INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Solução de Voice Portals para Marcação de Consultas

André Manuel Ribeiro Sabino

Dissertação para obtenção do Grau de Mestre em

Engenharia de Redes de Comunicações

Júri

Presidente: Prof. Luís Eduardo Teixeira Rodrigues
Orientador : Prof. Mário Serafim dos Santos Nunes
Co-Orientador : Prof. Pedro Manuel Moreira Vaz Antunes de Sousa
Vogal: Prof. Alberto Manuel Ramos da Cunha

Setembro de 2008

Agradecimentos

Gostaria de apresentar os meus sinceros agradecimentos, a todos os que, de uma forma mais ou menos activa, contribuíram positivamente para o resultado final do trabalho, que neste documento é apresentado.

Agradeço ao André Lourenço, consultor da Link Consulting, que me transmitiu alguma da sua experiência em projectos semelhantes ao que é abordado neste documento, dando assim um contributo precioso. Da mesma forma agradeço à Raquel Pinto, devido à ajuda na fase inicial da realização deste trabalho, com as diversas linhas orientadores fornecidas.

Quero agradecer ao Orientador e ao Co-Orientador deste trabalho, respectivamente o Professor Mário Serafim dos Santos Nunes e o Professor Pedro Manuel Moreira Vaz Antunes de Sousa, devido à sua colaboração e disponibilidade ao longo da realização do mesmo.

Agradeço igualmente, pelos contributos dados da parte do Professor Paulo Rogério Barreiros d'Almeida Pereira, assim como do Professor Rui Santos Cruz, que me ajudaram a perceber melhor certos aspectos teórico-práticos de diversos temas abordados directa ou indirectamente neste trabalho. Exemplos e referências a trabalhos realizados na área, a nível mundial, foi também um contributo importante para a melhor compreensão do ambiente externo.

Um agradecimento sincero aos amigos e colegas que se disponibilizaram para testar o sistema desenvolvido, dando um contributo muito importante para a componente de avaliação, a esse mesmo sistema.

Por fim, mas de forma alguma menos importante, o agradecimento à minha família mais próxima, que em determinadas alturas se viu privada da minha presença, junto dos vários elementos que a constituem. A eles um obrigado e um pedido de desculpas.

Abstract

The Interactive Voice Response systems (IVR) [1] associated potential is big [11] [13] [15] [16], once it is possible to optimise processes, combining existing technology, either in recognition or in synthesis. It acknowledge of a specific area of action, in order to create solutions to ensure the best possible action from the technology.

Link Consulting is a company that has developed several solutions in the area of voice systems, not only in a different context of what is involved, also with different technologies. It is a reality that once working with people with experience in this area naturally enriches the result of the work.

This document describes the work done, develop a system of voice response, enabling patients to a particular health institution and on this specific case of Social Services Medical Assistance (SAMS), make appointments, using the phone (or mobile) or any application of Voice Over Internet Protocol (VoIP), through voice, or using the keyboard of these devices or applications.

As an innovation associated with this work, besides the technology and what it can withdraw from, compared to others in the market, it describes a rules system, easily configured, these rules can be applied over the working flow regarding this application.

Based on the importance associated with each of the rules, it contributes to the final decision, associated with each of the different phases that already exists in the flow, allowing the optimisation of the functioning base, not just taking advantage of information retained in historical, but also the configuration system so as to take even better advantage of the system in the future, either regarding the time or the success of the work time factor, or factor in the success rate in recognition.

Keywords

Voice Portal [29], Automatic Speech Recognition, Speech Synthesis, Dialogue Definition Techniques [30], Voice User Interface [19] [21], VoiceXML [24], SALT [23], Windows Workflow Foundation, Dual Tone Multi-Frequency, Voice over Internet Protocol.

Resumo

A potencialidade associada a sistemas de Interactive Voice Response (IVR) [1] é elevada [11] [13] [15] [16], uma vez que é possível optimizar processos, conjugando a tecnologia existente, quer no reconhecimento, quer na síntese, e o conhecimento de uma determinada área de actuação, que permita criar soluções que tirem o melhor partido possível da tecnologia.

A Link Consulting é uma empresa que conta com várias soluções desenvolvidas, na área dos sistema de voz, não só em contextos diferentes do que se encontra associado a este trabalho, mas também com tecnologias diferentes. O facto de trabalhar com pessoas experientes na área, enriquece naturalmente o resultado final deste trabalho.

O presente documento relata o trabalho levado a cabo, no sentido de desenvolver um sistema de atendimento automático, que permita aos pacientes de uma determinada instituição ligada à saúde, para o caso específico do Serviço de Assistência Médica Social (SAMS), efectuarem marcação de consultas, recorrendo ao telefone (fixo ou móvel) ou a qualquer aplicação de Voice Over Internet Protocol (VoIP), através da fala, ou fazendo uso do teclado desses mesmos dispositivos ou aplicações.

Como inovação associada a este trabalho, para além da tecnologia e do que dela se pode retirar, face às restantes existentes no mercado, encontra-se descrito um sistema de regras, facilmente configurável, regras essas a aplicar ao longo do fluxo de execução da aplicação.

Com base no peso associado a cada uma das regras, estas contribuem para uma decisão final, impreterivelmente associada a cada um dos vários estados existentes no já referido fluxo. Permitem optimizar o funcionamento base, não apenas tirando partido de informação retida em históricos, mas também configurando o sistema, de forma a tirar cada vez melhor partido do mesmo ao longo do tempo, quer ao nível do factor tempo, quer ao nível do factor taxa de sucesso em reconhecimentos.

Palavras Chave

Portal de Voz [29], Reconhecimento Automático de Fala, Síntese de Fala, Técnicas de Definição de Diálogo [30], Interface de Voz [19] [21], VoiceXML [24], SALT [23], Windows Workflow Foundation, Dual Tone Multi-Frequency, Voz sobre Protocolo de Internet.

Índice

Índice de Figuras

Índice de Tabelas

Definições e Acrónimos

SAMS – Serviço de Assistência Médica Social

DTMF – Dual Tone Multi-Frequency

IVR – Interactive voice response

ASR – Automatic speech recognition

TTS – Text to speech

WWF – Windows Workflow Foundation

OCS – Office Communications Server

IP – Internet Protocol

VoIP – Voice Over IP

OOG – Out Of Grammar

1. Introdução

O termo *Voice User Interface* (VUI) é utilizado para classificar a ferramenta que permite de forma fácil, criar uma interacção natural entre homem e máquina. Associadas à interacção, podem estar acções ou eventos, que permitam iniciar, terminar ou controlar diversos processos, com base no decorrer dessa mesma interacção, de forma automatizada.

A tecnologia de definição e suporte a este género de sistemas, já existe há algum tempo, no entanto, pode ser considerado um difícil desafio, definir e implementar uma VUI que tenha uma boa aceitação por parte dos utilizadores.

A sensação associada à típica expressão de "falar para uma máquina", é definitivamente algo que não agrada a uma significativa parte dos utilizadores. Pode ocorrer, aliar-se a essa prévia sensação, uma má experiência dos contactos estabelecidos com este tipo de sistemas, devido a falhas de implementação, que podem levar a não reconhecimentos, falsos reconhecimentos ou ainda situações de confusão por parte do utilizador, devido ao facto de o diálogo não o guiar devidamente.

Com tudo isto, chega-se à conclusão que este tipo de sistemas não tem sucesso significativo, devido ao facto de as pessoas terem receio deste género de sistemas, mas também devido ao facto de nem sempre, as situações às quais se pretende aplicar um sistema com uma interface de voz, serem devidamente estudadas, no sentido de precaver situações de dúvida ou incerteza, quer para o sistema, quer para o utilizador, que culminam impreterivelmente num mau final, para uma experiência que supostamente deveria ajudar e beneficiar, ambas as partes envolvidas.

Mesmo com a qualidade elevada atribuída às plataformas, quer de suporte, quer de desenvolvimento deste tipo de sistemas, é necessário ter a noção que é imprescindível, uma vez que se visa atingir o sucesso, reunir diversos tipos de valências, não apenas da área tecnológica, mas também de área linguística e até mesmo de área psicológica.

É essencial ter a noção do contexto no qual será aplicado um sistema desta natureza, se num ambiente com utilizadores que farão um uso muito especifico e que de certa forma, conhecem o modo como o sistema funciona, ou se por outro lado, os utilizadores estão a lidar com algo desconhecido, pelo menos aos primeiros contactos, requerendo assim de ajuda e de algo que os guie na realização de uma determinada tarefa.

No âmbito deste trabalho, são estudadas as tecnologias existentes, que uma vez aliadas ao conhecimento do contexto de utilização, adquirido por profissionais de uma determinada área de actuação, culminam numa proposta de solução, para o problema que é a implementação de um sistema IVR. O sistema deve permitir aos utilizadores/pacientes do SAMS, sendo estes mais ou menos experientes, efectuarem a marcação de consultas médicas, através do telefone (fixo ou móvel) ou através da utilização de qualquer aplicação de *VoIP*, fazendo uso da sua voz ou das teclas existentes nos equipamentos/aplicações.

Como optimização ao sistema que soluciona o problema base, tenciona-se que exista um mecanismo que vise não só, optimizar o funcionamento normal do sistema, em termos de tempo e taxa de sucesso associada aos reconhecimentos, com base em informação passada associada a cada utilizador, mas também a possibilidade de fazer parametrizações a esse mesmo mecanismo, no sentido de melhorar progressivamente a performance do sistema, na sua globalidade.

Considera-se importante permitir efectuar parametrizações, quer ao mecanismo de regras, quer a alguns parâmetros relacionados com o funcionamento geral do sistema, no entanto essa parametrização deve puder ser efectuada por alguém sem conhecimentos aprofundados de tecnologia. Neste sentido, deve ser utilizada uma interface amigável, que permita a configuração dos já referidos parâmetros.

Este trabalho visa estender o sistema que a Link Consulting implementou nesta instituição, no sentido de conceder uma alternativa ao sistema de marcação de consultas por e-mail e via Web browser.

2. Estado da arte

2.1 Tecnologia Voice Portal (Interactive Voice Response)

Um *Voice Portal*, também designado por *Interactive Voice Response*, é um sistema que permite o acesso por parte de um utilizador, a informação localizada na internet através de, entre outras possibilidades, uma interface de telefone. São utilizadas tecnologias *[22]*, descritas em maior detalhe nas seguintes secções, como *Automatic Speech Recognition* (ASR) ou *Text to Speech* (TTS), no sentido de permitir que o utilizador consulte, modifique ou crie novos conteúdos, através de comandos de voz ou através da utilização de teclas, com uso de *Dual Tone Multi-Frequency* (DTMF).

Um dos conceitos inerentes ao *Voice Portal*, é o facto de ser possível aos utilizadores acederem, em qualquer local e a qualquer hora, aos conteúdos e serviços disponibilizados pelo sistema, independentemente de os utilizadores utilizarem telefones típicos (redes móveis ou fixas) ou utilizarem o seu computador, através de um *Voice Browser* [1] [2] ou qualquer outra aplicação de comunicação *Voice over Internet Protocol* (VoIP).

2.1.1 Acesso ao Voice Portal

Apenas se torna possível aceder a um *Voice Portal* através de redes telefónicas, devido à utilização de interfaces telefónicas. O principal propósito destas interfaces é permitir aos utilizadores, ligar para um *Voice Portal* através do recurso a um telefone normal, faça este parte de uma rede móvel ou de uma rede fixa. Podem existir funcionalidades adicionais, como identificação do número que efectua a chamada ou transferência de chamadas.

É a interface telefónica que efectua todos os processamentos necessários, no sentido de garantir que a informação é enviada ao serviço nos formatos correctos, assim como que toda a informação de retorno do mesmo serviço é devidamente enviada ao utilizador.

2.1.2 Tecnologias de diálogo

Um elemento chave para o desenvolvimento de um *Voice Portal* é a componente que envolve as tecnologias utilizadas no diálogo, a qual actualmente se divide em três módulos: *Text to Speech (TTS)*, *Automatic Speech Recognition (ASR)* e *Speech Authentication (Speaker Recognition)*. De seguida serão focados alguns pontos relativamente a cada um destes módulos.

2.1.2.1 Text To Speech (TTS – Speech Synthesis)

Falar em *Text To Speech*, é o mesmo que falar em Síntese de Fala (*Speech Synthesis*), que consiste na conversão de uma entrada de texto, para uma saída de áudio, gerando automaticamente voz sintetizada. Computacionalmente, o processo de síntese é mais leve que o processo de reconhecimento [3]. Os sistemas e soluções que existem actualmente e que permitem fazer síntese de fala com qualidade razoável e inteligível, não conseguem ainda assim introduzir uma grau de naturalidade a essa voz sintetizada, que possa ser considerado aceitável.

Alguns exemplos concretos da utilização desta tecnologia, são as aplicações de *Email-to-Voice* e *Document-to-Voice*, ou outro género de aplicações, que serão referenciadas mais à frente.

2.1.2.2 Automatic Speech Recognition (ASR)

Reconhecimento automático de fala, é o processo através do qual um sistema faz o mapeamento de um sinal de voz para texto. Existem vários tipos de sistemas de reconhecimento de fala, nomeadamente os que são dependentes do utilizador (*Speaker Dependent*), sendo desenvolvidos para serem utilizados por alguém em específico; os que são independentes do utilizador, prontos para utilização por vários utilizadores distintos (*Speaker Independent*); e os que são adaptáveis (*Speaker Adaptive*), desenvolvidos com o intuito de permitir aperfeiçoamento e adaptação, através de treino.

Os sistemas também podem ser caracterizados com base na dimensão do seu vocabulário, classificados como *Small* (dezenas), *Medium* (centenas), *Large* (milhares) e *Very Large* (dezenas de milhar) e com base no tipo de operação realizada, sendo *Continuous*, quando o sistema opera para uma sequência da palavras ligadas entre si, ou *Isolated Word*, quando o sistema opera com base em uma única palavra em cada instante de tempo, com a necessidade de existência de uma pausa entre diferentes palavras [4].

Um sistema de *Voice Portal* implica que exista independência face ao utilizador (*Speaker Independent*), no entanto, pode operar mediante simples comandos de voz (*Isolated Word*) ou com base num conjunto de palavras interligadas entre si (*Continuous*). A dimensão do vocabulário utilizado, depende do contexto da aplicação em causa.

Problemas no reconhecimento *[16] [20]*

Muito embora a tecnologia de reconhecimento de fala tenha vindo progressivamente a ser melhorada, continuam a existir problemas para os quais a tecnologia existente ainda não apresenta qualquer solução. O facto de as pessoas se deslocarem entre diferentes ambientes, onde pode existir mais ou menos ruído de fundo, de forma constante ou com variações ao longo do tempo, é um dos problemas existentes. Paralelamente aos ambientes,

existe o problema da diferente forma de falar, de pessoa para pessoa, nomeadamente ao nível das pronuncias ou dialectos, quando demasiado acentuados, ou para a mesma pessoa, em diferentes períodos de tempo. Existe também o problema das pessoas, que devido a hábitos existentes, utilizam a conjugação de duas palavras, por vezes até de línguas diferentes. A juntar aos já referidos, existe o problema da ligação com os dispositivos, que pode levar a interpretações erradas da entrada fornecida pelo utilizador, devido a ruídos que possam existir na respectiva ligação.

2.1.2.3 Speech Authentication (Speaker Recognition)

Autenticação por fala, ou reconhecimento do utilizador que fala, é um processo de reconhecimento automático de quem está a falar, com base em informação individual incluída nos sinais de voz, que caracteriza de forma única, cada individuo. O processo de *Speaker Recognition* pode ser divido em duas partes, *Speaker Identification* e *Speaker Verification*.

Na primeira fase, o utilizador identifica-se de entre a totalidade de utilizadores conhecidos pelo sistema. Na segunda fase, essa identificação é aceite ou rejeitada, mediante o facto de o utilizador ser ou não, quem realmente diz ser, após análise da informação existente no sinal de voz registado no momento, face aos registos anteriormente guardados, relativamente a esse mesmo individuo [5].

2.2 Interface de diálogo com utilizador

2.2.1 Aplicação – Utilizador

A interface de utilizador é apresentada recorrendo à utilização de *prompts* áudio gravadas anteriormente, em ficheiros de áudio ou através do recurso a voz sintetizada (TTS). Por norma, a utilização de voz sintetizada tem vindo a reduzir, devido à falta de naturalidade existente nas actuais soluções. No entanto, com as evoluções significativas registadas ao nível da qualidade, a sua utilização permite atribuir uma maior flexibilidade na escolha dos mecanismos a utilizar nas aplicações, uma vez que se pode inclusivamente conjugar as diversas opções, com *prompts* pré-gravadas e TTS. Desta forma, deve-se tirar partido da flexibilidade existente, caso isso seja considerado uma mais valia, face à aplicação em causa.

A definição de workflows, com Windows Workflow Foundation numa óptica de *Managed Code* ou Voice Extensible Markup Language (VoiceXML) e Speech Application Language Tags (SALT), numa óptica de desenvolvimento *Web-Based Code*, são os mecanismos abordados neste documento, relativamente à criação e gestão de diálogos, devido ao facto de serem os mais utilizados, para o desenvolvimento deste tipo de sistemas.

Nas figuras seguintes são mostradas as abordagens possíveis de adoptar no desenvolvimento de aplicações de voz, nomeadamente *Managed Code* e *Web-Based Code* [6].

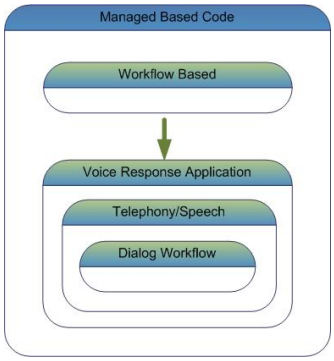

Figura 1 – Modelo de desenvolvimento Managed Based Code.

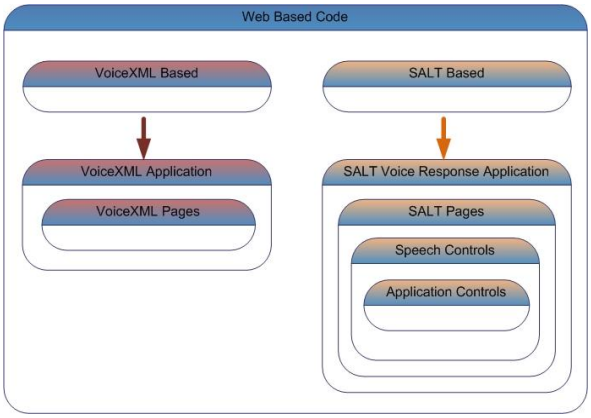

Figura 2 – Modelo de desenvolvimento Web Based Code.

Seguidamente será efectuada uma comparação entre os dois modelos de desenvolvimento, com posterior descrição das diferentes tecnologias de suporte à sua implementação.

Face ao VoiceXML e ao SALT, será efectuada uma comparação entre as duas tecnologias, nomeadamente ao nível das topologias típicas adoptadas, da origem e das funcionalidades associadas.

2.2.2 Utilizador – Aplicação

Uma aplicação telefónica, por norma tem duas formas de obter informação do utilizador, sendo elas a utilização do teclado (0-9, #, *,..), com DTMF ou através de comandos de voz. Os comandos de voz, devido às evoluções ao nível do reconhecimento, tornam possível e mais natural a interacção com o utilizador, em detrimento da vantagem em termos de simplicidade de implementação de DMTF.

2.2.2.1 Managed Code Development Model

O conceito de *Managed-Code*, aplica-se devido à integração da tecnologia do Windows Workflow Foundation, neste caso com Speech Server, ambas propriedade tecnológica da Microsoft. Esta integração, permite aos programadores desenvolverem aplicações de voz, através da definição de workflows, que se executam num ambiente controlado. Os workflows suportam uma aproximação composta na criação de aplicações e permitem efectuar modelação de processos mais complexos.

As *Simple Activities* implementam a execução de tarefas específicas, ao passo que as *Composite Activities*, implementam a execução de uma ou mais subtarefas. As actividades representadas, podem ser executadas pelos utilizadores ou por funções do próprio sistema, em que estas podem ser utilizadas para efectuar o controlo de vários aspectos, nomeadamente o fluxo de execução da aplicação, a concorrência, a sincronização, o tratamento de excepções e eventuais interacções com outras aplicações.

O desenvolvimento de IVR's recorrendo à utilização de workflows, oferece alguns benefícios, de entre os quais se destacam os seguintes:

- o código fonte da aplicação encontra-se centralizado, podendo assim ser compilado num único *assembly*;
- as aplicações podem mais facilmente ser estendidas, porque é seguida uma abordagem de uma aplicação composta por diversos componentes;
- é mais fácil modificar as aplicações, porque estas são mais transparentes;
- aplicações simples, podem ser desenvolvidas rapidamente, mesmo por programadores que tenham pouca experiência em linguagens como o C# ou o Visual Basic, porque existe uma plataforma gráfica para o desenvolvimento das mesmas;
- os processos existentes na aplicação, as dependências face a dados externos e as interacções do utilizador com a aplicação, são mais fáceis de controlar, porque todo o código da aplicação se encontra no lado do servidor.

Windows Workflow Foundation

O Windows Workflow Foundation consiste numa estrutura extensível para o desenvolvimento de soluções de workflow numa plataforma Windows. O Windows Workflow Foundation fornece uma API e todas as ferramentas necessárias ao desenvolvimento e à execução de aplicações baseadas num fluxo de trabalho. É oferecido um modelo unificado, para a criação de soluções de ponta a ponta, que abranjam várias categorias de aplicações, incluindo fluxos de trabalho humanos e de sistema.

Uma das finalidades desta tecnologia, é promover a extensibilidade a todos os níveis. As soluções desenvolvidas com recurso a Windows Workflow Foundation, são compostas por componentes interligados, que têm suporte de código em Microsoft .NET e são executados numa *host application*.

Embora seja possível desenvolver um workflow recorrendo unicamente a código, é também possível fazê-lo de forma gráfica. Um workflow é um modelo de um processo humano ou de sistema, que é definido como um conjunto de actividades mapeadas. Uma actividade é uma etapa de um workflow e é a unidade de execução, reutilização e composição de um workflow.

O mapa de actividades expressa as regras, as acções, os estados e as suas relações. Uma vez compilado, o processo descrito no workflow desenvolvido, pode ser executado em qualquer máquina Windows, desde *console applications*, *Windows services*, *web sites ASP.Net* e *web services*.

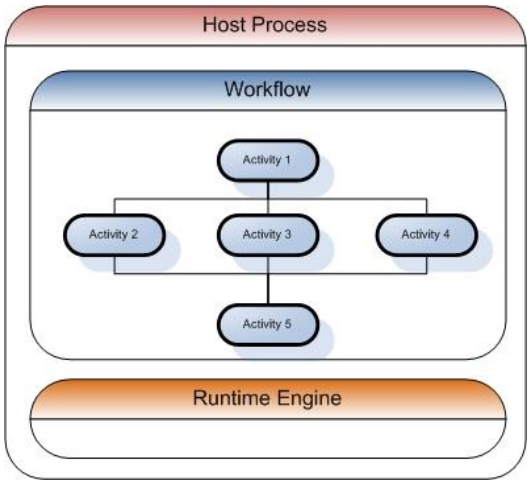

Figura 3 – Estrutura base de uma Workflow Application.

2.2.2.2 Web-Based Development Model

O modelo de desenvolvimento *web-based*, permite aos programadores o desenvolvimento fácil de IVR's, utilizando quer diálogos simples, quer diálogos mais complexos, existindo a possibilidade de recorrer a várias formas de preenchimentos de formulários e várias estratégias de confirmação.
Este modelo de aplicações é baseado num sistema distribuído, podendo assim ser dividido em duas camadas principais:

- camada lógica e de dados, que efectua o processamento das regras de negócio e guarda informação relativa aos acessos aos dados;

- uma camada de apresentação, que interage com as entradas de voz e DTMF, produzindo as saídas respectivas. Esta camada designa-se por *Voice User Interface* (VUI).

Quer SALT, quer VoiceXML, permitem fazer o desenvolvimento de aplicações que utilizam a voz como entrada ou saída. Foram desenvolvidas por diferentes consórcios, VoiceXML Forum e SALT Forum, e ambas contribuíram para a normalização dos *standars* de voz/fala por parte do W3C. *[26]*

Ambas as tecnologias, foram não só criadas com propósitos diferentes, mas também em fases diferentes do ciclo de vida da Web (*web life cycle*). O VoiceXML surgiu da necessidade de definir diálogos sobre a rede telefónica, por volta de 1999, quando muitos dos componentes da Web, tal como conhecidos hoje, ainda não se encontravam em tal estado de maturidade. O SALT surgiu da necessidade de permitir a criação de diálogos, para uma gama mais abrangente de equipamentos, desde Telefones, Telemóveis, PDA's e computadores, permitindo simultaneamente uma interface única de voz, ou uma interface multimodal (voz e gráfica) [10]. SALT foi concebida em 2002, quando algumas tecnologias chave da Web já se tinham estabelecido, das quais são exemplo eXtensible Markup Language (XML), *Document Object Model* (DOM), *XPath*, entre outros.

Arquitectura

A arquitectura base de qualquer sistema de suporte a *Voice Portal* é idêntica, mesmo quando se faz variar as tecnologias utilizadas. Ao contrário de qualquer servidor telefónico ou *Back-end Server*, que podem ter a necessidade de incorporar vários componentes, por exemplo, de acesso a documentos ou a bases de dados, uma plataforma servidor para aplicações de *Voice Portal*, é composta por três componentes essenciais, sendo eles um componente de ASR, um de TTS e uma plataforma de implementação, que se divide em vários módulos que controlam a chamada e a execução dos *script*s da aplicação de voz.

A Figura 4, ilustra de forma resumida a estrutura de uma plataforma gateway SALT.

Quando um utilizador executa uma aplicação SALT, o servidor web fará com que o script seja disponibilizado e executado pela *SALT Engine*, que utilizará gramáticas e outros servidores, eventualmente com ligação a bases de dados. A *TTS Engine* e a *ASR Engine*, podem ser invocadas caso isso seja necessário, nomeadamente no caso de se tratar de um IVR.

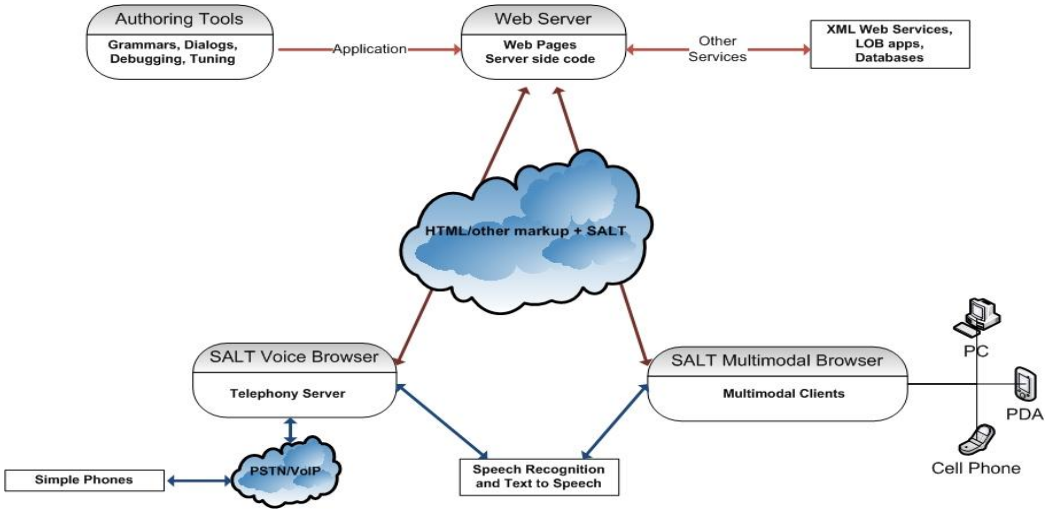

Figura 4 - Arquitectura base utilizando SALT.

Na Figura 5, é mostrada a arquitectura base para uma plataforma gateway VoiceXML. Tal como na arquitectura SALT, também será a *VoiceXML Engine* a responsável por disponibilizar os scripts a serem executados, quando uma chamada for efectuada. A plataforma gateway executará o *script* e invocará, caso seja necessário, a *TTS Engine*, a *ASR Engine*, ou outros componentes que existam, como áudio ou DTMF.

Os diferentes recursos comunicam através de mensagens, com uma API definida, sendo estas controladas pela *VoiceXML Engine.*

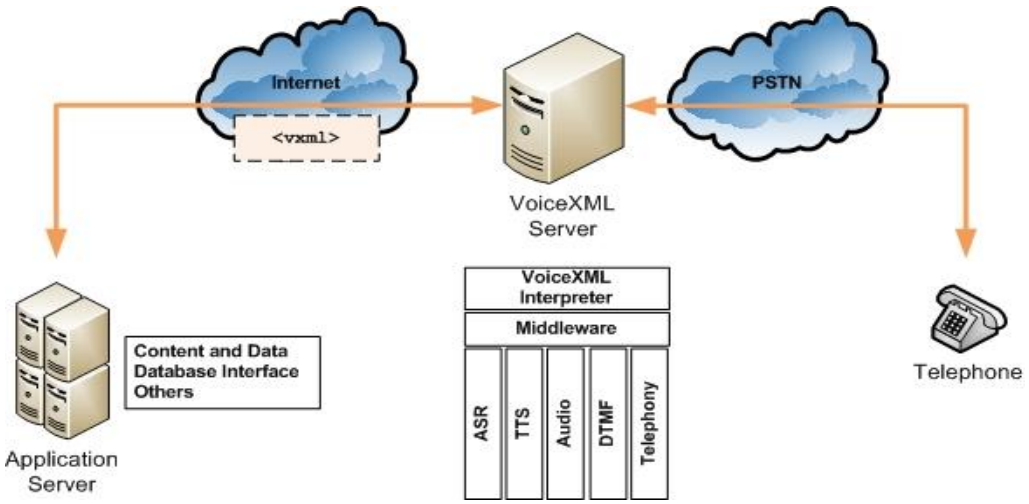

Figura 5 - Arquitectura base utilizando VoiceXML.

SALT

Origem

A 23 de Outubro de 2001, a Microsoft anunciou o seu suporte à principal tecnologia de reconhecimento de fala, o SALT [7]. A principal motivação que deu origem ao SALT, foi o facto de a Microsoft pretender permitir aos programadores, criarem *tags* para HTML, xHTML

e XML. O SALT, tal como foi anunciado, facilitaria a incorporação do uso de fala aos programadores web, sendo suportado em Internet Explorer, Pocket IE, ASP.net e Visual Studio.

Dia 24 de Outubro do mesmo ano, após o forte impulso introduzido pela Microsoft, outras empresas juntaram-se e isso levou à criação do SALT Forum, foram elas a Cisco, a Comverse, a Intel, a Philips e a SpeechWorks. O principal objectivo era desenvolver uma *Speech Markup Language* para introduzir noutras *Markup Languages*. Este grupo manifestou a sua intenção de desenvolver uma ferramenta livre, independente de qualquer plataforma, que tornasse possível o acesso à informação, aplicações e serviços, através de interfaces multimodais, com origem num computador, num telefone, num tablet PC, num PDA ou em qualquer outro dispositivo [8].

A versão 1.0 da especificação SALT, foi disponibilizada a 17 de Julho de 2002.

Funcionalidades

SALT é, tal como VoiceXML, uma *Speech Interface Markup Language*. Consiste num conjunto de elementos XML, com atributos associados, propriedades DOM, eventos e métodos, que permitem aplicar uma interface de voz às páginas web. O SALT pode ser utilizado com HTML, xHTML, entre outros standards, para desenvolver interfaces de voz (*voice-only*) ou aplicações multimodais.

O SALT segue uma abordagem de orientação a objectos, na qual algumas acções são iniciadas, perante a ocorrência de determinado tipo de eventos.

Com a utilização de SALT, é possível inserir valores em *form*'s através de comandos de voz e accionar os *input field*'s, para fazer com que as acções devidas sejam executadas.

O SALT é essencialmente desenhado para aplicações web, no entanto, pode perfeitamente ser utilizado num servidor telefónico, efectuando controlo de chamadas.

A Figura 6, representa um script em SALT, cujo comportamento pretendido é efectuar a reserva de bilhetes de avião. As *tags* de *input* são standard em qualquer código HTML, no entanto, o script SALT embebido, permite obter esses valores de entrada através de comandos de voz, fornecidos pelo utilizador. A gramática também é utilizada para os dados de entrada utilizados pelo *script*.

```
<input type="button" onclick="recoFromTo.Start()" value="Say From and To Cities" />

<input name="txtBoxOriginCity" type="text" />
<input name="txtBoxDestCity"   type="text" />

<salt:listen id="recoFromTo">
    <salt:grammar src="FromToCity.xml" />
    <salt:bind targetElement="txtBoxOriginCity" value="/result/originCity" />
    <salt:bind targetElement="txtBoxDestCity" value="/result/destCity" />
 </salt:listen>
```

Figura 6 – Exemplo de um script SALT.

VoiceXML

Origem

O *VoiceXML* [9] foi criado pela AT&T Bell Laboratories, surgindo da investigação num projecto designado PhoneWeb. Quando em 1996 a AT&T e a Lucent se separaram, continuaram em diferentes caminhos a investigação na área de uma *Phone Markup Language*.

A AT&T desenvolveu uma plataforma robusta de suporte a uma *Phone Markup Language*, utilizada na altura para o desenvolvimento de diversos tipos de aplicações. Desde então, aumentou o interesse pelo conceito de ser possível utilizar uma *Markup Language* standard, para aceder a aplicações baseadas na web, através de comandos de voz.

Em busca de soluções para a criação de interacções de fala de forma fácil, a Motorola entrou também nesta indústria. Uma vez que a sua produtividade se focava num serviço móvel, demonstrou o seu interesse em investir num acesso de mão livres. Isto significaria que os utilizadores estariam aptos a utilizar a voz, paralelamente às teclas, para efectuar determinado tipo de operações.

O VoiceXML é um produto do VoiceXML Forum, um consórcio de empresas, de entre as quais AT&T, IBM, Lucent (actualmente Alcatel-Lucent) e Motorola.

O forum foi criado com o intuito de desenvolver e promover os standards necessários que permitissem potenciar o desenvolvimento de aplicações baseadas na voz. Em Agosto de 1999, o forum disponibilizou a versão 0.9 da sua especificação, em Março de 2000 a versão 1.0. Cerca de 6 meses depois foi anunciada a versão 2.0, tendo saído em Julho de 2007 a recomendação para a versão 2.1.

Funcionalidades

O VoiceXML foi projectado para criar diálogos, recorrendo a áudio sintetizado, a reconhecimento de fala e reconhecimento DTMF. O seu maior objectivo é transportar as vantagens associadas ao desenvolvimento Web, para as aplicações que visam estabelecimento de diálogo interactivo.

Desde a especificação da versão 1.0, o VoiceXML tem-se desenvolvido de forma significativa, tornando-se uma linguagem de *scripting* bastante estável. De entre o vasto grupo de aplicações que podem usufruir deste tipo de tecnologia, é possível referir aplicações de compra de bilhetes de avião, ou aplicações mais complexas, envolvendo transferências e controlo de chamadas.

A Figura 7, mostra um exemplo de código em VoiceXML que permite efectuar a reserva de bilhetes de avião. As duas *tags* "*field*", representam as entradas esperadas pela aplicação, representando a origem e o destino, na viagem que o utilizador pretende reservar. O resultado da interacção será uma acção de *submit*, cujas entradas fornecidas pelo utilizador, serão processadas pelo ficheiro "*processTicket.jsp*".

```
<?xml version=”1.0”?>
<vxml version=”2.0”>
  <form>
    <field name=”from”>
       <prompt> Where would you like to fly from? </prompt>
           <grammar src=”from_cities.xml”/>
           <filled>
                You said <value expr=”from” />
           </filled>
    </field>
    <field name=”to”>
       <prompt> Where would you like to fly to? </prompt>
           <grammar src=”to_cities.xml”/>
           <filled>
               You said <value expr=”to” />
           </filled>
    </field>
    <submit next=”processTicket.jsp” namelist=”from to”/>
  </for</vxml>
```

Figura 7 – Exemplo de um script VoiceXML.

2.3 Actualmente

Actualmente, existem diversos tipos de aplicações, que se enquadram em contextos muito específicos, nos quais muitas vezes, nem existe processamento de escrita.

Alguns exemplos, são as aplicações de notícias, nas quais a navegação entre categorias pode ser feita através de comandos de voz, ou recorrendo a DTMF.

Um serviço idêntico, é o serviço de informação de trânsito, o qual fornece informação ao utilizador que efectua a chamada, com base em dados actualizados em alguma página de internet, utilizada como fonte.

Serviços mais complexos também já existem, como permitir efectuar compras, preenchendo um carrinho de compras, sendo que no final do processo, serão recebidos os produtos encomendados em casa. Tal como nos exemplos referidos anteriormente, a navegação entre categorias e subcategorias, pode ser feita através de comandos de voz ou, tal como maioritariamente acontece, através de DTMF.

Independentemente da tecnologia utilizada, existem actualmente aplicações com um certo grau de complexidade, que permitem desta forma às pessoas, tirar partido de um determinado serviço.

Uma das principais vantagens deste tipo de aplicações, é permitir usufruir de um determinado serviço, independentemente da localização, do equipamento a utilizar e com requisitos de tempo, aquando o acesso ao respectivo serviço.

Desta forma, o principal entrave à evolução de forma mais massificada deste tipo de soluções, seja qual for o mercado ao qual se destinem, é de facto o receio por parte das pessoas em lidar com um sistema desta natureza.

2.4 Soluções Tecnológicas

Na Tabela 1, são apresentadas algumas possibilidades relativamente ao fornecimento de toda a tecnologia necessária, incluindo os diversos componentes, considerados imprescindíveis para o desenvolvimento de um *Voice Portal*. De entre as várias possibilidades existentes no mercado, são apenas avaliadas as que suportam reconhecimento para Português (Portugal).

Como exemplo de critérios a ter em consideração, na escolha definitiva de uma solução, são apresentados os seguintes:

- facilidade de integração dos diversos componentes necessários ao funcionamento do sistema (ASR, TTS, etc..);
- suporte para língua portuguesa, com a melhor percentagem possível ao nível dos reconhecimentos;
- apoio e colaboração durante e após o desenvolvimento;
- requisitos de software ou hardware necessários para o funcionamento do sistema;
- não ser dependente do utilizador e permitir efectuar melhoramentos ao nível do reconhecimento;
- robustez face ao ruído;

Tabela 1 – Tecnologias de suporte.

	ASR	TTS	Voice Gateway
Loquendo	X	X	X
Nuance	X	X	X
Microsoft (Speech Server 2007)	X	X	X
Philips (SpeechMagic)	X		

2.5 Solução Adoptada

Tendo em conta os critérios referidos anteriormente, que deveriam ser tidos em consideração, aquando a escolha de uma solução, que permitisse efectuar o desenvolvimento de um sistema de *Voice Portal*, assim como, as possibilidades existentes quanto à criação e gestão de diálogos [13], abordadas anteriormente, enumeram-se de seguida alguns objectivos que se pretendem ver satisfeitos, de forma genérica:

- facilidade de integração entre todos os componentes necessários, nomeadamente ASR e TTS;

- garantia de uma percentagem elevada de sucesso ao nível do reconhecimento, neste caso, com língua portuguesa;
- possibilidade de desenvolver uma solução escalável, que ao mesmo tempo possa ser parametrizada, com o menor número de implicações possível, na estrutura já existente do sistema;

De entre os vários fornecedores, sobre os quais se fez levantamento de características, de modo a que fosse possível comparar de forma genérica as várias soluções existentes, retiram-se algumas conclusões.

O facto de a solução a adoptar permitir o uso de TTS, é uma mais valia, pois possibilita que sejam conjugadas *prompts* previamente gravadas, com o mecanismo de TTS da solução.

Desta forma, a melhor solução, uma vez que permite satisfazer todos os objectivos estabelecidos, passa pela adopção da plataforma da Microsoft, nomeadamente Microsoft Speech Server 2007, a qual contém de forma totalmente integrada, todos os componentes necessários.

Relativamente à forma como se definem e gerem os diálogos, a melhor solução passa pela adopção da solução da Microsoft, nomeadamente Windows Workflow Foundation. Esta componente, embora seja proprietária, permite a utilização/integração das outras duas possibilidades existentes, ao nível da criação e gestão de diálogos, SALT e VoiceXML.

A adopção de uma solução de *managed code programming model*, face a uma *web-based programming model*, oferece algumas vantagens, das quais é exemplo o facto de não ser necessário a existência de código do lado do cliente, logo a gestão é mais facilmente efectuada, pois tudo é feito do lado do servidor.

Para além destes pontos, os seguintes destacam-se igualmente, por serem uma mais valia na adopção de uma solução baseada em *managed code*, pesando assim na escolha tomada:

- flexibilidade de programação;
- maior facilidade na activação e gestão do fluxo;
- capacidade de sair do fluxo de execução (interligação com outros sistemas).

Flexibilidade de Programação

A utilização de uma abordagem *managed code*, concede a possibilidade de fazer o *wrapping* de componentes de *voice response* em actividades, que não só permite fazer uma abstracção sobre o código que é executado, mas também permite reutilizar esses mesmos componentes.

Desta forma, o workflow pode ser visto como uma estrutura hierárquica de actividades. Actividades estas que podem ser elementos simples (primitivas), ou tornar-se *containers* de mais actividades.

A camada de *Dialog Activities* da API do Speech Server, disponibiliza uma série de classes de objectos primitivos, de entre os quais se encontram os seguintes:

- *QuestionAnswerActivity* – execução de pergunta resposta;
- *StatementActivity* – execução de comando único;
- *RecordAudioActivity* – execução de uma *prompt*, com gravação da resposta do utilizador.

Para além destas actividades, outras são disponibilizadas, com o intuito de permitir efectuar um certo nível de abstracção, face às primitivas de voice response, e simultaneamente tornar possível a reutilização de componentes, ao longo do fluxo de execução.

Maior facilidade na activação e gestão do fluxo

Cada *container* faz a gestão do diálogo entre os diversos componentes com contém. Quando um workflow é activado, a ordem de execução das actividades é a mesma pela qual estas se encontram definidas. Após a execução de uma actividade, terá inicio uma próxima, e assim sucessivamente, até não existirem mais actividades. O fluxo pode ser dinâmico, com uma condução semântica, ou processual, como um modelo de uma máquina de estados finita.

Capacidade de sair do fluxo de execução

Alguns cenários de diálogos requerem que o fluxo seja interrompido, activando de certa forma um *switch* para uma tarefa diferente na aplicação. Os comandos invocados pelos utilizadores são um bom exemplo disso.

No SALT, os comandos são tratados pelo fluxo principal da aplicação. Se o utilizador resolver interromper o diálogo, este terá de ser retomado onde foi interrompido. Numa situação destas, em que o programador concede a possibilidade de o utilizador interromper, recomeçando ou voltando atrás no fluxo, são necessários cuidados adicionais, nomeadamente com os valores das variáveis, que podem já ter sido actualizadas.

Usando workflows, o programador pode criar uma aplicação, que em caso de interrupção, solicitada por parte do utilizador, esta possuirá um diálogo independente do anterior, com actividades, também elas independentes das do diálogo anterior. Tem ainda assim, a possibilidade de reiniciar todo o diálogo, com a garantia de que nenhumas variáveis guardam estados anteriores.

3. Especificação

3.1 Casos de Estudo

Em diversas situações, ao longo deste documento, são tomadas como exemplo um conjunto de tarefas, que permitam avaliar a resolução do problema descrito anteriormente, com diversos níveis de optimização, que se pretende que sejam atingidos.
Numa fase inicial, nomeadamente ao nível da especificação, foi realizado trabalho junto dos participantes e dos responsáveis por esta área de projectos, no sentido de levantar e modelar as actividades que são parte integrante do fluxo geral da aplicação, e por conseguinte, das tarefas que vão ser utilizadas como exemplo.

Tal como já foi referido anteriormente, até ao momento da realização deste trabalho, a tarefa de efectuar a marcação de uma consulta médica, para os utentes do SAMS, podia ser realizada de diversas formas:

- Balcão – o utente desloca-se fisicamente ao centro clínico, para o qual pretende a sua consulta, efectuando a marcação com um(a) dos(as) funcionários(as) presentes no local;
- Web Browser – o utente acede à área privada do Web site do SAMS, autentica-se e segue as instruções disponibilizadas, no sentido de marcar a sua consulta;
- E-mail – o utente envia um e-mail com toda a informação necessária relativamente à consulta que tenciona marcar, para um endereço destinado a este processo.

No sentido de perceber qual seria a melhor forma de direccionar o utente, ao longo da tarefa de efectuar a marcação de uma consulta no novo módulo disponível, foi estudada a aplicação de marcação de consultas via Web browser.
Foi possível perceber, que o utente se vê cingido a começar todo o processo de marcação de uma nova consulta, através da especialidade, sendo seguidamente guiado no sentido de se perceber qual o médico, local e horário pretendido, por esse mesmo utente.

Como situação de exemplo a testar nos próximos três sub-tópicos, consideremos o seguinte:

O contexto

- O utente com o número de beneficiário ***22222222***, com o nome **André Sabino**, tenciona efectuar a marcação de uma consulta, através do sistema de IVR disponível;
- O utente já referido, tenciona efectuar a marcação de uma consulta para a médica **Leonor Bento**, nomeadamente para a especialidade de **Cardiologia**;

- O utente tenciona ser consultado no **Hospital** central do Serviço de Assistência Médica Social;
- O utente tem preferência pelos **Horários** disponíveis da parte da **Manhã**, devido a razões não especificadas.

Informação (A)

- Nos últimos três meses, o utente consultou a médica **Leonor Bento**, mais do que qualquer outro médico(a);
- Nos últimos três meses, o utente foi consultado na especialidade de **Cardiologia**, mais de 3 vezes;
- Nos últimos três meses, o utente deslocou-se sempre ao **Hospital** Central do Serviço de Assistência Médica Social, para comparecer nas consultas;
- Em todas as consultas às quais o utente compareceu, todas elas decorreram da parte da **Manhã**.

Informação (B)

- O **Médico** da última consulta do utente, foi **Horácio Silva**;
- A **Especialidade** da consulta à qual o utente foi da última vez, foi **Audiometria**;
- Da última vez, o **Local** onde o utente foi consultado, foi o **Centro Clínico** do SAMS;
- A sua última consulta, decorreu num **Horário** da parte da **Manhã**.

Pode existir mais informação associada ao paciente em causa, no entanto, para efeitos de teste, apenas se refere **A** e **B**.

Seguidamente são abordados três exemplos, dos quais o primeiro representa todas as situações onde nada se sabe acerca do histórico informativo de um dado paciente, ou mesmo que se saiba, não se utiliza essa mesma informação. No segundo caso, são representadas todas as situações nas quais, com base no histórico de informação associado a um dado paciente, se consegue fazer uma previsão daquilo que eventualmente ele vai desejar na interacção que vai ter inicio, no entanto, a percentagem de certeza associada a essa mesma previsão não é demasiadamente elevada, que leve a formular uma única questão. No terceiro caso, são representadas todas as situações nas quais se faz uso do histórico informativo e se consegue uma percentagem de certeza relativamente elevada, face aquilo que provavelmente o paciente vai desejar realizar na interacção que vai ter inicio, permitindo assim, formular uma única questão.

3.1.1 Fluxo Base

Esta situação representa de forma genérica, todos os casos em que não é utilizado qualquer mecanismo, que vise optimizar o funcionamento geral do sistema base. O facto de não ser utilizado qualquer mecanismo, pode derivar de diversas razões:

- **Sistema foi configurado para não fazer uso desses mesmos mecanismos;**
- Não existe informação associada a um determinado utente (histórico), que permita tirar partido dos já referidos mecanismos.

O comportamento do sistema, quer se trate do primeiro, ou do segundo tópicos anteriormente referidos, é exactamente o mesmo.

Uma vez que foi referida, como exemplo a utilizar ao longo deste documento, uma situação em que existe informação associada a um determinado paciente/utente, vamos partir do principio, que neste caso, o sistema foi configurado para não fazer uso de quaisquer mecanismos que visem a optimização ao funcionamento base.

O fluxo base consiste na marcação de uma consulta, sendo necessário que o utente refira explicitamente todos os campos que são necessários à marcação dessa mesma consulta, nomeadamente **<médico>**, **<especialidade>**, **<local>** e **<horário>**.

O utente vai sendo questionado, acerca de cada um dos campos que são necessários, no sentido de se efectuar a marcação desejada.

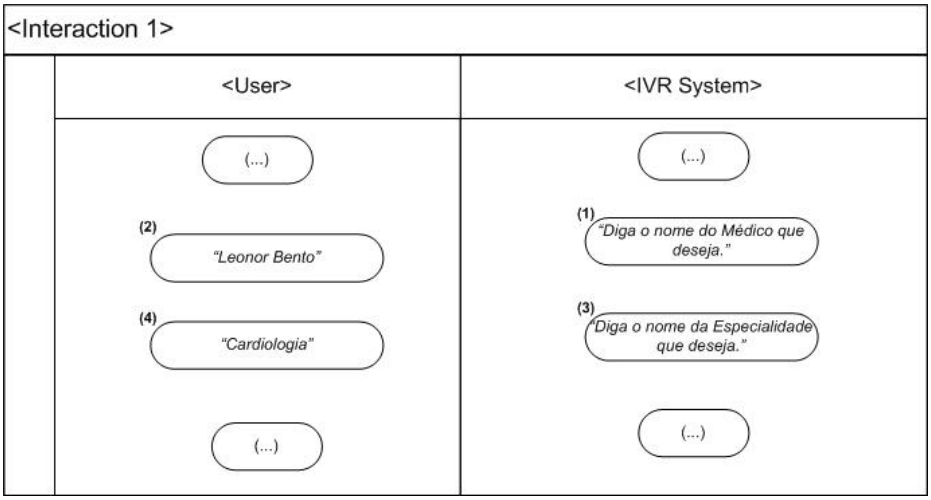

Figura 8 – Fluxo de Diálogo com Modo de Funcionamento Base.

Com base neste fluxo, o resultado final consiste na marcação de uma consulta, sem utilização de informação eventualmente associada a um determinado utente, sendo que é necessário, que seja o próprio utilizador a dizer todos os campos necessários à marcação de uma consulta, nomeadamente **<médico>**, **<especialidade>**, **<local>** e **<horário>**. Considera-se este caso, o mais simples da sequência de três, uma vez não é utilizado qualquer mecanismo, ou qualquer tipo de informação, que vise optimizar o funcionamento base do sistema.

3.1.2 Fluxo Optimizado 1

Esta situação representa de forma genérica, todos os casos em que é possível utilizar a informação armazenada ao longo dos tempos, associada a um determinado utilizador, tentando assim tirar partido dessa mesma informação, no sentido de optimizar o funcionamento base do sistema.
Ao passo que no ponto anterior, o sistema tinha sido configurado para não utilizar mecanismos de optimização, nesta situação, existe informação armazenada e essa informação vai ser utilizada ao longo da execução do sistema.
Associada ao utente em causa, existe informação que pode levar o sistema a optar por dois caminhos, sendo eles, nomeadamente:

- Tudo o que se passou nos últimos três meses;
- Tudo o que se passou da última vez.

Nesta situação, assume-se que perante as duas hipóteses, as quais possuem um peso idêntico ao nível da contribuição para a decisão final, o caminho **A** é o que prevalece, porque de certa forma, se considera que mais provavelmente que **B**, contribuirá para fazer as perguntas certas ao utente. Podia existir muito mais informação, para além da indicada.

Sabe-se que:

- o utente consultou a médica **Leonor Bento**, mais do que qualquer outro **Médico(a)**, desta forma, o sistema pode questioná-lo directamente, sobre a possibilidade de se efectuar uma nova marcação, para essa mesma **Médica**;
- o utente consultou a médica já referida, para a **Especialidade** de **Cardiologia**, mais de três vezes, nos últimos três meses, sendo assim, é possível sugerir uma nova consulta, para a mesma **Especialidade**;
- todas as consultas às quais o utente compareceu, decorreram no **Hospital** Central do SAMS, logo o sistema pode sugerir efectuar a marcação para o mesmo **Local**;
- todas as consultas às quais o utente compareceu, decorreram no **Horário** da **Manhã**, desta forma, o sistema pode questioná-lo sobre a possibilidade de efectuar a marcação para o próximo **Horário** disponível, que seja da parte da **Manhã**.

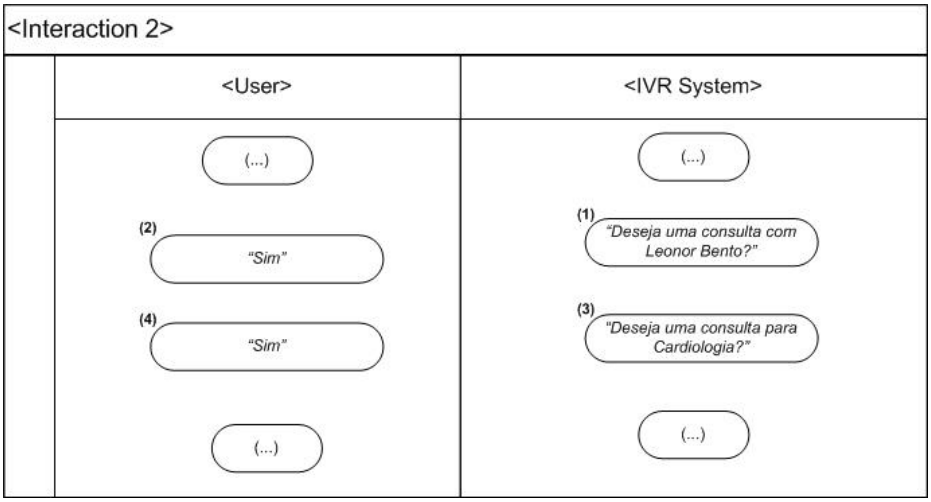

Figura 9 – Fluxo de Diálogo com Modo de Funcionamento Optimizado de nível 1.

Comparativamente ao fluxo anterior, é possível perceber que:

- em média, se consegue reduzir o tempo necessário à marcação de uma consulta médica, uma vez que o utilizador apenas necessita de confirmar aquilo que lhe é directamente questionado, independentemente de o número de estados se manter o mesmo;
- em média, se consegue aumentar a taxa de sucesso ao nível do reconhecimento, associada a cada um dos diversos estados do fluxo da aplicação, uma vez que o utilizador não diz o nome do Médico/Especialidade/Local/Horário, mas sim, confirma aquilo que lhe é sugerido.

3.1.3 Fluxo Optimizado 2

Tal como no caso anterior, toda a informação associada a um determinado utente, vai ser utilizada no sentido de optimizar o funcionamento geral do sistema.
Tal como já foi referido no tópico anterior, existe um conjunto de informação, que pode ajudar o sistema a optimizar diversos aspectos, nomeadamente o tempo necessário à realização da tarefa, mas também a taxa de sucesso ao nível dos reconhecimentos.
Ao passo que no tópico anterior, existiam duas hipóteses, em que uma delas se destacava ligeiramente da segunda, vamos assumir que nesta situação, **A** tem um peso significativamente superior a **B**, ou seja, considera-se que a informação **A** é muito mais provável de gerar as perguntas certas, face a **B**.

Sabe-se que:

- o utente consultou a médica **Leonor Bento**, mais do que qualquer outro **Médico(a)**;
- o utente consultou a médica já referida, para a **Especialidade** de **Cardiologia**, mais de três vezes, nos últimos três meses;

- todas as consultas às quais o utente compareceu, decorreram no **Hospital** Central do SAMS;
- todas as consultas às quais o utente compareceu, decorreram no **Horário** da **Manhã**;

Com base nesta informação, e tendo em conta a importância atribuída à mesma, o sistema pode optar por formular uma única pergunta ao utente, que junte toda a informação, que no tópico anterior foi confirmada separadamente e agora seria confirmada em conjunto.

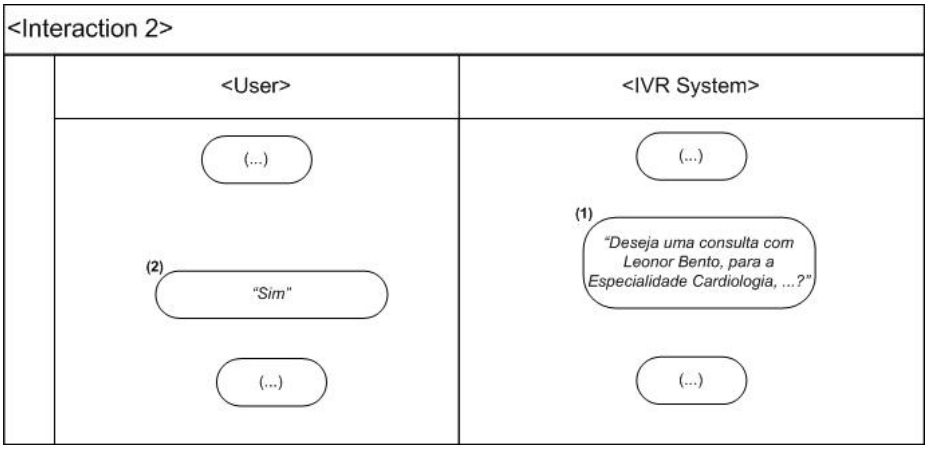

Figura 10 – Fluxo de Diálogo com Modo de Funcionamento Optimizado de nível 2.

Comparativamente ao fluxo anterior, é possível perceber que:

- em média, se consegue reduzir o tempo necessário à marcação de uma consulta médica, uma vez que o utilizador apenas necessita de responder a uma única pergunta, que com elevada probabilidade, corresponderá aquilo que o utente realmente necessita;
- em média, se consegue aumentar a taxa de sucesso ao nível do reconhecimento, associada a cada um dos diversos estados do fluxo da aplicação, uma vez que o utilizador não diz o nome do Médico/Especialidade/Local/Horário, mas sim, confirma aquilo que lhe é sugerido, neste caso, através de uma única questão.

3.2 Considerações (Normas e Metodologias)

Neste subcapítulo, são abordados alguns tópicos *[20] [27]* que devem ser levados em conta, aquando a definição e construção de diálogos. Face a cada um destes tópicos, são apontadas medidas relativas à implementação do já referido diálogo.

1. **Ter noção da capacidade de memorização humana**

Um dos principais factores, que deve ser levado em consideração na construção do diálogo, é precisamente o facto de que se está a lidar com pessoas, com capacidade de memorização limitada e que varia de utilizador para utilizador.
Torna-se desta forma necessário, minimizar a informação que é necessário o utilizador não esquecer.
Uma das formas de fazer com que a probabilidade de o utilizador se esquecer ou se perder, no diálogo com o sistema, diminua, pode ser então:

- Construir *prompts* pequenas;
- Não apresentar demasiadas opções em cada estado;
- Dar a possibilidade de ouvir novamente, ou pedir ajuda.

2. **Desenhar, tendo a noção de quem é o utilizador final**

Conhecer minimamente o universo de potenciais utilizadores do sistema, pode ajudar em muito no que respeita à definição do diálogo, sendo que directa ou indirectamente contribui para o sucesso do próprio sistema.
O sistema deve ser pensado, com base naquilo que os potenciais utilizadores esperam e não com base naquilo que aparentemente parece ser o mais correcto, para a equipa de desenvolvimento. Neste sentido, as pessoas devem ser questionadas sobre o que pensam ser o melhor, devendo também ser parte integrante, durante as fases de testes que se considerem necessárias, no sentido de atingir os objectivos especificados.
A utilização de palavras simples, fáceis de entender, ajuda o utilizador a perceber mais rapidamente aquilo que se pretende, e evita situações de erro ou confusão, por parte do utilizador.

3. **Deixar o utilizador conduzir a conversação**

Dar ao utilizador, o controlo da evolução do fluxo de execução da aplicação o mais cedo possível, não só pode contribuir para o factor “satisfação”, como também permite perceber o mais cedo possível, aquilo que o utilizador tenciona fazer.
A utilização de perguntas abertas, é um exemplo, no entanto, é necessário ter em conta as limitações da tecnologia em utilização e os riscos associados à utilização de perguntas, que permitam o utilizador agir como bem entender.

4. **Avançar sempre no fluxo**

Uma vez que um dos factores críticos neste tipo de sistemas, é o tempo necessário à realização das tarefas em causa, o diálogo deve ser desenhado de forma a evitar o mais

possível, retrocessos no fluxo de execução, a não ser quando explicitamente solicitado, nomeadamente pelo utilizador.
A utilização de mensagens de erro explicativas, é uma forma de prevenir situações de confusão por parte do utilizador.
Um sistema de ajuda que permita ao utilizador, saber a cada momento, aquilo que é dele esperado por parte do sistema, é um ponto igualmente importante na construção do diálogo.
O sistema deve conseguir oferecer ao utilizador, diversas formas de obter informação necessária para avançar no fluxo de execução, mesmo que as mensagens de erro e o sistema de ajuda, não tenham sido suficientes para elucidar devidamente o utilizador. A passagem a um operador ou a reformulação da questão, podem ser caminhos a seguir.

5. Desenhar o diálogo com base no maior grupo com interesses comuns

Uma vez estudado e devidamente entendido o possível universo de utilizadores, o diálogo deve ser definido, com base naquilo que representa a melhor solução para a grande maioria desses utilizadores.
Neste sentido, o sistema deve ser desenhado, tendo em conta que não se deve encaminhar todos os utilizadores para um caminho, que apenas alguns beneficiam.
Devem ser tidas em consideração, as operações mais vezes realizadas, tirando eventualmente partido dessa informação, para dar a conhecer aos utilizador aquilo que é possível fazer através do sistema. Um exemplo concreto, pode ser, listar as opções de menu, com base no número de pedidos por parte dos utilizadores, à realização de cada uma das acções, associadas às opções listadas. Este tipo de funcionamento, pode ser dinâmico em vez de estático, evoluindo com o passar do tempo, adaptando-se às circunstâncias existentes numa determinada altura.

6. Deixar o sistema fazer algum processamento

Tal como deve ser tido o maior cuidado possível, na definição do diálogo que melhor satisfaça as necessidades reais do sistema, deve igualmente ser tido cuidado com todos os mecanismos que, directa ou indirectamente, permitem melhorar a performance geral sistema.

6.1 Fluxo de Execução

Uma vez definido o fluxo geral de execução da aplicação, pode existir a necessidade de tomar decisões, que independentemente de variarem consoante os utilizadores, não devem dar origem a mais uma questão a formular a esse mesmo utilizador, e mesmo que isso seja realmente necessário, o resultado dessa interacção deve ser guardado.

É desta forma de evitar, questionar o utilizador sobre um determinado tema, mais que uma vez, assim como obter informação directamente do utilizador, quando esta podia ser obtida por outros meios, sem que o utilizador tivesse de intervir directamente.

6.2 Sistema de Reconhecimento

A utilização dos níveis de confiança associados a cada reconhecimento, em conjunto com as hipóteses associadas a cada um desses reconhecimentos, permite melhorar a performance do sistema em geral, para os casos em que o sistema não tem a certeza, acerca daquilo que foi dito pelo utilizador.
Directamente ligado ao conceito de nível de confiança e hipóteses de reconhecimento, está o conceito de gramática. A gramática é então composta por todas as hipóteses de resposta possíveis, para um determinado estado do fluxo de execução.
As hipóteses, constituem todas as respostas que o sistema considerou como possíveis, após ter reconhecido aquilo que foi dito pelo utilizador. O nível de confiança, encontra-se associado a cada uma dessas hipóteses, permitindo assim listar de forma crescente ou decrescente, aquelas que o sistema considera mais prováveis de terem sido realmente ditas, em detrimento das restantes.

Neste sentido, devem ser tidos em conta diversos pontos, aquando a criação das gramáticas:

- Devem conter apenas o essencial;
- A boa definição das *prompt's,* deve levar o utilizador a dizer os termos que fazem parte das gramáticas;
- As respostas não reconhecidas, ou seja, cujos termos não fazem parte das gramáticas utilizadas (Out Of Grammar - OOG), devem ser tratadas de forma cuidada, no sentido de tentar reencaminhar o utilizador e fazer com que este utilize os termos que fazem parte da gramática, nomeadamente com *prompt's* mais especificas, ou com sistemas de ajuda devidamente explícitos.

Com base nos seis tópicos anteriores, pressupõe-se que sejam tidos diversos cuidados, ao nível do desenvolvimento deste tipo de aplicações, nomeadamente relacionados com o desenho do diálogo.

- Tentar atingir o equilíbrio, entre aquilo que são as necessidade de negócio e aquilo que representa a experiência dos potenciais utilizadores, conciliando ambos os pontos, para que seja possível um meio termo, através de alguma flexibilidade;
- Realizar vários testes de utilização, sempre que possível, no sentido de garantir o mais possível, a boa definição do diálogo do sistema em causa;
- Utilizar uma versão piloto, para fazer testes reais com utilizadores, no sentido de perceber qual o real comportamento dos utilizadores e efectuar os devidos ajustes,

sempre que tal se verifique necessário, devido a situações de erro ou confusão por parte dos utilizadores.

3.3 Visão Geral

Funcionalidade

A aplicação visa permitir efectuar a marcação de consultas médicas, neste caso concreto, no SAMS. Toda a informação necessária, é recolhida ao longo de um fluxo de execução da já referida aplicação, de modo a que seja possível formular uma questão final, com base no Médico, Especialidade, Local e Horário, necessariamente a confirmar pelo utilizador/paciente.

Recursos

As opções de resposta estarão limitadas à gramática utilizada, desde Médicos, Especialidades, Locais, Horários e todas as gramáticas adicionais, necessárias ao correcto funcionamento do sistema.
O utilizador será guiado ao longo da aplicação, com perguntas que visem obter da melhor forma possível, a informação necessária à criação de um novo registo de consulta.

Utilizadores

Os utilizadores do sistema, serão todos os utentes do SAMS, independentemente de ser ou não a primeira vez, que usam o sistema de marcação.

Fluxo de diálogo geral

Após serem dadas as boas vindas ao utilizador, por parte do sistema, serão solicitados, quer o número de beneficiário, quer o código pessoal, para que possa ser efectuada a validação desses mesmos dados. Caso exista erro na validação, a interacção terminará, após mensagem de despedida e após se terem esgotado as três tentativas concedidas a cada utilizador. Caso contrário, são listados os comandos aos quais o utilizador pode recorrer, ao longo da execução da aplicação. De seguida, é dada ao utilizador a hipótese de efectuar a marcação por médico ou especialidade, dizendo o nome do médico ou da especialidade. Tendo sido obtida informação relativa ao médico e à especialidade, obter-se-á informação quanto ao local e posteriormente quanto ao horário. Com a formulação da questão final, é solicitado ao utilizador que confirme, com base em toda a informação recolhida até então. Com sucesso, será efectuado um novo registo, em caso de falha, serão dadas as hipóteses de o utilizador voltar ao início ou terminar a sua interacção com o sistema.

Exemplo de diálogo

(Anexo A)

3.4 Especificação Funcional

3.4.1 Arquitectura do Sistema

Na Figura 11, é mostrada uma possível estrutura de rede, com acesso a uma plataforma de *Voice Portal*. A plataforma pode servir clientes com acesso via internet, através de um computador ou outro dispositivo móvel, mas também telefones móveis e telefones fixos, através da rede dos respectivos operadores, móveis ou fixos, respectivamente.

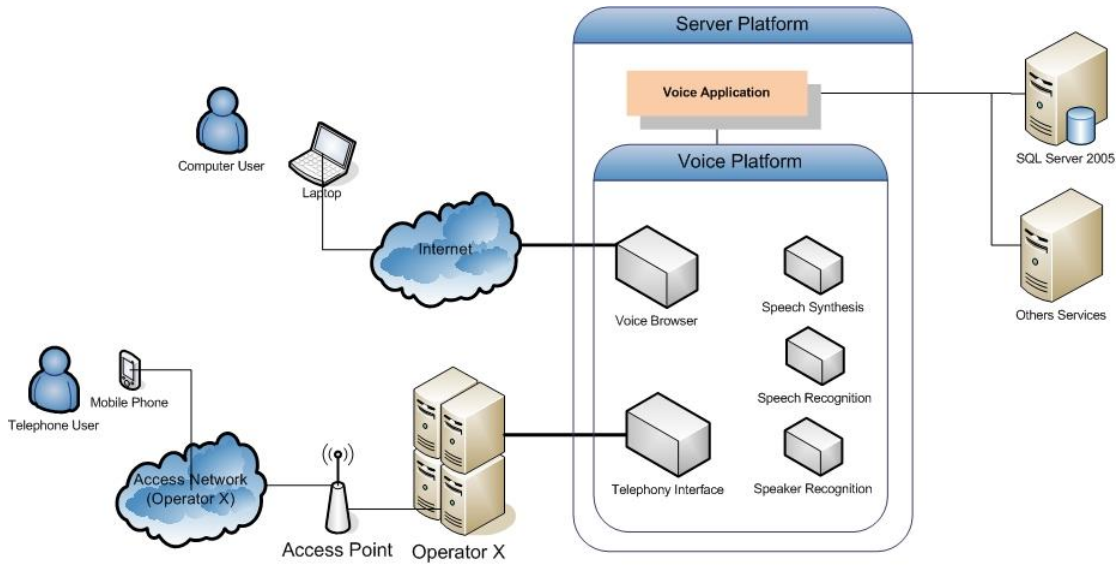

Figura 11 - Arquitectura para um sistema de Voice Portal.

Tal como já foi referido, os três módulos base da plataforma de voz, são *Speech Synthesis*, *Speech Recognition [12] [18]* e *Speaker Recognition*, sobre os quais funcionam outros dois módulos, que permitem o acesso por parte dos diversos tipos de clientes, *Voice Browser* e *Telephony Interface*.

A plataforma de voz, pode ser vista como *gateway,* que permite ligar os dispositivos telefónicos à internet, servindo de ponte entre ambientes diferentes.
O *Voice Browser* é um componente chave numa plataforma de voz, sendo um factor diferenciador face aos típicos IVR's. O *Voice Browser* faz a gestão dos diálogos com os utilizadores e obtém toda a informação necessária ao longo desse mesmo diálogo, quer seja de web services, bases de dados, ou qualquer outra fonte de informação possível.

3.5 Especificação de Diálogo

3.5.1 Fluxo de Diálogo

Notação Utilizada

A notação utilizada ao longo deste documento, é a representada na figura 12.
Uma vez que a Link Consuling tem alguns projectos realizados na área dos IVR's, embora com tecnologias diferentes e também para contextos diferentes, foi aconselhada a utilização de toda a notação anteriormente utilizada, nos já referidos projectos.
Com base em documentos, que constituíram parte da formação dada aos profissionais da Link Consulting, nomeadamente pela Nuance, foram retiradas notas e seguidas essas mesmas linhas de elaboração deste tipo de documentos.

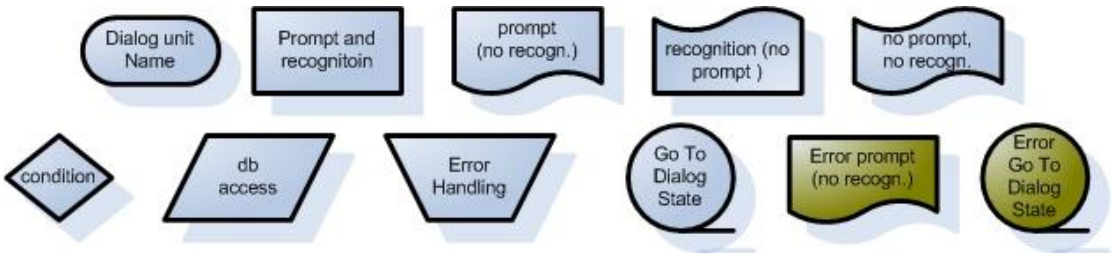

Figura 12 – Terminologia utilizada na Especificação de Diálogo.

Fluxos

Os restantes fluxos, em falta neste subcapítulo, encontram-se definidos no Anexo B – Fluxos de Execução.

InitialFase

Fase inicial do fluxo de execução geral. Nesta fase, são dadas as boas vindas ao utilizador/paciente, são listadas todas as opções/comandos que o paciente pode invocar a qualquer altura, ao longo da sua interacção com o sistema, nomeadamente pedir para *Repetir* instruções, para *Cancelar* uma opção tomada anteriormente, para solicitar *Ajuda* ou ainda, para *Terminar* completamente a interacção com o sistema.
O número de beneficiário e o código pessoal são solicitados ao paciente, sendo feita a sua validação. Caso a validação seja efectuada com sucesso, o processo seguirá normalmente, no sentido de efectuar a marcação da consulta, caso contrário, o paciente será notificado de que ocorreu um erro de autenticação e o processo terminará após a mensagem de despedida. São concedidas três oportunidades de autenticação a cada utilizador/paciente.

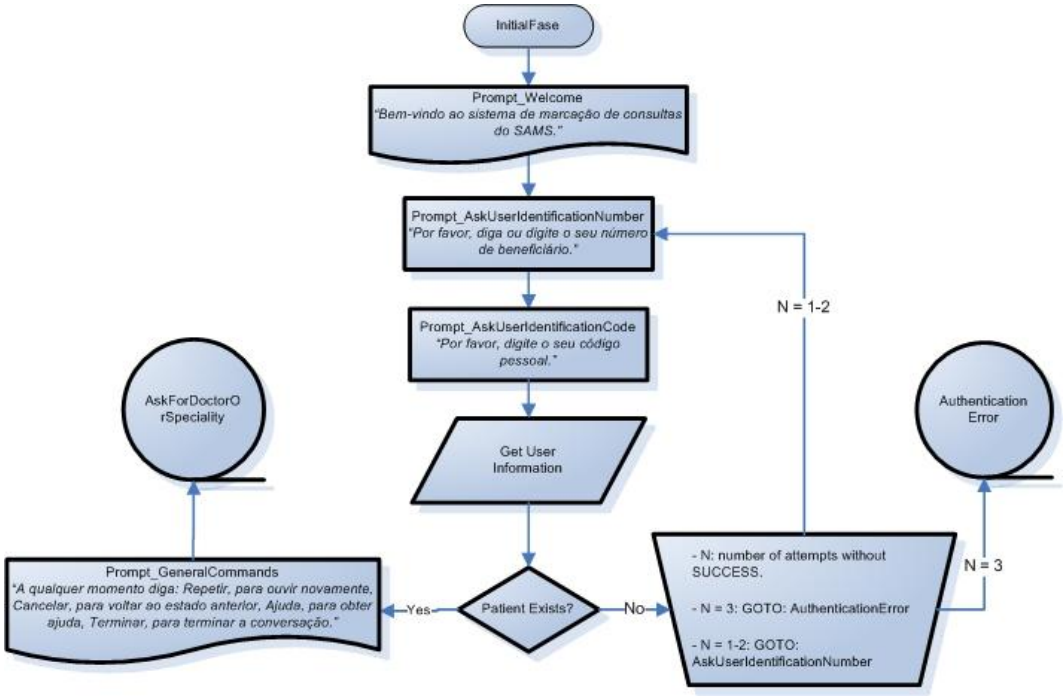

Figura 13 – Fase Inicial do fluxo de execução. Verificação da Identidade do Utilizador.

AskForDoctorOrSpeciality

A validação dos dados é efectuada com sucesso, sendo listadas as opções. O paciente pode escolher efectuar a marcação começando por indicar o nome do Médico ou o nome da Especialidade.

- Optando por Médico, serão obtidas as Especialidades desse Médico. No caso de não existirem Especialidades, o paciente será notificado da situação, no caso de serem mais que uma, estas serão listadas e será pedido ao paciente que diga a que pretende, no caso de ser apenas uma, o processo seguirá para a selecção da Local.
- Optando por Especialidade, serão obtidos os Médicos dessa Especialidade. No caso de não existirem Médicos, o paciente será notificado da situação, no caso de serem mais que um, estes serão listados e será pedido ao paciente que diga o que pretende, no caso de ser apenas um, o processo seguirá para a selecção da Local.

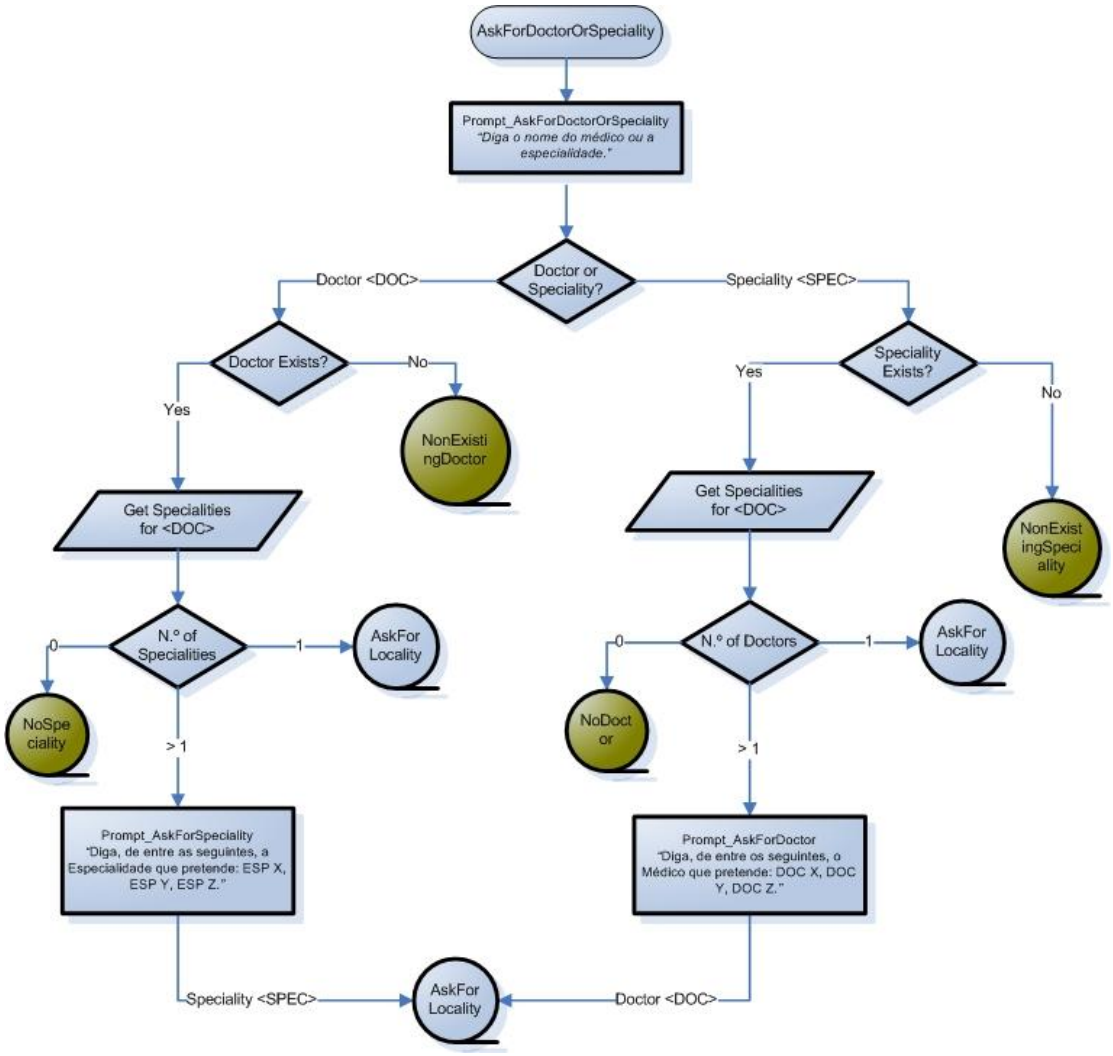

Figura 14 – Obtenção de informação relativa ao Médico e à Especialidade.

AskForLocality

Obter informação relativamente ao Local, para a qual se pretende efectuar a marcação da consulta, tendo em conta que já se sabe qual o Médico e qual a Especialidade. São obtidos todos os locais onde o Médico escolhido, dê consultas da Especialidade escolhida. No caso de não existirem Locais, o paciente será notificado da situação, no caso de serem mais que um, estes serão listados e será pedido ao paciente que diga o que pretende, no caso de ser apenas um, o processo seguirá para a selecção do Horário.

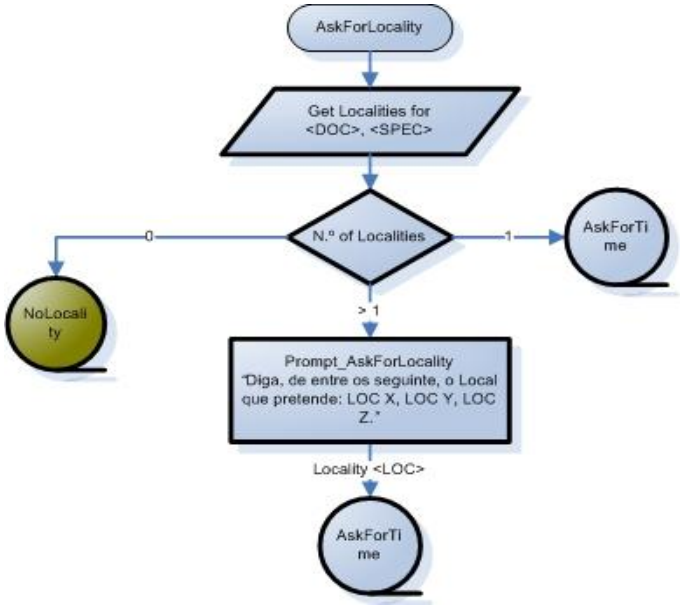

Figura 15 – Obtenção de informação relativa ao Local.

AskFinalQuestion

Formulação da questão final, a apresentar ao utilizador. Solicitação de confirmação por parte do utilizador, face aos dados recolhidos até então. Caso a confirmação seja efectuada com sucesso, será registada uma nova marcação, caso contrário, o utilizador será questionado sobre a hipótese de efectuar uma nova marcação. Em caso afirmativo, o processo voltará ao início, onde é dada a hipótese de escolha entre Médico ou Especialidade, caso contrário, o processo termina.

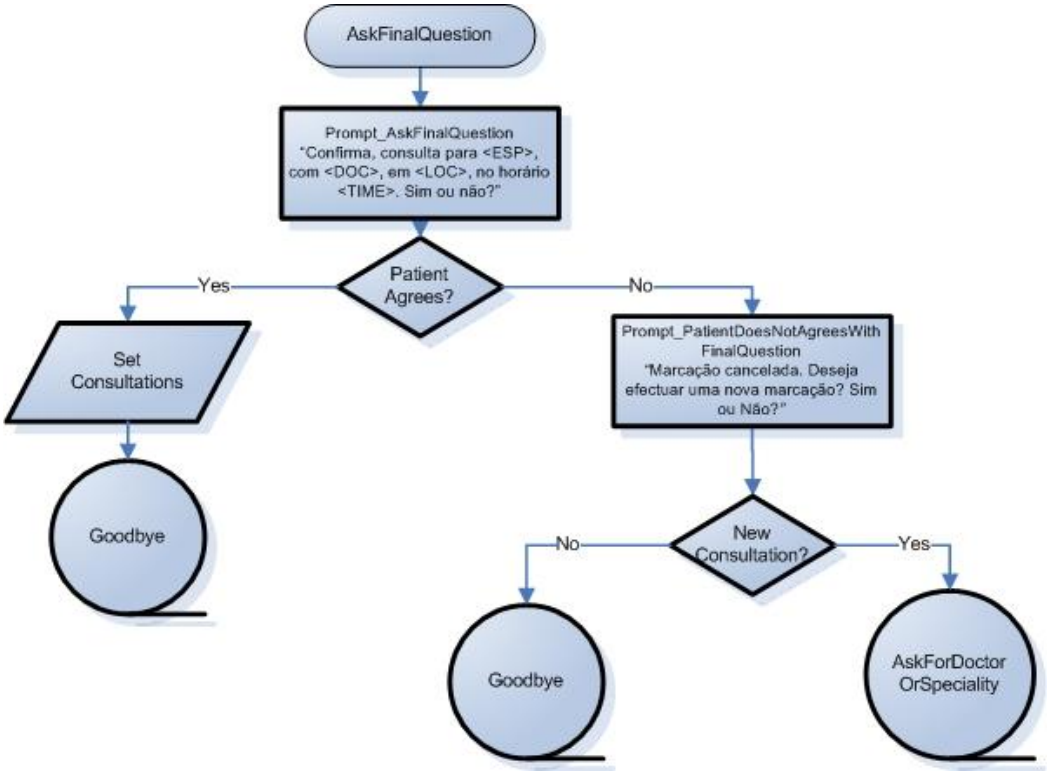

Figura 16 – Processo de confirmação de toda a informação recolhida até à altura.

3.5.2 Comandos Universais

Comandos Universais

Conjunto de comandos, que estão disponíveis ao longo de todo o fluxo de execução, que podem servir para o utilizador ouvir de novo uma determinada mensagem, para cancelar, voltando ao estado anterior, para obter ajuda, consoante o estado em que se encontre, ou para terminar a interacção com o sistema.

Na tabela seguinte, encontram-se alguns exemplos de expressões que podem ser utilizadas.

Tabela 2 – Comandos Universais.

Expressões de Exemplo	NL Slots, Sample Values
"repita (por favor)"	<**command** repeat > repeat current prompt
"repetir (por favor)"	<**command** repeat > repeat current prompt
"ouvir (novamente) (por favor)"	<**command** repeat > repeat current prompt
“(quero) cancelar (por favor)”	<**command** cancel> go to start prompt
“(quero) voltar (por favor)”	<**command** cancel> go to start prompt
“início”	<**command** cancel> go to menu prompt
“recomeçar”	<**command** cancel> go to menu prompt
“principio”	<**command** cancel> go to menu prompt
"(quero) ajuda (por favor)"	<**command** help > context sensitive help
"opções (por favor)"	<**command** help > context sensitive help
"ajuda-me"	<**command** help > context sensitive help
"(quero) terminar (por favor)"	<**command** exit / goodbye > finish interaction
"adeus" / “sair” / “terminar”	<**command** exit / goodbye > finish interaction

3.5.3 Estados de Diálogo

Os restantes estados de diálogo, em falta neste subcapítulo, encontram-se definidos no Anexo D – Estados de Diálogo.

GeneralPrompts

Conjunto de prompts de um género especifico (*silence, repeat*), que podem ser reutilizadas ao longo do fluxo de execução, sem haver a necessidade de definir cada uma delas, para cada estado do respectivo fluxo.

Tabela 3 – Welcome.

Name	*Prompt_Welcome*
Description	*Mensagem de boas vindas ao sistema de marcação de consultas.*
Prompts	*Prompt_Welcome.MainPrompt → “Bem-vindo ao sistema de marcação de consultas do SAMS.”*
Grammars	-
Universal Commands	*Repeat → Prompt_Welcome_Repeat* *Cancel → Prompt_Welcome_Cancel* *Help → Prompt_Welcome_Help* *Goodbye → GOTO: “Prompt_Goodbye”*

Tabela 4 – GeneralCommands.

Name	*Prompt_GeneralCommands*
Description	*Listagem dos comandos, que podem ser ditos a qualquer momento, ao longo do fluxo de execução.*
Prompts	*Prompt_GeneralCommands.MainPrompt → "A qualquer momento diga: Repetir, para ouvir novamente, Cancelar, para voltar ao estado anterior, Ajuda, para obter ajuda, Terminar, para terminar a conversação."*
Grammars	*-*
Universal Commands	*Repeat → Prompt_GeneralCommands_Repeat* *Cancel → Prompt_GeneralCommands_Cancel* *Help → Prompt_GeneralCommands_Help* *Goodbye → GOTO: "Prompt_Goodbye"*

Tabela 5 – AskUserIdentificationNumber.

Name	*Prompt_AskUserIdentificationNumber*
Description	*Solicitação do número de beneficiário ao utente.*
Prompts	*Prompt_AskUserIdentificationNumber.MainPrompt → "Por favor, diga ou digite o seu número de beneficiário."* *Prompt_AskUserIdentificationNumber.SilencePrompt → GeneralPrompts.Silence* *Prompt_AskUserIdentificationNumber.NoRecognitionPrompt → "Formato inválido."*
Grammars	*Voice, DTMF*
Universal Commands	*Repeat → Prompt_AskUserIdentificationNumber_Repeat* *→ GOTO: "Prompt_AskUserIdentificationNumber"* *Cancel → Prompt_AskUserIdentificationNumber_Cancel* *→ GOTO: "Prompt_Goodbye"* *Help → Prompt_AskUserIdentificationNumber_Help* *→ "Ajuda: Por favor, diga ou digite o seu número de beneficiário, composto por 8 dígitos. Um digito de cada vez."* *→ GOTO: "Prompt_AskUserIdentificationNumber"* *Goodbye → GOTO: "Prompt_Goodbye"*

Tabela 6 – AskForDoctorOrSpeciality.

Name	*Prompt_AskForDoctorOrSpeciality*
Description	*Listagem das opções permitidas pelo sistema de marcação de consultas.*
Prompts	*Prompt_AskForDoctorOrSpeciality.MainPrompt → "Diga o nome do médico ou a especialidade."* *Prompt_AskForDoctorOrSpeciality.SilencePrompt → GeneralPrompts.Silence* *Prompt_AskForDoctorOrSpeciality.NoRecognitionPrompt* *→ "O nome que indicou não foi reconhecido."* *→ "Não entendi."*
Grammars	*Voice*
Universal Commands	*Repeat → Prompt_AskForDoctorOrSpeciality_Repeat* *→ GOTO: "Prompt_AskForDoctorOrSpeciality"* *Cancel → Prompt_AskForDoctorOrSpeciality_Cancel* *→ GOTO: "Prompt_Goodbye"* *Help → Prompt_AskForDoctorOrSpeciality_Help* *→ "Por favor, diga o nome do Médico se pretender efectuar uma marcação por Médico, ou o nome da Especialidade, se pretender efectuar uma marcação por*

	Especialidade." *→ GOTO: "Prompt_AskForDoctorOrSpeciality"* *Goodbye → GeneralPrompts.Goodbye* *→ GOTO: "Prompt_Goodbye"*

Tabela 7 – AskFinalQuestion.

Name	*Prompt_AskFinalQuestion*
Description	*Formular a questão final, solicitando confirmação da parte do utente.*
Prompts	*Prompt_AskFinalQuestion.MainPrompt → "Confirma, consulta para <ESP>, com <DOC>, em <LOC>, no horário <TIME>. Sim ou Não."* *Prompt_AskFinalQuestion.SilencePrompt → GeneralPrompts.Silence* *Prompt_AskFinalQuestion.NoRecognitionPrompt* *→ "A sua resposta não foi reconhecida."* *→ "Não entendi."*
Grammars	*Voice*
Universal Commands	*Repeat → Prompt_AskFinalQuestion_Repeat* *→ GOTO: "Prompt_AskFinalQuestion"* *Cancel → Prompt_AskFinalQuestion_Cancel* *→ GOTO: "Prompt_AskForTime"* *Help → Prompt_AskFinalQuestion_Help* *→ "Por favor, diga SIM, se confirma com a questão formulada, NÃO, se não confirma com a questão formulada.* *→ GOTO: "Prompt_AskFinalQuestion"* *Goodbye → GOTO: "Prompt_Goodbye"*

Tabela 8 – PatientDoesNotAgrees.

Name	*Prompt_PatientDoesNotAgrees*
Description	*Indicar ao paciente, que o processo foi interrompido.*
Prompts	*Prompt_PatientDoesNotAgrees.MainPrompt → "Marcação Cancelada. Deseja efectuar uma nova marcação? Sim ou Não?"*
Grammars	*Voice*
Universal Commands	*Repeat → Prompt_PatientDoesNotAgrees_Repeat* *Cancel → Prompt_PatientDoesNotAgrees_Cancel* *Help → Prompt_PatientDoesNotAgrees_Help* *Goodbye → Prompt_PatientDoesNotAgrees_Goodbye* *GOTO: "Prompt_AskForDoctorOrSpeciality"*

3.5.4 Modelo de Dados

O sistema actualmente em utilização no SAMS, que permite a marcação de consultas de diversas formas (web e e-mail), disponibiliza web services, que é suposto puderem ser utilizados em situações de teste reais, pela aplicação desenvolvida.

Devido ao risco associado à utilização de web services, que actuam sobre informação real, optou-se por replicar parte dessa mesma informação, utilizada pelo sistema actualmente em

funcionamento, escolhendo-se bases de dados, para efectuar o armazenamento dos dados replicados.

Estruturalmente, a aplicação permite que a fonte de dados seja alterada, neste caso de bases de dados, para web services, sendo que todo o comportamento do sistema se mantém inalterado, uma vez que foi utilizado o padrão de desenho *Strategy*. Desta forma, é possível devolver à camada de aplicação, os valores de que esta necessita, sabendo que isso acarreta procedimentos distintos, estando a actuar sobre uma base de dados, sobre web services ou sobre qualquer outra fonte de informação.

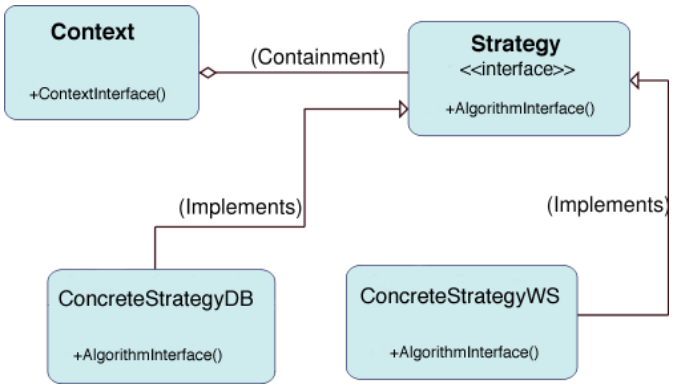

Figura 17 – Padrão de desenho Strategy.

Seguidamente, é efectuada uma pequena descrição, acerca de cada uma das tabelas necessárias para armazenamento de informação, que possa ser utilizada como dados de teste ao sistema em estudo.

Tabelas Base

- ***Doctor*** – Contém informação sobre todos os Médicos, existentes no sistema de marcação de consultas;
- ***Locality*** – Contém informação sobre todos os Locais, para os quais existe a possibilidade de efectuar marcações de consultas;
- ***Patient*** – Contém informação sobre todos os Pacientes, para os quais é admissível a marcação de consultas, pelo facto de estes se encontrarem registados no sistema;
- ***Speciality*** – Contém informação sobre todas as Especialidades, para as quais é possível efectuar marcação de consultas;
- ***Rules*** – Contém a configuração de todas as regras definidas no sistema, independentemente de estarem ou não a ser utilizadas pelo mesmo, uma vez que podem estar activas ou inactivas.

Tabelas de Relação

- ***Consultations*** – Contém informação sobre todas as consultas marcadas, para efeitos de optimização do processo de marcação de consultas e é onde serão registadas novas marcações. Efectua a relação entre Paciente, Médico, Especialidade e Local, registando

também a data da consulta;

- ***DoctorLocalities*** – Contém informação que permite relacionar Médicos e Locais, no sentido de perceber onde um Médico se encontra registado, mas também, quais os Médicos que se encontram registados num determinado Local;
- ***DoctorSpecialities*** – Contém informação que permite relacionar Médicos e Especialidades, no sentido de perceber quais as Especialidades de um determinado Médico, mas também, quais os Médicos que dão consultas de uma determinada Especialidade;
- ***SpecialityLocalities*** – Contém informação que permite relacionar Especialidades e Locais, no sentido de perceber quais os Locais onde se podem efectuar marcações para uma determinada Especialidade, mas também, quais as Especialidades que existem em cada um dos Locais.

3.6 Mecanismo de Regras

O mecanismo de regras, cuja implementação foi proposta, consiste em algo que permita incorporar optimizações ao fluxo base da aplicação, com base em regras que podem ser aplicadas ao longo das diversas fases desse mesmo fluxo. Este mecanismo, deve permitir o sistema funcionar na sua versão base, ou em versão optimizada, sendo desta forma, parametrizável a este e a outros níveis.
A introdução deste mecanismo, deve funcionar como a incorporação de um novo componente à versão base do sistema, implicando assim, poucas alteração estruturais na aplicação.
Como requisito associado a este mecanismo, considera-se a possibilidade de parametrizar essas mesmas regras, tornando desta forma possível, tornar o sistema geral de marcação de consultas, em algo adaptativo, uma vez que passa a ser possível influenciar directamente o funcionamento do sistema, através do mecanismo de regras.

Os principais factores, que se visam optimizar com a incorporação de um mecanismo de regras ao sistema base, são:

- **O Tempo necessário à realização da tarefa**

  Um dos principais pontos a optimizar, é precisamente o tempo que é necessário para efectuar a marcação de uma consulta, no sistema do SAMS.
  De entre os factores que levam à tentativa de satisfazer este objectivo, estão questões monetárias, ou até mesmo, questões associadas à disponibilidade, não só dos utilizadores, mas também do sistema em si.

- **A Taxa de sucesso associada aos reconhecimentos**

Outro factor essencial, que se visa atingir, é uma boa percentagem de sucesso, associada ao reconhecimento.
No sentido de satisfazer este requisito, tenta-se então aliar as boas práticas ao nível da definição de diálogos, com as possibilidades colocadas ao dispor, pela tecnologia utilizada.
A utilização de gramáticas, que contenham a informação essencial, é um dos caminhos a seguir, sendo que pode ser útil, essas mesmas gramáticas, não terem sempre a mesma dimensão ou o mesmo conjunto de palavras, podendo assim ser designadas de gramáticas dinâmicas (*Dynamic Grammars*), uma vez que podem ser modificadas, ao longo do fluxo de execução da aplicação.

Partindo do exemplo do Anexo C, que consiste na aplicação de regras num contexto de pesquisa de serviços numa determinada zona, tente-se agora passar para o contexto associado a este trabalho em especifico.

Imagine-se um utente, que liga para o sistema de marcação de consultas, sendo que o sistema consegue saber que, com uma percentagem de certeza de 100%, o utente irá tentar efectuar a marcação de uma consulta com "*Leonor Bento*", podendo, tal como no exemplo em anexo, efectuar uma pergunta directa, "Deseja uma consulta com Leonor Bento?", à qual o utente responde "Sim". Numa situação como esta, pode não fazer sentido, associar à pergunta seguinte no fluxo de execução, "Qual a Especialidade que deseja?", todas as especialidades que existam no SAMS, mas apenas aquelas através das quais, pode ser estabelecida uma relação, com o médico (a), anteriormente confirmado.

O mecanismo de regras, é então aquilo que tornará possível ao sistema base, comportar-se de uma ou de outra forma, consoante a informação que é gerada por esse mesmo mecanismo.
O mecanismo de regras "sugere" qual será o melhor caminho a seguir, a partir do estado actual no fluxo de execução, não apenas com base num utente especifico, mas também com base na análise do histórico de informação, associado a esse mesmo utente.

Assumindo que, uma vez configurado o sistema para se fazer uso do mecanismo de regras, em adição ao funcionamento base do sistema, a primeira informação que se pretende confirmar com o utente é, qual o <Médico> ao qual este tenciona ir.
Associado ao paciente em causa, existe um histórico de informação, que indica todas as consultas, com os respectivos médicos e especialidades, locais e horários, associados a essas mesmas consultas.

Designe-se então por estado, cada um dos pontos ao longo do fluxo de execução da aplicação, onde se pretende obter "informação crítica", ou seja, <Médico>, <Especialidade>, <Local> ou <Horário>.

Cada regra encontra-se associada a um estado, dando assim indicação a esse mesmo estado, sobre qual pensa ser a hipótese mais provável, que vise satisfazer as necessidades do paciente.

Uma vez que podem existir várias regras associadas a um mesmo estado, é necessário atribuir pesos a cada uma delas, ou seja, qual a percentagem/peso atribuída à sua sugestão, quando esta for agrupada com todas as outras sugestões, para ser tomada uma decisão final, relativamente a um estado.

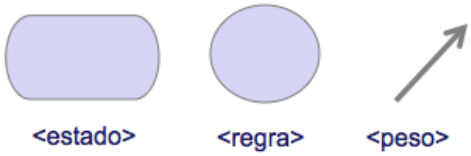

Figura 18 – Elementos base para exemplificar conceptualmente, o funcionamento do mecanismo de regras.

<estado>: representa uma fase especifica do fluxo de execução da aplicação, na qual existe um problema para resolver, ou seja, obter "informação crítica".

<regra>: processo que ajuda a resolver um determinado problema, associado a um <estado>.

<peso>: contribuição (%) de uma determinada <regra>, para a resolução de um problema, associado a um <estado>.

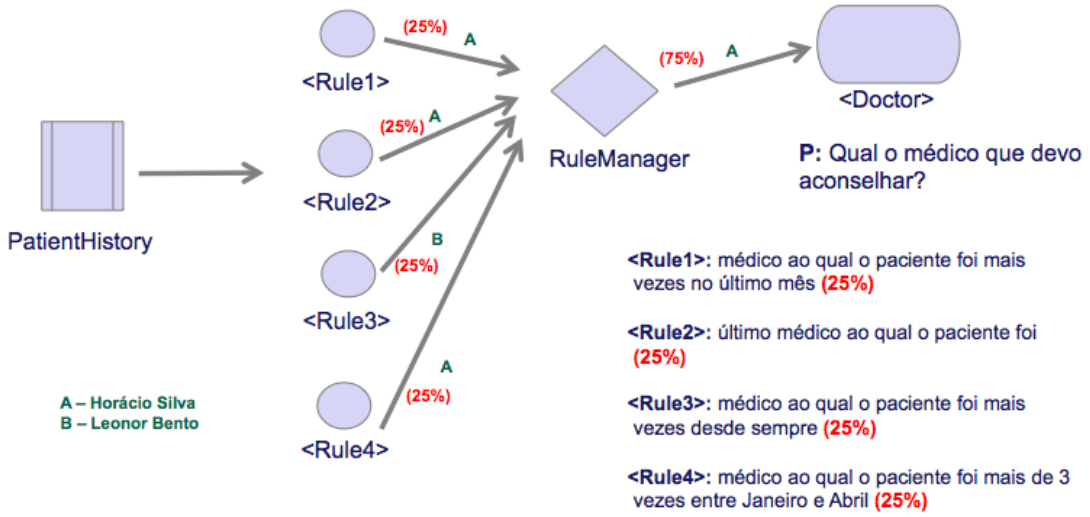

Figura 19 – Aplicação do mecanismo de regras ao estado onde se pretende obter informação acerca do Médico.

No exemplo anterior, para o estado <Médico/Doctor>, estavam definidas quatro regras, todas elas com pesos iguais entre si, uma vez que se pensa que a sugestão dada por cada uma

delas, para a tomada de decisão final, tem igual importância, independentemente de ser ou não igual.
Verificou-se que, com a análise do histórico realizada por cada uma das regras, no sentido de obter a informação que a cada uma delas compete, três regras deram a mesma sugestão, nomeadamente o **Médico A**, perfazendo uma unanimidade entre 75% das partes.
Considerando 75% um peso aceitável, é então tomada uma decisão final, sendo definitivamente sugerido o **Médico A**, como a melhor possibilidade, e que muito provavelmente, corresponderá aquilo que na realidade o paciente procura.

Com este mecanismo de regras, existe a possibilidade real, de haver conjugação de diferentes modos de funcionamento, no sentido de recolher o mais rapidamente possível, toda a informação necessária à criação de um novo registo. Supondo que o utilizador, apenas concorda com parte da informação que lhe é sugerida pelo sistema, por exemplo, apenas deseja o médico e a especialidade sugeridos. Caso não aceite o local ou o horário igualmente sugeridos, assiste-se a uma passagem do fluxo optimizado, neste caso o fluxo optimizado 1, para o fluxo base, sem perder a informação que já foi confirmada pelo utilizador, continuando a partir daí.
No Anexo E, encontra-se outro exemplo da aplicação do mecanismo de regras, que coloca lado a lado, as principais sequências possíveis, cada uma delas abordada genericamente, relativamente ao fluxo normal de execução da aplicação.
No sentido de tirar o melhor partido deste mecanismo, é necessário conjugar esforços com as pessoas que realmente podem confirmar, qual a informação que realmente faz sentido recolher do histórico do paciente, e qual a importância atribuída a essa mesma informação, não só no sentido de gerar mais/menos regras, mas também, regras que realmente contribuam para a tomada de decisão mais acertada.

3.6.1 Primitivas

Neste subcapítulo, vai ser abordado aquilo que está intimamente ligado ao conceito de Regra, que é o conceito de Primitiva.
Cada primitiva não é mais do que uma função. Cada função, de entre as listadas na seguinte tabela, recebe como argumentos diversos parâmetros, permitindo assim alterar o comportamento da própria função, aquando a variação dos parâmetros de entrada.
Por requisito imposto a este trabalho, as regras deveriam ficar armazenadas na Base de Dados. Existe então um mapeamento entre a estrutura (tuplos) da base de dados e as classes ao nível da parte aplicacional do mecanismo de regras, que permite direccionar o funcionamento da aplicação, consoante os parâmetros existentes na base de dados.

Tabela 9 – Descrição das Primitivas.

Primitiva	Parâmetros	Descrição de Parâmetros
MORE_THAN		
Função que permite verificar se algo aconteceu mais do que N vezes, quer no período considerado "padrão" (último mês), quer num período que pode ser parametrizado, com início e fim.		
	TYPE	Indica qual a fonte da qual será recolhida informação: Médicos, Especialidades, Locais, Horários.
	FLAG	0 ou 1: indicam se é para considerar o "último mês" ou um período de tempo definido, respectivamente. Utilizando 1, são utilizadas também as variáveis START e END.
	STAR	Início do período a considerar.
	END	Término do período a considerar.
	VALUE	Valor que serve de patamar, para as comparações que devem ser efectuadas.
MORE_OFTEN		
Função que permite verificar se algo, ocorreu mais vezes do que qualquer outra coisa, quer no período considerado "padrão" (último mês), quer num período que pode ser parametrizado, com início e fim.		
	TYPE	Indica qual a fonte da qual será recolhida informação: Médicos, Especialidades, Locais, Horários.
	FLAG	0 ou 1: indicam se é para considerar o "último mês" ou um período de tempo definido, respectivamente. Utilizando 1, são utilizadas também as variáveis START e END.
	STAR	Início do período a considerar.
	END	Término do período a considerar.
LAST_ONE		
Função que permite verificar, qual o último acontecimento ocorrido, quer no período considerado "padrão" (último mês), quer num período que pode ser parametrizado, com início e fim.		
	TYPE	Indica qual a fonte da qual será recolhida informação: Médicos, Especialidades, Locais, Horários.
	FLAG	0 ou 1: indicam se é para considerar o "último mês" ou um período de tempo definido, respectivamente. Utilizando 1, são utilizadas também as variáveis START e END.
	STAR	Início do período a considerar.
	END	Término do período a considerar.

Com base na informação da tabela anterior, é possível perceber que cada primitiva pode dar origem a muitas regras diferentes, pois basta para isso alterar os parâmetros da primitiva, para se criar uma nova regra, associada ao mesmo ou a outro estado.

Foram definidas três primitivas especificas, que visam sobretudo fazer uma análise da ocorrência de um determinado acontecimento, quer seja a última ocorrência, quer seja o número de ocorrências, num determinado período de tempo passado.

3.6.2 Modelo de Gestão

Com o intuito de utilizar as regras anteriormente definidas, como adição ao sistema base, é necessário ter noção que fazer a gestão das mesmas, não pode constituir um problema para quem o irá fazer.
Deve ser possível gerir as regras existentes, criar e eliminar novas regras, com recurso a uma aplicação fácil de utilizar e que seja familiar para quem a irá utilizar, provavelmente com conhecimentos de informática, apenas ao nível de utilizador normal.

Paralelamente à gestão do sistema de regras, é útil puder fazer a gestão de tudo aquilo que pode ser parametrizável, mesmo que da aplicação base se trate.
Através da aplicação, deve então ser possível configurar o sistema ao nível da utilização/não utilização do mecanismo de regras, níveis inferiores e superiores associados aos reconhecimentos para cada estado da aplicação, limiares de percentagem que fazem a separação entre fluxo base, fluxo optimizado com perguntas directas e fluxo optimizado com uma única pergunta directa, entre outros parâmetros gerais.

4 Desenvolvimento

4.1 Tecnologia

Para a realização deste trabalho, foi utilizada apenas tecnologia Microsoft, sendo que, tal como já foi referido anteriormente, era a única solução que permitia efectuar o completo desenvolvimento, apenas recorrendo a um único fornecedor.

Todo o desenvolvimento foi realizado recorrendo à criação de máquinas virtuais, nas quais se pudessem fazer testes a todos os componentes separadamente, mas também à aplicação final, sem que se pudesse comprometer o bom funcionamento da máquina host, fornecida pela Link Consulting, para a realização deste trabalho. Como ferramenta para a criação e utilização de máquinas virtuais, foi utilizado VMWare, ferramenta gratuita, sendo opção face às restantes, como por exemplo Virtual PC, apenas devido ao facto de já ter sido utilizada anteriormente, mas também por ser a maioritariamente utilizada na Link Consulting.

Como sistema operativo de suporte, na plataforma servidor, foi utilizado Windows Server 2003 R2 Enterprise Edition, por ser um requisito ao desenvolvimento deste tipo de aplicações, embora pudessem ter sido utilizadas versões anteriores.
Para o desenvolvimento da aplicação, foi utilizado Microsoft Visual Studio 2005 (MVS 2005), uma vez que as extensões necessárias para a criação de aplicações de voz, ainda não eram compatíveis com MVS 2008, à data do início do desenvolvimento da aplicação.
No sentido de permitir efectuar o desenvolvimento com recurso a workflows, foi necessária a instalação das extensões para Windows Workflow Foundation, as quais contém de raiz várias *activities*, ligadas ao desenvolvimento de aplicações de voz. É possível desenvolver novas *activities*, que estendam as existentes, ou realizem mesmo actividades totalmente distintas.
Uma vez todos os pré-requisitos cumpridos, foi possível instalar com sucesso a plataforma que iria suportar a aplicação de voz, o Microsoft Speech Server 2007 (SS 2007) *[14]*. Uma vez que de raiz, não existe a Língua Portuguesa (Portugal), foi necessário adicionar um *package* que permitisse fazer uso dessa mesma língua. O *package* encontra-se disponibilizado pelo Microsoft Language Development Center Portugal.

Sendo o SS 2007, um componente do Office Communications Server 2007 (OCS 2007), não deixa de ser possível desenvolver este tipo de aplicações, recorrendo apenas a este componente, uma vez que este pode funcionar de forma independente dos restantes componentes do OCS 2007. Para efeitos de testes, podem ser utilizadas aplicações ou mesmo dispositivos telefónicos físicos, directamente ligados ao SS 2007, no entanto, é possível anexar a aplicação desenvolvida a uma instalação de OCS 2007, tendo assim a possibilidade de testar de forma mais realista, o funcionamento da aplicação. Existindo um

grupo de trabalho na Link Consulting, dedicada a OCS 2007, teria sido interessante fazer essa mesma integração, no entanto, não foi possível.

A figura seguinte, ilustra quais os componentes que fazem parte de uma plataforma servidor com OCS 2007, assim como algumas das interacções existentes entre os vários componentes.

Independentemente do facto de a aplicação ser desenvolvida recorrendo às extensões para Windows Workflow Foundation, através da criação de workflows a serem executados no referido ambiente, encontram-se representados os módulos que permitem efectuar a integração com SALT e VoiceXML, sendo que ao nível da plataforma ASP.NET, todos os componentes partilham a mesma API.

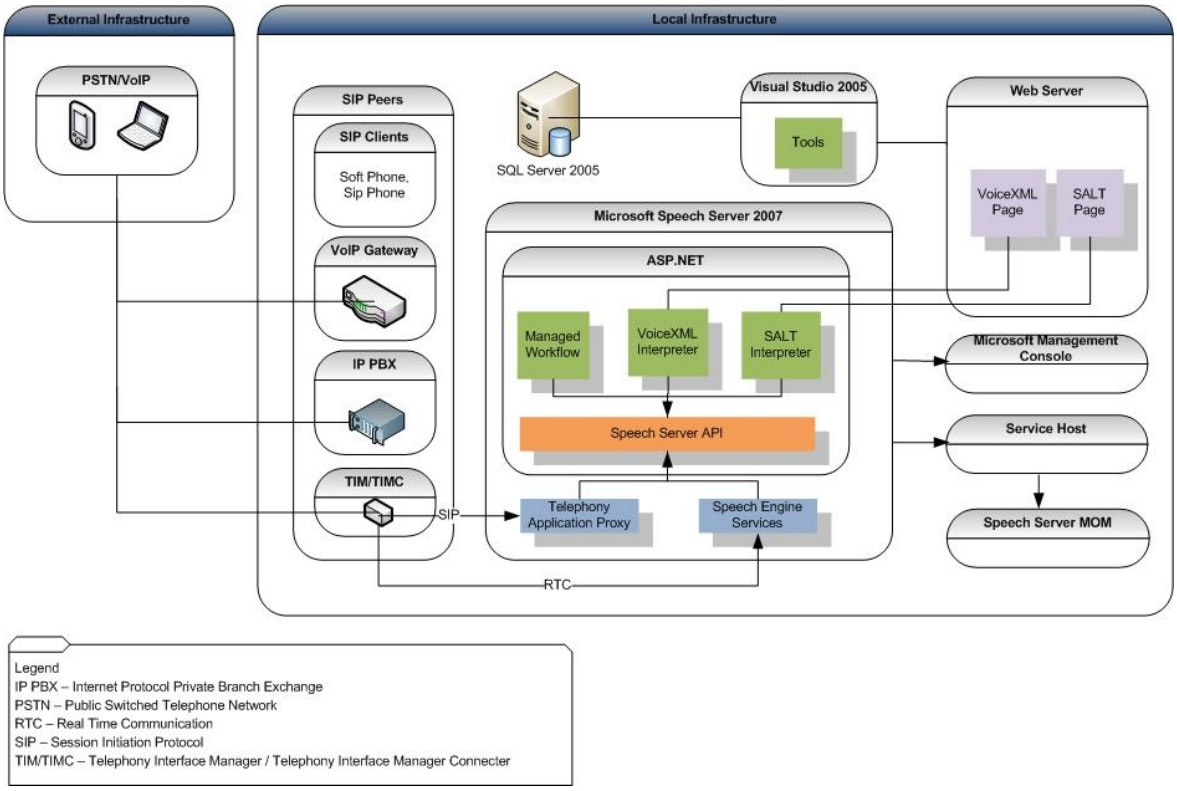

Figura 20 – Estrutura da plataforma COS 2007, com o componente SS 2007.

Ao desenvolvimento deste tipo de aplicações, são aconselhados alguns requisitos que se devem tentar cumprir, quer a nível de hardware, quer a nível de software, no sentido de garantir que se consegue obter do sistema, aquilo que se espera em circunstâncias normais de funcionamento do mesmo.

A plataforma servidor consiste então numa máquina, com requisitos de hardware e software suficientes, de modo a permitir o funcionamento do *OCS 2007 (ou apenas SS 2007)*.

São de seguida listados os requisitos que devem, segundo indicações dos fornecedores da tecnologia em utilização, ser preenchidos.

Software

- Sistema Operativo
    - Mínimo: Windows Server 2003, SP1R
    - Recomendado: Windows Server 2003 R2
- Office Communications Server 2007
    - .NET framework 2.0.
- SQL Server
    - Mínimo: SQL Server 2000 SP4 ou SQL Server 2005, SP1
    - Recomendado: SQL Server 2005, SP2

Hardware

- CPU: Dual processor 3.2 GHz ou equivalente
- Memória: 2 GB
- Disco: suficiente para a plataforma servidor e para as aplicações
- Cache: 1 MB
- Network: GBit NIC

Devido às condições colocadas ao dispor, por parte da Link Consulting, todas as condições relativas ao software foram disponibilizadas, no entanto, os requisitos de hardware, ficaram um pouco abaixo daquelas que eram as recomendações.

Como partes activas e componentes essenciais deste trabalho, existem duas entidades fundamentais, cuja tecnologia coloca ao dispor dos programadores, sendo elas:

Estados/Actividades (*Activities*)

Consiste numa Speech Application, cujos estados ao longo da aplicação são definidos recorrendo às funcionalidades incorporadas no Visual Studio 2005, quer ao nível das Speech Applications simples, quer através da integração com as extensões já referidas, que permitem desta forma definir todo o fluxo de execução, representando cada estado desse mesmo fluxo, como sendo uma *activity*.

A utilização de workflows, concede a possibilidade de fazer o *wrapping* de componentes de *voice response*, em actividades, que não só permitem fazer uma abstracção sobre o código que é executado, mas também permitem reutilizar esses mesmos componentes.
Desta forma, o workflow pode ser visto como uma estrutura hierárquica de actividades. Actividades estas que podem ser elementos simples (primitivas), ou tornar-se *containers* de mais actividades.

A camada de *Dialog Activities* da API do Speech Server, disponibiliza uma série de classes de objectos primitivos, de entre os quais se encontram os seguintes:

- *QuestionAnswerActivity* – execução de pergunta-resposta;
- *StatementActivity* – execução de comando único;
- *RecordAudioActivity* – execução de uma prompt, com gravação da resposta do utilizador.

Para além destas actividades, outras são disponibilizadas, com o intuito de permitir efectuar um certo nível de abstracção, face às primitivas de *voice response*, e simultaneamente tornar possível a reutilização de componentes, ao longo do fluxo de execução.

Gramáticas (*Grammars*)

Consoante o tipo associado a cada estado, pode haver ou não a necessidade de criação de gramáticas, sejam elas para utilizar em reconhecimento de fala ou em reconhecimento de DTMF.
Existe também a possibilidade de criar gramáticas recorrendo ao Grammar Builder, ao Grammar Creator ou à conjugações dos dois mecanismos.
A criação de gramáticas dinâmicas, definidas à medida que a aplicação se executa, consoante as necessidades para um dado estado, é também uma possibilidade.

4.2 Definição de Diálogo

Neste capítulo, é demonstrado como foram implementados alguns pontos abordados no capítulo da Especificação. Não é possível demonstrar todos os pontos anteriormente abordados, devido à dimensão do código necessário para implementar algumas das partes da aplicação. Tentar-se-á complementar o código, com explicações textuais, que visam melhor dar a entender, aquilo que se pretende.
No seguinte subcapítulo, tomar-se-á como exemplo uma interacção simples, para que possa ser mostrado algum código associado a estados essenciais, do fluxo principal da aplicação. Neste mesmo subcapítulo, apenas serão listados excertos de código, ao passo que no subcapítulo Estados de Diálogo, serão listados excertos de código, mas também estados do workflow da aplicação.

4.2.1 Fluxo de Diálogo

O seguinte excerto de código, mostra como é preparada a mensagem de boas vindas ao utilizador, que tenciona utilizar o sistema.
Por uma questão de monitorização e até avaliação do funcionamento do sistema, é utilizado um mecanismo de *logs,* que podem ser consultados na aplicação de configuração, de modo

a que seja possível perceber qual o fluxo de execução geral em cada interacção, de utilizadores com o sistema, quais as regras que foram aplicadas e a razão da sua aplicação, entre outras informações associadas ao funcionamento geral do sistema.

```
private void Welcome_TurnStarting(object sender, TurnStartingEventArgs e){
       this.Welcome.MainPrompt.ClearContent();
       this.Welcome.MainPrompt.AppendText("Bem-vindo ao sistema de
marcação de consultas do SAMES.");
       _context.logger.WriteMessage("[" +
       this.ApplicationHost.TelephonySession.Id + "] " + "[WELCOME]");
}
```

Figura 21 – Excerto de código correspondente ao estado Welcome.

O seguinte excerto de código, faz parte do processo de validação da identidade de um dado utilizador, que tenta aceder ao sistema. Uma vez solicitado o número de beneficiário e o código pessoal, procede-se à tentativa de obter da fonte de dados, algum registo, cujo número de beneficiário, seja igual ao referido pelo utilizador. Caso exista utilizador, é feita a validação ao código pessoal fornecido.

```
NewPatient = _context.GetPatient(PatientId);

if ((NewPatient != null) && (PatientCode == NewPatient.CodePatient))
{
   this.PatientExists = true;
}
else {
   this.PatientExists = false;
}
_context.logger.WriteMessage("[" +
this.ApplicationHost.TelephonySession.Id + "] " + "[USER VALIDATION]");
_context.logger.WriteMessage("[" +
this.ApplicationHost.TelephonySession.Id + "] " + "[Patient Id: " +
PatientId.ToString() + " Password: " + PatientCode.ToString() + "]");
```

Figura 22 – Excerto do código utilizado para validar a informação dada pelo utilizador.

Uma vez dadas três tentativas de autenticação perante o sistema, o utilizador é convidado a ligar mais tarde. Caso a autenticação seja realizada com sucesso, o fluxo continua normalmente, sendo listadas as opções gerais, que o utilizador pode invocar a qualquer instante, ao longo do decorrer da interacção.

```
this.GeneralCommands.MainPrompt.ClearContent();
this.GeneralCommands.MainPrompt.AppendText("A qualquer momento diga:");
this.GeneralCommands.MainPrompt.AppendText("Repetir, para ouvir
novamente.");
this.GeneralCommands.MainPrompt.AppendText("Cancelar, para voltar ao
estado anterior.");
this.GeneralCommands.MainPrompt.AppendText("Ajuda, para obter ajuda.");
this.GeneralCommands.MainPrompt.AppendText("Terminar, para terminar a
conversação.");
```

Figura 23 – Definição das prompts utilizadas no estado GeneralCommands.

Em todos os estados do fluxo da aplicação, se recorre a várias primitivas, no sentido de fazer com que não decorram erros, nas variadas hipóteses possíveis, que existem para percorrer

esse mesmo fluxo. Exemplo disso, é limpar sempre o estado anterior, exceptuando variáveis de sessão, que podem ser utilizadas para identificar o utilizador, ou ter qualquer outra função essencial, para a interacção em curso. Isso pode ser feito, recorrendo ao método `ClearContent()`.

Seguidamente, solicita-se ao utilizador, que indique o nome do médico ou da especialidade que pretende.

Uma vez que a tecnologia utilizada, permite a criação e gestão de gramáticas de forma dinâmica, tentou-se tirar partido disso, para permitir que o sistema de marcação, desse a possibilidade de efectuar uma pesquisa por nome, ou seja, caso o utilizador não se recorde do nome completo do médico que pretende, pode simplesmente dizer o componente do qual se recorda, que o sistema analisará as hipóteses possíveis, sugerindo cada uma delas ao utilizador, ou assumir automaticamente a resposta, caso exista apenas um registo, passando desde logo ao passo seguinte.

No código seguinte, são definidas as *prompts* que permitem questionar o utilizador, quanto à forma através da qual, ele pretende dar início ao processo de recolha de informação, começando pelo Médico ou pela Especialidade.

```
//Add contents to this state.
this.GACAskForDoctorOrSpeciality.ConfirmationMainPrompt.AppendText("");
this.GACAskForDoctorOrSpeciality.MainPrompt.AppendText("Diga o nome do
médico ou a especialidade.");
this.GACAskForDoctorOrSpeciality.Prompts.SilencePrompt.AppendText("Por
favor, é necessário que, diga o nome do médico ou a especialidade.");
this.GACAskForDoctorOrSpeciality.Prompts.NoRecognitionPrompt.AppendText("N
ão entendemos o que disse, por favor, queira repetir.");

//Help and Repeat Prompts, that will be used with universal commands.
this.GACAskForDoctorOrSpeciality.Prompts.RepeatPrompt.AppendText("Diga o
nome do médico ou a especialidade.");
this.GACAskForDoctorOrSpeciality.Prompts.HelpPrompt.AppendText("Sistema de
Ajuda: ");
this.GACAskForDoctorOrSpeciality.Prompts.HelpPrompt.AppendText("Por favor,
diga o nome do médico, caso pretenda efectuar uma marcação por médico. Ou
o nome da especialidade, caso pretenda efectuar uma marcação por
especialidade.");

Grammar DoctorOrSpecialityGrammar = new
Grammar(DynamicGrammars.CreateGrammarDoctorOrSpeciality(_context));

this.GACAskForDoctorOrSpeciality.Grammars.Add(DoctorOrSpecialityGrammar);
```

Figura 24 – Definição da prompt principal e das prompts de Ajuda, Repetição, Silêncio e Não Reconhecimento.

O seguinte excerto de código, mostra uma das formas possíveis de saber o que o utilizador disse, se uma Especialidade ou um Médico, sendo que o nome do médico pode ter sido recolhido com base no sistema de pesquisa, ou por outro lado, pode logo ter sido dito correctamente pelo utilizador.

```
string value = null;

if (!AutoCompleteNames)
       value = this.GACAskForDoctorOrSpeciality.RecognitionResult.Text;
else if ((AutoCompleteNames) && (CompletedName))
       value =
       this.GACAskForDoctorOrSpecialityAutoComplete.RecognitionResult.Text;
else if ((AutoCompleteNames) && (!CompletedName) && (NamesCompleted > 1))
       value =
       this.GACAskForDoctorOrSpecialityAutoCompletePos.RecognitionResult.Text;
else if ((AutoCompleteNames) && (!CompletedName) &&
              (NamesCompleted == 1))
       foreach (string name in doctorsNamesCompleted)
              value = name;

if (DynamicGrammars.GetDoctor(_context).Contains(value)){
       SaidDoctor = true;
}
else if (DynamicGrammars.GetSpeciality(_context).Contains(value)){
       SaidSpeciality = true;
}
```

Figura 25 – Perceber o que foi dito pelo utilizador.

Uma vez que já se sabe ao certo, aquilo que o utilizador tencionava dizer, Médico ou Especialidade, é altura de recolher a informação que ainda falta, para que possa ser efectuada uma nova marcação. Assumindo que o utilizador disse o nome de um Médico, torna-se necessário começar a criar estruturas que permitam guardar toda a informação, recolhida ao longo da interacção.

```
NewDoctor = new Doctor();
NewDoctor.IdDoctor = doc.IdDoctor;
NewDoctor.NameDoctor = doc.NameDoctor;
```

Figura 26 – Criação de uma nova instância da classe Doctor.

A partir do momento em que se sabe qual o médico desejado, uma vez que o utilizador optou por efectuar a marcação por Médico, é necessário saber qual a Especialidade, para a qual o utilizador pretende a consulta.

Seguidamente, é mostrado como se obtém as várias Especialidades associadas a cada médico, uma vez que o sistema real, contempla a hipótese de cada Médico, puder ter mais que uma Especialidade. Os Identificadores (ID's) das Especialidades são recolhidos, utilizando o ID do Médico, anteriormente definido.

```
List<DoctorSpecialities> doctorSpecialities =
                              _context.GetAllDoctorSpecialities();
NewDoctorSpecialities = new List<DoctorSpecialities>();
foreach (DoctorSpecialities docspec in doctorSpecialities) {
       if (docspec.IdDoctor == NewDoctor.IdDoctor)
              NewDoctorSpecialities.Add(docspec);
}
```

Figura 27 – Obtenção das Especialidades de um dado Médico.

Caso exista uma única Especialidade associada ao Médico escolhido, essa é desde logo assumida, no entanto, se existir mais do que uma, é necessário que o utilizador diga qual

pretende. No caso de existirem até três Especialidades, todas elas são listadas, caso existam mais de três, por requisito imposto à aplicação, no sentido de evitar listar muitos nomes, apenas se solicita ao utilizador que diga o nome da Especialidade que pretende. O mesmo aconteceria, para uma Especialidade, à qual estivessem associados vários Médicos. A listagem das hipóteses é realizada, como exemplifica o seguinte código.

```
//MAIN PROMPT
//There are 3 or less than 3 Specialities, so all are listed.
if (specialitiesForDoctor.Count <= maxElementsToList) {
       this.GACAskForSpeciality.MainPrompt.AppendText("Diga, de entre as
       seguintes, o nome da Especialidade que pretende. ");
       foreach (string spec in specialitiesForDoctor) {
              this.GACAskForSpeciality.MainPrompt.AppendText(spec);
              this.GACAskForSpeciality.MainPrompt.AppendText(". ");
       }
}
//There are more than 3 Specialities and the user say which he wants.
if (specialitiesForDoctor.Count > maxElementsToList) {
       this.GACAskForSpeciality.MainPrompt.AppendText("Diga, o nome da
       Especialidade que pretende. ");
}
```

Figura 28 – Listagem todas as Especialidades ou solicitação de Especialidade.

Por questões do ambiente onde o utilizador se encontre, por pronuncias mais ou menos acentuadas que possa ter, ou por qualquer outra razão, pode acontecer nem sempre o sistema reconhecer de imediato, com um nível de confiança aceitável, aquilo que foi dito pelo utilizador. Nesse sentido, após cada questão, pode ser necessário confirmar se aquilo que o sistema entendeu, corresponde mesmo ao que foi dito pelo utilizador.

Uma vez que, para todos os estados do fluxo de execução, podem acontecer este tipo de situações, torna-se necessária a criação de métodos específicos, que permitam fazer essa mesma confirmação, em cada um dos diferentes estados. Como objecto da nova pergunta de confirmação, é utilizado o valor reconhecido pelo sistema na pergunta anterior, podendo assim servir para confirmar esse mesmo valor, ou descartar essa hipótese da lista das possibilidades, *this.GACAskForSpeciality.RecognitionResult.Alternates.*

```
private void GACAskForSpeciality_ConfirmationTurnStarting(object sender,
TurnStartingEventArgs e) {
       this.GACAskForSpeciality.ConfirmationMainPrompt.ClearContent();
       this.GACAskForSpeciality.ConfirmationMainPrompt.AppendText("Disse {0}.
       Certo?", this.GACAskForSpeciality.RecognitionResult.Text);
}
```

Figura 29 – Confirmação, caso necessária, de que o sistema entendeu correctamente.

Uma vez confirmado o Médico e a Especialidade, torna-se necessário recolher informação quanto ao Local, para o qual o utilizador tenciona efectuar a marcação da sua consulta.

No sentido de obter da fonte de dados, as reais possibilidades quanto ao Local, para que estas possam ser listadas ao utilizador, torna-se agora necessário estabelecer uma relação entre Médico, Especialidade e Local, retirando apenas os Locais, onde o Médico escolhido, dê consultas da Especialidade escolhida, uma vez que se contempla a hipótese de cada

Médico, não ter de dar consultas de todas as suas Especialidades, em cada um dos Locais onde trabalhe.

```
//All Localities where exists wanted Speciality
List<SpecialityLocalities> localitiesForSpeciality =
       _context.GetAllSpecialityLocalities();
List<SpecialityLocalities> tempSpecLoc = new List<SpecialityLocalities>();
foreach (SpecialityLocalities specloc in localitiesForSpeciality){
       if (specloc.IdSpeciality == NewSpeciality.IdSpeciality) {
              SpecialityLocalities NewSpecLoc = new SpecialityLocalities();
             NewSpecLoc.IdSpeciality = NewSpeciality.IdSpeciality;
             NewSpecLoc.IdLocality = specloc.IdLocality;
             tempSpecLoc.Add(NewSpecLoc);
       }
}
//All Localities where exists wanted Doctor
List<DoctorLocalities> localitiesForDoctor =
       _context.GetAllDoctorLocalities();
tempDocLoc = new List<DoctorLocalities>();
foreach (DoctorLocalities docloc in localitiesForDoctor) {
       if ( (docloc.IdDoctor == NewDoctor.IdDoctor) ) {
              DoctorLocalities NewDocLoc = new DoctorLocalities();
              NewDocLoc.IdDoctor = NewDoctor.IdDoctor;
              NewDocLoc.IdLocality = docloc.IdLocality;
              NewDocLoc.TimePeriodEnd = docloc.TimePeriodEnd;
              NewDocLoc.TimePeriodStart = docloc.TimePeriodStart;
              tempDocLoc.Add(NewDocLoc);
       }
}
```

Figura 30 – Obtenção de Locais para a conjugação Médico/Especialidade.

Uma vez obtidos todos os Locais, onde existe a Especialidade desejada e todos os Locais, onde existe o Médico desejado, é necessário relacionar a informação obtida em separado.

```
NewDoctorLocalities = new List<DoctorLocalities>();

foreach (SpecialityLocalities specloc in tempSpecLoc) {
       foreach (DoctorLocalities docloc in tempDocLoc) {
              if ((specloc.IdSpeciality == NewSpeciality.IdSpeciality) &&
              (docloc.IdDoctor == NewDoctor.IdDoctor)) {

                     (……)

                     NewDoctorLocalities.Add(docLocality);
                     break;
              }
       }
}
```

Figura 31 – Obtenção de Locais para a conjugação Médico/Especialidade.

Sabendo o Médico, a Especialidade e o Local, para os quais o utilizador deseja efectuar uma nova marcação, é necessário confirmar um Horário disponível.

Uma vez que, no âmbito deste projecto, não foi possível aceder ao real formato e organização da informação, na aplicação actualmente em funcionamento no SAMS, para este trabalho assumiu-se que os horários que eventualmente não tenham sido ocupados no passado recente, se mantém na fonte de dados, até que um sistema paralelo ao sistema abordado neste trabalho, trate de eliminar da fonte de dados, todos os registos que não são

mais necessários. Sendo que no actual sistema em utilização no SAMS, as marcações são efectuadas apenas para a semana seguinte, com o período de marcação que consiste na semana em curso, optou-se por fazer com que, na informação contida na fonte de dados, estejam registos da semana seguinte, legitima semana para a qual se aceitam marcações, mas também horários que não tenham sido ocupados na semana em curso, durante o respectivo período de marcações.

```
finalDocLoc = new List<DoctorLocalities>();
foreach (DoctorLocalities docloc in tempDocLoc) {
	if ((docloc.IdDoctor == NewDoctor.IdDoctor) &&
	(docloc.IdLocality == NewLocality.IdLocality) &&
	(docloc.TimePeriodStart.DayOfYear > DateTime.Now.DayOfYear)) {
		DoctorLocalities NewDocLoc = new DoctorLocalities();
		NewDocLoc.IdDoctor = NewDoctor.IdDoctor;
		NewDocLoc.IdLocality = docloc.IdLocality;
		NewDocLoc.TimePeriodEnd = docloc.TimePeriodEnd;
		NewDocLoc.TimePeriodStart = docloc.TimePeriodStart;
		finalDocLoc.Add(NewDocLoc);
	}
}
```

Figura 32 – Selecção de Horários possíveis.

Relativamente ao estado, onde se tenciona recolher informação acerca do horário, uma vez que se trata de um problema diferente face à recolha do nome do Médico, da Especialidade e do Local, optou-se por fazer uma listagem dos vários horários disponíveis, antecedidos de um número, sendo que o utilizador em vez do Horário que pretende, diz o número associado a esse mesmo horário.

```
timesForDoctorAndSpecialityAndLocality =
DynamicGrammars.GetTimesForDoctorAndSpecialityAndLocality(finalDocLoc);

(……)
this.AskForTime.MainPrompt.AppendText("Caso pretenda um dos horários listados
de seguida, diga o número correspondente ao que pretende.");

int index = 0;
foreach (string time in timesForDoctorAndSpecialityAndLocality) {
	index++;
	this.AskForTime.MainPrompt.AppendText(index.ToString());
	this.AskForTime.MainPrompt.AppendText(": ");
	this.AskForTime.MainPrompt.AppendText(time);
	this.AskForTime.MainPrompt.AppendText(". ");
}
this.AskForTime.Prompts.SilencePrompt.AppendText("Por favor, é necessário
que, diga o número correspondente ao horário que pretende.");
this.AskForTime.Prompts.NoRecognitionPrompt.AppendText("Não entendemos o que
disse, por favor, queira repetir.");
```

Figura 33 – Listagem de Horários possíveis.

Da mesma forma, a gramática criada e que se encontra associada a este estado, contém o intervalo de números, entre 1 e N (`index`), sendo que N é o número de horários disponíveis, que foram listados ao utilizador, para que este pudesse escolher qual queria.

```
Grammar TimesForDoctorAndSpecialityAndLocalityGrammar = new
Grammar(DynamicGrammars.CreateGrammarTimesForDoctorAndSpecialityAndLocality(i
ndex));

this.AskForTime.Grammars.Add(TimesForDoctorAndSpecialityAndLocalityGrammar);
```

Figura 34 – Gramática utilizada na listagem de Horários.

A partir do momento, em que o sistema contém informação para todos os campos que são necessários preencher, para que um novo registo de consulta possa ser efectuado, é necessário obter a confirmação final do utilizador, formulando assim uma questão única, contendo Médico, Especialidade, Local e Horário, à qual o utilizador deve dizer se confirma ou não confirma.

```
this.AskFinalQuestion.MainPrompt.AppendText("Confirma a marcação: ");
this.AskFinalQuestion.MainPrompt.AppendText("Para ");

this.AskFinalQuestion.MainPrompt.AppendText(NewSpeciality.NameSpeciality);
this.AskFinalQuestion.MainPrompt.AppendText(". Com ");

this.AskFinalQuestion.MainPrompt.AppendText(NewDoctor.NameDoctor);
this.AskFinalQuestion.MainPrompt.AppendText(". Em ");

this.AskFinalQuestion.MainPrompt.AppendText(NewLocality.NameLocality);
this.AskFinalQuestion.MainPrompt.AppendText(".");
this.AskFinalQuestion.MainPrompt.AppendText(". No dia ");

this.AskFinalQuestion.MainPrompt.AppendText(NewDoctorLocality.TimePeriodStart
.ToString("d-MMMM-yyyy H:mm"));
this.AskFinalQuestion.MainPrompt.AppendText(". Sim ou Não?");
this.AskFinalQuestion.Prompts.SilencePrompt.AppendText("Por favor, diga Sim
se estiver de acordo, diga Não, caso contrário.");
this.AskFinalQuestion.Prompts.NoRecognitionPrompt.AppendText("Não entendemos
o que disse, por favor, queira repetir.");
```

Figura 35 – Confirmação final.

Caso o utilizador confirme, a interacção terminará após a confirmação dada pelo sistema, de que uma nova consulta foi agendada.

```
this.PatientAgreesWithFinalQuestion.MainPrompt.ClearContent();

this.PatientAgreesWithFinalQuestion.MainPrompt.AppendText("Marcação efectuada
com sucesso.");
```

Figura 36 – Mensagem de confirmação de nova marcação efectuada.

Caso o utilizador não confirme, é-lhe dada a hipótese de escolher entre, efectuar uma nova marcação ou terminar a interacção com o sistema.
Se o utilizador resolver tentar efectuar uma nova marcação, o processo será reiniciado, eliminando toda a informação até então armazenada, excepto a informação relativa à identidade do utilizador.

```
this.PatientDoesNotAgreesWithFinalQuestion.MainPrompt.AppendText("Marcação
cancelada. Deseja efectuar uma nova marcação? Sim ou Não?");
this.PatientDoesNotAgreesWithFinalQuestion.Prompts.SilencePrompt.AppendText("
Por favor, diga Sim se pretender efectuar uma nova marcação, diga Não para
terminar.");
this.PatientDoesNotAgreesWithFinalQuestion.Prompts.NoRecognitionPrompt.Append
Text("Não entendemos o que disse, por favor, queira repetir.");

//Help and Repeat Prompts, that will be used with universal commands.
this.PatientDoesNotAgreesWithFinalQuestion.Prompts.HelpPrompt.AppendText("Sis
tema de Ajuda: ");
this.PatientDoesNotAgreesWithFinalQuestion.Prompts.HelpPrompt.AppendText("Por
favor, diga Sim se pretender efectuar uma nova marcação, diga Não para
terminar.");
this.PatientDoesNotAgreesWithFinalQuestion.Prompts.RepeatPrompt.AppendText("M
arcação cancelada. Deseja efectuar uma nova marcação? Sim ou Não?");
```

Figura 37 – Mensagem de confirmação de cancelamento, a pedido do utilizador.

Caso o utilizador não tencione efectuar uma nova marcação, desejando assim terminar a interacção com o sistema, ser-lhe-á dada a mensagem de despedida e a interacção terminará de seguida.

```
this.Goodbye.MainPrompt.ClearContent();
this.Goodbye.MainPrompt.AppendText("Obrigado. Adeus.");
```

Figura 38 – Mensagem de despedida. Fim de interacção.

4.2.2 Comandos Universais

Tal como foi descrito no capitulo da Especificação, a aplicação dispõe de cinco comandos universais, sendo eles o que dá acesso ao sistema de ajuda, o que permite repetir uma *prompt*, o que permite voltar ao início da interacção, com listagem das opções de menu, o que permite voltar ao estado onde se escolhe um médico ou uma especialidade e ainda o comando que permite terminar a interacção com o sistema.

Dos cinco comandos referidos, o comando que dá acesso ao sistema de ajuda e o comando que permite repetir algo que foi dito anteriormente, no estado onde a interacção se encontra, são dependentes do contexto, pois quer a ajuda dada ao utilizador pelo sistema, quer a repetição de algo que foi dito anteriormente, e que por alguma razão o utilizador tenciona ouvir novamente, dependem do estado onde a interacção se encontra, sendo assim necessário, ter a constante noção da evolução de cada interacção existente.

O seguinte excerto de código, mostra como é possível definir as *prompts RepeatPrompt* e *HelpPrompt*, de forma a que sempre que um dos comandos universais seja invocado, as *prompts* possam ser utilizadas.

```
//Clear all the content for this state.
this.AskUserIdentificationNumber.Prompts.HelpPrompt.ClearContent();
this.AskUserIdentificationNumber.Prompts.RepeatPrompt.ClearContent();

(.....)

//Help and Repeat Prompts, that will be used with universal commands.
this.AskUserIdentificationNumber.Prompts.HelpPrompt.AppendText("Sistema de
Ajuda: ");
this.AskUserIdentificationNumber.Prompts.HelpPrompt.AppendText("O seu número
de benefeciário é composto por 8 digitos. Deve dizer ou digitar, um a um.");
this.AskUserIdentificationNumber.Prompts.RepeatPrompt.AppendText("Por favor,
diga ou digite o seu número de benefeciário.");

(.....)
```

Figura 39 – Definição de prompts de Ajuda e Repetição.

Os comandos que permitem retroceder na interacção e o comando que permite terminar a interacção, não são dependentes do contexto, sendo desta forma de mais fácil execução, uma vez que independentemente do estado onde o utilizador se encontre, ao longo da interacção, a acção a realizar será sempre um *goTo*, para um estado previamente definido, seja ele para a listagem das opções, para a selecção de médico ou especialidade ou para a *prompt* de despedida. Devido às facilidades oferecidas pelas extensões para WWF, é possível fazê-lo recorrendo às entidades já definidas, sendo apenas necessário a alteração de alguns parâmetros. A imagem seguinte mostra uma *activity* já existente, parametrizada para passar ao estado *Goodbye*, sempre que o utilizador solicitar a terminação do processo de marcação de consultas.

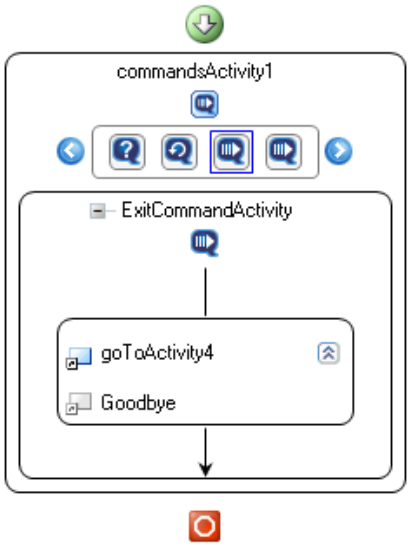

Figura 40 – Definição do comando universal Sair/Adeus.

4.2.3 Estados de Diálogo

O primeiro estado de maior relevância do workflow da aplicação, é o que permite efectuar a validação dos dados do utilizador, surgindo daí a passagem ao processo de marcação de consultas, caso os dados sejam reais e correctos, ou a terminação da interacção, após três tentativas sem sucesso de validação.

A imagem seguinte, representa o processo de autenticação, composto por três *activities*, das quais a primeira solicita o número de beneficiário ao utilizador, composto por oito dígitos, dispondo assim de gramáticas que permitam efectuar o reconhecimento desse mesmo número, quer através de DTMF, quer através da voz. A segunda *activity*, solicita o código pessoal ao utilizador, que por questões de segurança, apenas pode ser introduzido via DTMF. Uma vez fornecidos os dados necessários, a terceira *activity*, designada de *UserValidation*, confirmará esses mesmos dados. Caso a validação seja realizada com sucesso, o processo seguirá para a escolha de um Médico ou de uma Especialidade, logo depois da *activity* que efectua a listagem das opções de menu, às quais o utilizador poderá recorrer a qualquer altura, designada por *GeneralCommands*.

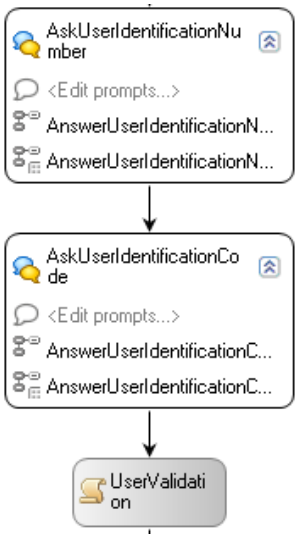

Figura 41 – Obtenção de Número de Beneficiário e Código Pessoal.

Tendo sido a autenticação realizada com sucesso, é dada a altura de questionar o utilizador, acerca da forma pela qual ele tenciona começar o processo de marcação de uma nova consulta, processo do qual já foi mostrado um excerto de código no subcapítulo 4.2.1 – Fluxo de Diálogo.

Na figura seguinte, é mostrado o processo executado após a resposta dada pelo utilizador, quanto ao Médico ou à Especialidade.

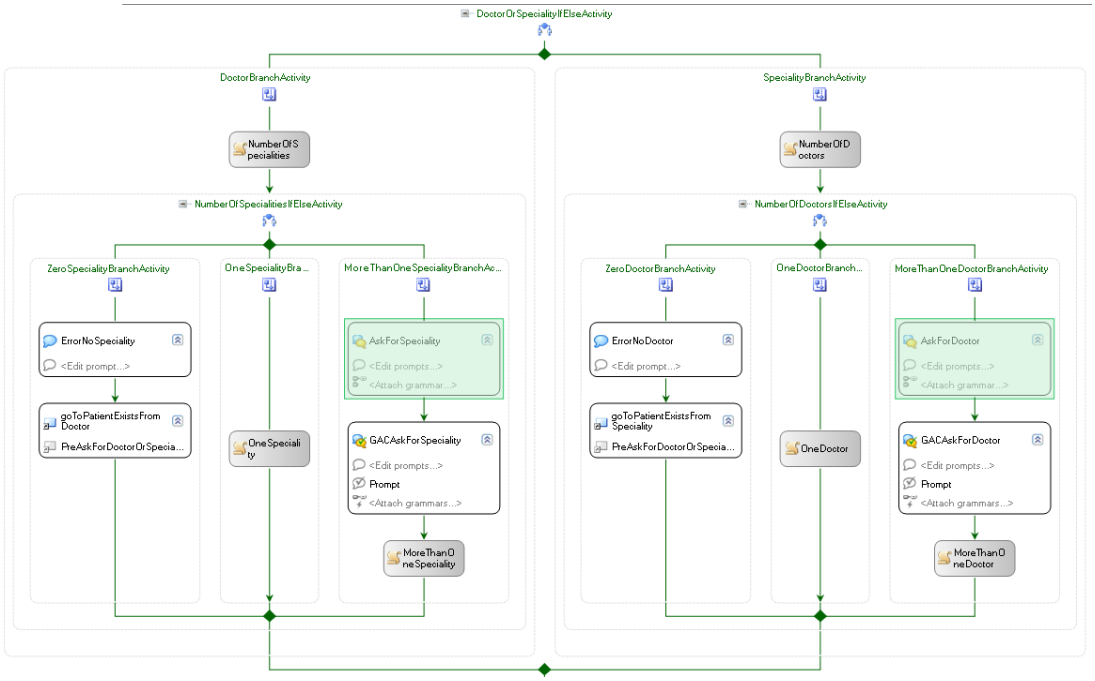

Figura 42 – Solicitação de Médico ou Especialidade.

Após cada resposta dada pelo utilizador, quando questionado acerca do Local ou do Horário, estados estes, também já referidos com “principais”, desenrola-se um processo de decisão idêntico ao mostrado na figura anterior, mas um pouco mais simples, devido ao facto de no estado onde se tenta obter o nome do Médico ou o nome da Especialidade, se tratar de uma pergunta aberta, para a qual o utilizador pode indicar o nome de um Médico ou o nome de uma Especialidade, sendo que seguidamente será obtida a informação em falta, ou seja, se indicar inicialmente o nome de um Médico, será questionado de seguida quanto Especialidade, ou vice-versa.

Para cada estado, existem três possibilidades de caminho a seguir, sendo eles, avisar o utilizador de erro, relativamente ao que foi por ele dito, devido ao facto de não fazer parte da gramática utilizada; perceber automaticamente aquilo que o utilizador deseja, devido ao facto de ter havido *matching* com apenas uma possibilidade, assumindo essa como verdadeira; ou ainda, ter sido feito o *matching* com várias possibilidades, sendo necessário questionar o utilizador sobre qual a que ele realmente pretende. Por exemplo, o utilizador disse o nome de uma Especialidade, para a qual existem quatro Médicos possíveis. Para a situação referida anteriormente, é directamente perguntado ao utilizador qual o Médico que deseja, no entanto, por requisito imposto à aplicação, tal só acontece quando existe correspondência com um número de possibilidades superior a três, caso sejam apenas três ou menos, o número de hipóteses, todas elas são listadas, sendo igualmente solicitado ao utilizador, que diga qual é a que pretende.

A partir do momento em que se sabe qual o Médico e qual a Especialidade, questiona-se o utilizador acerca do Local, e analisa-se a resposta dada pelo utilizador à questão realizada, como mostra a seguinte figura.

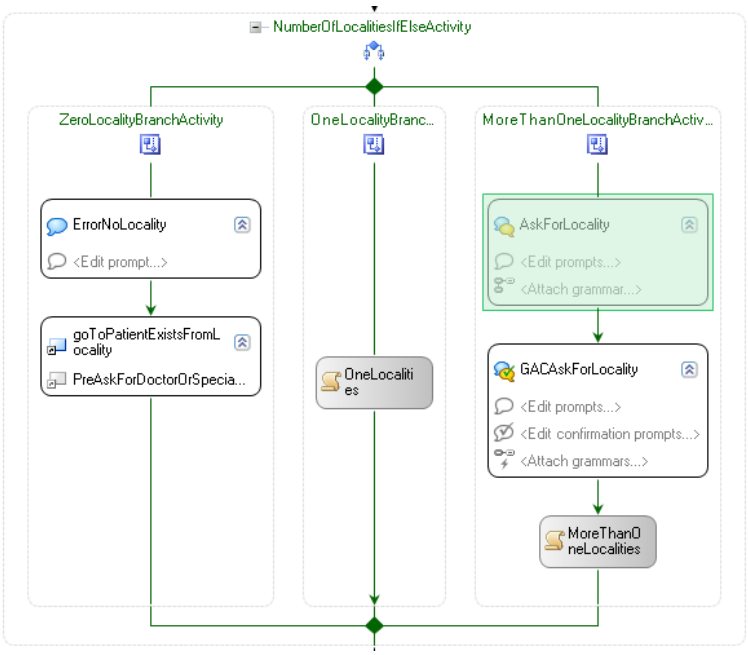

Figura 43 – Solicitação de Local, admissão do único existente ou mensagem de erro.

Após ser questionado acerca do Local, será questionado acerca do Horário, sendo o processo de decisão, idêntico ao da figura anterior.

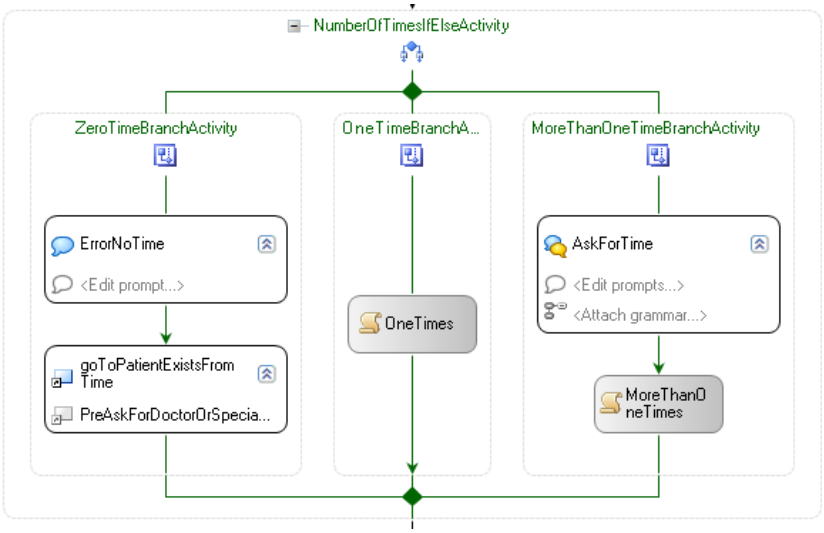

Figura 44 – Solicitação de Horário, admissão do único existente ou mensagem de erro.

Uma vez obtida toda a informação necessária à marcação de uma nova consulta, toda a informação fornecida será repetida ao utilizador, pedindo que este confirme a mesma. Caso o utilizador confirme, será efectuada uma nova marcação, caso contrário, é-lhe dada a hipótese de voltar ao início, para efectuar uma nova marcação, ou terminar a interacção com o sistema. Esta fase de decisão, é mostrada na figura seguinte.

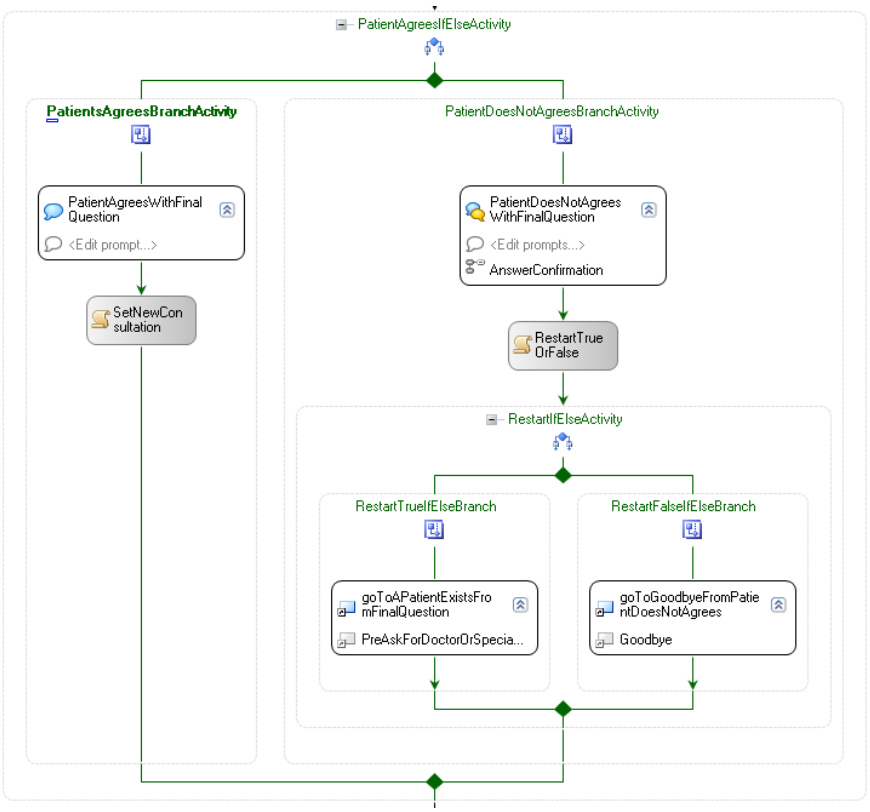

Figura 45 – Confirmação final.

4.2.4 Modelo de Dados

O modelo de dados representado na figura 46, corresponde à estruturação da informação adoptada, para efeitos de teste da plataforma de desenvolvimento e de uma aplicação de demonstração, sobre essa mesma plataforma, tal como já tinha sido referido no capitulo de Especificação.

Na figura seguinte, é dada a visão geral da estrutura de dados utilizada, tomando como fonte de informação uma base de dados, assim como as relações existentes entre as diversas tabelas, dessa mesma base de dados.

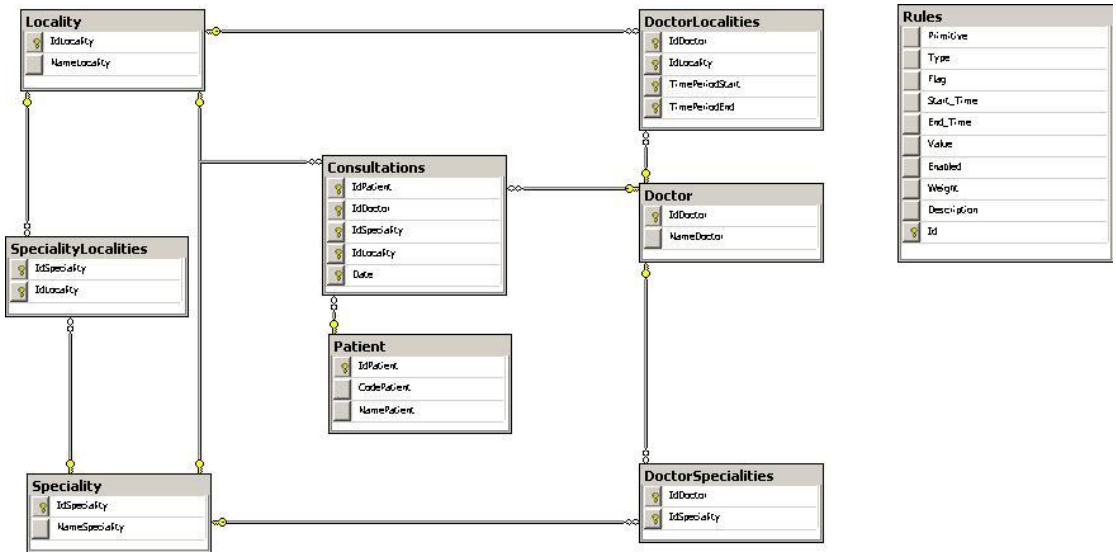

Figura 46 – Modelo de dados do sistema.

Tabelas Base

- ***Doctor*** – *IdDoctor (Int)*: Identificador do Médico; *NameDoctor (String)*: Nome do Médico.
- ***Locality*** – *IdLocality (Int)*: Identificador do Local; *NameLocality (String)*: Nome do Local.
- ***Patient*** – *IdPatient (Int)*: Identificador do Paciente, ou número de beneficiário; *CodePatient (Int)*: Código de identificação pessoal do Paciente perante o sistema; *NamePatient (String)*: Nome do Paciente.
- ***Speciality*** – *IdSpeciality (Int)*: Identificador da Especialidade; *NameSpeciality (String)*: Nome da Especialidade.
- ***Rules*** – *Primitive (String)*: Primitiva utilizada em cada uma das regras; *Type (Int)*: Tipo/Estado ao qual se aplica a regra (Médico / Especialidade / Local / Horário); *Flag (Int)*: Indica se é para analisar um período de tempo especifico ou para analisar a globalidade dos registos armazenados; *StartTime (DateTime)*: Inicio do período a analisar pela regra em causa; *EndTime (DateTime)*: Fim do período a analisar pela regra em causa; *Value (Int)*: Representa o valor do número de ocorrências que se procura encontrar; *Enabled (Boolean)*: Indica se a regra se encontra ou não activa; *Weight (Int)*: Representa o peso, em percentagem, da regra em causa, na tomada de decisão final relativamente à melhor escolha para um determinado estado; *Description (String)*: Descrição textual da regra; *Id (Int)*: Identificador da regra.

Tabelas de Relação

- ***Consultations*** – *IdPatient (Int)*: Identificador do Paciente, ou número de beneficiário*; IdDoctor (Int)*: Identificador do Médico; *IdSpeciality (Int)*: Identificador da Especialidade; *IdLocality (Int)*: Identificador do Local; *Date (DateTime)*: Horário da consulta.

- ***DoctorLocalities*** – *IdDoctor (Int)*: Identificador do Médico; *IdLocality (Int)*: Identificador do Local; *TimePeriodStart (DateTime)*: Início do período de tempo atribuído à consulta; *TimePeriodEnd (DateTime)*: Fim do período de tempo atribuído à consulta.
- ***DoctorSpecialities*** – *IdDoctor (Int)*: Identificador do Médico; *IdSpeciality (Int)*: Identificador da Especialidade.
- ***SpecialityLocalities*** – *IdSpeciality (Int)*: Identificador da Especialidade; *IdLocality (Int)*: Identificador do Local.

4.3 Mecanismo de Regras

Tal como foi abordado no capitulo de Especificação, o mecanismo de regras tem como objectivo principal, permitir prever o maior número de informação possível, aquando uma nova interacção com um utilizador, com base no seu histórico de interacções com o sistema de marcação de consultas.
Neste sentido, foram implementadas as funções necessárias, que permitissem devolver os campos mais prováveis, com base no já referido histórico informativo, desde o Médico, a Especialidade, o Local e a preferência de Horários, de manhã ou de tarde.
Tentou-se fazer com que não seja de forma alguma necessário, alterar a aplicação ao nível do código, para que se consiga um comportamento diferente. Da mesma forma, tentou-se fazer com que existisse um mapeamento entre as classes de código ao nível da aplicação e as possibilidades de parametrização de cada uma das regras, definidas na fonte de dados.

Tal como foi referido anteriormente, a introdução de um mecanismo de regras, deveria implicar o menor número de alterações possível à versão base da aplicação, consistindo desta forma em módulos, introduzidos em locais específicos, do fluxo da aplicação, podendo ou não encontrar-se activos, sendo que, quando activos, deveriam permitir evitar percorrer certos percursos, em detrimento de alternativas mais viáveis.
Nesse sentido, com a introdução do mecanismo de regras, passa a existir um módulo idêntico ao representado na figura seguinte, antes de cada um dos principais estados, ou seja, antes de se perguntar o Médico ou a Especialidade, antes de se perguntar o Local e antes de se perguntar o Horário.
Com a introdução destes novos estados, é possível o sistema decidir entre, caminho com aplicação de regras e caminho base, sendo que, escolhendo caminho com aplicação de regras, pode realizar várias perguntas directas, caso o nível de certeza não seja acima do limiar definido, ou fazer uma única pergunta directa, caso esse mesmo limiar seja ultrapassado.

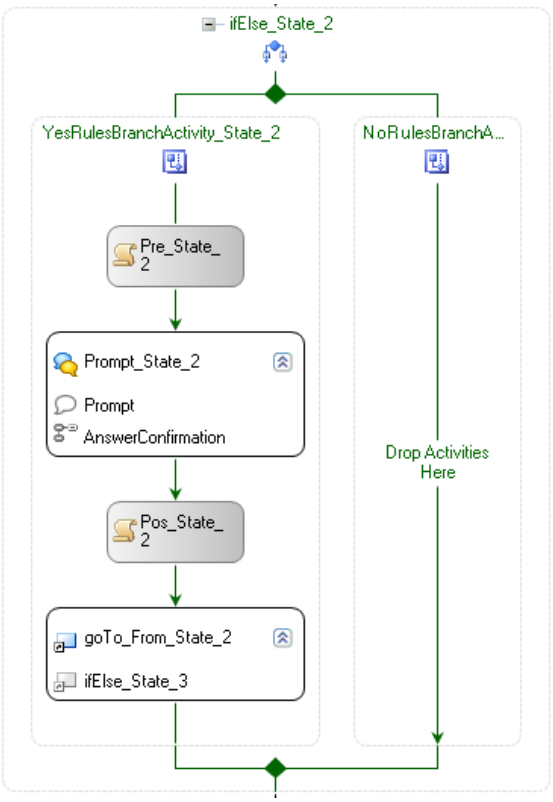

Figura 47 – Representação visual de cada módulo do mecanismo de regras.

Cada estado tem uma fase de pré processamento, onde são tomadas as decisões necessárias, anteriormente referidas, uma fase onde se questiona o utilizador, uma fase de pós processamento, onde se analisa a resposta dada pelo utilizador, terminando com uma fase, na qual, com base na resposta dada pelo utilizador, se passa para um novo estado.

4.3.1 Primitivas

Um dos objectivos relativamente à implementação do sistema de regras, era fazer com que fosse genérico, mas ao ponto de conseguir optimizar significativamente o sistema, para o caso concreto de marcação de consultas.

Neste sentido, fez-se com que as primitivas pudessem ser utilizadas, independentemente do estado ao longo do fluxo de execução, que as estivesse a executar, fazendo apenas variar os parâmetros de entrada em cada uma das funções, que representam cada um das primitivas.

```
public static object More_Than(int type, int flag, int start_time, int
end_time, int value, int weight, int patientId, int doctorId, int
specialityId, int localityId) {

        (……)

        //Doctor
        if (type == 1) { (……) }

        //Speciality
        if (type == 2) { (……) }

        //Locality
        if (type == 3) { (……) }

        //DoctorLocality
        if (type == 4) { (……) }

        (……)
}

public static object More_Often(int type, int flag, int start_time, int
end_time, int value, int weight, int patientId, int doctorId, int
specialityId, int localityId) { (……) }

public static object Last_One(int type, int flag, int start_time, int
end_time, int value, int weight, int patientId, int doctorId, int
specialityId, int localityId) { (……) }
```

Figura 48 – Estrutura genérica das funções que representam cada uma das primitivas.

Cada tuplo/registo da base de dados, representa uma regra associada ao sistema. Cada uma das regras existentes, tem uma primitiva como um dos seus parâmetros de entrada.

No sentido de fazer a gestão de todas a regras a serem aplicadas ao longo da execução do sistema, é necessária a existência de um módulo, que pondere todas as propostas efectuadas, por cada uma das regras activas no sistema, tomando assim uma decisão, quanto à melhor proposta para cada um dos estados do fluxo de execução.

A classe `RuleManager` faz a recolha da informação fornecida por cada uma das regras existentes, organizando as opções sugeridas por cada uma delas, tendo em consideração o peso que cada regra tem para a decisão final, quanto ao caminho a seguir em cada um dos estados.

A classe que permite representar o resultado da aplicação de uma determinada regra, e desta forma, uma das opções para o resultado final, com o peso da regra associado a essa mesma opção, é a classe `Weight`, cujo construtor se encontra a seguir. O campo `Weight.obj`, contém a decisão da aplicação de uma regra, o campo `Weight.weight`, contém o peso desse resultado, para a decisão final de um determinado estado.

O campo `Weight.obj` pode assim ter associado um Médico, uma Especialidade, um Local ou um Horário, consoante o estado para o qual o `RuleManager` esteja à procura de uma solução.

```
public Weight() {
        obj = new object();
        weight = 0;
}
```

Figura 49 – Estrutura utilizada para guardar um objecto e o peso associado.

No seguinte excerto de código, é mostrado como o `RuleManager` está estruturado, no sentido de dar resposta ao sistema, acerca de qual será a melhor opção ao nível do Médico a sugerir ao Paciente, neste caso representado pelo parâmetro `patientId`. Após a recolha de cada um dos resultados fornecidos por cada uma das regras, alinhados com os respectivos pesos, é devolvido o valor que tiver conseguido maior consenso, de entre todas as regras aplicadas.

```
public static object GetDoctor(Context _context, int patientId) {

        List<Rules> rulesDoctor = new List<Rules>();
        rulesDoctor = GetRulesByType(_context,1);

        foreach (Rules rule in rulesDoctor) {
                  nDoc = new Doctor();
                  weight = new Weight();

                if (rule.Primitive.Equals("MORE_THAN")) { (.....) }

                if (rule.Primitive.Equals("MORE_OFTEN")) { (.....) }

                if (rule.Primitive.Equals("LAST_ONE")) { (.....) }

                (.....)
        }
        (.....)
}
```

Figura 50 – Obtenção de regras relativas ao Médico, aplicação das primitivas.

No seguinte excerto de código, é mostrado o caso idêntico ao anterior, no entanto, a fase de busca pela melhor opção para o médico já foi ultrapassada, o que leva a utilizar essa mesma informação, para conseguir sugerir a melhor Especialidade, com base no Médico que foi sugerido anteriormente.

O mesmo se passa para os restantes estados do fluxo da aplicação, uma vez que a informação recolhida até um determinado estado, é utilizada na obtenção de mais informação, num estado posterior.

```
public static object GetSpeciality(Context _context, int patientId, int
doctorId) {
       List<Rules> rulesSpeciality = new List<Rules>();
       rulesSpeciality = GetRulesByType(_context,2);

       foreach (Rules rule in rulesSpeciality) {
                 nSpec = new Speciality();
                 weight = new Weight();

              if (rule.Primitive.Equals("MORE_THAN")) { (.....) }
              if (rule.Primitive.Equals("MORE_OFTEN")) { (.....) }
              if (rule.Primitive.Equals("LAST_ONE")) { (.....) }
              (.....)
       }
       (.....)
}
```

Figura 51 – Obtenção de regras relativas à Especialidade, aplicação das primitivas.

4.3.2 Modelo de Gestão

Tal como foi anteriormente referido no capítulo da Especificação, é considerado importante ser possível fazer a gestão geral do sistema, de forma simples e por alguém que pode não ter conhecimentos aprofundados das tecnologias utilizadas.

Foi então desenvolvido, um protótipo de aplicação de gestão e posteriormente implementada essa mesma aplicação, recorrendo às mesmas tecnologias utilizadas no restante trabalho.

A aplicação permite fazer a gestão de aspectos como os limiares associados aos reconhecimentos, ou seja, "abaixo de" – descarta automaticamente, "acima de" – aceita automaticamente, "entre os limiares" – confirma, questionando o utilizador.

Através da aplicação, é possível fazer a gestão de aspectos gerais relativos ao sistema de regras, como os níveis de concordância necessários para seguir ou não, por um caminho mais rápido. Seguidamente são apresentadas algumas imagens da aplicação.

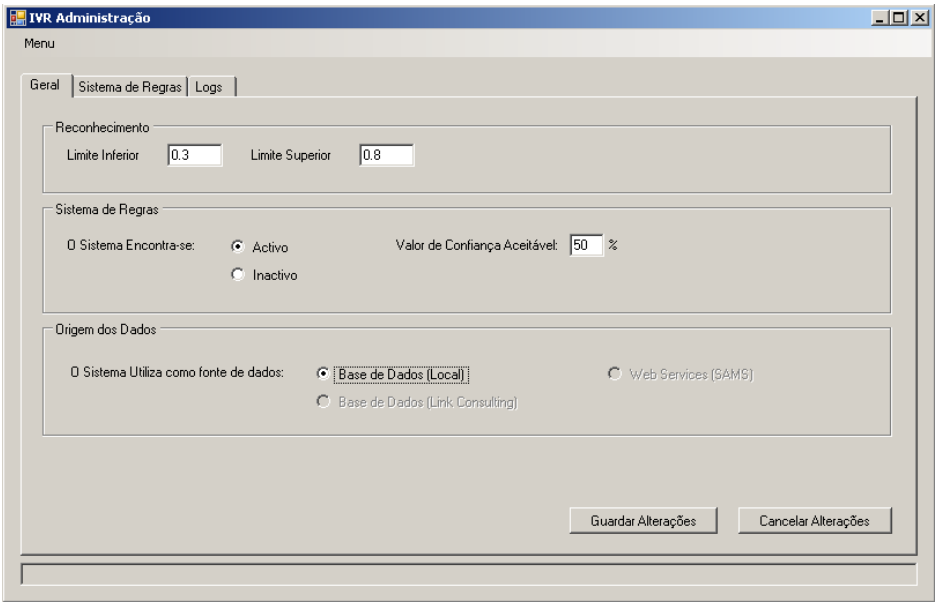

Figura 52 – Menu "Geral" da aplicação de gestão.

A aplicação permite fazer a completa gestão das regras existentes, alterando os seus parâmetros. Permite igualmente, criar e eliminar novas regras.

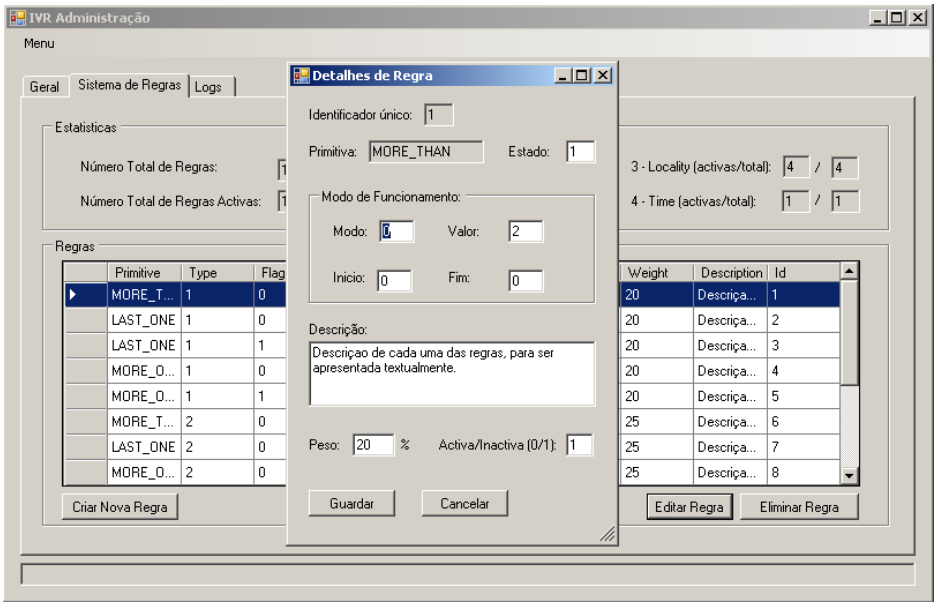

Figura 53 – Menu "Sistema de Regras". Algumas estatísticas. Ecrã de edição/criação de regras.

Uma vez que foi utilizado um mecanismo de *logs*, que visa sobretudo poder analisar os dados de interacções passadas, tendo assim a possibilidade de ajustar o sistema, para deste modo conseguir tirar um melhor partido, é igualmente possível aceder e consultar todos os *logs*, através da aplicação.

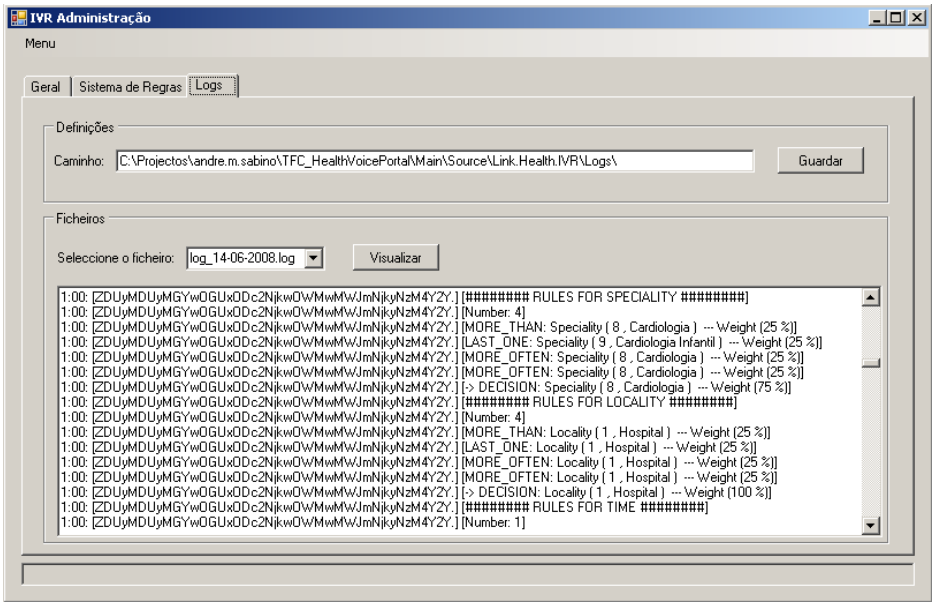

Figura 54 – Menu "Logs". Definição de caminho e abertura de ficheiro seleccionado.

5 Resultados e Avaliação

Neste capítulo, serão abordadas situações de teste reais com utilizadores, efectuados ao sistema de marcação de consultas, no sentido de avaliar o bom funcionamento do mesmo, mas também, se este cumpre ou não com as expectativas de cada um dos utilizadores, que fez parte do grupo de testes.

No sentido de avaliar o sistema, em situações o mais próximas do real possível, foram definidos vários cenários de teste, em vários ambientes, uma vez que se considera do maior interesse, planear e avaliar devidamente este tipo de sistemas *[17] [25] [28]*, cujo funcionamento pode variar, mesmo que as "entradas" não variem.

Primeiramente, o sistema foi testado por utilizadores experientes, no sentido de criar alguns valores de referência, que permitissem então prever, com base em cada uma das tarefas a realizar, o tempo que utilizadores normais, levariam na realização dessas mesmas tarefas.

Seguidamente será realizada uma análise aos resultados e tiradas algumas conclusões, acerca dos mesmos.

Com a realização destes testes ao funcionamento do sistema, tenciona-se avaliar qual a optimização que se consegue ao nível de dois aspectos essenciais, o tempo necessário à realização das tarefas, neste caso a marcação de uma consulta em vários ambientes, mas também a optimização ao nível das taxas de sucesso, associadas aos reconhecimentos.

5.1 Testes com Utilizadores

Tal como foi referido anteriormente, existem duas situações de teste. Uma delas que permitirá criar valores de referência, sendo realizada por utilizadores experientes com o sistema em causa. Uma segunda situação de teste, será realizada por utilizadores considerados normais, aos quais foi dada informação relativa ao funcionamento geral sistema, mas dos quais se espera que façam um uso o mais próximo da realidade possível.

Os testes foram realizados recorrendo a dois utilizadores considerados experientes e dez utilizadores considerados normais.

- **Situação 1** (o mais real possível)
  Contexto: Utilizadores Experientes, vários ambientes de teste;
  Universo: 2 Utilizadores;
  Tarefa: efectuar uma consulta para Leonor Bento, na especialidade Cardiologia, no Hospital, sem restrição a impor ao horário. Existe um histórico regular para o paciente em causa.
- **Situação 2** (o mais real possível)
  Contexto: Utilizadores Não Experientes, vários ambientes de teste;
  Universo: 10 Utilizadores;

Tarefa: efectuar uma consulta para Leonor Bento, na especialidade Cardiologia, no Hospital, sem restrição a impor ao horário. Existe um histórico regular para o paciente em causa.
Nota: É fornecido um guia de introdução ao sistema.

Paralelamente às duas situações de teste referidas, foram definidos vários ambientes de teste, uma vez que um sistema desta natureza, deve poder ser utilizado em qualquer ambiente, independentemente do ruído existente.
Neste sentido, foram definidos três ambientes de teste distintos, que diferem quanto à intensidade do ruído existente, sendo eles os seguintes:

- **Ambiente 1:** ambiente sem ruído → Ex: sala sem televisores ou outros aparelhos ligados;
- **Ambiente 2:** ambiente com ruído médio → Ex: jardim, com ruído do vento e barulho longínquo de pessoas a falar;
- **Ambiente 3:** ambiente com ruído elevado → Ex: qualquer zona perto de uma via pública, com carros e pessoas;

Uma vez descritos genericamente os diferentes ambientes de teste, é necessário referir que existem três modos principais para o funcionamento do sistema, sendo eles:

- **Modo 1:** Modo "base", sem optimizações ao fluxo normal para efectuar a marcação de uma consulta, por médico ou especialidade, ou seja, sem utilização do sistema de regras;
- **Modo 2:** Modo "base", com optimizações de nível 1, onde é utilizada informação do histórico de consultas de um determinado paciente, no sentido de conseguir prever qual o Médico, Especialidade, Local e Horário, com um nível de certeza não elevado (abaixo dos limiares definidos);
- **Modo 3:** Modo "base", com optimizações de nível 2, onde é utilizada informação do histórico de consultas de um determinado paciente, no sentido de conseguir prever qual o Médico, Especialidade, Local e Horário, com um nível de certeza elevado (acima dos limiares definidos).

Os valores obtidos pelos utilizadores experientes, são descritos de seguida. Estes valores servem de referência, como sendo o melhor caso possível, uma vez que se trata de utilizadores conhecedores do funcionamento do sistema, de forma um pouco mais detalhada que os utilizadores considerados normais, que vão utilizar o sistema pela primeira vez. Os valores do seguinte gráfico, são a média dos tempos obtidos (segundos) por 2 utilizadores.

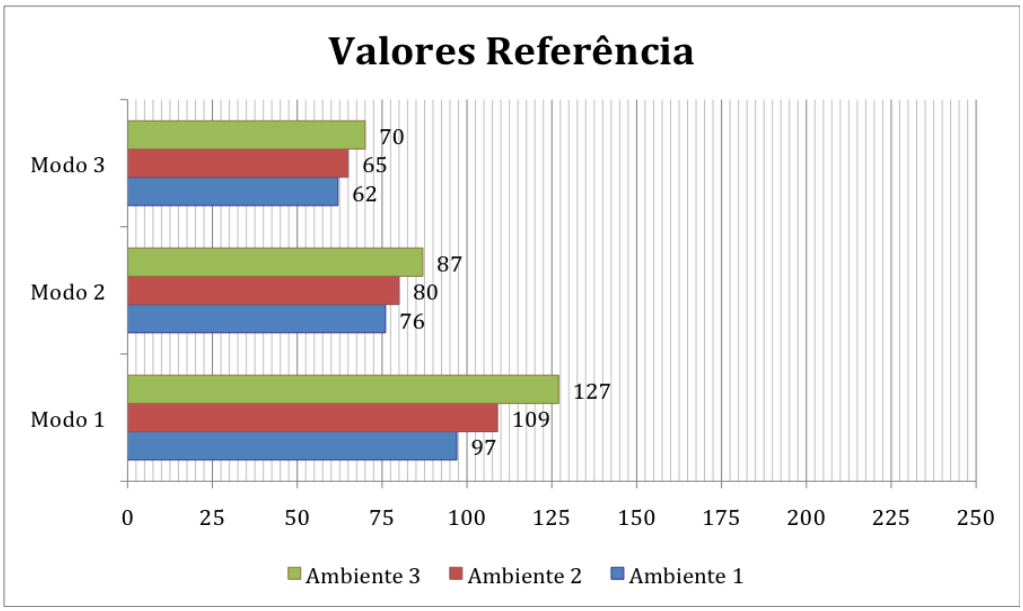

Figura 55 – Valores de referência. Utilizadores experientes..

Com base nos valores do gráfico anterior, são de seguida apontados os valores esperados para a realização das tarefas descritas anteriormente, por utilizadores normais. São tidos em conta pedidos de ajuda, pedidos de repetição de *prompts* e outros aspectos que potencialmente ocorrerão com utilizadores normais, significando um acréscimo no tempo necessário à realização das tarefas.

Os valores do gráfico da Figura 56, encontram-se em segundos e representam o tempo esperado para a realização da tarefa anteriormente indicada, em cada um dos modos de funcionamento e em cada um dos diferentes ambientes.

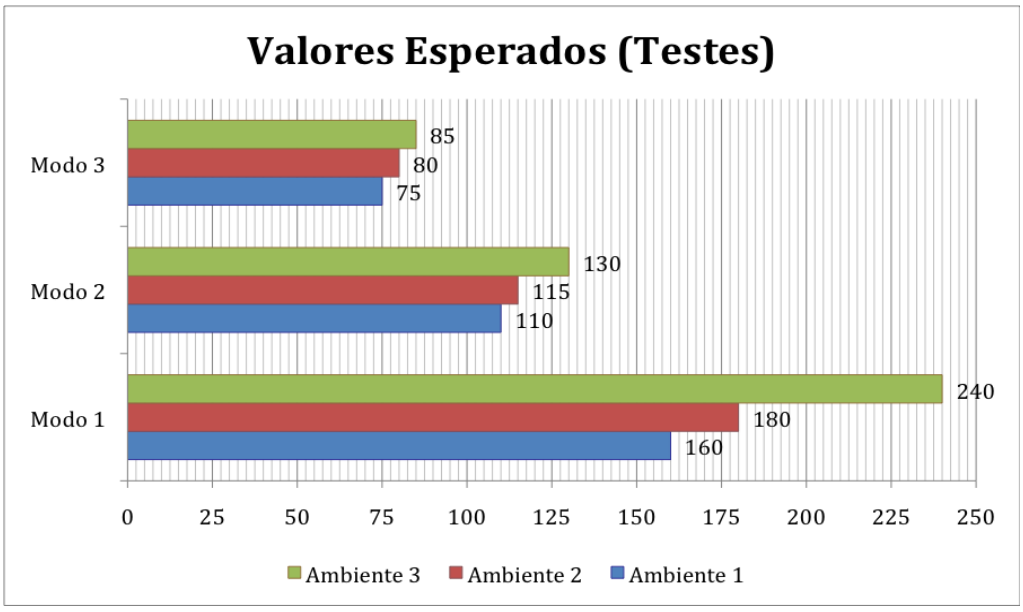

Figura 56 – Valores estimados. Utilizadores inexperientes..

O somatório dos tempos individuais, indica que a totalidade seria de aproximadamente 19.33 minutos, no entanto, existem tempos associados à mudança de ambiente, preparação da aplicação, recolha de valores, estimando que o tempo total necessário à realização dos

testes, por cada utilizador, será no máximo de 35 minutos, contemplando assim todas as situações indicadas anteriormente.

No seguinte gráfico, estão representados os valores médios em segundos, obtidos para cada uma das situações de teste, enunciadas anteriormente.

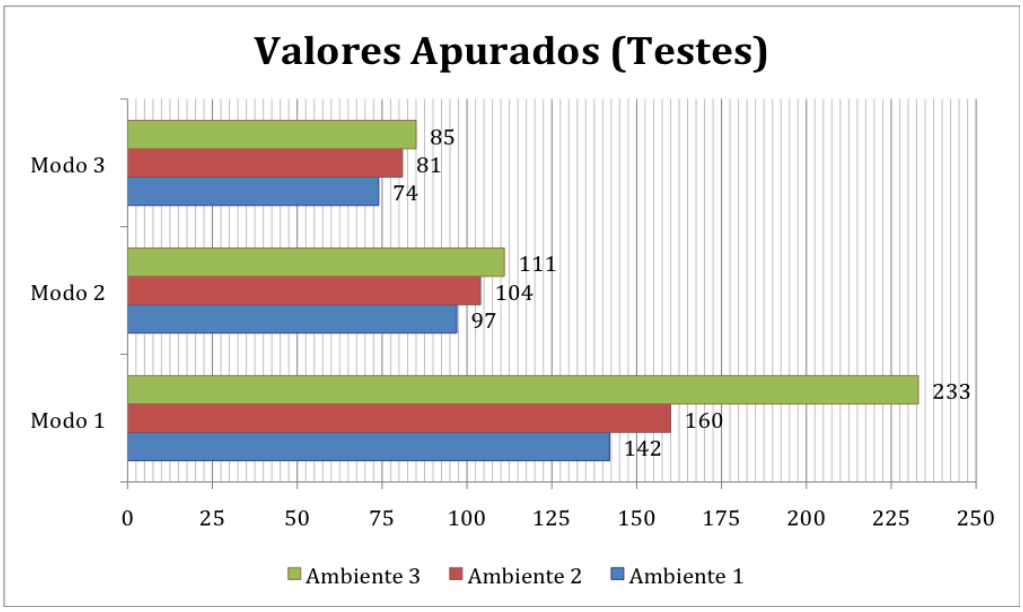

Figura 57 – Valores obtidos. Utilizadores inexperientes..

Uma vez que este trabalho, visa conseguir desenvolver um sistema de marcação de consultas, incluindo uma versão base e versões de optimização á primeira, é de referir que as optimizações conseguidas ao nível do reconhecimento do que é dito pelo utilizador, são conseguidas através de modificações do diálogo utilizado, e consequentemente das gramáticas, que em conjunto com o sistema de regras e outros aspectos directamente relacionados com a aplicação, permitem atribuir ao sistema geral, um funcionamento mais dinâmico. O motor de reconhecimento utilizado, plataforma da Microsoft, não permite afinações realizadas pelo cliente, neste caso pela Link Consulting, sendo que não é possível optimizar o funcionamento geral do sistema, afinando o reconhecedor.

Desta forma, a principal tarefa associada ao objectivo de reduzir a taxa de "Não Reconhecimentos", consiste em evitar, através de todos os artifícios possíveis, situações de "Não Reconhecimento", e não realizar melhorias ao nível do processo de reconhecimento.

Seguidamente são apresentados os resultados obtidos quanto aos reconhecimentos, reconhecimentos com confirmação e não reconhecimentos.

Entenda-se cada uma das situações, como o seguinte:

- **Reconhecimento**: o sistema reconhece o que foi dito pelo utilizador, com um grau de confiança (GC) igual ou superior, àquele que foi definido para uma aceitação automática, por definição GC >= 0.8;
- **Reconhecimento Com Confirmação**: o sistema reconhece o que foi dito pelo utilizador, com um nível de confiança acima do mínimo exigido, mas abaixo do limiar a partir do qual pode aceitar automaticamente, por definição 0.3 <= GC < 0.8. Nestas

situações, é necessário questionar o utilizador, sobre aquilo que se entendeu que ele tinha dito anteriormente;

- **Não Reconhecimento**: o sistema pode ou não ter reconhecido o que foi dito pelo utilizador, no entanto, mesmo que tenha reconhecido, foi com um grau de confiança inferior ao mínimo aceitável, para que se possa confirmar isso com o utilizador, por definição GC < 0.3.

Os resultados apresentados, resultam da avaliação individual de cada uma das situações de teste, atribuídas a cada um dos utilizadores.

No sentido de conseguir mostrar a validade das optimizações realizadas à versão base do sistema, pensa-se ser relevante avaliar sobretudo, até que nível se consegue reduzir as situações de "Não Reconhecimento" ou de "Reconhecimento com Confirmação".

O seguinte gráfico, contém as percentagens obtidas em cada uma das nove situações de teste, realizadas pelos utilizadores. Pretende-se representar "Ambiente 1", por A1, sendo que o mesmo se aplica aos "Modos de Funcionamento", ou seja, "Modo de Funcionamento 1", por M1.

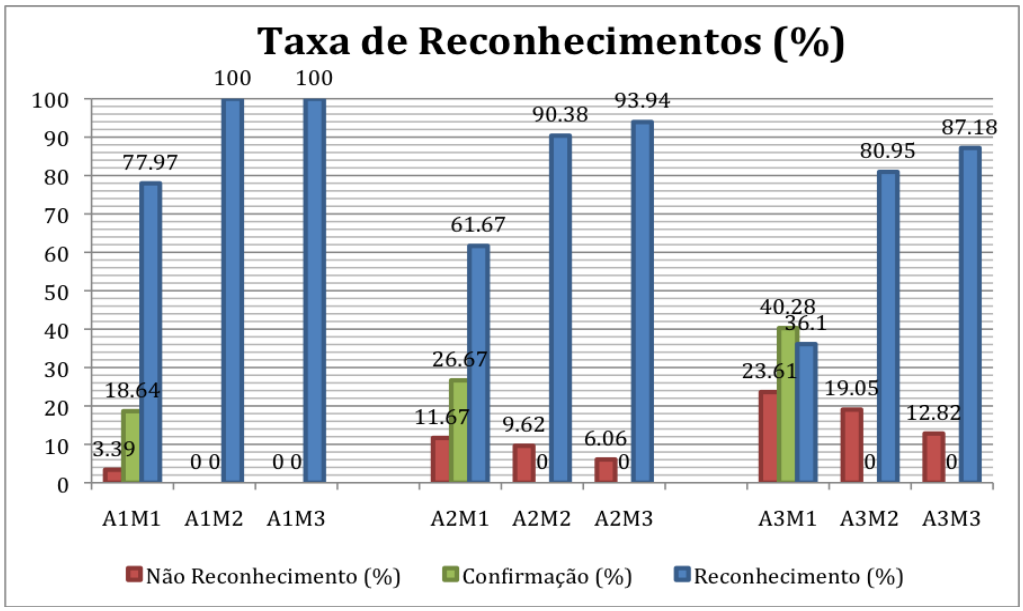

Figura 58 – Reconhecimento, Reconhecimento com Confirmação e Não Reconhecimento.

Na tabela do Anexo F, encontra-se resumida a informação que deu origem às percentagens expressas no anterior gráfico. É também possível analisar o número de vezes, em média, que cada utilizador "fala para o sistema". Quanto menos vezes o utilizador falar para o sistema, menor é a probabilidade de ocorrência de "não reconhecimentos", assim como maior é a probabilidade de se conseguir reduzir o tempo total, necessário à realização da tarefa.

5.2 Análise de Resultados

Relativamente aos resultados apresentados no subcapítulo anterior, é possível constatar que a variação é constante nos modos 2 e 3, mesmo com o aumento do ruído entre os diferentes ambientes de teste.

A principal razão, que terá levado à variação constante entre os modos 2 e 3, foi a utilização de perguntas directas ao utilizador, permitindo direcciona-lo, mas sem lhe tirar o controlo, quanto à evolução da interacção. Com isto, é possível controlar a dispersão do utilizador, relativamente ao fluxo que tencionamos que ele siga, mas que ao mesmo tempo tencionamos que ele tenha a ideia, de que é ele que decide ir, por esse mesmo caminho.

Outra das razões que terá levado à variação constante, terá sido a utilização de gramáticas mais pequenas ("Sim/Não/..."), do que no modo de funcionamento 1, em paralelo com a formulação de perguntas directas.

Deparamo-nos com uma variação significativa, quando passamos do ambiente 2 para o ambiente 3, no modo de funcionamento 1. A principal causa destes resultados, terá sido a utilização de gramáticas grandes, aliadas ao ruído do ambiente em causa.

É possível constatar que existe um bom comportamento em todos os modos de funcionamento, independentemente de se tratar do ambiente 1 ou do ambiente 2. Isto prova que o sistema, mesmo com gramáticas grandes, face à melhor optimização possível, comporta-se bem, tolerando bem o ruído moderado, do ambiente 2.

Relativamente ao tempo necessário, para realização da tarefa de marcar uma consulta, é possível desde logo perceber que existe uma diferença significativa, entre os tempos obtidos por utilizadores experientes e utilizadores inexperientes, o que aliado ao facto de que, por norma, os utilizadores se vão habituando aos sistemas que utilizam, a tendência é claramente irem reduzindo o tempo de que necessitam para a realização da tarefa. Exemplo disso, é a redução de aproximadamente 45% do tempo necessário, para o Ambiente mais ruidoso, no Modo de Funcionamento Base. O mesmo se verifica para as restantes situações de teste, embora não de forma tão acentuada.

Ponderando apenas sobre a optimização que se consegue obter, com a transição entre Modos de Funcionamento, consegue-se perceber que, num ambiente de ruído moderado, existe uma diminuição de 35%, com a passagem do Modo de Funcionamento 1, para o Modo de Funcionamento 2 e de aproximadamente 22% do Modo 2 para o Modo 3, passando de 160 segundos para 104 segundos e de 104 segundos para 81 segundos, respectivamente. Caso se coloque o caso de uma passagem do Modo 1 para o Modo 3, consegue-se uma optimização de aproximadamente 49%, no tempo necessário à realização da tarefa de marcar uma consulta.

Na tabela seguinte, é possível analisar as percentagens em termos de tempo, ganhas em cada um dos Ambientes, fazendo variar o Modo de Funcionamento.

Tabela 10 – Ganhos nas várias conjugações Ambiente e Modo de Funcionamento.

	Modo 1	**Modo 2 (1 → 2)**	**Modo 3 (2 → 3 / 1 → 3)**
Ambiente 1	142s (100%)	97s (~ -31.7%)	74s (~ -23.7% / ~ -47.9%)
Ambiente 2	160s (100%)	104s (-35%)	81s (~ -22.1% / ~ -49.4%)
Ambiente 3	233s (100%)	111s (~ -52.4%)	85s (~ -23.4%/ ~ -63.5%)

Relativamente aos valores obtidos quanto aos reconhecimentos, é possível constatar diversos aspectos, os quais já esperados:

1. aumento do número de não reconhecimentos e por sua vez, a diminuição do número de reconhecimentos, quando se faz variar os ambientes, entre Ambiente 1 e Ambiente 3;
2. diminuição do número de vezes em que o utilizador "fala para o sistema", quando se faz variar os modos de funcionamento, entre o Modo 1 e o Modo 3, mas por outro lado, um aumento desse mesmo valor quando se faz variar o Ambiente, para um mesmo modo de funcionamento.

Relativamente ao gráfico da figura 58 e à tabela 25 (Anexo F), é possível perceber que o valor de "Reconhecimentos com Confirmação", para os Modos de Funcionamento 2 e 3, tem o valor 0 em todos os Ambientes. Isto acontece devido ao facto de, na passagem do Modo 1 para o Modo 2, e posteriormente para o Modo 3, existir optimização ao diálogo, passando essa optimização pelo formulação de perguntas directas ao Utilizador, tendo este apenas de confirmar com "Sim/Não/Certo/...", e não dizer ele próprio, aquilo que pretende, como "Cancerologia/Clínica Geral/...", o que devido à facilidade de reconhecimento atribuída ao primeiro e segundo caso, permite reduzir o número de "Reconhecimento com Confirmação", o qual podia não ser zero, incrementando assim directamente o número de reconhecimentos.
É possível constatar que o número médio de vezes que o utilizador fala para o sistema, aumenta com a variação entre o Ambiente 1, 2 e 3, por esta ordem, muito devido ao aumento do ruído, que leva o utilizador a solicitar uma repetição, em grande parte dos casos.
Um tópico que não se encontra descrito, quer no gráfico, quer na tabela, é a distinção entre os "Reconhecimentos com Confirmação" certos e errados, ou seja, os casos em que o sistema reconheceu acertadamente o que foi dito pelo utilizador, mesmo que com grau de confiança reduzido, e os casos em que reconheceu erradamente o que foi dito pelo utilizador, mas também com grau de confiança reduzido. O número de Reconhecimentos com Confirmação Errados é nulo, devido ao facto de as gramáticas não serem demasiado grandes, mas também devido ao facto de não existirem registos demasiado idênticos nessas gramáticas.

Como análise final e conjunta dos vários aspectos anteriormente abordados, pensa-se ser claro o beneficio que se pode retirar, da optimização dos dois principais factores em discussão neste capítulo, sendo eles o tempo necessário à realização da tarefa, assim como

o aumento da taxa de reconhecimento, e consequente diminuição da taxa de não reconhecimentos e reconhecimentos com confirmação.

Com a optimização a estes níveis, é possível tornar mais fácil o processo de marcação de consultas e ajudar a contribuir para a diminuição do número de casos, em que o utilizador adquire alguma frustração no decorrer da interacção, por não ser devidamente guiado, ou por demorar demasiado tempo para realizar a tarefa que deseja.

6 Conclusão

6.1 Trabalho Efectuado

Após a realização deste trabalho, é grande a satisfação de perceber que os requisitos definidos para o mesmo, foram cumpridos e que desta forma se conseguiu desenhar e implementar um sistema de IVR, que permite efectuar a marcação de consultas médicas, no caso concreto para o SAMS, recorrendo à tecnologia colocada ao dispor pela Link Consulting. Com não menos satisfação, se constata que muito embora não tenham sido impostos, requisitos muito específicos quanto ao mecanismo de regras, que visava optimizar o sistema de marcação de consultas, na sua versão base, se conseguiu desenvolver um módulo, o qual pode ou não ser utilizado com a versão base do sistema, de fácil parametrização, que consegue, em certos casos, optimizações a rondar os 50% de redução de tempo para realização da tarefa, que é efectuar a marcação de uma consulta, assim como de taxas de 100%, associadas ao reconhecimento efectuado com sucesso.

A proposta de solução apresentada, resultou do trabalho de pesquisa, sobre todas as notas encontradas, que se pudessem aplicar ao contexto do problema apresentado, assim como do trabalho individual, de tentar conceber uma solução, que não apenas resolva o problema existente, mas que o resolva de forma inovadora, tirando partido de todas as ferramentas colocadas ao dispor.
Para além do trabalho individual, decorreu um trabalho conjunto com profissionais da área, na Link Consulting, uma vez que a empresa já conta com algumas soluções ao nível dos IVR's, dispondo assim de profissionais competentes, que foram aconselhando quais os rumos a seguir, ao longo das várias iterações do processo de avaliação ao trabalho realizado, que tinham lugar através de reuniões periódicas.
Paralelamente ao principal objectivo associado à realização deste trabalho, existe a geração de *know-how* para a empresa, onde o mesmo foi realizado. Ao longo da evolução do trabalho, com o surgimento de problemas e resolução dos mesmos, era gerada documentação, no sentido de garantir uma mais fácil resolução de problemas idênticos, que viessem a surgir no futuro. Não apenas problemas e notas associadas à sua resolução eram documentados, mas também toda a informação considerada útil ao desenvolvimento deste tipo de aplicações.

No sentido de avaliar a solução apresentada, tentou-se realizar testes à mesma, que correspondessem o mais possível à realidade existente, quando se fala de um sistema de marcação de consultas, através de um IVR. Foi de tremenda utilidade, conseguir pessoas dispostas a despender de mais de trinta minutos, para que fosse possível testar o sistema em vários ambientes e através de vários níveis de optimização. Uma situação de testes desta natureza, permite não só perceber o correcto ou incorrecto funcionamento do sistema,

mas também perceber, até que ponto a tecnologia utilizada se encontra devidamente madura, face às restantes apresentadas no capítulo Estado de Arte.
Para além das ilações que podem ser tiradas pessoalmente, existe um acréscimo nos benefícios de testar um sistema desta natureza, com várias pessoas, e durante algum tempo com cada pessoa. Um dos principais benefícios, consiste nas notas que cada uma das pessoas vai referindo, à medida que vai testando o sistema, que pode levar à realização de ajustes ao próprio sistema.

Após a realização deste trabalho, existe uma versão testada, que cumpre com os requisitos definidos inicialmente, no sentido de permitir realizar uma avaliação ao sistema, aquando este estivesse terminado. Desta forma, este trabalho pode ser visto como um ponto de partida, para novos trabalhos nesta área, com a tecnologia nele utilizada, uma vez que poucos são os projectos conhecidos, tornados públicos, que utilizem estas tecnologias da Microsoft.

Foi um trabalho que permitiu aprofundar alguns conhecimentos ao nível da tecnologia utilizada, mas sobretudo, permitiu perceber aspectos essenciais, relativamente às técnicas de definição de diálogos, para sistemas a utilizar por pessoas. As características podem diferir muito de pessoa para pessoa, não apenas em termos de fala, o que pode levar a comportamentos diferentes por parte do sistema, mas também relativamente ao que para umas faz mais sentido, ao passo que para outras pode não fazer assim tanto sentido, nomeadamente ao nível do fluxo base do sistema.

6.2 Trabalho Futuro

Integração com Sistema de Regras Externos - BizTalk Business Rule Engine (BizTalk BRE)

A integração da solução encontrada, com sistemas de regras externos como o BizTalk BRE, permitirá conferir à solução um grau de extensibilidade superior, uma vez que passa a ser o BRE o responsável pela decisão a tomar, com base nos resultados da aplicação das diversas regras definidas no sistema. É ainda possível a gestão das próprias regras, através da interface disponibilizada pelo BRE.
Paralelamente ao facto, de ser possível transpor para um sistema externo, a aplicação e a gestão das regras a aplicar ao longo da execução do sistema, passa a ser possível disponibilizar essas mesmas regras, para outras aplicações, que naturalmente se encontrem no mesmo contexto do sistema abordado ao longo deste documento, nomeadamente a marcação de consultas no Serviço de Assistência Médica Social.

Incorporar actualizações ao Speech Server 2007

A incorporação das actualizações, quer ao nível do reconhecedor, quer ao nível do sintetizador, ao Microsoft Speech Server 2007, permitirão melhorar significativamente o resultado final de qualquer aplicação, que utilize esta tecnologia.
A taxa de reconhecimento passará para níveis mais aceitáveis, o que aliado às boas práticas para a definição de diálogos, poderá conferir um nível de satisfação final superior, comparativamente ao actual. O melhoramento ao nível do reconhecedor, deve-se ao enriquecemento do vocabulário utilizado, permitindo assim, melhorar as taxas de reconhecimento com sucesso.
As melhorias ao nível do sintetizador, passam pela incorporação de uma voz mais natural, comparativamente à actualmente utilizada pela plataforma, que de certa forma contribuirá também, para o aumento do nível de satisfação do utilizadores, nas soluções que façam uso destas tecnologias.

Sistema IVR para confirmação das consultas agendadas

No sentido de desenvolver um sistema mais completo, embora um pouco fora do âmbito do trabalho proposto, surgiu no decorrer de algumas reuniões realizadas na Link Consulting, a proposta de incorporar ao sistema de marcação de consultas, que recebe chamadas efectuadas pelos utilizadores, que tencionam efectuar uma nova marcação, um outro sistema paralelo, que vai confirmando todas as consultas marcadas pelo anterior, sendo que desta vez, é o sistema a realizar a chamada para o utilizador, a informá-lo da existência de uma consulta agendada.
O sistema deveria ser configurável, constituindo uma parte do sistema geral, podendo funcionar em conjunto, no entanto deveria também funcionar como um módulo independente, a actuar sobre uma fonte da dados especifica, tal como o projecto detalhado neste documento.

7 Referências

1. **Speech Technology.** Introduction and Overview of W3C Speech Interface Framework. [Online] http://en.wikipedia.org/wiki/Voice_browser.
2. **Voice Browser.** Applying Web technology to enable users to access services from their telephone via a combination of speech and DTMF. [Online] http://www.w3.org/Voice/
3. **Speech Synthesis (Speech at CMU).** Speech Synthesis Introduction. [Online] http://www.speech.cs.cmu.edu/comp.speech/FAQ5.html, http://en.wikipedia.org/wiki/Speech_synthesis
4. **Speech Recognition (Speech at CMU).** Speech Recognition Introduction. [Online] http://www.speech.cs.cmu.edu/comp.speech/FAQ6.html, http://en.wikipedia.org/wiki/Speech_recognition
5. **Speaker Recognition (Speech Authentication).** Speaker Recognition Introduction. [Online] http://www.speech.cs.cmu.edu/comp.speech/Section6/Q6.6.html, http://en.wikipedia.org/wiki/Speaker_recognition
6. **Programming Models.** Programming Models Introduction. [Online] http://msdn2.microsoft.com/en-us/library/bb801420.aspx
7. **SALT Forum. SALT Specification.** [Online] http://www.saltforum.org/, http://xml.coverpages.org/salt.html
8. **Cisco and Microsoft, Voice News.** [Online] http://newsroom.cisco.com/dlls/prod_101501.html, http://www.microsoft.com/presspass/features/2002/mar02/03-05standards.mspx
9. **VoiceXML Forum.** [Online] http://www.voicexml.org, http://www.voicexml.org/tech_bkgrnd.html
10. **Multimodal Requirements for Voice Markup Languages.** [Online] http://www.w3.org/TR/multimodal-reqs
11. **Gunnar Fiedler, Peggy Schmidt.** *Developing Interactive Voice Response Interfaces for Large Information Systems.* [Paper] Institute for Computer Science and Applied Mathematics, Germany.
12. **Trancoso, Isabel, Diamantino Caseiro, Rui Amaral, Frederico Rodrigues, A. Serralheiro, Fernando Perdigão, Eduardo Sá Marta, Carlos Espain, Vítor Pera, Luís Moreira.** *An Overview of the REC Project - Speech Recognition Applied to Telecommunications.* [Paper] INESC-Lisboa, IT-Coimbra, FEUP.
13. **Rudžionis, A., K. Ratkevičius, R. Maskeliūnas, V. Rudžionis.** *Review of Voice Dialogues in Telecommunications.* [Paper] Speech Research Laboratory, Kaunas University of Technology, Dept.of Informatics, Kaunas Humanities Faculty of Vilnius University. Kaunas, Lithuania.

14. **Shi, Hao, Sebastian Auer.** *Telephony Applications using Microsoft Speech Server.* [Paper] School of Computer Science and Mathematics Victoria University, Melbourne, Australia. IJCSNS International Journal of Computer Science and Network Security, VOL.6 No.8A, August 2006.
15. **Xiaoqing, Yang.** *Speech-Enabled Tourist Information System.* [Paper] Brief Studies in Computer Science (Fall 2003). P. 78 - 89. Department Of Computer Sciences University Of Tampere. Tampere 2004.
16. **Scheele, Art.** *Voice Self-Service Aligns with Web Architectures to Reduce TCO, Increase Flexibility and Ensure Content Consistency*. [Paper] Customer-Centric Strategies. Yankee Group. 2006.
17. **Lane, Ian R., Shinichi Ueno, Tatsuya Kawahara.** *Cooperative Dialogue Planning with User and Situation Models via Example-based Training.* [Paper] School of Informatics, Kyoto University Yoshida-Hommachi, Sakyo-ku, Kyoto 606-8501, Japan.
18. **Dey, Anind K., Lara D. Catledge, Gregory D. Abowd and Colin Potts.** *Developing Voice-only Applications in the Absence of Speech Recognition Technology.* [Paper] Graphics, Visualization, and Usability Center. Georgia Institute of Technology. Atlanta, USA.
19. **Schnelle, Dirk, Fernando Lyardet.** *Voice User Interface Design Patterns.* [Paper] Telecoperation Group, Darmstadt University of Technology, Darmstadt, Germany.
20. **Choularton, Stephen.** *Handling Speech Recognition Errors in Spoken Language Dialogue Systems.* [Paper] Center for Language Technology Macquarie University, Sydney.
21. **Mecanovic, Daniel, Hao Shi.** *Voice User Interface Design for a Telephone Application Using VoiceXML.* [Paper] School of Computer Science and Mathematics, Victoria University. Australia.
22. **Yamazaki, Yasushi, Hitoshi Iwamida, Kazuhiro Watanabe.** *Technologies for Voice Portal Platform.* [Paper] FUJITSU Sci. Tech. J. 2004.
23. **SALT Forum.** *Speech Application Language Tags (SALT) Technical White Paper.* [Paper] SALT Forum.
24. **VoiceXML Forum.** *Voice eXtensible Markup Language – VoiceXML Specification.* [Paper] VoiceXML Forum.
25. **Walker, Marilyn A., Diane J. Litman, Candace A. Kamm, Alicia Abella.** *Evaluating Interactive Dialogue Systems: Extending Component Evaluation to Integrated System Evaluation.* [Paper] AT&T Labs Research, Florham Park. USA.
26. **Kwok, Derek.** *Developing a Voice Application: A Comparison Between VoiceXML and SALT.* [Paper] VoiceGenie Technologies Inc. Toronto, Ontario.
27. **Duggan, Bryan, Mark Deegan.** *Considerations In The Usage Of Text To Speech (TTS) In The Creation Of Natural Sounding Voice Enabled Web Systems.* [Paper] School of Informatics, National College of Ireland, Ireland. School of Computing, Dublin Institute of Technology, Ireland.

28. **Facco (1), A., D. Falavigna (1), R. Gretter (1), M.Viganò (2).** *On the Development of Telephone Applications: Some Practical Issues and Evaluation.* [Paper] (1) ITC-Irst, Instituto per la Ricerca Scientifica e Tecnologica, Trento. (2) Reitek Spa, Milano. Italy.
29. **Steinbiss, V.,** *Speech Dialogue Systems In The Telecom Network.* [Paper] Philips Speech Processing Kackertstr. 10, 52072 Aachen. Germany.
30. **Fiedler (1), Gunar, Bernhard Thalheim (1), Peggy Schmidt (1), Thomas Schwanzara-Bennoit (2).** *State-, Hhml-, And Object-Based Dialog Design For Voice-Web Applications.* [Paper] (1) Computer Science Institute, University Kiel, Germany. (2) Computer Science Department, Databases and Information Systems Group, Brandenburg University of Technology at Cottbus, Germany.

8 Anexos

Anexo A – Exemplos de fluxos de diálogo (textual)

Nota: As situações de teste são limitadas, devido à quantidade limitada de informação existente na base de dados.

########################## --- AUTENTICAÇÃO --- ##########################

- **Erro de Autenticação**

IVR: *“Bem-vindo ao sistema de marcação de consultas do SAMS.”*
IVR: *"Por favor, diga ou digite o seu número de beneficiário."*
USER: *“2222”*
IVR: *“Por favor, digite o seu código pessoal.”*
USER: *“2322”*
IVR: *"Erro de autenticação. Número de beneficiário, ou código inválido. Dispõe de: duas tentativas."*
IVR: *"Por favor, diga ou digite o seu número de beneficiário."*
USER: *“2222 2222”*
IVR: *“Por favor, digite o seu código pessoal.”*
USER: *“2322”*
IVR: *" Erro de autenticação. Número de beneficiário, ou código inválido. Dispõe de: uma tentativa."*
IVR: *"Por favor, diga ou digite o seu número de beneficiário."*
USER: *“2222”*
IVR: *“Por favor, digite o seu código pessoal.”*
USER: *“2322”*
IVR: *“Erro de autenticação. Número de beneficiário, ou código inválido. Esgotou o número de tentativas. Por favor, tente mais tarde.”*
IVR: *“Obrigado. Adeus.”*

- **Autenticação com sucesso**

IVR: *“Bem-vindo ao sistema de marcação de consultas do SAMS.”*
IVR: *"Por favor, diga ou digite o seu número de beneficiário."*
USER: *“2222”*
IVR: *“Por favor, digite o seu código pessoal.”*
USER: *“2222”*

IVR: *“A qualquer momento diga: Repetir, para ouvir novamente. Cancelar, para voltar ao estado anterior. Ajuda, para obter ajuda. Terminar, para terminar a conversação.”*
IVR: *"Diga o nome do médico ou a especialidade."*
(...)

########## --- ESCOLHA DE MÉDICO --- ##########

- **Médico sem Especialidade**

(...)
USER: “*Tânia Barbosa”*
(IVR: *“Disse Tânia Barbosa. Certo?”)*
(USER: *“Sim”)*
IVR: *“Para Tânia Barbosa. Não existem especialidades.”*
IVR: *“Diga o nome do médico ou a especialidade.”*
(...)

- **Médico com uma Especialidade, vários Locais e um Horário.**

(...)
USER: “*Horácio Silva”*
(IVR: *“Disse Horácio Silva. Certo?”)*
(USER: *“Sim”)*
IVR: *“Diga, de entre os seguintes, o nome do Local que pretende: Hospital, Centro Clínico.”*
USER: *“Centro Clínico”*
(IVR: *“Disse Centro Clínico. Certo?”)*
(USER: *“Sim”)*
IVR: *“Confirma a marcação: para Audiometria, com Horário Silva, em Centro Clínico, no dia 24 de Março de 2008 (24-03-2008), dez e trinta (10:30). Sim ou Não?“*
USER: *“Não”*
IVR: *"Marcação cancelada. Deseja efectuar uma nova marcação? Sim ou Não?"*
USER: *“Sim”*
IVR: *"Diga o nome do médico ou a especialidade."*
(...)

- **Médico com várias Especialidades, vários Locais e vários Horários.**

(...)
USER: “*Leonor Bento”*
(IVR: *“Disse Leonor Bento. Certo?”)*
(USER: *“Sim”)*
IVR: *“Diga, de entre as seguintes, o nome da Especialidade que pretende: Cardiologia, Cardiologia Infantil.”*

USER: *"Cardiologia"*

(IVR: *"Disse Cardiologia. Certo?")*

(USER: *"Sim")*

IVR: *"Diga, de entre os seguintes, o nome do Local que pretende: Hospital, Centro Clínico."*

USER: *"Hospital"*

(IVR: *"Disse Hospital. Certo?")*

(USER: *"Sim")*

IVR: *"Caso pretenda um dos horários listados de seguida, diga o número correspondente ao que pretende. UM: 24 de Março de 2008 (24-03-2008), catorze e trinta (14:30). DOIS: 24 de Março de 2008 (24-03-2008), quinze e trinta (15:30)"*

USER: *"Dois"*

IVR: *"Confirma a marcação: para Cardiologia, com Leonor Bento, em Hospital, no dia 24 de Março de 2008 (24-03-2008), quinze e trinta (15:30). Sim ou Não?"*

USER: "*Sim"*

IVR: *"Marcação efectuada com sucesso."*

IVR: *"Obrigado. Adeus."*

########## **--- ESCOLHA DE ESPECIALIDADE ---** ##########

- **Especialidade sem Médico**

(...)

USER: "*Alergologia"*

(IVR: *"Disse Alergologia. Certo?")*

(USER: *"Sim")*

IVR: *"Para Alergologia. Não existem médicos."*

IVR: *"Diga o nome do médico ou a especialidade."*

(...)

- **Especialidade com um Médico, vários Locais e um Horário.**

(...)

USER: "*Cancerologia"*

(IVR: *"Disse Cancerologia. Certo?")*

(USER: *"Sim"*

IVR: *"Diga, o nome do Local que pretende."*

USER: *"Hospital"*

(IVR: *"Disse Hospital. Certo?")*

(USER: *"Sim")*

IVR: *"Confirma a marcação: para Cancerologia, com Gabriela Lopes, em Hospital, no dia 24 de Março de 2008 (24-03-2008), dez e trinta (10:30). Sim ou Não?"*

USER: *"Não"*

IVR: *"Marcação cancelada. Deseja efectuar uma nova marcação? Sim ou Não?"*

USER: *"Sim"*

IVR: *"Diga o nome do médico ou a especialidade."*

(...)

- **Médico com várias Especialidades, vários Locais e vários Horários.**

(...)

USER: "*Cardiologia Infantil"*

(IVR: *"Disse Cardiologia Infantil. Certo?")*

(USER: *"Sim")*

IVR: *"Diga, de entre os seguintes, o nome do Médico que pretende: Leonor Bento, João Macedo."*

USER: *"Leonor Bento"*

(IVR: *"Disse Leonor Bento. Certo?")*

(USER: *"Sim")*

IVR: *"Diga, de entre os seguintes, o nome do Local que pretende: Hospital, Centro Clínico."*

USER: *"Hospital"*

(IVR: *"Disse Hospital. Certo?")*

(USER: *"Sim")*

IVR: *"Caso pretenda um dos horários listados de seguida, diga o número correspondente ao que pretende. UM: 24 de Março de 2008 (24-03-2008), catorze e trinta (14:30). DOIS: 24 de Março de 2008 (24-03-2008), quinze e trinta (15:30)"*

USER: *"Dois"*

IVR: *"Confirma a marcação: para Cardiologia Infantil, com Leonor Bento, em Hospital, no dia 24 de Março de 2008 (24-03-2008), quinze e trinta (15:30). Sim ou Não?"*

USER: "*Sim"*

IVR: *"Marcação efectuada com sucesso."*

IVR: *"Obrigado. Adeus."*

########## **--- EXTRA ---** ##########

- **Situação 1**

(...)

IVR: *"Diga o nome do médico ou a especialidade."*

USER: "*Barbosa"*

(IVR: *"Disse Barbosa. Certo?")*

(USER: *"Sim")*

IVR: *"Supomos que se refere a: Tânia Barbosa."*

IVR: *"Para Tânia Barbosa. Não existem especialidades."*

IVR: *“Diga o nome do médico ou a especialidade.”*

(...)

- **Situação 2**

(...)

IVR: *"Diga o nome do médico ou a especialidade."*

USER: “*Lopes”*

(IVR: *“Disse Lopes. Certo?”)*

(USER: *“Sim”)*

IVR: *“Diga, de entre os seguintes, o médico ao qual se refere: Lopes Menezes, Gabriela Lopes, André Lopes.”*

USER: “*André Lopes”*

(IVR: *“Disse André Lopes. Certo?”)*

(USER: *“Sim”)*

IVR: *“Para André Lopes. Não existem especialidades.”*

IVR: *“Diga o nome do médico ou a especialidade.”*

USER: “*Sair”*

IVR: *“Obrigado. Adeus.”*

Anexo B – Fluxos de Execução

AuthenticationError

A validação relativa ao número de beneficiário e código pessoal indicado falha. São dadas três tentativas ao paciente. Ao final da terceira tentativa, o paciente é alertado face ao sucedido e o fluxo de execução termina, após a mensagem de despedida.

Figura 59 – (Anexo) Estado AuthenticationError.

AskForTime

Obter informação relativamente ao Horário, para o qual se pretende efectuar a marcação da consulta, tendo em conta que já se sabe qual o Médico, qual a Especialidade e qual o Local. São obtidos todos os Horários, num dado Local, onde o Médico escolhido, dê consultas da Especialidade escolhida. No caso de não existirem o paciente será notificado da situação, no caso de serem mais que um, estes serão listados e será pedido ao paciente que diga o número correspondente ao Horário que pretende, no caso de ser apenas um, o processo seguirá para a formulação da questão final, tendo em conta toda a informação recolhida até então.

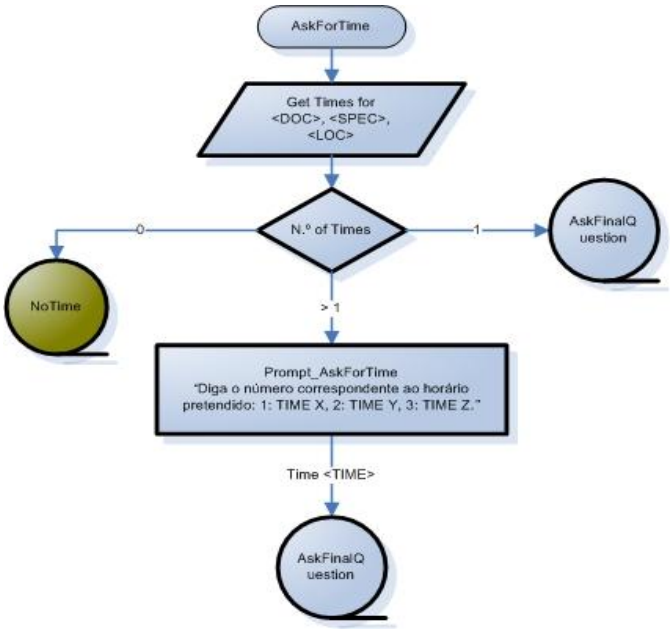

Figura 60 – (Anexo) Obtenção de informação relativa ao Horário

Goodbye

Estado final do fluxo de execução. Mensagem de despedida do sistema de marcação de consultas. Término do fluxo de execução.

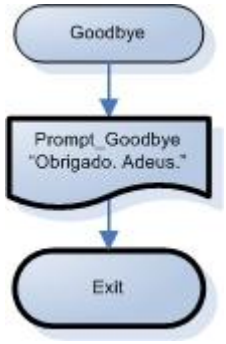

Figura 61 – (Anexo) Mensagem de despedida. Fim de interacção.

NonExistingDoctor

Situação de erro, após ter sido dito pelo paciente, um nome inválido para um Médico. Depois de alertar o paciente, o processo volta ao início, onde é dada a hipótese de escolher entre Médico ou Especialidade.

Figura 62 – (Anexo) Estado NonExistingDoctor.

NonExistingSpeciality

Situação de erro, após ter sido dito pelo paciente, um nome inválido para uma Especialidade. Depois de alertar o paciente, o processo volta ao início, onde é dada a hipótese de escolher entre Médico ou Especialidade.

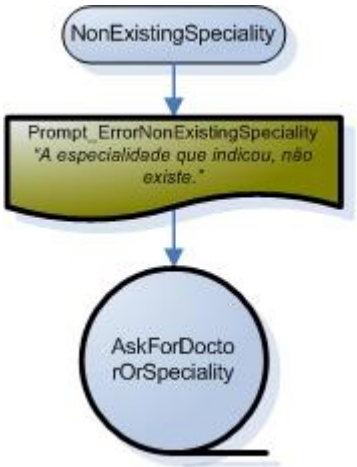

Figura 63 – (Anexo) Estado NonExistingSpeciality.

ErrorNoDoctor

Situação de erro, na qual não existem Médicos para a Especialidade escolhida pelo paciente. Após alertar o paciente, o processo volta ao início, onde é dada a hipótese de escolher entre Médico ou Especialidade.

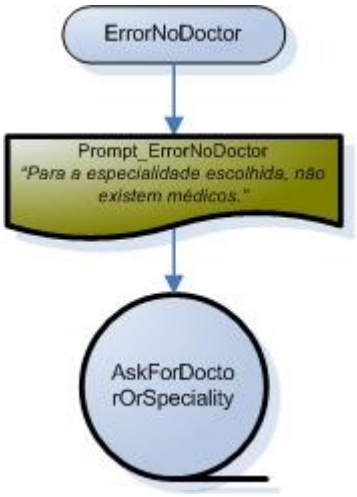

Figura 64 – (Anexo) Estado ErrorNoDoctor.

ErrorNoSpeciality

Situação de erro, na qual não existem Especialidades para o Médico escolhido pelo paciente. Após alertar o paciente, o processo volta ao início, onde é dada a hipótese de escolher entre Médico ou Especialidade.

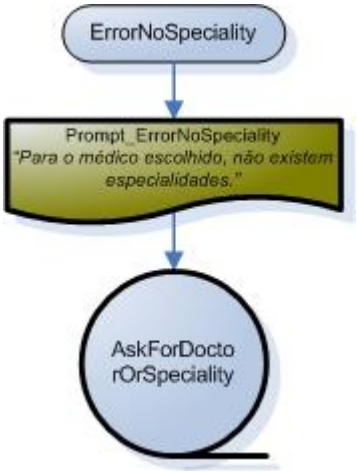

Figura 65 – (Anexo) Estado ErrorNoSpeciality.

ErrorNoLocality

Situação de erro, na qual não existem Locais, onde o Médico escolhido, dê consultas da Especialidade escolhida. Após alertar o paciente, o processo volta ao início, onde é dada a hipótese de escolher entre Médico ou Especialidade.

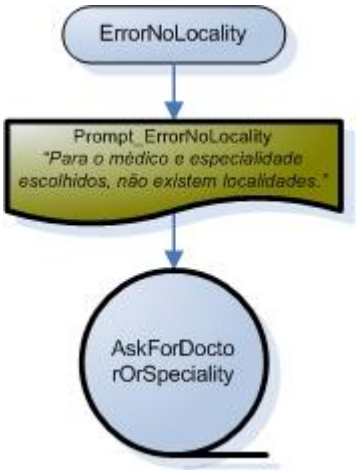

Figura 66 – (Anexo) Estado ErrorNoLocality.

ErrorNoTime

Situação de erro, na qual não existem Horários para o Local escolhido, onde o Médico escolhido, dê consultas da Especialidade escolhida. Após alertar o paciente, o processo volta ao início, onde é dada a hipótese de escolher entre Médico ou Especialidade.

Figura 67 – (Anexo) Estado ErrorNoTime.

Fluxo Geral

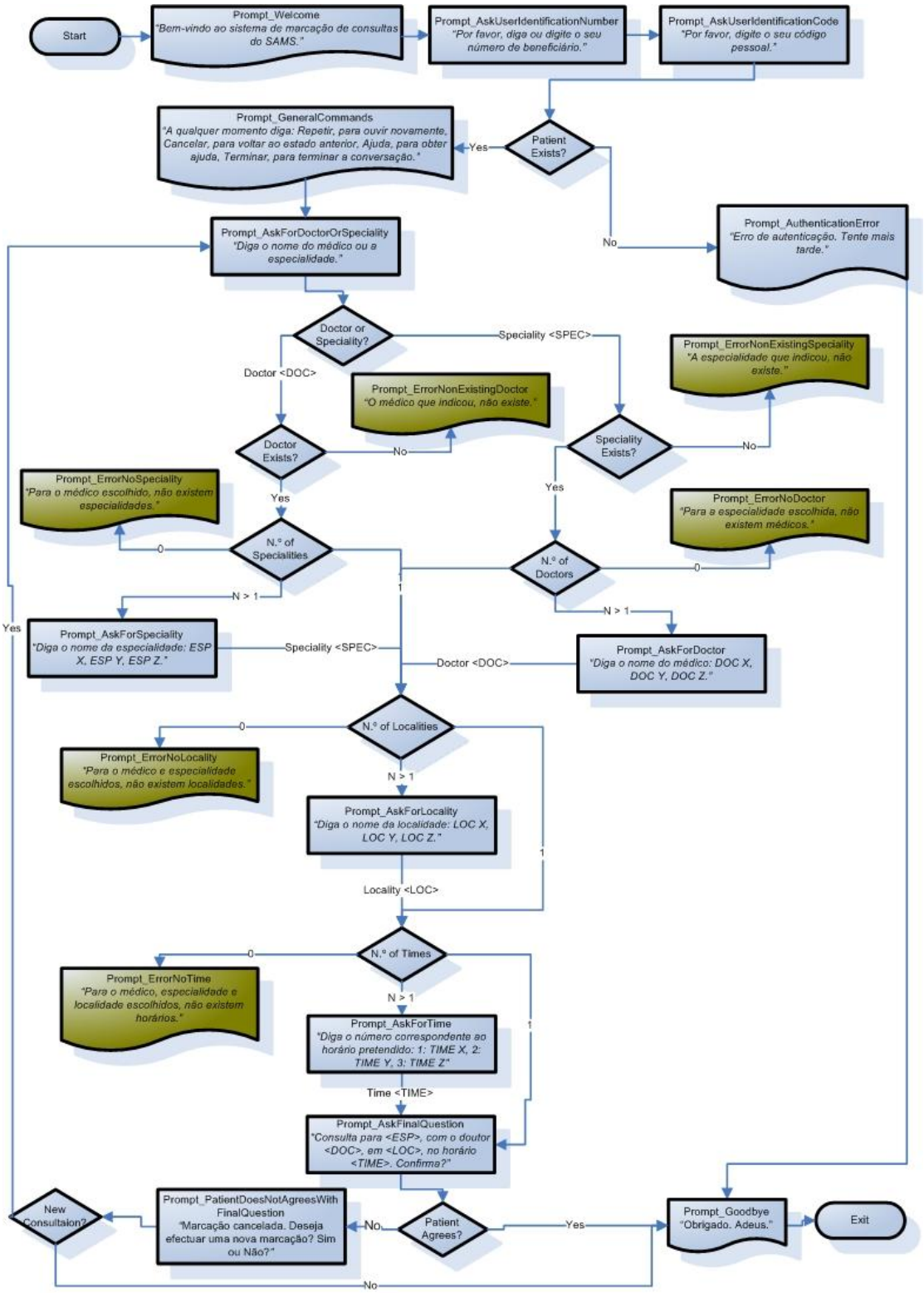

Figura 68 – (Anexo) Visão geral do fluxo de interacção

Anexo C – Exemplo de aplicação de regras, num contexto de pesquisa de serviços por zona

Imagine-se um utilizador que liga a um sistema de informações, o qual, com recurso a módulos internos, cuja especificação não interessa para o caso, conseguiria prever que o utilizador iria pesquisar serviços em "*New York*", como resposta à pergunta "*Diga a cidade na qual tenciona efectuar uma pesquisa de serviços.*". Se o sistema dispunha de uma percentagem de certeza de 100%, relativamente à resposta do utilizador, não faria sentido juntar na gramática "*Newark*", pois poderia levar a não reconhecimentos ou falsos reconhecimentos, na hora de o utilizador dizer a cidade na qual pretendia efectuar uma pesquisa de serviços.

Como alternativa, uma vez que a percentagem de certeza é de 100%, o sistema podia inclusivamente perguntar ao utilizador, "Deseja efectuar uma pesquisa em New York?". Ao contrário da situação em que o utilizador tem de dizer o nome da cidade, aqui bastava-lhe confirmar, com "Sim/Não", reduzindo assim a probabilidade de não reconhecimento, mas também o tempo necessário à realização da tarefa.

Anexo D – Estados de Diálogo

Tabela 11 – (Anexo) GeneralPrompts.

Name	*GeneralPrompts*
Description	*Conjunto de prompts comuns a vários estados do workflow.*
Prompts	*GeneralPrompts.Silence → "Tem de escolher uma das opções."* *GeneralPrompts.Repeat → "Por favor, repita."*
Grammars	-
Universal Commands	-

Tabela 12 –(Anexo) AskForSpeciality.

Name	*Prompt_AskForSpeciality*
Description	*Pedir ao paciente que indique a ESPECIALIDADE desejada, sabendo quem é o MÉDICO.*
Prompts	*Prompt_AskForSpeciality.MainPrompt → "Diga, de entre as seguintes, a especialidade que pretende: ESP X, ESP Y, ESP Z."* *Prompt_AskForSpeciality.SilencePrompt → GeneralPrompts.Silence* *Prompt_AskForSpeciality.NoRecognitionPrompt* *→ "A especialidade que indicou não foi reconhecida."* *→ "Não entendi."*
Grammars	*Voice (Gramática dinâmica com os nomes das ESPECIALIDADES de um determinado MÉDICO)*
Universal Commands	*Repeat → Prompt_AskForSpeciality_Repeat* *→ GOTO: "Prompt_AskForSpeciality"* *Cancel → Prompt_AskForSpeciality_Cancel* *→ GOTO: "Prompt_AskForDoctorOrSpeciality"* *Help → Prompt_AskForSpeciality_Help* *→ "Por favor, diga qual a Especialidade para a qual pretende efectuar a marcação com <DOC>."* *→ GOTO: "Prompt_AskForSpeciality"* *Goodbye → GOTO: "Prompt_Goodbye"*

Tabela 13 – (Anexo) AskForDoctor.

Name	*Prompt_AskForDoctor*
Description	*Pedir ao paciente que indique o MÉDICO desejado, sabendo qual é a ESPECIALIDADE.*
Prompts	*Prompt_AskForDoctor.MainPrompt → "Diga, de entre os seguintes, o Médico que pretende: DOC X, DOC Y, DOC Z."* *Prompt_AskForDoctor.SilencePrompt → GeneralPrompts.Silence* *Prompt_AskForDoctor.NoRecognitionPrompt* *→ "O médico que indicou não foi reconhecido."* *→ "Não entendi."*
Grammars	*Voice (Gramática dinâmica com os nomes dos MÉDICOS de uma determinada ESPECIALIDADE)*
Universal Commands	*Repeat → Prompt_AskForDoctor_Repeat* *→ GOTO: "Prompt_AskForDoctor"* *Cancel → Prompt_AskForDoctor_Cancel* *→ GOTO: "Prompt_AskForDoctorOrSpeciality"* *Help → Prompt_AskForDoctor_Help* *→ "Por favor, diga qual o Médico para o qual pretende efectuar a marcação, na*

	Especialidade <ESP>." *→ GOTO: "Prompt_AskForDoctor"* *Goodbye → GOTO: "Prompt_Goodbye"*

Tabela 14 –(Anexo) AskUserIdentificationCode.

Name	*Prompt_AskUserIdentificationCode*
Description	*Solicitação do código pessoal ao utente.*
Prompts	*Prompt_AskUserIdentificationCode.MainPrompt → "Por favor, digite o seu código pessoal."* *Prompt_AskUserIdentificationCode.SilencePrompt → GeneralPrompts.Silence* *Prompt_AskUserIdentificationCode.NoRecognitionPrompt → "Formato inválido."*
Grammars	*DTMF*
Universal Commands	*Repeat → Prompt_AskUserIdentificationCode_Repeat* *→ GOTO: "Prompt_AskUserIdentificationCode"* *Cancel → Prompt_AskUserIdentificationCode_Cancel* *→ GOTO: "Prompt_Goodbye"* *Help → Prompt_AskUserIdentificationCode_Help* *→ "Ajuda: Por favor, digite o seu código pessoal, composto por 4 dígitos. Um digito de cada vez."* *→ GOTO: "Prompt_AskUserIdentificationCode"* *Goodbye → GOTO: "Prompt_Goodbye"*

Tabela 15 – (Anexo) AuthenticationError.

Name	*Prompt_AuthenticationError*
Description	*Mensagem de alerta, pelo facto de a autenticação não ter sido efectuada com sucesso .*
Prompts	*Prompt_AuthenticationError.MainPrompt → "Erro de autenticação. Tente mais tarde."*
Grammars	*-*
Universal Commands	*Repeat → Prompt_AuthenticationError_Repeat* *Cancel → Prompt_AuthenticationError_Cancel* *Help → Prompt_AuthenticationError_Help* *Goodbye → Prompt_AuthenticationError_Goodbye*

Tabela 16 – (Anexo) AskForLocality.

Name	*Prompt_AskForLocality*
Description	*Pedir ao paciente que indique o LOCAL desejado, sabendo qual é a ESPECIALIDADE e o MÉDICO.*
Prompts	*Prompt_AskForLocality.MainPrompt → "Diga, de entre os seguintes, o Local que pretende: LOC X, LOC Y, LOC Z."* *Prompt_AskForLocality.SilencePrompt → GeneralPrompts.Silence* *Prompt_AskForLocality.NoRecognitionPrompt* *→ "O local que indicou não foi reconhecido."* *→ "Não entendi."*
Grammars	*Voice (Gramática dinâmica com os nomes dos LOCAIS, onde um determinado MÉDICO dá consultas de uma determinada ESPECIALIDADE)*
Universal Commands	*Repeat → Prompt_AskForLocality_Repeat* *→ GOTO: "Prompt_AskForLocality"* *Cancel → Prompt_AskForLocality_Cancel* *→ GOTO: "Prompt_AskForDoctor" OR "Prompt_AskForSpeciality"*

	Help → Prompt_AskForLocality_Help *→ "Por favor, diga qual o Local onde pretende efectuar a marcação, com <DOC> , para a Especialidade <ESP>."* *→ GOTO: "Prompt_AskForLocality"* *Goodbye → GOTO: "Prompt_Goodbye"*

Tabela 17 – (Anexo) AskForTime.

Name	*Prompt_AskForTime*
Description	*Pedir ao paciente que indique o HORÁRIO desejado, sabendo qual é a ESPECIALIDADE, o MÉDICO e o LOCAL.*
Prompts	*Prompt_AskForTime.MainPrompt → "Diga o número correspondente ao horário pretendido: 1: TIME X, 2: TIME Y, 3: TIME Z."* *Prompt_AskForTime.SilencePrompt → GeneralPrompts.Silence* *Prompt_AskForTime.NoRecognitionPrompt* *→ "O horário que indicou não foi reconhecido."* *→ "Não entendi."*
Grammars	*Voice (Gramática dinâmica com os HORÁRIOS de um LOCAL, onde um determinado MÉDICO dá consultas de uma determinada ESPECIALIDADE)*
Universal Commands	*Repeat → Prompt_AskForTime_Repeat* *→ GOTO: "Prompt_AskForTime"* *Cancel → Prompt_AskForTime_Cancel* *→ GOTO: "Prompt_AskForLocality"* *Help → Prompt_AskForTime_Help* *→ "Por favor, diga qual o Horário que pretende para efectuar marcação, em <LOC>, com <DOC> , na Especialidade <ESP>."* *→ GOTO: "Prompt_AskForTime"* *Goodbye → GOTO: "Prompt_Goodbye"*

Tabela 18 –(Anexo) Goodbye.

Name	*Prompt_Goodbye*
Description	*Mensagem de despedida do sistema de marcação de consultas.*
Prompts	*Prompt_Goodbye.MainPrompt → "Obrigado. Adeus."*
Grammars	*-*
Universal Commands	*Repeat → Prompt_Goodbye_Repeat* *Cancel → Prompt_Goodbye_Cancel* *Help → Prompt_Goodbye_Help* *Goodbye → Prompt_Goodbye_Goodbye*

Tabela 19 – (Anexo) ErrorNonExistingSpeciality.

Name	*Prompt_ErrorNonExistingSpeciality*
Description	*Indicar ao paciente que não existe a ESPECIALIDADE indicada.*
Prompts	*Prompt_ErrorNonExistingSpeciality.MainPrompt → "A especialidade que indicou, não existe."*
Grammars	*-*
Universal Commands	*Repeat → Prompt_ErrorNonExistingSpeciality_Repeat* *Cancel → Prompt_ErrorNonExistingSpeciality_Cancel* *Help → Prompt_ErrorNonExistingSpeciality_Help*

	Goodbye → GeneralPrompts.Goodbye *GOTO: "Prompt_AskForDoctorOrSpeciality"*

Tabela 20 – (Anexo) ErrorNonExistingDoctor.

Name	*Prompt_ErrorNonExistingDoctor*
Description	*Indicar ao paciente que não existe o MÉDICO indicado.*
Prompts	*Prompt_InitialMenu.MainPrompt → "O médico que indicou, não existe."*
Grammars	-
Universal Commands	*Repeat → Prompt_ErrorNonExistingDoctor_Repeat* *Cancel → Prompt_ErrorNonExistingDoctor_Cancel* *Help → Prompt_ErrorNonExistingDoctor_Help* *Goodbye → GeneralPrompts.Goodbye* *GOTO: "Prompt_AskForDoctorOrSpeciality"*

Tabela 21 – (Anexo) ErrorNoDoctor.

Name	*Prompt_ErrorNoDoctor*
Description	*Indicar ao paciente que não existe MÉDICO, sabendo qual é a ESPECIALIDADE.*
Prompts	*Prompt_ErrorNoDoctor.MainPrompt → "Para a especialidade escolhida, não existem médicos."*
Grammars	-
Universal Commands	*Repeat → Prompt_ErrorNoDoctor_Repeat* *Cancel → Prompt_ErrorNoDoctor_Cancel* *Help → Prompt_ErrorNoDoctor_Help* *Goodbye → Prompt_ErrorNoDoctor_Goodbye* *GOTO: "Prompt_AskForDoctorOrSpeciality"*

Tabela 22 – (Anexo) ErrorNoSpeciality.

Name	*Prompt_ErrorNoSpeciality*
Description	*Indicar ao paciente que não existe ESPECIALIDADE, sabendo qual é o MÉDICO.*
Prompts	*Prompt_ErrorNoSpeciality.MainPrompt → "Para o médico escolhido, não existem especialidades."*
Grammars	-
Universal Commands	*Repeat → Prompt_ErrorNoSpeciality_Repeat* *Cancel → Prompt_ErrorNoSpeciality_Cancel* *Help → Prompt_ErrorNoSpeciality_Help* *Goodbye → Prompt_ErrorNoSpeciality_Goodbye* *GOTO: "Prompt_AskForDoctorOrSpeciality"*

Tabela 23 – (Anexo) ErrorNoLocality.

Name	*Prompt_ErrorNoLocality*
Description	*Indicar ao paciente que não existe LOCAL, sabendo qual é o MÉDICO e a ESPECIALIDADE.*
Prompts	*Prompt_ErrorNoLocality.MainPrompt → "Para o médico e especialidade escolhidos, não existem locais."*
Grammars	-
Universal Commands	*Repeat → Prompt_ErrorNoLocality_Repeat* *Cancel → Prompt_ErrorNoLocality_Cancel* *Help → Prompt_ErrorNoLocality_Help* *Goodbye → Prompt_ErrorNoLocality_Goodbye*

	GOTO: "Prompt_AskForDoctorOrSpeciality"

Tabela 24 – (Anexo) ErrorNoTime.

Name	*Prompt_ErrorNoTime*
Description	*Indicar ao paciente que não existe HORÁRIO, sabendo qual é o MÉDICO, a ESPECIALIDADE e o LOCAL.*
Prompts	*Prompt_ErrorNoTime.MainPrompt → "Para o médico, especialidade e local escolhidos, não existem horários."*
Grammars	-
Universal Commands	*Repeat → Prompt_ErrorNoTime_Repeat* *Cancel → Prompt_ErrorNoTime_Cancel* *Help → Prompt_ErrorNoTime_Help* *Goodbye → Prompt_ErrorNoTime_Goodbye* *GOTO: "Prompt_AskForDoctorOrSpeciality"*

Anexo E – Comparação dos três principais fluxos (genericamente)

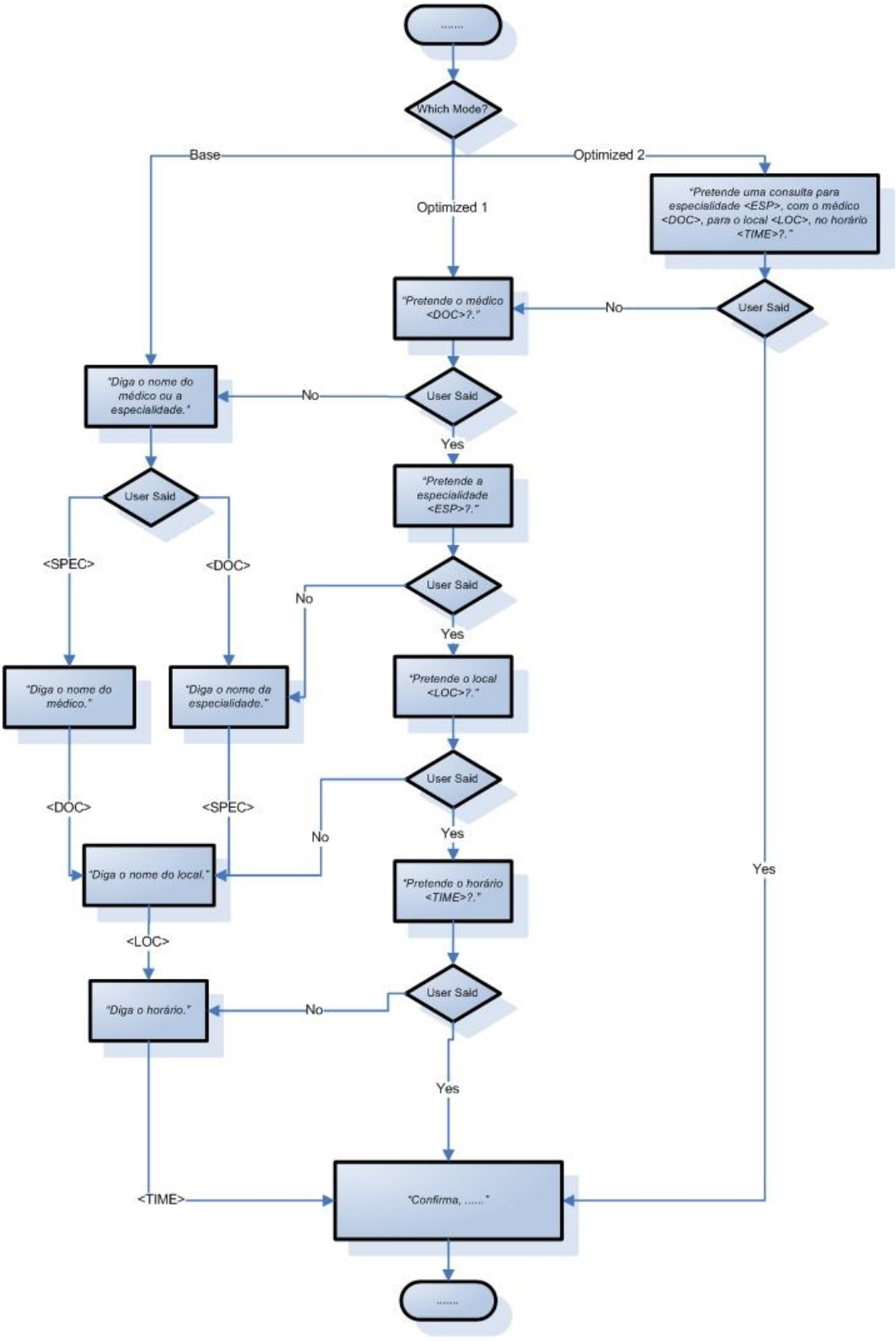

Figura 69 – (Anexo) Múltiplos Modos de Funcionamento.

Anexo F – Valores Médios de Reconhecimento com e sem confirmação e de Não Reconhecimentos

Da tabela seguinte, entenda-se a designação de cada um dos campos:

- O valor **n** corresponde ao número de interacções, realizadas no Ambiente 1 (A1) segundo o Modo de Funcionamento 1 (M1), ou seja, em A1M1, e desta forma ao número de utilizadores existentes, uma vez que existe uma interacção por utilizador;
- **N**: número médio de não reconhecimento, para um determinado modo de funcionamento, num determinado ambiente, por todas as interacções realizadas pelos utilizadores envolvidos.

  Designando por **NRx**, o número de não reconhecimentos obtidos pelo utilizador **x**, obtém-se: **N = ((NR1 + NR2 + ... + NRn) / n);**
- **RC**: número médio de reconhecimentos, com necessidade de confirmação por parte do utilizador. O processo de cálculo é idêntico ao anterior, desta vez designando por **RCCx**, o número de reconhecimentos com necessidade de confirmação obtidos pelo utilizador **x**, obtém-se: **RC = ((RCC1 + RCC2 + ... + RCCn) / n)**
- **R**: número médio de reconhecimentos imediatos efectuados pelo sistema. O processo de cálculo é idêntico ao anterior, desta vez designando por **RIx**, o número de reconhecimentos obtidos pelo utilizador **x**, obtém-se: **R = ((RI1 + RI2 + ... + RIn) / n);**
- **T**: valor médio do número de vezes que cada utilizador "fala para o sistema", respondendo a perguntas ou solicitando algo, como ajuda ou repetição de *prompts*. O processo de cálculo é idêntico ao anterior, desta vez designando por **FSx**, o número de vezes que utilizador **x** fala para o sistema, obtém-se: **T = ((FS1 + FS2 + ... + FSn) / n)**

Tabela 25 – (Anexo) Valores médios de Reconhecimento com e sem confirmação e de Não Reconhecimentos.

		N	RC	R	T
A1	**M1**	0.2	1.1	4.6	5.9
	M2	0	0	4.7	4.7
	M3	0	0	2.8	2.8
A2	**M1**	0.7	1.6	3.7	6
	M2	0.5	0	4.7	5.2
	M3	0.2	0	3.1	3.3
A3	**M1**	1.7	2.9	2.6	7.2
	M2	1.2	0	5.1	6.3
	M3	0.5	0	3.4	3.9