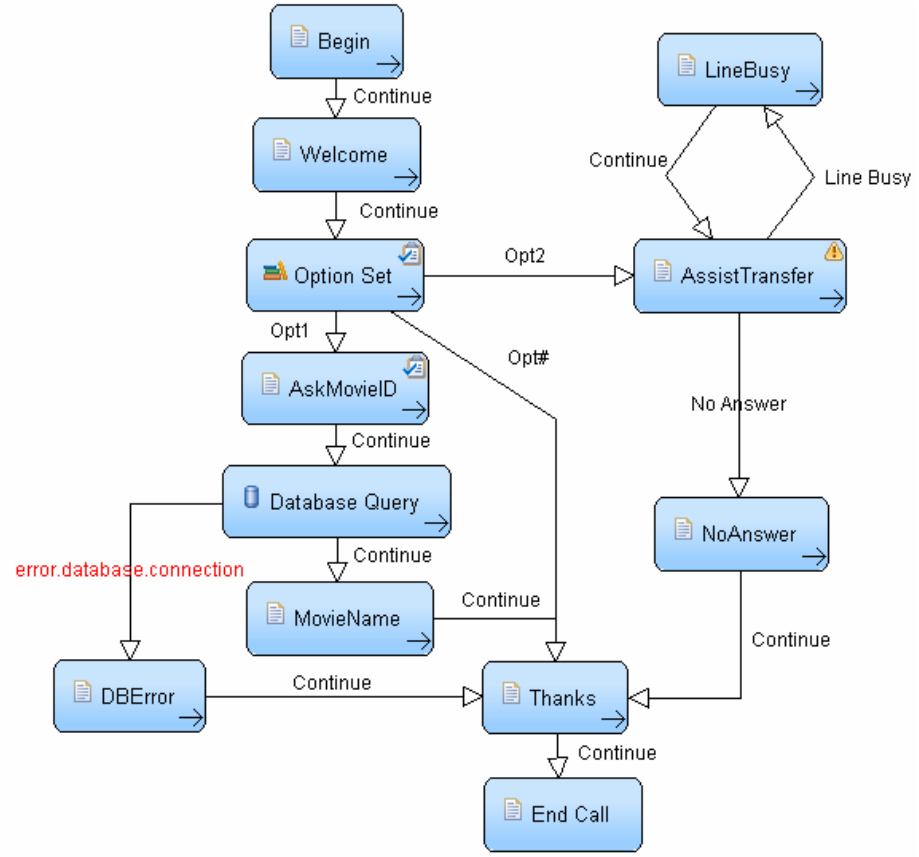

Figura 33 - Fluxo do serviço "Find Your Movie"

O *openVXML* utiliza os *Plug-Ins* WTP para fazer a publicação do fluxo da chamada, desenvolvido como uma *Web Application*. O WTP tem por finalidade prover um ambiente de desenvolvimento normalizado para a *Web* sobre a plataforma *Eclipse*. É constituido por um conjunto de APIs para JEE, aplicações *Web* e ferramentas para suporte de instalação e testes de aplicações *Web*. Para que o serviço estabeleça correctamente a ligação com a base de dados, foram especificados na fase de exportação os parâmetros de ligação à base de dados *PostegreSQL*.

Para cada serviço interactivo é guardado o desenho do fluxo da chamada. Isto é feito através da identificação de todos os *Speech Objects*, qual o seu posicionamento de cada um no fluxo e quais as ligações que tem. Para cada *Speech Object* o *openVXML* gera um *script* de XML onde estão todas as configurações. Cada *Speech Object* é visto como um *primitive-elemen* e dentro das *tags primitive-element* temos a definição dos atributos para cada *Speech Object*.

Prompt Block

```
<primitive-element
            id="3f2e9ddb89fd4bd8adedd4d6ff7b001b"
            name="Welcome"
            type="org.eclipse.vtp.desktop.editors.core.playPrompt">
          ...
               <text-content
                   xmlns="http://www.eclipse.org/vtp/media/content"
                   dataType="static">
                 Ben vindo ao servico de pesquisa de filmes.
               </text-content>
          ...
```

```
          </bindings>
          <custom-config
              xmlns="http://www.eclipse.org/vtp/namespaces/config" />
</primitive-element>
```

Existem três atributos comuns a todos os *Speech Objects*: *id*, *name* e *type*. O *id* identifica univocamente o objecto em todo o fluxo da chamada, utilizando-se o atributo *name* para a ilustração do *Speech Object* na edição visual. O atributo *type* identifica o tipo de *Speech Object* (*playPrompt*, *advancedTransfer*, *Record*, *databaseQuery*, etc) associado ao *<primitive-element>*.

Data Base Query

```
  ...
<primitive-element
            id="41608e9df7fd4a00a992c904602ff4fe"
            name="Database Query"
            type="org.eclipse.vtp.desktop.editors.core.databaseQuery">
          <custom-config
              xmlns="http://www.eclipse.org/vtp/namespaces/config">
            <settings
                db-name="VideoBlog"
                db-result-limit="-1"
                db-table="video_rectable"
                var-exists="false"
                var-multi="0"
                var-name="consult_video_rectable"
                var-type="video_rectable">
              <mappings>
                <mapping name="id" type="1" value="id" />
                <mapping name="name" type="1" value="name" />
                <mapping name="n_visits" type="1" value="n_visits" />
                <mapping name="creation" type="1" value="creation" />
                <mapping name="filename" type="1" value="filename" />
              </mappings>
              <criteria>
                <criterium comp="0" name="id" type="-1" value="" />
                <criterium comp="-1" name="name" type="-1" value="" />
                <criterium comp="-1" name="n_visits" type="-1" value="" />
                <criterium comp="-1" name="creation" type="-1" value="" />
                <criterium comp="-1" name="filename" type="-1" value="" />
              </criteria>
            </settings>
          </custom-config>
        </primitive-element>
  ...
```

Na *tag <settings>* são definidas as propriedades do *Data Base Query Block*, isto é qual a base de dados (*VideoBlog*) e qual a tabela a usar na consulta (*video_rectable*). Para além destes atributos, é definida a variável (*consult_video_rectable*) e o tipo (*video_rectable*) onde será guardada a informação que resulta da consulta.

Na *tag <mappings>* é definido o tipo de mapeamento que será feito entre os campos da base de dados e os campos do *Business Object* (objecto que faz o mapeamento entre a informação contida na base de dados e o resultado da consulta). Na *tag <criteria>* é definido qual o tipo de consulta, isto é, quais os parâmetros da base de dados que serão utilizados para fazer a consulta. O atributo *comp* (0 ou -1) define se o atributo *name* está a ser utilizado na consulta, e o atributo *value* define qual o valor desse atributo.

Advanced Transfer Block

```
  ...
        <primitive-element
```

```
            id="e78ec9aa2d0e45149add9af40dac3b30"
            name="AssistTransfer"
            type="org.eclipse.vtp.desktop.editors.core.advancedtransfer">
          ...
            <property-binding
                name="destination">
              <item
                  key="Default">
                sip:luis@10.112.136.14
              </item>
            </property-binding>
          </bindings>
          ...
        </primitive-element>
  ...
```

No *Advanced Transfer Block* é guardado na *tag item* o número de destino para a transferência da chamada.

O desenho do fluxo da chamada no *Canvas* é também guardado, através da identificação de todos os *Speech Objects*, qual o posicionamento que tem cada um no fluxo e quais as ligações.

CallDesign.xml

```
<canvas
          id="1"
          name="Main Canvas"
          orientation="2"
          paper-size="org.eclipse.vtp.desktop.editors.core.Letter">
        <ui-element
            id="cbb0fc79382a4f039e6f8eb5fef8b1cd"
            x="283"
            y="73" />
        <ui-element
            id="3f2e9ddb89fd4bd8adedd4d6ff7b001b"
            x="283"
            y="144" />
         ...
```

Após a exportação do serviço interactivo para o *Tomcat Web Server* é necessário registar o URL do serviço no interpretador *InoVXML*. O *InoVXML* está registado no *InoVox-IP* como um serviço e, aquando da recepção de uma chamada pelo *InoVox-IP* referente ao serviço interactivo, o *InoVox-IP* passa a gestão da chamada para o *InoVXML*.

InoVXML.conf

```
...
defaultScript = default.xml
numberOfServices = 2
...
url1 = http://10.112.136.204:8080/openbd/openbd
mask1=openbd
...
url2 = http://10.112.136.204:8080/ptin/ptin
mask2=ptin
...
```

Os serviços interactivos estão encapsulados numa *Web Application* no *Tomcat Web Server*. Esta aplicação tem como objectivo disponibilizar conteúdos estáticos ou dinâmicos através dos *scripts* de *VoiceXML* ao *InoVox-IP*. Conforme foi referido anteriormente, os *scripts* de *VoiceXML* são tratado como páginas *Web*, ou seja, através de um *Web Browser* conseguimos aceder e visualizar o *VoiceXML* gerado pelo *openVXML*. A navegação entre as várias etapas

do fluxo de uma chamada é feita através de diferentes porções de *scripts* de *VoiceXML*. Na Figura 34 é apresentado o *script* gerado para tomar o *prompt Welcome*.

```
<vxml version="2.0">
 <form id="OutputMessageForm" scope="document">
  <block name="OutputMessageBlock">
    <prompt bargein="true" xml:lang="pt-PT">
          Ben vindo ao serviço de pesquisa de filmes.
    </prompt>
    <goto next="/openbd/-/next"/>
  </block>
</form>
<catch event="connection.disconnect.hangup">
  <goto next="/openbd/-/abort"/>
</catch>
</vxml>
```

Figura 34 - Script VoiceXML do Welcome

A navegação entre as etapas é feita através da *tag <goto next= URL />*. Após a interpretação por parte do *InoVXML* do script *VoiceXML* da *prompt Welcome* as novas instruções de *VoiceXML* encontram-se *<goto next="/openbd/-/next"/>*. Com a *tag goto* é redireccionada a ordem de execuções de *scripts*. Isto é feito, para que a execução dos *scripts* seja feita de forma modular e estruturada, pois, cada acção sobre o fluxo da chamada tem um *script* de *VoiceXML* correspondente. Na Figura 35 é apresentado o *script* de *VoiceXML* para o menu principal.

```
<vxml version="2.0">
<menu id="selection" scope="document">
<property name="bargein" value="true"/>
<property name="timeout" value="5s"/>
<property name="inputmodes" value="dtmf"/>
<property name="com.telera.speechenabled" value="false"/>
<property name="termtimeout" value="3s"/>
<property name="termchar" value=""/>
<prompt bargein="true" xml:lang="pt-PT">
      Prima um para consultar o filme na base de dados.
      Prima dois para falar com o assistente. Prima cardinal para sair.
</prompt>
<choice dtmf="1" next="/openbd/
       -/next?7a50d4f78c9842888a0df6487830f59f=success.filled&selection=Opt1"/>
<choice dtmf="2" next="/openbd/
       -/next?7a50d4f78c9842888a0df6487830f59f=success.filled&selection=Opt2"/>
<choice dtmf="#" next="/openbd/
       -/next?7a50d4f78c9842888a0df6487830f59f=success.filled&selection=Opt%23"/>
<noinput>
  <goto next="/openbd/
       -/next?7a50d4f78c9842888a0df6487830f59f=error.input.noinput"/>
</noinput>
<nomatch>
  <goto next="/openbd/
       -/next?7a50d4f78c9842888a0df6487830f59f=error.input.nomatch"/>
</nomatch>
</menu>
<catch event="connection.disconnect.hangup">
  <goto next="/openbd/-/abort"/>
</catch>
</vxml>
```

Figura 35 - Script VoiceXML do Option Set

A Figura 36 descreve, de uma forma muito genérica, (sem descriminar os métodos invocados), as sequências na troca de mensagens na utilização do serviço.

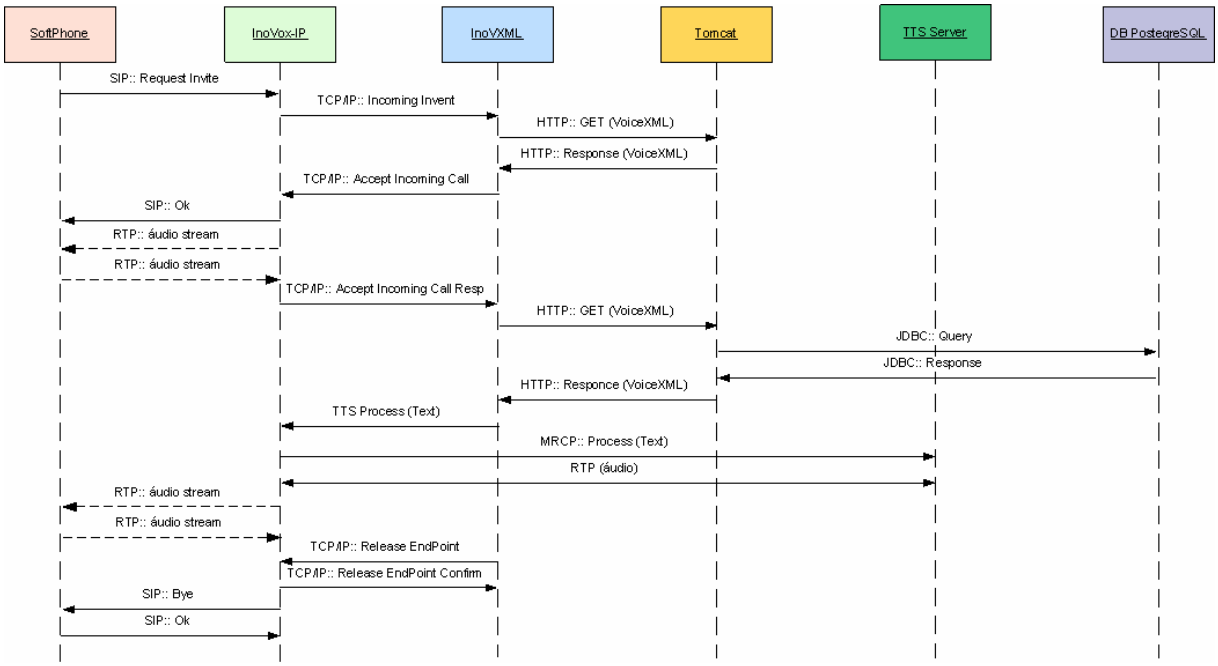

Figura 36 - Diagrama de sequência

Na Figura 36 são apresentados os protocolos de comunicação e a sequência de mensagens para o *playPrompt* de uma *prompt* de texto, resultado de uma consulta à base de dados. O utilizador usa um *softphone* e faz uma chamada (protocolo SIP) para um serviço interactivo registado no *InoVox-IP*. O *InoVox-IP* detecta que a chamada é para um serviço registado pelo *InoVXML* e então passa o controlo da chamada ao *InoVXML*. A comunicação entre o *InoVox-IP* e o *InoVXML* é feita com recurso às instruções da *InoAPI*, utilizando o protocolo TCP/IP. Entre o *InoVXML* e o *Tomcat* existe uma troca de pedidos (*GET/Response*) de *scripts* de *VoiceXML* com as instruções sobre a chamada. Aquando da existência no serviço de uma consulta à base de dados, é feita a consulta entre o *Tomcat* e a base de dados *PostregrSQL* através do protocolo JDBC. Quando o *Tomcat* tem a *string* resultante da consulta é construído o *script VoiceXML* com a *prompt* de texto e é enviada ao *InoVXML*. O *InoVXML* interpreta o *script* e envia a instrução ao *InoVox-IP* para o processamento do texto através das funções da *InoAPI*. O *InoVox-IP* faz o pedido do processamento do texto (TTS) ao servidor de TTS (utilizando o protocolo MRCP). Após o processamento do texto pelo servidor de TTS e do *InoVox-IP* ter recebido toda a *stream* de áudio, o *InoVox-IP* envia o áudio ao utilizador através do protocolo RTP. No final é desligada a chamada e o *InoVox-IP* notifica o *InoVXML* de que já não existem mais pedidos de *VoiceXML*.

5.4 Testes

Nesta secção vão ser descritos os testes que foram realizados à solução *openVXML*. Foram efectuados testes às funcionalidades e configurações dos *Speech Objects*, execução e validação dos *scripts VoiceXML* e testes de usabilidade da interface gráfica disponibilizada pela *openVXML*.

Idealmente seria necessário efectuar testes unitários a todos os requisitos definidos no capítulo 4, tal não foi feito de alguns requisitos satisfazem à partida as necessidades por eles descritos. Foram então realizados testes unitários às principais funcionalidades e testes de usabilidade à plataforma *openVXML*.

Os testes unitários foram especificados e executados pelo autor desta dissertação, os testes de usabilidade foram especificados pelo autor e executados por cinco pessoas seleccionadas aleatoriamente. Essas pessoas efectuaram a instalação e configuração da plataforma *openVXML* e desenvolveram um serviço interactivo proposto pelo autor. No final deste processo, foram recolhidas algumas impressões e sugestões em termos de usabilidade da solução *openVXML*.

5.4.1 Testes unitários

Na Tabela 9 são apresentados os vários testes unitários feitos aos *Speech Objects*, da comunicação com *Web Services*, da comunicação com a base de dados *PostgreSQL* e navegação entre os vários *Speech Objects*.

Tabela 9 - Testes realizados aos Speech Objects

Código	Teste	Descrição
TS01	***Play Prompt***	**Efectua a correcta definição e execução de *prompts* de texto, referência, número, dígitos, ordinal, quantias monetárias, caracteres, ficheiro de áudio, ficheiro de vídeo.**
TS02	***Variable assignment***	**Efectuar a definição de variáveis e reutilização e alteração dos seus valores ao longo do fluxo da chamada.**
TS03	***Desicion***	**Estabelecimento da variável de decisão e o termo comparativo.**
TS04	***Question***	**Definição da *prompt* referente à questão, definição do *User Input* e guardar a opção introduzida pelo utilizador.**
TS05	***Option Set***	**Definição de várias opções e análise da árvore de decisão perante o *User Input*.**
TS06	***Record***	**Efectuar a gravação de *User Input* (áudio) e apresentar ao utilizador a gravação.**
TS07	***Transfer***	**Foi testado a transferência da chamada entre dois clientes *Softphones*.**
TS08	***Advanced Trasfer***	**Foi testado o novo *Speech Object* relativo à transferência da chamada entre dois clientes Softphones.**
TS09	***SubDialog***	**Foi testado a interacção de o fluxo da chamada entre um fluxo principal e um fluxo secundário.**
TS10	***Script Block***	**Através do *Script Block* foram efectuados o acesso e inserção a base de dados e alterações das variáveis globais.**
TS11	***Web Service Call***	**Através do *Web Servies Call* foi estabelecida a ligação com *Web Servers* para a troca de informação.**
TS12	***Data Base Query***	**Foi efectuado o teste de consulta a uma base de dados de desenvolvimento em *PostgreSQL*.**
TS13	**Ligação entre *Speech Objects***	**As ligações entre os vários *Speech Objects* foram testadas, como forma de validar os atributos passados entre os vários objectos.**

Na Tabela 10 são apresentados os vários testes unitários feitos às diferentes configurações existentes no *openVXML*.

Tabela 10 - Testes realizados às configurações

Código	Teste	Descrição
TC01	***Media Files***	**Gestão de vários ficheiros multimédia (*.vox, .wav, .3gp, .ivx, .al, .ul*).**
TC02	***Data Bases***	**Definição da ligação a uma base de dados *PostgreSQL*.**
TC03	***Business Objects***	**Definição de objectos de transacção de informação entre o fluxo da chamada e uma base de dados *PostgreSQL*.**
TC04	***Dependencies***	**Através *Script Block* foram efectuadas chamadas a bibliotecas externas ao *Java* e adicionadas à pasta de dependências.**
TC05	***Web Services***	**Foi efectuada a definição de um *Web Service*.**

No cômputo geral, podemos afirmar que os testes unitários realizados tiveram bastante sucesso em termos de execução e resultados esperados *versus* obtidos. De uma forma geral, todos os *Speech Objects* operaram de forma correcta e continham os atributos e parâmetros necessários para a sua utilização no desenvolvimento da maior parte dos serviços interactivos. Apenas existiram falhas em dois *Speech Objects*: o *Record* e o *Web Service*.

O teste ao *Record* não se efectuou com sucesso, não pela existência de problemas na geração do *script* de *VoiceXML* por parte do *openVXML*, mas devido a problemas do interpretador *InoVXML* em fazer *parsing* das *tags* do elemento *record*. O *script* de *VoiceXML* gerado pelo *openVXML* está correctamente definido, mas o interpretador não está consoante a norma definida pela *VoiceXML 2.0*. Este problema está actualmente em fase de resolução.

Relativamente aos testes unitários efectuados às configurações, o grande problema encontrado foi na configuração de *Web Services*. O *openVXML* não consegue fazer a correcta interpretação dos ficheiros WSDL utilizados nos testes, o que fez com que existissem dois problemas, um no *parsing* do ficheiro WSDL e outro na definição dos parâmetros no *Web Service Block*. Não é possível estabelecer uma correcta definição dos parâmetros de *Input* e *Output* na comunicação com um *Web Service*.

5.4.2 Testes de usabilidade

A usabilidade é uma questão muito importante quando se faz o desenvolvimento de software recorrendo a interfaces gráficas. A usabilidade mede a relação das características do sistema e o grau de satisfação do utilizador final **[Nielsen 04 A]**. Como não era praticável no tempo útil para a elaboração da dissertação focar de forma intensiva ou aprofundar os testes de usabilidade da plataforma *openVXML*, optou-se por efectuar testes de usabilidade de uma forma muito informal, baseada na metodologia *Hallway testing*. Para tal foi elaborado um pequeno guião de utilização do *openVXML*, e, juntamente, foi especificado um fluxo de uma chamada de um serviço interactivo. No Anexo E é apresentado o fluxo da chamada proposto. Esta informação foi fornecida a cinco pessoas seleccionadas aleatoriamente, que procederem ao seu desenvolvimento. Após terminado o guião, foram recolhidas as avaliações numa escala de um a dez sobre os cinco atributos definidos como as métricas de usabilidade **[Nielsen 93 C]**. As opiniões recolhidas incidiram nos principais objectivos dos testes de usabilidade: fácil de aprender, usar, fácil de lembrar, fácil de recuperar de situações de erro e agradável de usar.

Apesar de terem sido seleccionadas cinco pessoas aleatoriamente, refira-se que o utilizador final do *openVXML* é um *designer*, alguém com conhecimentos no âmbito do desenvolvimento de serviços interactivos, o que, à partida, deverá ter uma maior facilidade em perceber a forma como operam os *Speech Objects*.

Tabela 11 - Métricas de usabilidade

Métrica	A	B	C	D	E	Média
Fácil de aprender	**5**	**4**	**6**	**7**	**4**	**4.3**
Eficiente para usar	**7**	**5**	**8**	**7**	**6**	**6.6**
Fácil de lembrar	**8**	**8**	**9**	**8**	**8**	**8.2**
Fácil de recuperar a situações de erro	**7**	**8**	**7**	**8**	**6**	**7.2**
Agradável de usar	**8**	**8**	**8**	**7**	**9**	**8**

O maior problema que se encontrou na utilização do *openVXML,* reside na característica “Fácil de Aprender”. Quando os utilizadores não têm os conhecimentos mínimos sobre o funcionamento dos serviços interactivos, a percepção do funcionamento global da aplicação

torna-se mais difícil. Um problema crítico encontrado pela maioria dos utilizadores foi a configuração da ligação com base de dados. Apesar de existir um *wizard* de configuração, torna-se difícil fazer a configuração destes elementos por parte de utilizadores que não têm conhecimento da forma como funcionam as bases de dados. Estes elementos de configuração, deveriam ser revistos de forma que a sua utilização fique mais acessível e intuitiva. No entanto, as restantes configurações *Media File* e *Dependecies* são de fácil compreensão.

Como avaliação final, apesar do *openVXML* ser uma aplicação ainda em fase de desenvolvimento, podemos afirmar que os resultados de uma forma geral foram bastante positivos. O *openVXML* é de utilização fácil e intuitiva e abrange as principais funcionalidades necessárias para a criação de serviços interactivos.

6 Comparação dos Processos de Desenvolvimento

A utilização das interfaces gráficas tem um impacto significativo nos processos de desenvolvimento usados pelas empresas de telecomunicações. No presente capítulo, vão ser estudadas e avaliadas as implicações que terão as interfaces gráficas nos processos de desenvolvimento de serviços interactivos.

A criação de serviços interactivos utilizando ambientes gráficos num ambiente de desenvolvimento integrado de aplicações tem como principal objectivo reunir todas as fases de desenvolvimento de um serviço num só IDE. Os componentes gráficos dão a capacidade de definir o fluxo da chamada através de *"drag and drop"* de objectos e vários componentes. O IDE dispõe das ferramentas necessárias para a execução de todas as fases de desenvolvimento dos serviços, esta evolução vem alterar significativamente os processos de desenvolvimento de serviços. Em vez de termos várias fases distintas e elaboradas com ferramentas apropriadas para cada fase de desenvolvimento, temos um IDE que dispões das ferramentas necessárias para fazer a especificação, o desenvolvimento, os testes e a publicação dos serviços interactivos.

Na secção 6.1 é descrito o processo de desenvolvimento mais utilizado pelas empresas de telecomunicações, designado por processo de desenvolvimento tradicional. Na secção 6.2 é apresentado a nova metodologia que é obtida com a utilização de interfaces gráficas, designado por processo de desenvolvimento utilizando ambientes gráficos

6.1 Processo de desenvolvimento tradicional

O modelo de desenvolvimento de software em Espiral, é o mais utilizado no desenvolvimento de serviços interactivos. Apesar de, actualmente, as empresas de telecomunicações utilizarem várias metodologias de desenvolvimento, o modelo em Espiral é aquele que tem obtido resultados mais positivos. O modelo em Espiral apresenta algumas fases genéricas para o desenvolvimento de software, tais como: Determinar Objectivos, Identificar e Resolver Riscos, Desenvolvimento e Testes e Planeamento para a próxima interacção. Este modelo é baseado numa sequência de fases que culminam em várias entregas ou protótipos incrementais. Em geral, modelos incrementais têm o objectivo de lidar melhor com um conjunto de requisitos incerto ou sujeito a alterações. Todavia, para cada tipo de projectos, estas fases são ajustadas perante as necessidades do desenvolvimento. Na Figura 37 é apresentado o modelo de desenvolvimento em Espiral ajustado ao desenvolvimento de serviços interactivos.

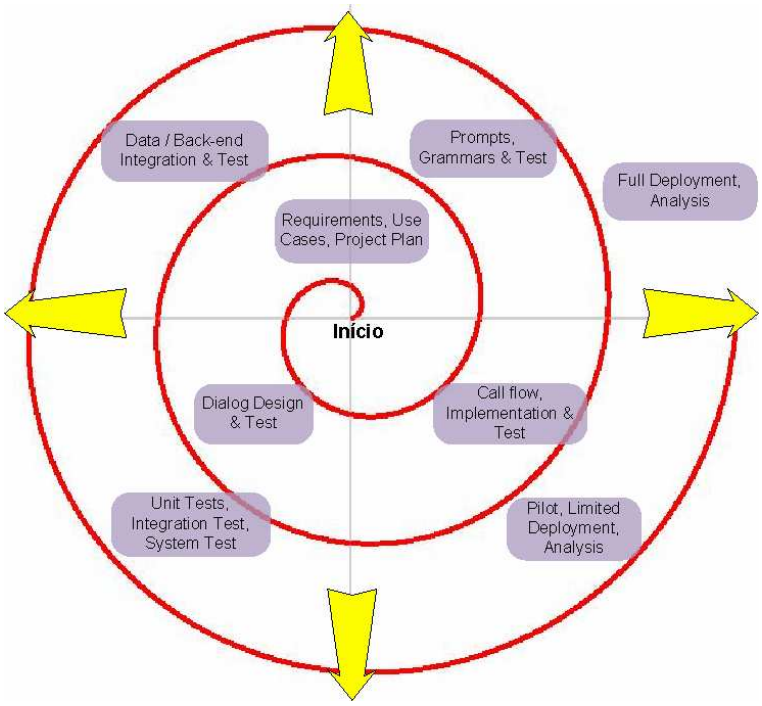

Figura 37 - Modelo de desenvolvimento de software Espiral

Nas próximas secção apresentadas as várias fases do modelo de desenvolvimento Espiral para o desenvolvimento de serviços interactivos.

Requirements, Use Cases, Project Plan

A primeira fase é executada junto dos clientes. É feita a recolha dos requisitos e casos de utilização que os clientes desejam para o serviço interactivo. Para além da especificação do serviço, é também elaborado um primeiro planeamento (rascunho ou previsão) para o projecto, a ser acordado com o cliente e demonstrado à equipa de desenvolvimento.

Dialog Design & Test

Nesta fase é feito o desenho do fluxo para os vários diálogos existentes ao longo do fluxo da chamada e feitos testes unitárias a esses diálogos que poderão ser feitos junto do cliente.

Call flow, Implementation, & Test

Com base nos diálogos definidos na fase anterior, esta fase é responsável por fazer a construção e implementação do fluxo da chamada e a execução dos testes unitários ao fluxo estabelecido.

Prompts, Grammars, & Test

Nesta fase é feita a geração e gravação das várias *prompts* e gramáticas utilizadas no decorrer do fluxo da chamada pelo serviço interactivo.

Data / Back-end Integration, & Test

Nesta fase é feita a integração com os restantes componentes que comunicam com os serviços interactivos, ou seja, no caso de existirem bases de dados, *Web Services*, interpretadores de *VoiceXML* ou tecnologias de voz associadas aos IVRs.

Unit Tests, Integration Test, System Test

Nesta fase são feitos os teste unitários às várias funcionalidades especificadas e testes de integração, com o objectivo de encontrar falhas provenientes da integração interna dos vários componentes, isto é, são feitos os testes ao funcionamento do sistema na sua globalidade. São também feitos testes ao sistema, com o intuído de executar o serviço sob ponto de vista do utilizador final, varrendo todas as funcionalidades em busca de falhas.

Pilot, Limited Deployment, Analysis

Como resultado da execução das fases anteriores, nesta fase é feita uma primeira entrega do serviço interactivo ao cliente, designado por *Pilot*. Normalmente, nas primeiras versões dos serviços não estão implementadas todas as funcionalidades especificadas, isto é, trata-se de um serviço interactivo *Pilot* ou *Prototype*. Este serviço poderá sofrer várias entregas até chegar ao produto final (fase seguinte) **[Pilot 08]**.

Full Deployment, Analysis

Esta é a última fase de desenvolvimento dos serviços interactivos utilizando o modelo em Espiral. Através da fase anterior, foram feitas várias entregas e análises ao funcionamento dos serviços enquanto que em simultâneo, eram efectuadas correcções e adicionadas novas funcionalidades até obtermos o serviço interactivo final. Esta é a fase em que se dá por terminado o desenvolvimento e ocorre a entrega do serviço desenvolvido ao cliente, apenas ficando por concluir, a partir de agora, a manutenção do serviço no caso da existência de problemas.

6.2 Processo de desenvolvimento utilizando ambientes gráficos

O desenvolvimento de serviços interactivos utilizando ambientes gráficos altera significativamente os processos de desenvolvimento utilizados pelas empresas de telecomunicações. Passa-se de um modelo de desenvolvimento tradicional como o de Espiral, para o âmbito do desenvolvimento rápido de aplicações, designado metodologia RAD (*Rapid Application Development*). Todas as fases do desenvolvimento são abrangidas pelo IDE num ambiente integrado de desenvolvimento, em que o *designer* tem ao seu dispor todas as ferramentas necessárias para proceder à especificação, ao desenvolvimento, aos testes e à publicação do serviço interactivo.

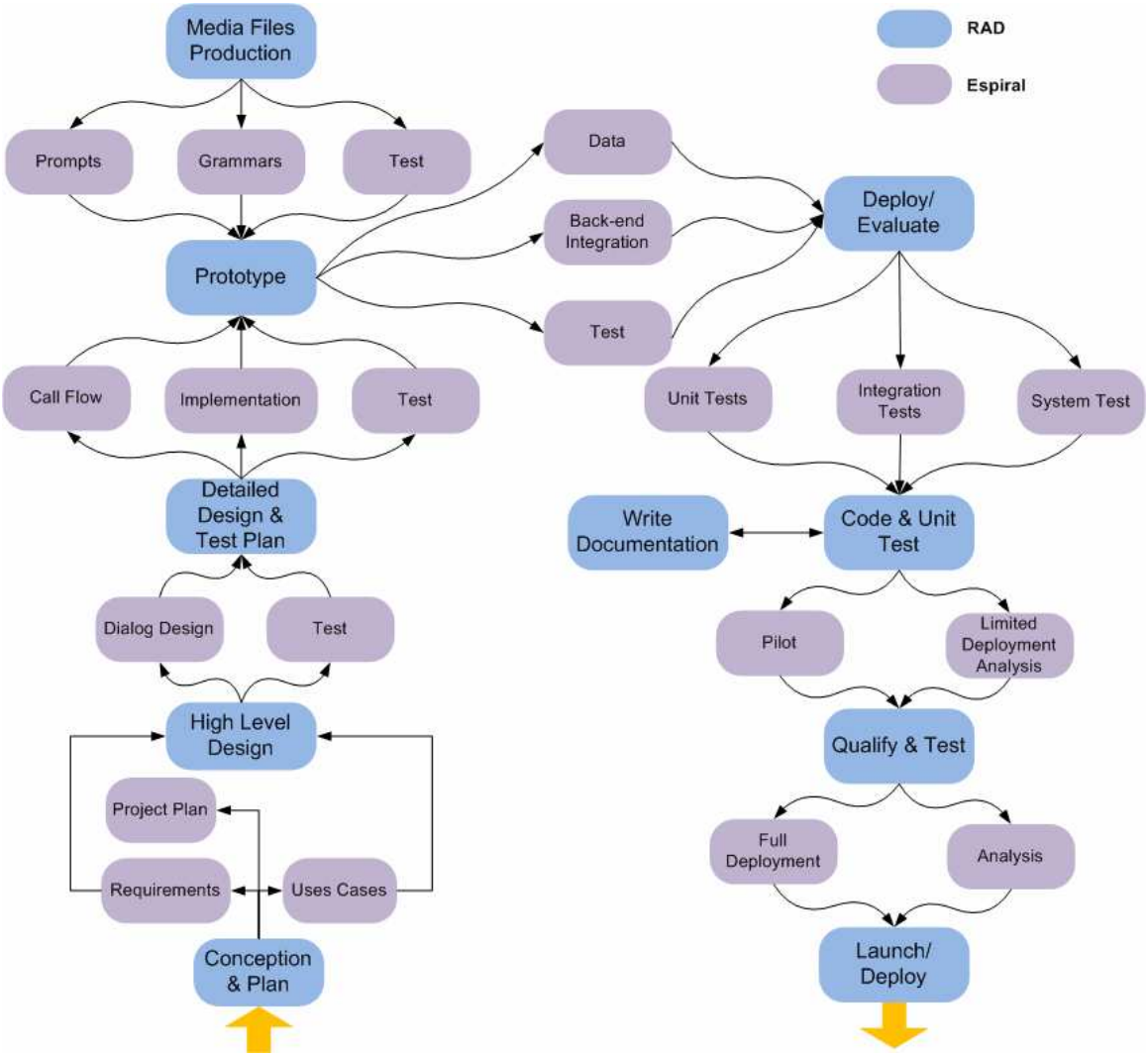

Figura 38 - Modelo de desenvolvimento de software RAD

Nas próximas secção apresentadas as várias fases do modelo de desenvolvimento RAD para o desenvolvimento de serviços interactivos.

Conception & Plan

A primeira fase de desenvolvimento engloba as fases de *Requirements*, *Uses Cases* e *Project Plan*, em que são definidos os requisitos e casos de utilização para o serviço interactivo. É também feita uma previsão para as várias entregas e para funcionalidades implementadas em cada entrega.

High Level Design

Numa primeira abordagem través dos *Requirements* e *Uses Cases* é feito um esboço *High Level Design* do fluxo da chamada e dos diálogos que o serviço irá conter.

Detailed Design & Test Plan

Após o *High Level Design*, são definidos os diálogos do fluxo da chamada *Dialog Design* e os respectivos testes. É elaborado o *Design* dos fluxos mais detalhados *Detailed Design* e feito o plano de testes às funcionalidades.

Prototype

Tendo a especificação dos fluxos detalhados *Detailed Design*, é feita a implementação (*Implementation*) do fluxo da chamada (*Call Flow*) para o serviço interactivo. Em simultâneo são construídas as *prompts* e gramáticas (*Media Files Production*) que vão ser usadas no serviço interactivo. Nesta fase existem também os testes às implementações feitas sobre o fluxo da chamada e testes para verificar a qualidade das *prompts* geradas.

Deploy/Evaluate

Através do protótipo desenvolvido, podemos fazer a primeira publicação do serviço e obter uma primeira avaliação do seu funcionamento. No entanto, para tal é necessário fazer a gestão de toda a informação gerada pela utilização do serviço e estabelecer, no caso de existirem bases de dados, *Web Services* para uma comunicação correcta.

Code & Unit Test

Após a publicação do serviço, são feitos os testes unitário às funcionalidades (*Unit Tests*), diálogos e interacções com o utilizador especificadas nas fases iniciais. Para além dos testes unitários, são feitos testes de integração *(Integration Tests*), com o objectivo de encontrar falhas provenientes da integração interna dos componentes existentes no todo o sistema (*System Tests*). Finalizadas as fases de *Unit Tests*, *Integration Tests* e *System Tests*, é dado início ao melhoramento ao código produzido no desenvolvimento do serviço e feitos novamente os testes unitários para verificar o funcionamento após as alterações.

Write Documentation

Em simultâneo com a fase anterior, é produzida toda a documentação referente ao serviço interactivo. Não é nesta fase que existe o primeiro contacto com a documentação, apenas se referiu nesta altura o processo de documentação com o intuído de realçar que muitas vezes aquando, os testes unitários, a especificação feita inicialmente sofre alterações significativas.

Qualify & Test

Como resultante das duas fases anteriores e antes da primeira entrega do serviço ao cliente, temos uma versão *Pilot* do serviço interactivo, que poderá não ter implementada em si todas as funcionalidades especificadas nas fases iniciais. Nesta fase, são feitos testes para qualificar o nível de aceitação do que já foi desenvolvido através da utilização do serviço, não sendo, normalmente, os testes executados pelo próprio *designer* mas sim por colegas de trabalho.

Launch/Deploy

Tendo as fases anteriores implementadas todas as funcionalidades no serviço interactivo, dá-se por terminado o desenvolvimento e procede-se a entrega do serviço ao cliente.

6.3 Comparação

Os dois modelos de desenvolvimento descritos anteriormente têm uma organização e forma de execução consideravelmente diferentes.

O modelo em Espiral é um modelo de desenvolvimento de software sequencial, no qual o desenvolvimento é visto como precedências de tarefas associadas às várias fases até chegar ao produto final. Para cada fase temos a execução de tarefas para as quais são utilizadas

ferramentas específicas, por exemplo, a utilização de uma ferramenta de desenho e documentação para as fases de *Requirements, Use Cases, Project Plan, Dialog Design e Call flow*. Estas tarefas são normalmente executadas por um analista.

As fases de geração e gestão de *Prompts* e *Grammars* são executadas pela pessoa que faz o desenvolvimento através de editores de áudio e vídeo. A fase de *Data / Back-end Integration* é executado por uma pessoa responsável pela gestão da rede e de conteúdos processados pelos vários serviços existentes. Normalmente, refere-se à integração com *Web Services* e base de dados. As especificações das fases de *Unit Tests, Integration Test* são feitas pelos programadores das vários componentes de um serviço e executadas por pessoas externas. A publicação do serviço nas fases *Pilot, Limited Deployment e Full Deployment* é feita por ferramentas independentes de todo o desenvolvimento, através da compilação e publicação dos vários componentes desenvolvidos. Esta fase é igualmente da responsabilidade de uma outra pessoa, que tem como função a publicação e manutenção dos vários serviços existentes.

Apesar do seu relativo sucesso e de ser a metodologia mais praticada actualmente pelas empresas de telecomunicações, a separação entre as fases de desenvolvimento, assim como, a separação entre as ferramentas de desenvolvimento, levam a atrasos e falhas de comunicação. Nem sempre a passagem de uma fase para a outra é feita da melhor forma. Isto porque, a comunicação entre a fase anterior e a fase subsequente não é fácil e difícil garantir transparência. O tempo que é despendido a resolver estas falhas de comunicação causa um maior atraso na entrega do produto final. A necessidade de utilização de diferentes ferramentas para cada fase do desenvolvimento, leva a que o custo final do desenvolvimento seja superior, devido ao facto de serem utilizadas ferramentas licenciadas.

No entanto, no modelo Espiral temos o desenvolvimento de serviços com um controlo total por parte dos programadores de todas as fases do desenvolvimento. Para além desta vantagem, o modelo Espiral é apropriado para o desenvolvimento de serviços com uma maior escala do que o RAD, isto é, serviços que englobam muitos componentes. Com o desenvolvimento independente dos vários componentes e com interfaces bem definidas, o grau de sucesso é maior do que com elementos pré-desenvolvidos. O custo de adaptação de um determinado componente para um novo serviço pode ser maior que o custo do desenvolvimento desse componente de raiz.

O desenvolvimento de serviços recorrendo a interfaces gráficas, com o recurso a *"drag and drop"* permite a criação de aplicações com uma enorme redução da programação os processos de *Design*, construção e redefinição do fluxo de uma chamada, permitindo realizar todo o ciclo de desenvolvimento de um serviço interactivo num IDE. Com estas alterações nos processos de criação de um serviço interactivo, passamos de um modelo de desenvolvimento em Espiral para um desenvolvimento rápido de aplicações.

As interfaces gráficas permitem ao *designer* o desenvolvimento rápido e a prototipagem de aplicações e enfatiza um ciclo de desenvolvimento para projectos com a reutilização de vários componentes e com o recurso de ferramentas de fácil acesso. Não existe a necessidade de licenciamento de mais aplicações, o que reduz o custo final dos serviços interactivos. O desenvolvimento é conduzido a um nível mais alto de abstracção sem ser necessário o conhecimento das camadas mais baixas de programação. Com isto, torna a conclusão do desenvolvimento feito mais rápido e mais cedo, conduzindo assim a uma maior flexibilidade a alterações e modificações dos requisitos definidos inicialmente.

Através da interface podemos obter de quase de imediato uma primeira versão do serviço interactivo. As fases *Conception & Plan*, definição dos requisitos e casos de utilização pode ser feita junto dos clientes com o recurso à interface gráfica. Tal como a fase de *High Level Design* e uma primeira versão do fluxo da chamada e os respectivos diálogos, pode ser apresentada no início da definição dos requisitos. Nas fases de *Detailed Design & Test Plan* são criados os diálogos do fluxo da chamada, este é um processo no qual o *designer* deverá estar no ambiente de desenvolvimento, para que seja feito o plano de testes às funcionalidades. Após esta fase, é obtido um primeiro protótipo do serviço interactivo, *Prototype*. Poderão não estar contempladas todas as funcionalidades necessárias ao serviço, mas já é oferecido um produto funcional ao cliente. Nas fases seguintes são adicionadas progressivamente novas funcionalidades, é feita a geração e gravação das *prompts* e gramáticas utilizadas e gerados novos protótipos que substituem as entregas anteriores. Para cada fase de entrega, para além da adição de novas funcionalidades (*Deploy/Evaluate*), são feitos testes (*Code & Unit*) às funcionalidades já implementadas, ficando assim o produto final mais testado do que o modelo em Espiral.

Paralelamente ao desenvolvimento rápido do serviço interactivo e aos testes, é feita a documentação das funcionalidades implementadas, *Write Documentation*. Nas fases finais do desenvolvimento e entrega do serviço são feitos os testes para qualificar o nível de aceitação do que já foi desenvolvido (*Qualify & Test*) através da utilização do serviço. Na fase final (*Launch/Deploy*) é feita a entrega do serviço interactivo e termina-se o desenvolvimento. A partir desta fase apenas fica a manutenção do serviço quando existirem problemas ou alterações aos requisitos definidos inicialmente.

O processo de criação de serviços interactivos utilizando interfaces gráficas reduz significativamente o tempo da entrega de um serviço, agiliza os processos de desenvolvimento e reduz significativamente a ocorrência de erros no desenvolvimento dos vários componentes do serviço. Podemos afirmar que o recurso às interfaces gráficas reduz aproximadamente para metade o tempo todas as fases do desenvolvimento.

7 Conclusões

No presente capítulo serão apresentadas as conclusões sobre o tema estudado, qual a perspectiva seguida, quais as soluções encontradas e quais os resultados obtidos. No final, serão apontadas sugestões para a possível prossecução do trabalho já feito.

7.1 Análise do trabalho realizado

O tema **Ambientes Gráficos para a Criação de Serviços Interactivos** revelou-se muito interessante e motivador, não só por ser inovador, mas também por poder constituir uma mais-valia para as empresas de telecomunicações.

Nesta nova era das comunicações, o mercado modifica-se e evolui de uma forma extremamente rápida, onde as redes de dados e de voz convergem. Existe uma competição feroz entre os operadores e prestadores de serviços pela conquista e manutenção dos clientes. Uma das maneiras para as empresas se diferenciarem e adquirirem novos clientes consiste no fornecimento de novas tecnologias que proporcionem a criação e disponibilização de serviços de valor acrescentado. As empresas de telecomunicações tentam acompanhar essa evolução, procurando soluções que minimizem os custos na criação e manutenção dos seus serviços interactivos.

A necessidade descrita na premissa anterior levou a que as empresas de telecomunicações recorram a interfaces gráficas para a criação de serviços interactivos. Utilizam interfaces amigáveis, assistentes de criação de operações e acções através de *"drag and drop"* de objectos que representam operações sobre o fluxo da chamada. Estas interfaces agregam num só IDE todas das funcionalidades disponibilizadas pelos Sistemas Interactivos de Voz, reunindo todas as funcionalidades necessárias ao desenvolvimento dos serviços tais como: gestão de ficheiros multimédia, interoperabilidade com sistemas externos, controlo de sessões de utilizadores e publicação dos serviços para as redes tradicionais e redes móveis.

Apesar das soluções estudadas serem capazes de satisfazer as necessidades da maior parte dos serviços interactivos simples, facilmente se encontram problemas quando pretendemos desenvolver serviços mais rebuscados. Contudo, ainda continua a existir uma enorme dificuldade em dotar as interfaces gráficas de meios capazes de suportar todas as funcionalidades necessárias para a criação de serviços interactivos. É difícil que as interfaces gráficas consigam contemplar todas as especificidades existentes no desenvolvimento destes serviços.

Apesar de existirem ferramentas que potenciam e facilitam significativamente o desenvolvimento de uma interface gráfica de raiz, a construção de uma solução de raiz não era viável no tempo de duração da dissertação. Foi então utilizada, a plataforma *openVXML*, que consiste na criação de um conjunto de *Plug-Ins* sob licença *open source* para a plataforma *Eclipse* e tem por base as ferramentas estudadas para a modelação dos fluxos da chamada e edição visual: EMF, GEF e GMF.

A especificação feita para uma interface gráfica é um ponto de partida para a construção de uma solução que reúna todas as vantagens encontradas em cada uma das soluções e colmate as falhas ou funcionalidades inexistentes nas próprias.

O **Ambiente Gráfico para a Criação de Serviços Interactivos** apresentado, utilizando o *openVXML,* é totalmente funcional. Pode-se concluir que a prova de conceito responde na totalidade aos objectivos definidos inicialmente. É também um bom ponto de partida para que as empresas de telecomunicações criem uma versão estável e funcional para a criação gráfica de serviços. Com os resultados dos testes feitos, podemos afirmar que existem algumas falhas de funcionamento no ambiente de desenvolvimento ou na plataforma *openVXML*, mas nunca impedindo que o seu funcionamento como um todo seja prejudicado. A versão *open source* do *openVXML* necessita ainda de muito desenvolvimento para atingir alguma maturidade, mas isso é algo perfeitamente possível, se tomarmos o exemplo da versão apresentada comercialmente pela *openMethods*.

Estamos perante mudanças significativas nas formas utilizadas pelas empresas de telecomunicações para a criação de serviços interactivos. A utilização das interfaces gráficas tem um impacto significativo nos processos de desenvolvimento usados pelas empresas de telecomunicações. As interfaces gráficas permitem ao *designer* o desenvolvimento rápido e a prototipagem de aplicações, e enfatiza um ciclo de desenvolvimento para projectos com a reutilização de vários componentes. Através da interface podemos obter quase de imediato uma primeira versão do serviço interactivo.

Após todo o trabalho realizado, conclui-se que, existe uma grande necessidade por parte das empresas de telecomunicações em adquirir soluções capazes de integrar a edição visual e ferramentas para o desenvolvimento de serviços num IDE, de forma a fazer face à agressiva concorrência e oferecendo aos seus clientes novas tecnologias que proporcionam a criação e disponibilização de serviços de valor acrescentado.

7.2 Perspectivas de trabalho futuro

Existem vários componentes que poderão ser adicionados de forma a melhorar a solução *openVXML,* para que esta cumpra na totalidade todos os requisitos necessários à criação de serviços sobre os Sistemas de Resposta Interactiva.

É de todo útil que, os testes de usabilidade tenham implicações na interface gráfica. Para tal, deverá ser feita uma melhor avaliação da usabilidade do *openVXML,* utilizando um método não tão formal como foi feito. Deverão ser recolhidos os resultados dos testes de usabilidade, o grau de aceitação por parte dos utilizadores e proceder às respectivas alterações ao nível da interface.

Seria uma mais valia o *openVXML* disponibilizar rotinas que permitissem a simulação do serviço interactivo. Para tal, é necessário adicionar componentes de interpretação do *VoiceXML*, conversão de texto para fala (TTS) e reconhecimento da fala (ASR).

Para tornar ainda mais completa a utilização do simulador do serviço interactivo seria, também, uma mais valia que *openVXML* fizesse a geração automática de testes ao fluxo. A detecção de problemas e a respectiva resolução seria mais fácil e rápida.

Apesar de o ambiente de desenvolvimento ter sido criado numa empresa, com a utilização de componentes comerciais, actualmente existem soluções no âmbito do *open source* capazes de criar um ambiente de desenvolvimento totalmente *open source*. Para a realização deste projecto, seria interessante criar uma parceria entre uma empresa de telecomunicações e um

grupo de investigação de uma faculdade. Existiria a passagem do conhecimento na área das telecomunicações por parte da empresa e o desenvolvimento de cada componente pelo grupo de investigação. Um exemplo de uma solução totalmente *open source* seria: *Asterisk PBX* (Sistema de Resposta Interactiva), o *openVXI* (para o interpretador de *VoiceXML*), *Tomcat* (para a publicação dos serviços) e *openVXML* para a criação visual de serviços interactivos.

Referências e Bibliografia

[Apache 06] **Apache Software Foundation**, The Apache Tomcat 5.5 Servlet/JSP Container, 2006.

[Armstrong 05] **The J2EETrademarked 1.4 Tutorial**, For Sun Java System Application Server Platform Edition 8.2, Eric Armstrong, Jennifer Ball, Stephanie Bodoff, Debbie Bode Carson, Ian Evans, Dale Green ,Kim Haase, Eric Jendrock, December 5, 2005.

[Avaya 07 B] **Avaya™ Interaction Center**, Release 6.0, Avaya Workflow Designer Guide, DXX-1012-02 Issue 1.0, June 2007.

[Bierman 07 A] **Extending GMF: Generating Domain-Specific Model Editors with Complex Editing Commands** - E. Biermann, A. Crema, K. Ehrig, C. Ermel, C. Kohler, R. Schmutzler and G. Taentzer - Technische Universitat Berlin, Germany, University of Leicester, United Kingdom, CWI Amsterdam, The Netherlands, Philipps-Universitat Marburg, Germany, 2007.

[Bierman 07 B] **Extending GMF: Generating Domain-Specific Model Editors with Complex Editing Commands** - E. Biermann, A. Crema, K. Ehrig, C. Ermel1, C. Kohler, R. Schmutzler, and G. Taentzer - Technische Universitat Berlin, Germany,University of Leicester, United Kingdom,CWI Amsterdam, The Netherlands,Philipps-Universit¨at Marburg, Germany, 2007.

[Butler 07] **Remote application design**, International Business Machines Corp. (Armonk, NY), Butler, Nicholas David; Bowden, Jacqueline; Hyatt, Steven John; Renshaw, David Seager; Wong, Yuk-Lun, (Oct-31-2000), 2007.

[Cerami 02] **Web Services Essentials** - Distributed Applications with XML-RPC, SOAP, UDDI & WSDL - By Ethan Cerami, February 2002

[Chappell 07] **Introducing Microsoft Windows Workflow Foundation** - David Chappell, Chappell & Associates, 2007.

[Ehrig 07 A] **Towards Graph Transformation Based Generation of Visual Editors using Eclipse** - Karsten Ehrig, Claudia Ermel, Stefan Hansgen, Gabriele Taentzer - Institut fur Softwaretechnik und Theoretische Informatik Technische Universitat Berlin Germany, 2007.

[Ehrig 07 B] **Generation of Visual Editors as Eclipse Plug-Ins** – Karsten Ehrig, Claudia Ermel, Stefan Hansgen, and Gabriele Taentzer - Technische Universit at Berlin, Germany, 2007.

[Inter Gui 05] **IBM WebSphere Voice Server V5.1.1/V5.1.2 and Avaya Interactive Response V1.3: An Interoperability Guide**, James Chamberlain, Bob Anderson, George Clelland, Mukund Chandrasekhar, Steve Sahakyan, Redpaper, 2005

[Julian Sinai 07] **Tool for Graphically Defining Dialog Flows and for establishing operational links between Speech Applications and Hypermedia content in an Interactive Voice Response environment,** Julian Sinai, Palo Alto, CA (US), Steven C. Ehrlich, Burlingame, CA (US), Rajesh Ragoobeer, San Mateo, CA (US), Nuance Communications, Menlo Park CA (US), 2007.

[KW Bill 06] **Speech Service Creation, An Overview of Speech Service Creation Tools**, K. W. (Bill) Scholz, NewSpeech, LLC, December, 2006.

[Kunins 02] **VoiceXML – Strategies and Techniques for Effective Voice Application Development with VoiceXML 2.0**, Professional Developr's Guide Series, Chetan Sharma and Jeff Kunins, 2002.

[Moore 07] **Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework** - RedBooks - Bill Moore, David Dean, Anna Gerber, Gunnar Wagenknecht, Philippe Vanderheyden, 2007.

[Nielsen 04 A] **Nielsen, Jakob (1994). Usability Engineering**. San Diego: Academic Press, 115-148. ISBN 0-12-518406-9, 2004.

[Nielsen 04 B] **Nielsen, J. (1994). Heuristic evaluation.** In Nielsen, J., and Mack, R.L. (Eds.), Usability Inspection Methods, John Wiley & Sons, New York, NY, 2004.

[Nielsen 93 C] **Nielsen, Jakob, and Landauer, Thomas K**.: "A mathematical model of the finding of usability problems," Proceedings of ACM INTERCHI'93 Conference (Amsterdam, The Netherlands, 24-29 April 1993), pp. 206-213, 1993.

[VTP News 05] **Voice Tools Project (VTP)**, Creation Review, VTP Newsgroup, 2005.

[Wiston 70] **Royce, Winston (1970), "Managing the Development of Large Software Systems",** Proceedings of IEEE WESCON 26 (August): 1-9, 1970.

[Yi-Xuan Li 07] **Voice Composer: A Development Tool for Voice Applications**, Yi-Xuan Li and Nai-Wei Lin - Department of Computer Science and Information Engineering - National Chung Cheng University, 2007.

Referências Web

[Avaya 07 A]	**Avaya solutions for IP Telephony** **http://www.avaya.com/**
[ApexVoice 08]	**ApexVoice - Voice Communications, Inc. OmniVox3D, OmniView and OmniVoXML** **http://www.apexvoice.com/**
[Aspect 08]	**Aspect** **http://www.aspect.com/**
[Audium 08]	**AudiumCorp** **http://www.audiumcorp.com/**
[DB 07]	**Database** **http://en.wikipedia.org/wiki/Data_base**
[EMF 07]	**Eclipse Modeling Framework (EMF)** **http://www.eclipse.org/emf**
[Eclipse Doc 07]	**Eclipse Documentation –** Archived Release Eclipse 3.2.1, September 21, 2007 **http://archive.eclipse.org/eclipse/downloads/drops/R-3.2.1-200609210945/index.php**
[EVTP 07]	**Eclipse Voice Tools Project** (Table of Contents) **http://www.eclipse.org/vtp/openvxml/index.php**
[GUI 08]	**GUI (Graphical User Interface)** **http://en.wikipedia.org/**
[GEF 07]	**Eclipse Graphical Editing Framework (GEF)** **http://www.eclipse.org/gef**
[GMF 07]	**Eclipse Graphical Modeling Framework (GMF)** **http://www.eclipse.org/gmf**
[GetVocal 08]	**GetVocal** **http://www.getvocal.com/**
[Genesys 08]	**Genesys** **http://www.genesys.com/**
[IDE 07]	Integrated Development Environment (IDE) **http://en.wikipedia.org/wiki/Integrated_development_environment**
[IBM 08]	**IBM Voice Systems** **http://www.ibm.com/**

[MDN 07]	**Microsoft Developer Network** **http://msdn2.microsoft.com/**
[Open Group 07]	**Open Group - Service-Oriented Architecture** **http://opengroup.org/**
[Oasis 07]	**OASIS Advanced Open Standards for the information Society** **http://www.oasis-open.org/**
[OpenMethods 08]	**OpenMethods – openVXML Solution** **http://www.openmethods.com/**
[SpeechMag 07]	**SpeechTechMag - Bringing Video to the Voice Arena** **http://www.speechtechmag.com/**
[Pilot 08]	**Understanding the Pilot Project** **http://www.microsoft.com/**
[Tim Barnes 07]	**Tim Barnes** – CEO and Managing Partner & Co-Founder, of OpenMethods **www.openmethods.com**
[VoiceXML 07]	**Voice Extensible Markup Language (VoiceXML) Version 2.0** **http://www.w3.org/TR/voicexml20/**
[Vicorp 07]	**Vicorp** **http://www.vicorp.com/**
[Voice Objects 08]	**Voice Objects** **http://www.voiceobjects.com/**
[Voice Dev 07]	**VoiceXML Developer Tools Roundup** **http://www.developer.com/**
[VTP 07]	**Voice Tools Project** **http://www.eclipse.org/proposals/eclipse-voicetools/**
[WS Activ 07 A]	**W3C – Web Services Activity – Architecture Domain** **http://www.w3.org/2002/ws/**
[WS Archit 07 B]	**W3C – Web Services Architecture** **http://www.w3.org/TR/ws-arch/**
[W3C-SRGS 07]	**W3C - Speech Recognition Grammar Specification version 1.0** **http://www.w3.org/TR/speech-grammar/**
[W3C-CCXML 07]	**W3C - Voice Browser Call Control (CCXML) Version 1.0 - W3C Working Draft 19 January 2007** **http://www.w3.org/TR/ccxml/**

[WWF 07] **Microsoft Windows Workflow Foundation**
http://netfx3.com/content/WFHome.aspx

[XML 07] **Extensible Markup Language (XML)**
http://www.w3.org/XML/

Anexo A

Requisitos para a edição visual

Tabela 12 - Requisitos de edição visual (completa)

Código	Requisito	Descrição
RFE01	**Inserção de *Speech Objects***	**Ao adicionar um novo *Speech Object* deverá existir a validação da inserção deste, se é possível a sua inserção e quais as ligações possíveis.**
RFE02		**Permitir a configuração das variáveis ou atributos do *Speech Object* tendo em conta pré-validação existente na altura da sua inserção.**
RFE03		**Associação de acções por defeito aos *Speech Objects* (ex. *noinput error, timeout, nomach*).**
RFE04	**Configuração de *Speech Objects***	**Colocar os valores mais usuais nos seus atributos por defeito (ex. *timeout* 5 segs).**
RFE05		**Sempre que possível fornecer uma *drop down list* com a lista de opções possíveis para os atributos.**
RFE06	**Interligação entre *Speech Objects***	**Deverá ser validada todas as ligações possíveis para cada *Speech Object*.**
RFE07	**Exemplos de aplicações *Dialog Objects***	**Vários exemplos de *Dialog Objects* e aplicações pré-definidas para serem tomadas como exemplo para o *designer*.**
RFE08	**Interligação entre *Dialog Objects***	**Cada *Dialog Object* é representado num *Canvas*, deverá ser possível efectuar a interligação entre diferentes *Dialog Objects*. Permitindo que o fluxo da chamada seja redireccionado para um diálogo diferente.**
RFE09	***Templates* de *Dialog Objects***	**Deverá existir vários *Templates* de *Dialog Objects* com algumas das funcionalidades mais usuais na criação do fluxo de um Serviço Interactivo.**
RFE10	***Templates* de pequenos fluxos**	**Deverá existir vários *Templates* de pequenos fluxos que o *designer* pode adicionar para o fluxo principal da definição do Serviço Interactivo.**
RFE11	**Aplicação por defeito**	**Ao iniciar uma nova aplicação deverá estar definido por defeito os *Speech Objects* de início e fim da chamada (e deverão ser fixos).**
RFE12	**Várias perspectivas**	**Como forma de agilizar o processo de criação do fluxo, e dar a possibilidade de ajuste do esquema de apresentação da interface, esta deverá permitir a configuração da estrutura em várias vistas pré-definidas e ajustáveis pelo *designer*.**
RFE13	**Várias *views***	***Palette* de operações sobre o fluxo de chamada.**
RFE14		***Palette* de operações sobre o *Design* (*Dialogs*, *Sub-Dialogs*, Interligação entre *Dialogs*, *Templates* de *Dialogs* e fluxos).**
RFE15		***Palette* de operações de comunicação e interligação com entidades externas (base de dados, *Web Services*, etc).**
RFE16		**Pallet de vista reduzida do fluxo da chamada (como forma de facilitar a visualização no caso de fluxos extensos).**
RFE17		***Canvas* para o fluxo da chamada.**
RFE18		***Browser* dos projectos existentes.**
RFE19		**Editor de *VoiceXML* em simultâneo que o editor visual (permitindo que o desenvolvimento seja efectuado paralelamente).**
RFE20	**Gestão dos ficheiros multimédia e gramáticas**	**Existência de vários idiomas (vários ficheiros multimédia).**
RFE21		***Prompts* de vídeo.**
RFE22		***Prompts* de voz.**

Código	Requisito	Descrição
RFE23		***Prompts* de texto.**
RFE24		**Gramáticas de DTMF.**
RFE25		**Gramáticas de voz.**
RFE26		**Adição de *prompts* através de *"drag and drop"* para a pasta de ficheiros multimédia.**
RFE27	**Ferramenta para a edição de ficheiros multimédia**	**Gravação de áudio.**
RFE28		**Edição de áudio.**
RFE29		**Conversão entre vários formatos de áudio.**
RFE30		**Gravação de vídeo.**
RFE31		**Edição de vídeo.**
RFE32		**Conversão entre vários formatos de vídeo.**
RFE833	**Desenho do fluxo**	**Invalidar as ligações entre *Speech Objects* erradas.**
RFE34		**Mensagem de erro no estabelecimento de uma ligação errada.**
RFE35		**Permitir a apresentação dos atributos possíveis de passagem entre *Speech Objects*.**

Anexo B

Requisitos para os *Speech Objects*

Tabela 13 - Requisitos para os Speech Objects (completa)

Código	Requisito	Descrição	
RFS01	***Prompt***	**A *prompt* pode ser: texto, referencia, número, dígitos, ordinal, quantias monetárias, caracteres, ficheiro de áudio, ficheiro de vídeo.**	
RFS02	***Name***	**Todos os objectos para além das suas características específicas deverão ter um campo para a definição do nome do *Speech Object*.**	
RFS03	**Definição de *User Input* (a ser aplicado sempre que for necessário interacção por parte de utilizador)**	**DTMF**	***Barg-in (True/False).***
RFS04			***Initial Input timeout.***
RFS05			***Interdigit timeout*.**
RFS06			***Termination timeout.***
RFS07			***Termination character* (caracter de terminação).**
RFS08		**Voz**	***Barg-in (True/False).***
RFS09			***Initial Input timeout.***
RFS10			***Speech incomplete timeout.***
RFS11			***Speech completion timeout.***
RFS12			***Maximum speech length.***
RFS13			***Minimum confidence level accepted.***
RFS14			***Typical caller environment.***
RFS15			***Speed vs Accuracy.***
RFS16		**DTMF e Voz**	**Todas as características definidas para os dois tipos de *input* (DTMF e voz).**
RFS17	***Begin***	**Definição do início do fluxo da chamada.**	
RFS18	***End Call***	**Terminação do fluxo da chamada e fim da chamada.**	
RFS19	***Play Prompt***	***Prompt*.**	
RFS20		**Definição do idioma.**	
RFS21		**Concatenação de várias *prompts* numa só *prompt*.**	
RFS22		**Edição e posicionamento das *sub-prompts* (mover para anterior, mover para o posterior, editar, eliminar e adicionar).**	
RFS23		***Barg-in (True/False*).**	
RFS24	***Variable assignment***	**Atribuição de um nome para a variável.**	
RFS25		**Definição do tipo da variável que é do tipo *prompt*.**	
RFS26		**Associação da variável a um idioma.**	
RFS27	***Desicion***	**Apresentar a lista de variáveis existentes.**	
RFS28		**Seleccionar a variável (A, B, C, D, …) para efectuar inúmeras comparações.**	
RFS29		**Seleccionar o termo de comparação entre variáveis (<, >, =<, >=, ==, !=, *Exists*, *Not Exists*).**	

Código	Requisito	Descrição	
RFS30		**Opção de saída do objecto (*True/False*), consoante a verificação ou não da condição definida.**	
RFS31	***Question***	**Deverá ser possível a definição e criação da variável onde será guardada a resposta do utilizador.**	
RFS32		**Definição da *prompt* referente à pergunta.**	
RFS33		**Definição (se aplicável) da gramática.**	
RFS34		**Possibilidade de adicionar Gramáticas pré-definidas (datas, números inteiros, número de telefone, horas, opções boleanas, dígitos ou um ficheiro de gramática existente).**	
RFS35		**Definição do *User Input*.**	
RFS36	***Option Set***	**Opção**	**Adicionar, editar e remover uma opção.**
RFS37			***Prompt* (a tocar na selecção da opção).**
RFS39			**Gramática (se aplicável).**
RFS40		**Definição de *User Input*.**	
RFS41	***Record***	***Prompt*.**	
RFS42		**Deverá ser possível a definição ou criação da variável onde será guardada o *User Input* do utilizador (áudio ou vídeo).**	
RFS43		***Barge-in (True/False).***	
RFS44		***Play Beep (True/False).***	
RFS45		**Terminação por DTMF (pode ser utilizado com *True*, *False* ou especificar um caracter).**	
RFS46		***Initial Input timeout*.**	
RFS47		***Termination timeout.***	
RFS48		***Maximum recordin time.***	
RFS49		***Type (*áudio/vídeo*).***	
RFS50	***Transfer***	***Destination*.**	
RFS51		***Bridge (True/False)***	
RFS52		***Connection Timeout.***	
RFS53	***SubDialog***	**URL do *Sub-Dialog* (entenda-se como redireccionamento para um outro Serviço Interactivo).**	
RFS54		***Inputs* (lista de variáveis que devem passar para o *Sub-Dialog*).**	
RFS55		***Outputs* (lista de variáveis que devem ser retornadas pelo *Sub-Dialog*).**	
RFS56	***Script Block*** (Blocos de código em *JavaScript*)	**Edição de código *JavaScript* num editor de texto.**	
RFS57		**Compilação em tempo real de edição, permitindo a sua correcção antes da publicação.**	
RFS58	***Web Service Call***	**Definir a variável onde deverá guardar a informação retornada pela invocação do *Web Service*.**	
RFS59		***Service***	**Selecção do *Web Service* a utilizar.**
RFS60			***portType* (definição das operações a serem tomadas e o tipo de mensagens).**
RFS61			***Message* (dados da mensagem de uma operação).**
RFS62			***Type* (tipo de dados usado pelo *Web Service*).**

Código	Requisito	Descrição	
RFS63			***Binding* (definição do formato das mensagens e os detalhes para cada portType).**
RFS64		**Conteúdo que será passado ao *Web Service*.**	
RFS65		**Conteúdo que será retornado pelo *Web Service*.**	
RFS66	***Data Base Query***	**Definir a variável onde deverá guardar a informação retornada pela consulta.**	
RFS67		**Base de Dados**	**Selecção da base de dados.**
RFS68			**Definição da tabela para efectuar a consulta.**
RFS69			**Selecção e definição dos campos da tabela.**
RFS70		**Selecção da operação (*SELECT, INSERT, UPDATE* e *DELETE*).**	
RFS71		**Selecção dos atributos (variáveis resultante do fluxo da chamada) a utilizar para a interacção com a base de dados.**	
RFS72	***Portal Entry***	**Transpor o fluxo da chamada para outro *Canvas*.**	
RFS73	***Portal Exit***	**Voltar o fluxo da chamada para o *Canvas* anterior.**	

Anexo C

Descrição da *InoAPI*

API para o desenvolvimento de serviços telefónicos sobre a plataforma *InoVox-IP*. Disponibiliza uma interface de alto nível para acesso e controlo dos vários recursos de telefonia e média. Abstrai a complexidade das operações decorrentes da utilização das APIs de baixo nível, disponibilizadas pelos fornecedores dos vários recursos suportados pelo *InoVox-IP* como telefonia, fax , etc.

Características

- Modos de funcionamento: síncrono e assíncrono
- Serviços
  - *Inbound* (recepção e atendimento de chamadas);
  - *Outbound* (geração de chamadas);
- Protocolo proprietário para a comunicação com o *InoVox-IP* (baseado em mensagens ASCII)
- Comunicação via TCP/IP com o *InoVox-IP*
- Possibilidade de ligação ao *InoVox-IP* a funcionar em *Cluster* ou *Stand-alone*
- Mecanismo de *Heartbeat* para verificar o correcto funcionamento da comunicação com o *InoVox*
- Capacidade de *logging* das mensagens enviadas e recebidas suportando vários níveis (*Debug*, *Info*, *Warn*, *Error*)

Principais métodos utilizados

Tabela 14 - Principais métodos da InoAPI

Métodos	Descrição
initialize()	**Inicialização da *InoApi* e estabelecimento da comunicação com a plataforma *InoVox-IP*.**
setICallMaskEx()	**Desregista uma máscara e o nome do serviço respectivo.**
claimTelResource()	**Solicita o endereço IP de um IVR do *cluster* com os recursos solicitados disponíveis.**
acceptICall()	**Aceita uma chamada de entrada no *InoVox-IP*.**
connect()	**Efectua o atendimento de uma chamada de entrada depois de aceite.**
attachCapabilities()	**Altera a capacidade para utilizar reconhecimento de fala.**
routeCall()	**Estabelece uma ligação entre os dois *endpoints*.**
release()	**Liberta um *endpoint* e recursos associados no *InoVox-IP*.**
suspendResume()	**Permite suspender uma chamada e voltar a activá-la.**
claimTelResource()	**Solicita um determinado recurso.**
claimCallEx()	**Requisita recursos para chamadas de *outbound* ao *InoVox-IP*.**
transferCall()	**Transfere um *endpoint* de uma aplicação para outra desde que a segunda esteja registada no mesmo IVR.**
makeCallEx()	**Estabelece uma chamada de saída no canal de voz associado ao *endpoint* especificado.**

Métodos	Descrição
resetCall()	**Pára toda a actividade num determinado *endpoint*.**
adjustAnnouncement()	**Ajusta o *play-speed* ou *play-volume* associado ao *endpoint* especificado.**
getDigits()	**Recolhe dígitos do utilizador.**
playAnnouncement()	**Toca um anúncio no canal de voz associado ao *endpoint* especificado.**
playData()	**Toca um determinado tipo de dados no canal de voz associado ao endpoint especificado. Tipo de dados: Data, Horas, quantias em dinheiro, números naturais, dígitos.**
playIndexFile()	**Toca alguns conteúdos de um ficheiro indexado no canal de voz associado ao *endpoint* especificado.**
recordVoice()	**Grava do canal de voz associado ao *endpoint* especificado para um ficheiro.**
recordVoiceEx()	**Idêntico ao anterior mas permite configurar o formato de gravação. Suporta concatenação dos ficheiros gravados.**
clearDigitBuffer()	**Limpa o *digit buffer* de um *endpoint*.**
stopOperation()	**Termina as operações de processamento associadas ao *endpoint* especificado.**
TTSProcess()	**Inicia a síntese de fala num determinado *endpoint.***
conferenceManager()	**Executa uma determinada operação de conferência sobre determinado *endpoint* tendo como parâmetros o identificador da conferência *(conference bridge)*. Caso a conferência não exista, esta será criada automaticamente.**

Anexo D

Prompts de texto utilizadas no serviço "*Find You Movie*".

Tabela 15 - Prompts do serviço "Find Your Movie"

prompt	Descrição
Welcome	**Bem vindo ao servico de pesquisa de filmes.**
Opt1	**Prima um para consultar o filme na base de dados**
Opt2	**Prima dois para falar com o assistente.**
Opt#	**Prima cardinal para sair.**
AskMovieID	**Qual o identificador do filme que deseja pesquisar.**
DBError	**Ocorreu um erro na ligacao a base de dados. Por favor tente mais tarde.**
MovieName	**O filme que procura é o <nome do filme>.**
LineBusy	**Por favor aguarde, Prevemos atender a sua chamada brevemente.**
NoAnswer	**De momento não é possível falar com o assistente. Por favor tente mais tarde.**
Thanks	**Obrigado por utilizar o servico de pesquisa de filmes.**

Anexo E

Fluxo da chamada do guião proposto aos utilizadores para os testes de usabilidade

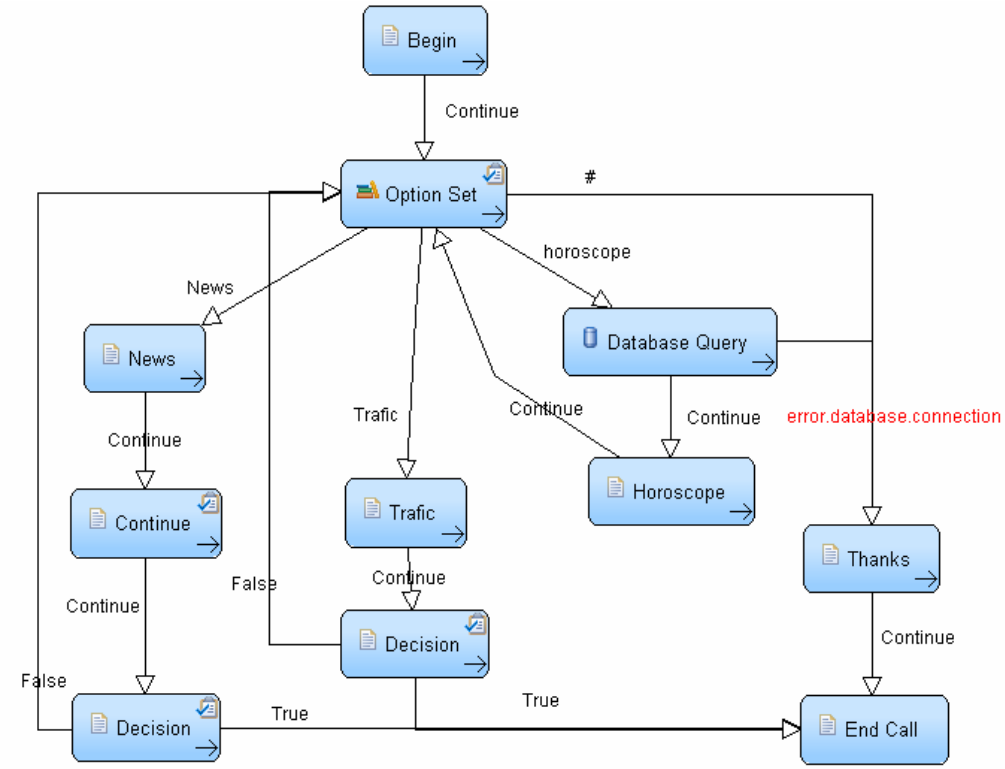

Figura 39 – Fluxo do serviço interactivo utilizado no guião

Prompts de texto utilizadas

Tabela 16 – Prompts do serviço interactivo utilizado no guião

prompt	Descrição
Option Set	**Bem vindo ao serviço interactivo. Prima um para notícias. Prima dois para notícias sobre trânsito. Prima três para ouvir o horóscopo. Prima cardinal para sair.**
News	**<<ficheiro de áudio>>**
Trafic	**<<ficheiro de áudio>>**
Horoscope	**<<prompt que resulta da consulta à base de dados>>**
Decision	**Deseja sair?**
Thanks	**Obrigado por utilizar o servico interactivo.**