UNIVERSIDADE DE ÉVORA
DEPARTAMENTO DE INFORMÁTICA

CRIAÇÃO DE UMA FRAMEWORK PARA DESENVOLVIMENTO DE SISTEMAS DE COMUNICAÇÃO AUMENTATIVA E ALTERNATIVA

Gonçalo José Cunha Fontes

Mestrado em Engenharia Informática

Dissertação orientada pelo:
Prof. Doutor Salvador Luís de Bethencourt Pinto de Abreu

Évora

2010

UNIVERSIDADE DE ÉVORA
DEPARTAMENTO DE INFORMÁTICA

CRIAÇÃO DE UMA FRAMEWORK PARA DESENVOLVIMENTO DE SISTEMAS DE COMUNICAÇÃO AUMENTATIVA E ALTERNATIVA

Gonçalo José Cunha Fontes

Mestrado em Engenharia Informática

Dissertação orientada pelo:
Prof. Doutor Salvador Luís de Bethencourt Pinto de Abreu

Évora

2010

Agradecimentos

Uma tese de mestrado é um trabalho árduo, exigente e que pode ser considerado algo solitário, no entanto, não seria possível atingir os objectivos traçados para um trabalho como o proposto sem um grande apoio e colaboração de inúmeras pessoas, que sem qualquer tipo de contrapartida me apoiaram prestando o seu contributo para a elaboração desta tese.

Ao Professor Doutor Salvador Luís de Bethencourt Pinto Abreu, orientador desta dissertação agradeço a partilha de conhecimentos e experiencias, os incansáveis incentivos e as valiosas contribuições para o trabalho. De salientar ainda todo o apoio prestado e encorajamento, quer para esta dissertação, quer para o artigo escrito e apresentado no âmbito da mesma. Por fim, agradeço ainda o interesse demonstrado na continuação da minha ligação a esta instituição.

Ao Luís que mesmo à distância, sempre mostrou a maior disponibilidade, quer pelo apoio demonstrado, quer pelos concelhos práticos e técnicos, tendo assim sido de uma grande ajuda, principalmente quando a clarividência já se encontrava ofuscada pelo cansaço.

Ao Luís Garcia, pela ideia, opiniões, e pela partilha de conhecimento sobre a área de trabalho em que é especialista agradecendo ainda todo o apoio e encorajamento dado durante todo o processo.

Ao Luís Bruno, pelas suas ideias e pela ágil coordenação do tempo de aulas em que partilhamos a co-docência de uma disciplina durante a elaboração desta tese.

A todos os membros do Laboratório de Sistemas de Informação e Interactividade do Instituto Politécnico de Beja, pelos preciosos contributos e ideias para este projecto.

À minha família, composta pelos meus pais, avós, irmãos, cunhados e sobrinhos, por todo o apoio e confiança demonstrada durante este processo.

Por fim gostaria de agradecer a minha esposa por toda a compreensão, encorajamento, ajuda e suporte, tendo sido muitas vezes a voz da minha razão, sem a qual isto não seria possível.

Resumo

Framework para desenvolvimento de sistemas de comunicação aumentativa e alternativa

Nesta dissertação descreve-se uma proposta de implementação de uma plataforma de desenvolvimento de Sistemas de Comunicação Aumentativa e Alternativa para programadores, com o objectivo de melhorar a produtividade e diminuir os tempos despendidos na implementação deste tipo de soluções. Esta proposta assenta numa estrutura composta por *widgets* configuráveis por código e integráveis em novas aplicações, numa filosofia de reaproveitamento de objectos e funcionalidades, permitindo ainda a uniformização da estrutura do código no desenvolvimento de *softwares* deste tipo. Esta plataforma pretende ainda dar flexibilidade aos programadores, através da possibilidade de introdução de novas funcionalidades e *widgets*, permitindo também que se testem novas abordagens ao *software* durante a investigação. A implementação em tecnologias *open source* independentes da plataforma, permitirá ainda utilizar os objectos deste *toolkit* em vários sistemas operativos.

Palavras-chave: *Framework*, *ToolKit*, Comunicação Aumentativa e Alternativa, Pessoas com necessidades especiais, Tecnologias de Apoio

Abstract

Framework for augmentative and alternative communication systems development

In this master thesis we describe an implementation proposal for an Augmentative and Alternative Communication Framework for developers, with the objective of improves the productivity and reduces the implementation times for these types of solutions. This proposal is based on a customized widgets structure that can be integrated in new applications, with the purpose of reuse common features of these applications, also allowing standardize the code structure in this kind of software development. This framework intends to provide flexibility to programmers giving them the possibility of introduce new functionalities and widgets, allowing them too test new approaches during research. The implementation based on open-source technologies, platform independent, allows the use of this toolkit in several different operating systems.

Keywords: *Framework*, *ToolKit*, Augmentative and Alternative Communication, People with special needs, Support Technology

Índice

Índice de figuras

Lista de abreviaturas e siglas

CAA - Comunicação Aumentativa e Alternativa

ASHA - *American Speech-Language-Hearing Association*

CE-e - Caderno Escolar Electrónico

WAACT - *Widget Augmentative and Alternative Communication Toolkit*

Capítulo 1

1. Introdução

A comunicação é uma prática e uma necessidade fundamental para todos os seres humanos. No entanto, por diversos motivos, muitos de nós sofrem de condições físicas que tornam a comunicação tradicional difícil ou até mesmo impossível.

Certas perturbações sensoriais, cognitivas ou motoras podem comprometer total ou parcialmente as capacidades comunicativas humanas. As estimativas apontam para que cerca de 10% da população mundial seja portadora de um qualquer tipo de deficiência, sendo que nesse grupo, uma percentagem bastante significativa é afectada por deficiências ao nível da comunicação [1].

Nestas circunstâncias pode-se recorrer à Comunicação Aumentativa e Alternativa (CAA).

Segundo a *American Speech-Language-Hearing Association* (ASHA), esta área de prática clínica tenta compensar de forma temporária ou permanente incapacidades de comunicação por parte de pessoas com dificuldades ao nível da fala ou escrita [2]. Ainda segundo a ASHA um sistema de CAA é, "*um grupo integrado de componentes, incluindo símbolos, ajudas, estratégias e técnicas usadas por indivíduos para melhorar a comunicação*" [1].

Os sistemas de informação e as tecnologias actuais podem ser um importante auxílio para os sistemas de CAA, tendo sido desenvolvidos nos últimos anos esforços nesse sentido com o desenvolvimento e a implementação de diversas soluções. No entanto, grande parte das soluções até agora desenvolvidas, não tem em consideração

qualquer integração com outros sistemas, não sendo assim aproveitado o trabalho já desenvolvido e o conhecimento já adquirido de forma a fazer mais e melhor.

No desenvolvimento de soluções deste tipo, é importante a incorporação de todo o discurso existente, vocalizações, gestos e sempre que necessário o recurso a elementos externos de apoio à comunicação (e.g. tabela com letras e frases). Neste sentido têm vindo a ser desenvolvidos diversos sistemas tecnológicos que auxiliam a composição e transmissão de mensagens escritas ou faladas.

O Laboratório de Sistemas de Informação e Interactividade (LabSI[2]), em conjunto com o Centro de Paralisia Cerebral de Beja (CPCB) e o Instituto de Engenharia de Sistemas e Computadores Investigação e Desenvolvimento em Lisboa (INESC-ID), têm vindo a desenvolver algumas ferramentas de apoio à CAA, entre as quais se destaca o "Eugénio – O Génio das Palavras" [3].

Este sistema é uma ferramenta de apoio à escrita de textos em Português Europeu que recorre a modelos estatísticos baseados em *n-gramas*[4] para sugerir um conjunto de palavras prováveis na sequência do texto já escrito. O Eugénio funciona no ambiente *MS Windows* e apoia a escrita de mensagens em qualquer aplicação deste sistema operativo.

Para pessoas incapacitadas de utilizar um teclado de computador, o sistema dispõe de um teclado de ecrã[1]. A selecção destes elementos pode ser efectuada através de métodos de acesso directo que utilizam, por exemplo, um dispositivo de ponteiro (e.g. rato, caneta, luva virtual, entre outros.), ou métodos de acesso indirecto que recorrem apenas a um ou dois interruptores. Para o reforço da interacção com o utilizador foi incorporado no sistema um agente de interface e um sintetizador de fala, podendo estes componentes da interface do sistema ser adaptados às necessidades particulares de cada pessoa.

[1] Componente que apresenta uma matriz contendo os vários caracteres disponíveis para a composição

No desenvolvimento de outros sistemas de apoio à CAA, como é o caso do Caderno Escolar - Electrónico (CE-e)[5] tem-se verificado a necessidade de reutilização de componentes de *software* já desenvolvidos para outros sistemas. Por exemplo, no CE-e, que pretende ser uma alternativa digital aos tradicionais cadernos de papel para alunos com necessidades especiais, verificou-se a utilidade de incorporação de algumas funcionalidades já desenvolvidas para o Eugénio, como a predição de palavras ou o teclado de ecrã. Além disso, se estas ferramentas adoptarem uma estrutura modular que facilite a substituição dos seus componentes, então poder-se-ão experimentar e avaliar de forma mais eficaz novas abordagens a determinadas técnicas de CAA, como é o caso da predição de palavras.

Contudo, a integração de componentes entre as duas aplicações anteriormente descritas não é tarefa fácil, visto que tais sistemas não foram desenvolvidos de forma modular e nem sequer na mesma tecnologia.

Neste contexto, o âmbito deste projecto propõe uma plataforma de desenvolvimento transversal ao sistema operativo que permite a reutilização de componentes de *software*, numa linguagem multiparadigma e de uso geral.

A solução que aqui se descreve, é implementada numa arquitectura modular que permite a reutilização de variados *widgets* ou a criação de novos componentes, podendo estes ser usados isoladamente, num novo projecto ou ainda integrados num projecto existente.

Devido ainda ao *middleware* utilizado na implementação deste sistema e a um módulo de gestão de eventos criado para o efeito, garante-se total liberdade na compilação e execução dos projectos para os sistemas operativos mais comuns do mercado.

O *middleware* que foi utilizado e permite tal flexibilidade é a *User Interface Framework QT*, que foi escolhida essencialmente devido à sua portabilidade, ao detrimento das suas concorrentes *.NET Framework* e MFC (*Microsoft Foundation Classes*) que apenas funcionam sobre plataforma *Windows*.

Por fim, devido ao facto das tecnologias utilizadas serem abrangidas pelo licenciamento LGPL, é garantida total liberdade no desenvolvimento e disponibilização dos componentes.

Assim, de forma a introduzir o trabalho desenvolvido, na secção 2 apresenta-se uma revisão do estado da arte, sendo na secção 3 apresentado e descrito o sistema. Na secção 4 será explicada a avaliação e o trabalho relacionado, e por fim, serão apresentadas as conclusões e direcções para trabalho futuro.

O presente trabalho deu ainda origem a um artigo que foi apresentado na conferência Inforum 2010 – Simpósio de Informática, no *track* de "Computação Gráfica" que decorreu a 9 e 10 de Setembro na Universidade do Minho em Braga.

Capítulo 2

2. Estado da Arte

É comum, no que diz respeito às tecnologias de informação, que no desenvolvimento de um *software* ou sistema, o público portador de um qualquer tipo de deficiência cognitiva ou motora, acabe por não ser contemplado pelos requisitos das várias fases do projecto.

Duas das razões para esta situação prendem-se com a especificidade que os sistemas devem ter neste tipo de casos e a variedade de situações entre cada utilizador e o seu estado.

No entanto, o desenvolvimento de *software* de apoio à CAA é imprescindível para as pessoas que conseguiram ganhar parcial ou total autonomia graças as tecnologias, bem como para futuros utilizadores.

2.1. Comunicação Aumentativa e Alternativa

Um dos principais objectivos desta área, é fornecer a ajuda necessária às pessoas incapazes de satisfazer as suas necessidades diárias de comunicação através de meios convencionais.

Ao contrário do que normalmente é sugerido, a Comunicação Aumentativa e Alternativa abrange todas as formas de comunicação, não sendo esta apenas um exclusivo da forma oral, sendo assim usada para a expressão de pensamentos, necessidades, vontades ou ideias. Todos usamos CAA, mesmo que inconscientemente, quando fazemos expressões faciais ou gestos, usamos símbolos

ou imagens, ou ainda quando escrevemos. Todas estas utilizações de CAA permitem-nos usufruir de uma prática fundamental, a comunicação.

As pessoas com graves dificuldades ao nível da fala ou expressão têm nestes métodos de comunicação, uma forma de complementar o seu discurso existente, ou, se este for inexistente, uma forma de o substituir.

Para se exprimir, estas pessoas dispõem de diversas ajudas aumentativas, como os quadros de comunicação por imagens ou símbolos, ou ainda equipamentos electrónicos, no entanto, mesmo com este tipo ajudas, é importante que os seus utilizadores nunca deixem de se exprimir de uma forma natural, se o conseguirem, devendo estes métodos ou mecanismos ser apenas uma melhoria na sua comunicação.

Contudo, se estas ajudas permitem uma melhoria significativa da comunicação dos seus utilizadores, estas também apresentam alguns problemas, tais como o facto de a sua utilização ser bastante lenta comparativamente com os métodos de expressão mais utilizados, a fala ou escrita.

Este é um dos principais problemas ligados ao desenvolvimento de soluções de auxílio a comunicação, sendo que os dois valores mais importantes expressados por pessoas que utilizam este tipo de sistemas são: (i) Dizer exactamente o que querem dizer; (ii) Dizê-lo o mais depressa que consigam [6].

Estas são duas características de difícil satisfação por parte dos *designers* deste tipo de soluções, visto que os utilizadores alvo destas aplicações podem ter uma variedade de necessidades, sendo que estas nem sempre são compatíveis com as de outros utilizadores, mesmo que estes sejam portadores da mesma deficiência.

Dependendo do nível de deficiência de um utilizador, poderá ser preferível um determinado tipo de dispositivo de *input* ao detrimento de outro. No entanto, este último poderá ser utilizado com maior eficácia por outro utilizador portador da mesma deficiência.

Por exemplo; no caso de um utilizador com paralisia cerebral com um nível de deficiência elevado, afectando as suas funções motoras, poderá ser mais vantajoso a utilização de um dispositivo interruptor associado a um sistema de varrimento ao detrimento da utilização de um rato ou do próprio teclado do computador.

Se neste exemplo imaginarmos que o referido utilizador está a utilizar um sistema de escrita de frases, e que os seus movimentos são muito limitados, este terá grandes dificuldades em utilizar quer um teclado físico, quer um teclado de ecrã, o que irá reduzir drasticamente a sua velocidade de escrita.

No entanto, se no mesmo cenário imaginarmos um utilizador portador da mesma condição, tendo este contudo, limitações motoras muito inferiores às do utilizador anteriormente descrito, poderemos tirar claras mais-valias da utilização de um teclado.

Não sendo justificável o desenvolvimento de uma aplicação específica para cada pessoa, é importante o desenvolvimento de sistemas flexíveis que permitam a maior abrangência possível de utilizadores. Contudo, na grande maioria das vezes tal não é possível devido ao largo número de variáveis a ter em conta, sendo que nesses casos é importante tentar novas abordagens que permitam validar novas realidades.

2.2. *Software* de Apoio a CAA

A introdução dos sistemas de informação e das tecnologias actuais vieram melhorar significativamente a eficácia dos sistemas de CAA.

Como já foi referido, ao logo dos últimos anos têm sido desenvolvidos muitos esforços da construção de soluções tecnológicas com o intuito de auxiliar os utilizadores com dificuldades comunicativas.

A grande maioria dos *softwares* de CAA desenvolvidos tem por base, de uma forma ou de outra, um teclado de ecrã, tanto na sua forma tradicional funcionando nos tradicionais computadores de secretária ou portáteis, como em interfaces físicas adaptadas (dispositivos do tipo "*handheld*"), sendo os mesmos utilizados para a obtenção de frases, podendo estas ser formadas por letras, palavras ou por sequências de símbolos pictóricos. Um exemplo deste tipo de sistemas é o Eugénio (Fig. 1.) [3]. Esta ferramenta que dispõe de uma funcionalidade de predição recorre a modelos estatísticos baseados em *n-gramas*[4] para sugerir um conjunto de palavras prováveis na sequência do texto.

A ideia por detrás das *n-gramas* ou modelo de *Markov* é prever a probabilidade da letra seguinte num determinado texto. Para isso são examinadas amostras de textos denominadas de textos de treino que, com base numa estimativa de parecenças, irão identificar a letra com maior probabilidade possível de ser a seguinte.

Para além da predição de palavras, o Eugénio dispõe de um sistema de varrimento da aplicação que permite aos utilizadores que tenham deficiências motoras, estando por isso impossibilitados de usar os tradicionais mecanismos de I/O, como é o caso dos dispositivos apontadores, usar métodos que recorrem a um ou dois interruptores, para que o utilizador não tenha que movimentar os membros podendo apenas seleccionar a opção desejada.

Fig. 1 – Interface do Eugénio

Como já foi também referido, um dos principais requisitos para os sistemas de CAA é o facto de o utilizador poder ser rápido na escrita da mensagem que pretende comunicar. Assim, para além dos mecanismos já descritos, o Eugénio dispõe ainda de um mecanismo de extensão de abreviaturas que permite ao utilizador escrever palavras ou frases digitando apenas a sua abreviatura. Esta funcionalidade recorre a um dicionário de abreviaturas predefinido, podendo este ser configurado para um incremento dos termos, permitindo uma melhor adaptação ao seu utilizador.

Isolando todas estas funcionalidades, podem-se extrair componentes dos mais diversos tipos, como a predição de palavras, o teclado de ecrã, o varrimento e a expansão de abreviaturas. Todos estes componentes que trabalham em conjunto foram implementados sobre uma estrutura hierárquica de classes. Todos eles seriam de uma grande utilidade em sistemas similares.

Para além do Eugénio, existem algumas outras ferramentas de CAA que permitem a expressão de palavras ou frases, quer de uma forma escrita, quer de uma forma falada, por intermédio de sintetizadores de voz. Um desses casos é o *Talkactive* (Fig. 2.)[7] que recorre a modelos de *High Frequency Vocabulary* ou *Core Vocabulary*: Este tipo de modelo define que um número relativamente pequeno de palavras pode constituir a grande maioria do que é dito em comunicações normais. Com algumas centenas de palavras uma pessoa pode dizer cerca de 80% de tudo o que é necessário em comunicações diárias [8].

Este é um sistema de utilização simples que recorre a símbolos pictóricos para identificar palavras, que conjugadas entre si permitem formar as frases que o utilizador pretende comunicar.

Assim, o utilizador apenas precisa de seleccionar as imagens representativas do que este pretende transmitir a outro interveniente, tendo a vantagem de num único *click* poder transmitir uma palavra, frase ou ideia.

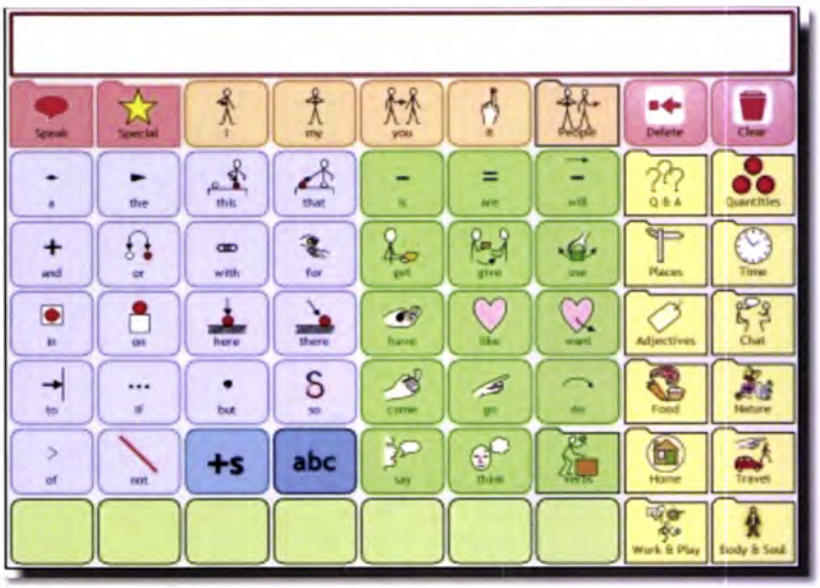

Fig. 2 – Interface do Talkactive

Devido em grande parte a sua simplicidade, estes sistemas de símbolos ganharam grande popularidade em vários países, sendo usados com grande sucesso por pessoas portadoras de paralisia cerebral [9][10].

Contudo, o Talkactive também possui claras desvantagens, como o facto de, devido a sua extensão, não ser possível disponibilizar num único ecrã todo o vocabulário de utilização diária ou de não estar disponível em português.

No entanto, à semelhança do que se passa com o Eugénio esta aplicação também é composta por diversos componentes distintos, tais como o teclado, o corpus de símbolos pictóricos e um editor de texto, que poderiam por sua vez também ser de grande utilidade em aplicações do mesmo género.

Por outro lado, estes dois sistemas acabam por se revelar bastante semelhantes ao nível da sua estrutura gráfica, bem como, ao nível de algumas das suas funcionalidades, sendo ambos baseados numa grelha de teclas permitindo a escrita de palavras e frases.

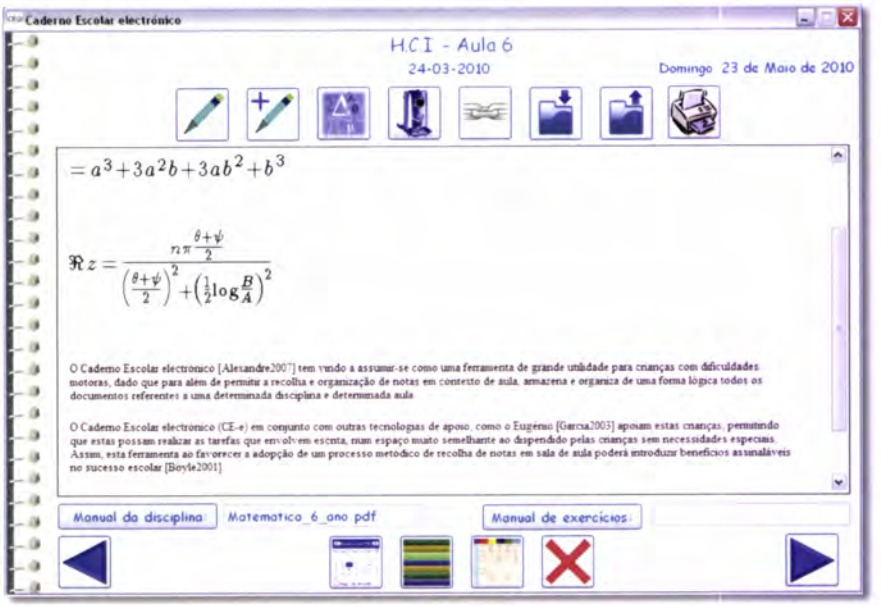

Fig. 3 – Interface do CE-e

Outro software de apoio a pessoas com necessidades especiais, mas desta vez essencialmente direccionado para estudantes, é o Caderno Escolar electrónico (CE-e) [5].

Este sistema que pretende ser um substituto dos tradicionais cadernos escolares em papel, disponibiliza aos seus utilizadores uma interface e variadas funcionalidades de *notetaking* organizado, permitindo a escrita de texto, inserção de imagens e hiperligações, *upload* de ficheiros e aplicação de fórmulas matemáticas, entre outras (Fig. 3.).

No entanto, ao contrário dos sistemas anteriores este não possui qualquer ferramenta de apoio ou aceleração da escrita, como o preditor de palavras, o teclado de ecrã com varrimento ou ainda a grelha de símbolos pictóricos, ferramentas estas, de grande importância para o aumento da produtividade deste tipo de utilizadores.

2.3. Sistemas com Paradigmas Semelhantes

A ideia da criação de uma plataforma de auxílio ao desenvolvimento de aplicações de CAA não é propriamente nova, tendo sido nos anos 90 desenvolvidos esforços nesse sentido por parte do Consórcio *Comspec*.

Este consórcio que reuniu uma equipa multidisciplinar constituída por educadores, engenheiros e programadores provenientes de diversos países europeus, tais como a Inglaterra, Portugal, Holanda, Suécia e Noruega [11], tinha por objectivo a criação de uma plataforma que permitisse plena transversalidade entre quatro tipos de utilizadores (programadores, integradores de sistema, facilitadores e utilizadores finais).

As metas a atingir eram bem claras e consistiam em:

- Proporcionar auxiliares alternativos à comunicação e sistemas de controlo para pessoas com algum tipo de deficiência comunicativa ou idosos;
- Possibilitar aos profissionais da reabilitação a criação e adaptação de sistemas de comunicação baseados em aplicações informáticas de interfaces *userfriendly*;
- Introduzir na indústria das tecnologias de reabilitação, um conjunto poderoso de ferramentas com características modulares.

As primeiras avaliações a arquitectura deste sistema foram feitas num simulador implementado em *SmallTalk V* e *HyperCard* para *Macintoch* [11]. No entanto, a primeira versão do Software Comspec foi desenvolvida na plataforma *OpenDoc* [12].

Após o cancelamento do suporte aos sistemas *OpenDoc*, o *software* foi redesenhado e novamente desenvolvido utilizando a tecnologia Java, tendo sido esta tecnologia escolhida com o intuito de fornecer ao sistema uma maior flexibilidade e transversalidade entre sistemas operativos.

Com já foi referido, este projecto pretendia transversalidade entre diversos tipos de utilizadores, tendo sido este requisito, uma das grandes dificuldades na implementação deste sistema, devido essencialmente ao facto de este necessitar de uma grande diversidade de perfis de utilização.

Os quatro *stakeholders* identificados na análise de requisitos tinham tarefas distintas e distribuídas (Fig. 3.) [13]:

- Os Programadores iriam criar componentes para as aplicações;
- Os Integradores de Sistema, iriam criar as aplicações para os utilizadores finais com base nos componentes desenvolvidos pelos programadores;
- Aos Facilitadores caberia a tarefa de optimizar as aplicações para os utilizadores finais;
- Os Utilizadores finais apenas teriam apenas que utilizar os sistema previamente adaptado pelos Facilitadores.

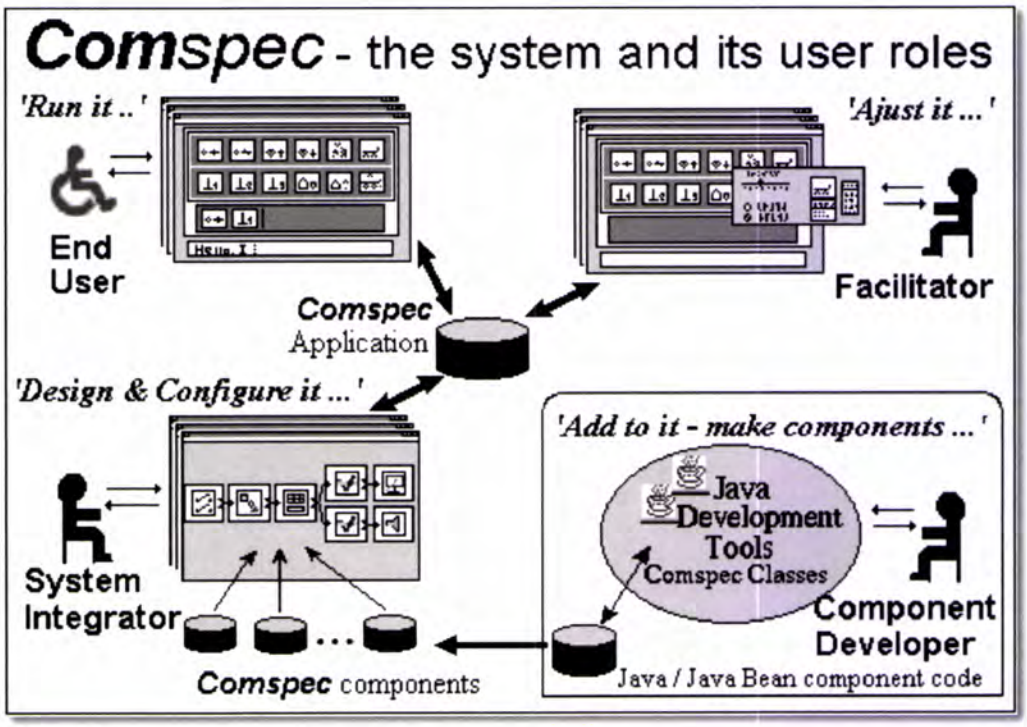

Fig. 4 – Utilizadores e Funções no Projecto *Comspec*

Contudo, esta diversidade dos papéis (roles) que a aplicação deveria permitir, acabou por condicionar a implementação da mesma, sendo que desta forma o sistema *Comspec* acabou por não ser mais que um gerador de *software* CAA.

Assim, devido a necessidade de garantir o cumprimento dos requisitos necessários a todos os utilizadores, acabaram por serem impostas grandes limitações na interface bem como na configuração de componentes, o que levou ao esquecimento do projecto, alguns anos mais tarde.

O projecto *Ulysses* tentou ser um pouco menos ambicioso que o *Comspec* tendo sido definidos apenas três tipos de utilizadores principais:

- Os designers e programadores, que a semelhança do *Comspec* iriam criar componentes para as aplicações;
- Os integradores que com base nos componentes desenvolvidos pelos *designers* e programadores implementaria as aplicações de ajuda a comunicação;
- Os utilizadores finais que apenas teriam que utilizar e tirar o melhor partido destas aplicações.

Por outro lado este projecto já tinha como referencia as fragilidades oriundas do *Comspec* tendo sido definida uma nova abordagem ao problema por forma a tentar melhorar as diversas situações identificadas., no entanto o desenvolvimento do sistema acabou por não conhecer grandes avanços, tendo acabado por ser abandonado [14].

Capítulo 3

3. Apresentação e Descrição do Sistema

Como foi dito no Capítulo 1, a maior parte dos sistemas de apoio à CAA foram desenvolvidos de forma monolítica, sendo a sua implementação, alteração ou integração noutros sistemas bastante difícil.

Este problema acaba por ter diversas origens, tais como o desenvolvimento de aplicações proprietárias, a falta de estrutura para reaproveitamento de código, ou ainda, a utilização de diferentes estruturas ou linguagens durante a sua implementação. São estes os problemas que este projecto pretende resolver.

Para isso, foi estudada e desenvolvida uma plataforma de desenvolvimento para programadores, denominada de *Widget Augmentative and Alternative Communication Toolkit* (WAACT), que lhes permitirá criar projectos de ferramentas de CAA, utilizando uma variedade de componentes gráficos predefinidos e configuráveis, comuns neste tipo de aplicações, permitindo assim aumentar a produtividade no desenvolvimento, bem como experimentar e avaliar novas abordagens a estas aplicações durante a investigação.

A avaliação de novas abordagens ao desenvolvimento de sistemas deste tipo é um aspecto que mereceu particular atenção durante a análise dos requisitos, tendo em conta que em muitos casos o desenvolvimento de um protótipo é moroso e pode, após avaliação, ser descartado por não satisfazer as necessidades ou especificidades do utilizador.

Para isso, foi necessário o pensamento de que um sistema com estas características deveria ser de fácil utilização, e que, com apenas algumas linhas de código pudesse fornecer uma base de trabalho aceitável para uma avaliação por parte dos *testers*.

3.1. Framework ou Toolkit

Analisando as suas características e estrutura podemos classificar esta ferramenta como *Toolkit* ou como *Framework.*

Existem varias definições para ambas as estruturas entre as quais se destaca, para os *Frameworks*, a de Ralph E. Johnson que diz que estes *"são a reutilização da totalidade ou de parte do desenho de um sistema que é representado por um conjunto de classes abstractas e pela forma como as suas instâncias interagem"*[15].

Ainda o mesmo autor diz que *"um Framework é um esqueleto de uma aplicação que pode ser configurada por um programador".*

Assim, quando se utiliza um *Framework* o corpo da aplicação está implementado, permitindo a sua reutilização, devendo apenas ser particularizadas as chamadas às funções, o que reduz significativamente as decisões de concepção, ao contrário do que é feito em sistemas como os *Toolkits* ou bibliotecas partilhadas(*dll's*), em que é codificado o corpo das aplicações e apenas se faz a chamada ao código que se quer reutilizar [16].

Não sendo as duas definições anteriormente descritas semelhantes, também não são contraditórias, acabando até por se completar uma a outra. Enquanto a primeira define a estrutura de um *Framework* a segunda define o seu objectivo.

O WAACT pode assim ser considerado um *Toolkit* com características de *Framework.*

3.2. Análise e Decisões

Durante a análise deste projecto foram surgindo diversas questões, tais como, a quem seria o sistema disponibilizado e quem o poderia desenvolver.

Assim, como o principal objectivo é a uniformização da estrutura das aplicações em questão bem como a reutilização de componentes já desenvolvidos e validados, verificou-se que a melhor forma de atingir esse objectivo seria através da participação no desenvolvimento de todos aqueles que utilizarão o sistema. Para isso, foram escolhidas ferramentas e linguagens de uso geral e abrangidos por licenciamento aberto, permitindo a utilização e possível desenvolvimento de todos.

Contudo, o uso de sistemas *Open Source* não era de todo suficiente, tendo em conta que a maior parte dos profissionais destas áreas acabam por trabalhar em sistemas e *softwares* proprietários, tendo sido assim necessário a utilização de tecnologias multiplataforma, bem como a criação de uma estrutura modular que permitisse um desenvolvimento contínuo e flexível.

O desenvolvimento do WAACT é feito em C++ e apoiado na *User Interface Framework*, QT [17], o que permite resolver alguns dos problemas atrás referidos.

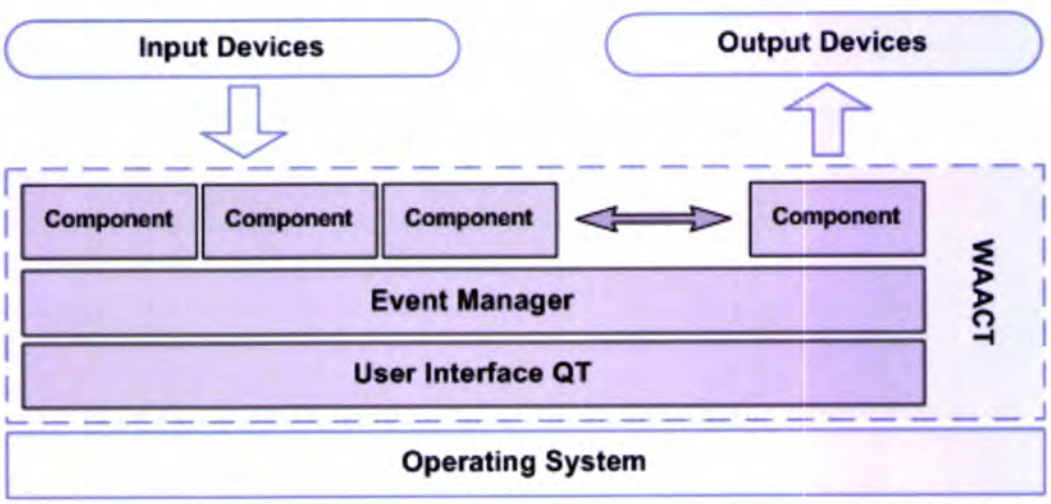

Fig. 5 –Estrutura de implementação do WAACT

Por outro lado, a necessidade de garantir a gestão dos eventos enviados por equipamentos de *Input* (podendo estes ser ratos, teclados, interruptores, entre outros.), processando-os e, se necessário, enviando-os para algum equipamento de *Output*, originou a criação de uma interface *Event Manager* que tem por missão tornar os módulos independentes dos eventos, permitindo o isolamento de cada *Widget*, podendo estes ser desenvolvidos com total independência uns dos outros (Fig. 5.).

3.2.1. Linguagem de Suporte

A escolha do C++ como linguagem de suporte para este projecto, deveu-se essencialmente ao facto de esta ser uma linguagem multiparadigma, abrindo assim um leque de possibilidades à direcção que o projecto poderia levar, e ao facto de esta ser de médio nível, combinando características de linguagens de baixo e alto nível.

Por outro lado, por esta tecnologia ser uma das mais populares desde os anos 90 e ter dado origem a outras linguagens mais recentes, tais como o JAVA e o C#, também estas objecto de grande popularidade, pensou-se que poderia ser uma mais-valia, uma tecnologia de conhecimento generalizado.

Por fim, o seu sistema de apontadores, que pode ser utilizado de forma explicita, podendo o programador alocar expressamente memória para determinado tipo de dados, ou de forma implícita, sendo o próprio sistema a alocar a memória que precisa (alocação dinâmica de espaço em memória), garante um excelente desempenho na execução.

3.2.2. *User Interface* QT

Com já foi referido, um dos grandes objectivos deste projecto, prendia-se com o facto de este sistema dever permitir um desenvolvimento multiplataforma. Contudo, devido essencialmente as diferentes formas como os vários sistemas

operativos tratam os eventos, foi necessário optar por uma estratégia de desenvolvimento que oferece-se garantias de continuidade.

As duas possibilidades eram; ou se programavam no *Event Manager* todos os eventos possíveis para os diversos sistemas operativos que se pretendessem abrangidos, ou se usava um *middleware* que permitisse a compilação em diversas plataformas.

Nesta primeira versão do WAACT, optou-se por utilizar a *User Interface* QT tendo em conta as inúmeras vantagens que esta fornece.

Sendo o QT desenvolvido em C++, este *middleware* permite uma grande liberdade e interoperabilidade entre sistemas operativos, podendo assim ser desenvolvidas aplicações para as diversas plataformas existentes no mercado, com recurso à versatilidade que o paradigma da programação orientada a objectos oferece.

Assim, as aplicações desenvolvidas não só podem ser implementadas com recurso às bibliotecas *standard* do C++ como podem ser programadas com recurso as bibliotecas específicas do QT.

O QT dispõe ainda de um mecanismo de *SLOTs* e *SIGNALs* que se podem definir como uma alternativa às técnicas de *callback*. Estas são as funcionalidades centrais deste sistema e o maior aspecto diferenciador de outros *Frameworks*. Esta funcionalidade permite ligar funções a eventos, ou funções a outras funções, o que permite despoletar acções em qualquer tipo de evento predefinido ou em qualquer evento programado.

Contudo, com esta opção não se fecham as portas a outras abordagens, podendo no futuro estas ser testadas sobre a base já desenvolvida.

3.3. Conceito

O conceito por detrás do WAACT é bastante simples e consiste na criação de *widgets* individuais que podem ser chamados numa aplicação sobre a forma *standard*[2] ou sobre a forma personalizada, permitindo assim ao programador do sistema em desenvolvimento, a configuração das características do referido *widget* (e.g. cor, dimensão, *labeling*, etc.).

Cada módulo, ou componente identifica o *widget* que este representa e é composto pelas funções necessárias a sua implementação num novo projecto ou num projecto existente.

Como já foi referido, os eventos fornecidos pelos utilizadores podem ter diversas origens, sendo que para a gestão dos mesmos foi implementado um módulo denominado de *Event Manager* que trata cada evento fornecido da forma que o programador desejar.

Tal situação, fez com que os *widgets* que enviam eventos para o *Event Manager* fossem catalogados em duas categorias: (i) Os que interagem directamente com a *Mainwindow* (e.g. Botão de saída da aplicação); (ii) Os que interagem com outros *widgets* (e.g. Teclado virtual a interagir com um Processador de texto).

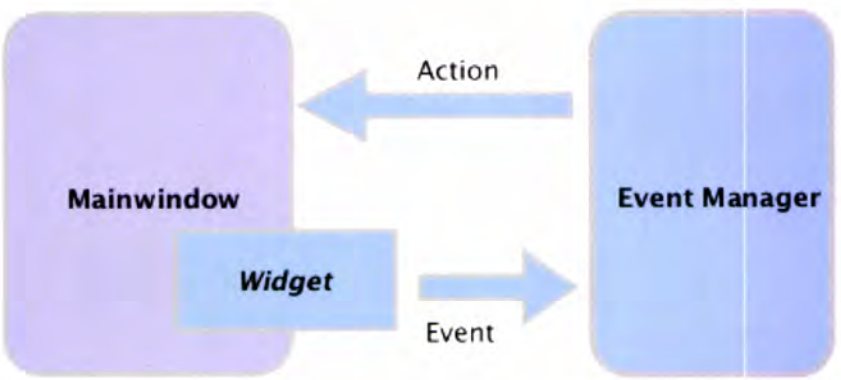

Fig. 6 – Diagrama de Interacção entre um *Widget* e a *Mainwindow*

[2] Pré-definida pelo programador do módulo

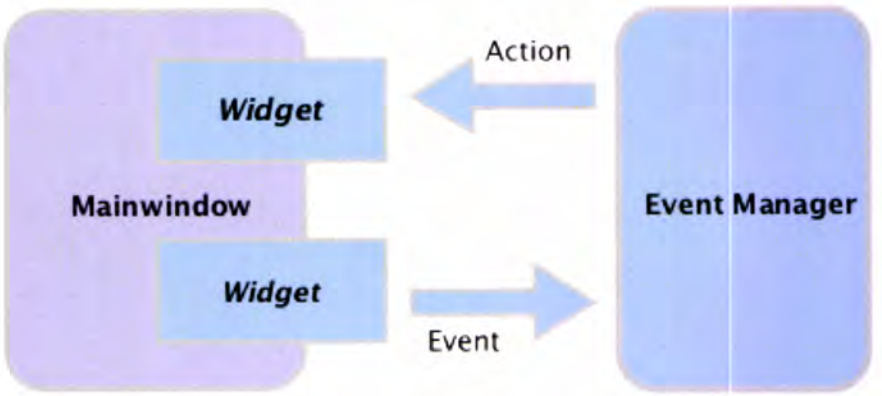

Fig. 7 – Diagrama de Interacção entre um *Widget* e outo *Widget*

No 1º caso apenas é necessário fazer uma chamada à função do *Event Manager* que trata o evento pretendido (e.g. evento *click*) e identificar o *widget* que deverá receber esse determinado evento, sendo que, neste caso, o focús da acção será sempre a *Mainwindow* (Fig. 6).

No 2º caso, também é feita uma chamada à função do *Event Manager* referente ao evento pretendido, no entanto, neste caso devem ser definidos, o *widget* que deve receber os eventos do tipo pré-definido e o que irá despoletar a acção na *Mainwindow* (Fig. 7).

Em qualquer um dos casos descritos, a intervenção do *EventManager* na utilização dos módulos do WAACT é essencial e deve ser imperativa, para permitir que possam ser desenvolvidos novos componentes independentes dos já existentes, mas que possam interagir com os mesmos, sem que contudo, haja necessidades de intervenção nos módulos já desenvolvidos.

3.4. Estrutura do Sistema

Desde o início que estava claro que a modularidade do sistema deveria ser o ponto de partida para a implementação do mesmo, no entanto, veio a verificar-se que

nem todos os módulos deveriam ser independentes, podendo ser construída uma hierarquia de classes permitindo a utilização de determinados componentes integrados em componentes mais abrangentes (Fig. 8).

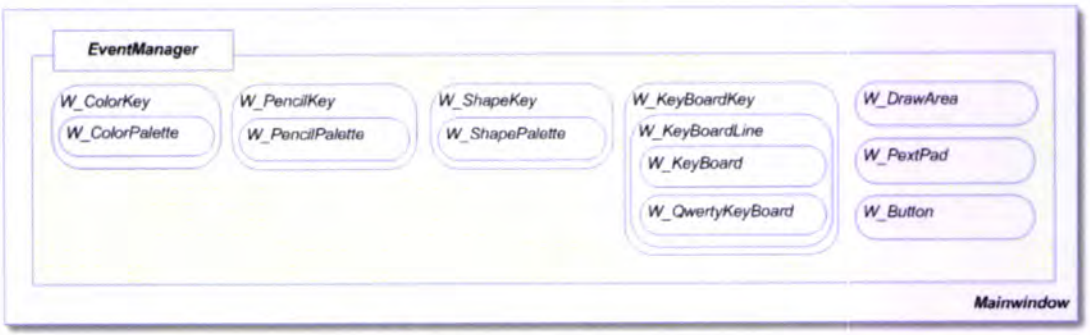

Fig. 8 – Hierarquia de Classes do WAACT

Por forma a melhor perceber a integração de determinados componentes, e as suas funcionalidades é apresentada a descrição e identificação dos componentes, sendo explicada a possível utilização para cada classe desenvolvida, bem como o diagrama de classes do sistema (Fig. 9):

- ***W_DrawArea*** – permite a instanciação de uma área de desenho;
- ***W_TextPad*** – permite a instanciação de uma área de escrita para notetacking;
- ***W_Button*** – permite a instanciação de botões de interacção com a Mainwindow;
- ***W_KeyBoardKey*** – permite a instanciação de uma tecla Stadard ou personalizada, podendo esta ser integrada num teclado ou não;
    - ***W_KeyBoardLine*** – permite a instanciação de uma linha de teclas descritas a alínea anterior (*W_KeyBoardKey*);
        - ***W_KeyBoard*** – permite a instanciação de um teclado formado por as linhas de teclas descritas a alínea anterior (*W_KeyBoardLine*);

        - ***W_QwertyKeyBoard*** – permite a instanciação de um teclado QWERTY pré-definido ao nível da estrutura;

- ***W_ShapeKey*** – permite a instanciação de um botão de figuras que podem ser desenhadas numa *W_DrawArea*;

    - ***W_ ShapePalette*** – permite a instanciação de uma palete de botões de figuras (*W_ShapeKey*) que podem ser desenhadas numa *W_DrawArea*;

- ***W_PencilKey*** – permite a instanciação de um botão de espessuras de desenho utilizado no *rendering* das figuras que podem ser desenhadas numa *W_DrawArea*;

    - ***W_ PencilPalette*** – permite a instanciação de uma palete de botões espessuras de desenho (*W_PencilKey*), standard ou personalizada, sendo estes utilizados no *rendering* das figuras que podem ser desenhadas numa *W_DrawArea*;

- ***W_ColorKey*** – permite a instanciação de um botão de cor que pode ser utilizado no *rendering* das figuras que podem ser desenhadas numa *W_DrawArea*;

    - ***W_ ColorPalette*** – permite a instanciação de uma palete de botões de cor (*W_ColorKey*), standard ou personalizada, sendo estes utilizados no *rendering* das figuras que podem ser desenhadas numa *W_DrawArea*.

Umas das situações de integração de componentes é o caso do teclado de ecrã que se aprofundarmos podemos decompor em vários possíveis módulos. Um desses módulos refere-se as teclas, sendo que é certo que cada uma delas tem a sua especificidade, no entanto, na sua essência, são apenas teclas.

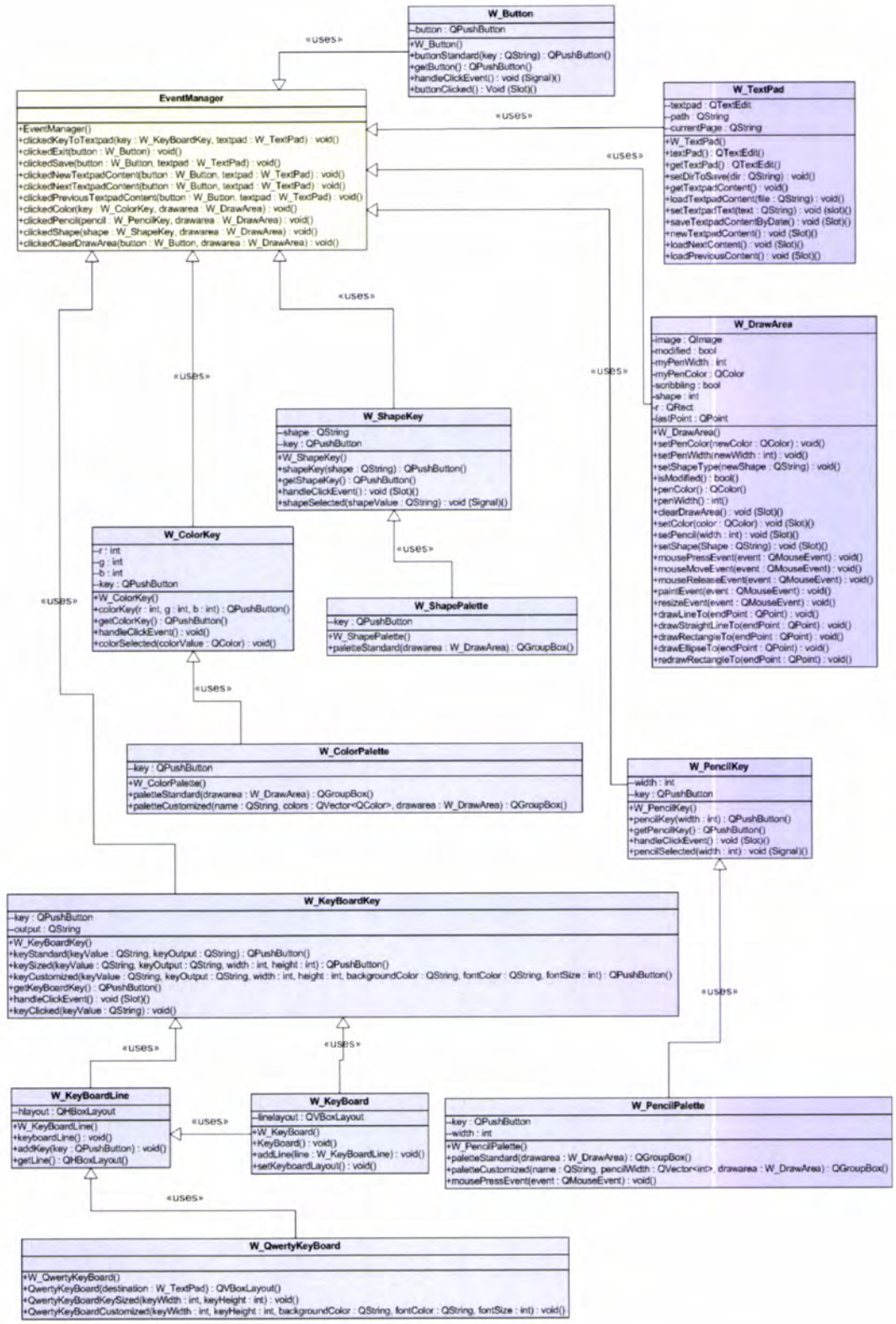

Fig. 9 – Diagrama de Classes do WAACT

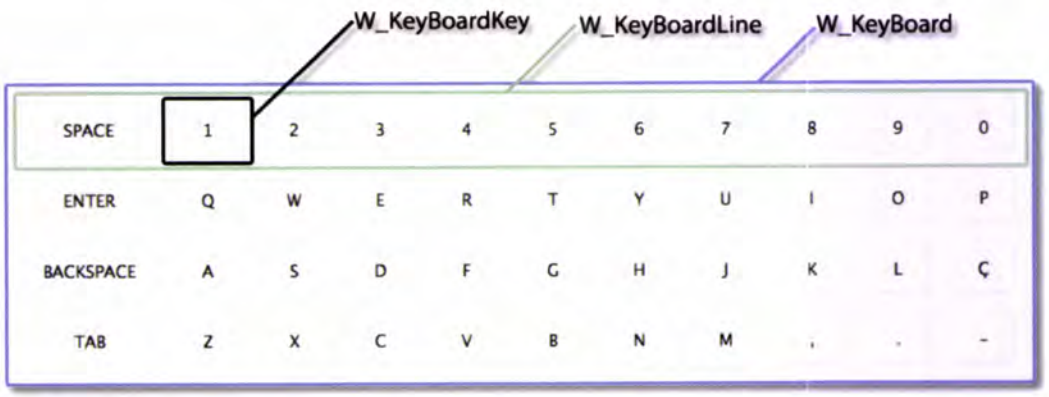

Fig. 10 – Esquema da composição de um teclado

Contudo, se avaliarmos a composição de teclas entre um teclado QWERTY[3] e outro tipo de teclado, verificamos que existem claras diferenças tanto ao nível da quantidade de teclas bem como ao nível da disposição da grelha das mesmas, o que pode dificultar a construção de um novo teclado, se não existirem mecanismos de personalização.

Na área da CAA a construção de teclados personalizados é uma prática corrente e essencial para que o utilizador possa tirar o maior partido possível do sistema que está a utilizar, sendo que nestas circunstâncias os componentes devem ser adaptáveis às especificidades e limitações de cada pessoa.

Assim, como poderia ser complexa a implementação e organização de um novo teclado, podendo este ter mais de 25 teclas, e nem sempre estando alinhadas, verificou-se ainda que para além de um componente tecla e um componente teclado iria ser necessário um componente intermédio que permitisse a implementação uma estrutura de várias de linhas de teclas por forma a formar a personalizar o teclado (Fig. 10).

Com base na mesma estrutura, foram implementadas outras classes simples as quais são integráveis em componentes mais abrangentes, como é o caso dos botões de

[3] Disposição de teclado adoptada pelos países ocidentais

cores e tamanhos de linha que podem ser implementados em componentes de paletes, podendo estas ser Standard ou personalizadas (Fig. 11).

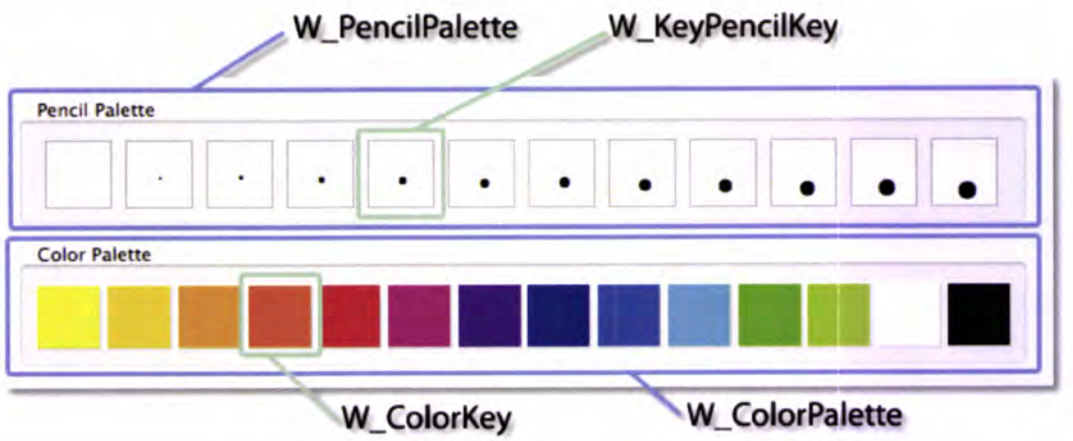

Fig. 11 – Esquema da composição das paletes de cores e espessura de linha

Por fim, por forma a poder diferenciar de uma forma clara e simples as funções do WAACT comparativamente com as funcionalidades do C++ ou do próprio QT, o nome das classes foi definido com uma estrutura específica, tendo todas elas o prefixo "W_" antes do nome que as descreve.

Assim, para além de haver uma clara distinção entre as classes que podem ser utilizadas, esta estrutura permite-nos, que utilizando um IDE de desenvolvimento sejam listados todos os *widgets* disponíveis no sistema.

3.5. Utilização e Funcionamento

Como já foi referido, foi necessária a utilização de um *middleware* que fosse multiplataforma, de forma a não restringir a utilização de software desenvolvido a um determinado sistema, tendo sido o QT a plataforma escolhida.

Esta escolha permite alguma flexibilidade ao programador, pois como também já foi referido, este poderá desenvolver utilizando varias tecnologias.

Por outro lado, o QT disponibiliza um IDE de desenvolvimento também ele multiplataforma e abrangido, se o programador assim o entender, pelo licenciamento GLPL.

O *QT Creator IDE*(Fig. 12) é uma plataforma de desenvolvimento de simples utilização e instalação, que permite a programação em C++ puro ou com recurso as bibliotecas especificas do QT, permitindo a criação de classes isoladas para integrar em projectos existentes, ou a criação de novos projectos de bibliotecas, aplicações de consola ou ainda aplicações gráficas, o que encaixa perfeitamente na filosofia de desenvolvimento pensada para o WAACT.

Contudo, a flexibilidade já referida, com a utilização de várias tecnologias pode ser confusa para o programador, o que em vez de aumentar a sua produtividade pode diminui-la.

Tendo em vista esta situação, pensa-se ser importante proporcionar aos utilizadores do WAACT uma metodologia de implementação mais direccionada e que não obrigue a utilização de todas as tecnologias disponíveis na plataforma.

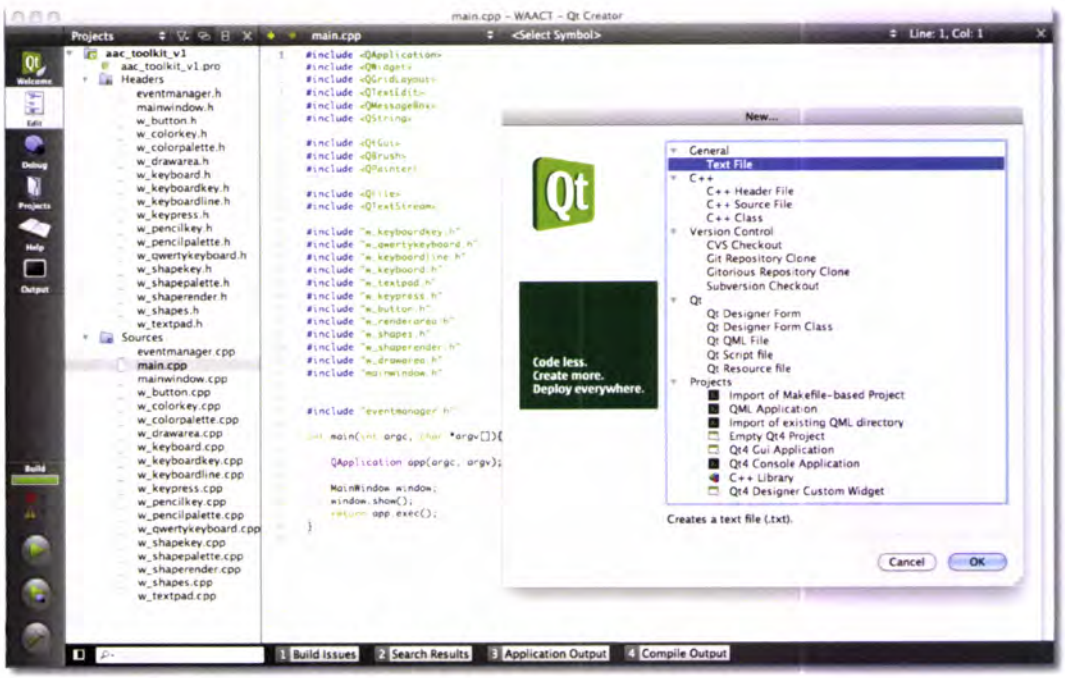

Fig. 12 – *QT Creator IDE*

Assim, numa óptica de programação orientada a objectos, foram envolvidas nas funções já existentes mecanismos que permitissem absorver as bibliotecas do QT, para que o programador apenas programe em C++ com recurso ao WAACT, libertando-o de mais uma tecnologia.

Nesta primeira versão acabaram por não ser absorvidas todas as funcionalidades do QT, tendo no entanto havido a preocupação de garantir que na utilização de componentes do WAACT apenas fosse necessário a utilização de funções próprias.

No entanto nesta versão, acabam por ser necessárias algumas funções do QT que se limitam apenas a garantirem a organização e "paginação" na *Mainwindow*.

A utilização do WAACT é muito similar a programação praticada noutras linguagens que recorrem a orientação por objectos.

Na Fig. 13 demonstra-se a instanciação de três objectos tecla que não estão integrados em nenhum teclado. Em cada uma destas teclas é aplicada, em seguida à sua instanciação, uma característica, sendo que no caso da primeira *W_KeyBoardKey*, que é designada por *k1*, apenas é definido que esta será uma tecla standard e que terá como *label* o caractere "A" e como output o caractere "a".

Na segunda *W_KeyBoardKey*, designada por *k2*, é aplicada a função *keySized()* que para além das características atrás definidas permite personalizar o tamanho da tecla.

```
50        // KeyBoardKey Standard
51        W_KeyBoardKey *k1 = new W_KeyBoardKey();
52        k1->keyStandard("A", "a");
53
54        // KeyBoardKey that allow a resized of the key
55        W_KeyBoardKey *k2 = new W_KeyBoardKey();
56        k2->keySized("B", "b", 50, 50);
57
58        // KeyBoardKey that allow a resized of the key and a look customization
59        W_KeyBoardKey *k3 = new W_KeyBoardKey();
60        k3->keyCustomized("C", "c", 70, 70, "yellow", "red", 22);
61
```

Fig. 13 – Código para a criação de uma tecla Standard, uma tecla dimensionada e uma personalizada

Por fim, no caso da *W_KeyBoardKey k3* a função *keyCostumized()* permite que para além das características já apresentadas seja feita uma personalização de factores tais como a cor, cor de letra e tamanho de letra.

A disponibilização de *widgets* completos é um dos principais objectivos deste projecto tendo em vista o aumento de produtividade dos programadores de softwares de CAA, no entanto, é importante não limitar os mesmos a uma ou outra característica especifica de um determinado controlo, permitindo a mais completa personalização possível, como no caso da Fig. 13.

Outro dos pontos já focados, é o facto de haver interoperabilidade, não só entre sistemas operativos, mas também entre funcionalidades, tendo sido para isso criado o *Event Manager*.

Na Fig. 14 é apresentado o código de um exemplo de uma aplicação simples programada no WAACT composta por uma área de texto e dois botões, dos quais um interage directamente com a *Mainwindow* (Fig. 6) e o segundo interage com outro controlo da aplicação (Fig. 7), neste caso um controlo de área de texto.

```
        // Text area
        W_TextPad *t = new W_TextPad();

        // Button standard with a label
        W_Button *b = new W_Button();
        b->buttonStandard("Sair!!!");

        // Button standard with a label
        W_Button *save = new W_Button();
        save->buttonStandard("Guardar!!!");

        // Event Manager
        EventManager *e = new EventManager();
        // Call the click event to exit and apply it to the "b" control
        e->clickedExit(b);
        // Call the click event of the "save" control to save the content of "t" control
        e->clickedSave(save, t);

        // QT function QHBoxLayout to create a layout
        QHBoxLayout *layout = new QHBoxLayout;
        // Apply "t" control to the layout
        layout->addWidget(t->textPad());
        // Apply "b" control to the layout
        layout->addWidget(b->getButton());
        // Apply "save" control to the layout
        layout->addWidget(save->getButton());
```

Fig. 14 – Código para a criação de uma aplicação simples composta por dois botões e uma área de texto

A instanciação dos objectos botão (*W_Button()*) e área de texto (*W_TextPad()*) é muito similar ao já apresentado na figura anterior, sendo assim criados nas primeiras linhas de código da Fig. 14, o controlo "t" que instancia a área de texto e os controlos "b" e "*save*" que instanciam botões de interacção.

Na linha seguinte a instanciação dum objecto *EventeManager()* permite a utilização de eventos aplicados aos controlos ou à *Mainwindow*.

No caso da utilização da função *ClickedExit()* como apenas estamos a passar um único parâmetro de entrada, sabemos que a interacção é feita entre o controlo e a *Mainwidow*, interacção esta que irá originar o fecho da mesma.

No caso da utilização da função *ClickedSave()* os dois parâmetros de entrada, indica-nos em primeiro lugar que estamos a tratar eventos de interacção entre dois controlos, neste caso um botão e uma área de texto. Por outro lado, podemos verificar que o controlo "*save*" origina a interacção recebendo os eventos de *click* sendo este reencaminhados para originar uma acção por parte do controlo "t".

Tanto num caso como no outro a utilização dos SIGNALs e SLOTs do QT é essencial para emissão de eventos no que diz respeito aos SIGNALs e para a execução das acções no que respeitas aos SLOTs.

Na Fig. 15 apresenta-se as duas funções de ligação do exemplo anterior, sendo que à primeira vista estas não parecem muito diferentes, no entanto, existe uma diferença substancial.

No QT a função *connect()* permite ligar SIGNALs a SLOTs ou até mesmo SIGNALs a outros SIGNALs, sendo isto, o que na pratica permite a ligação de funções à aplicação ou de funções a outras funções podendo estas ainda ser encadeadas umas nas outras.

```
//Create a click event for an Exit button
void EventManager::clickedExit(W_Button *button){

    connect(button, SIGNAL(buttonClicked()), qApp, SLOT(quit()));
}

//Create a click event for a Save button for a TextPad
void EventManager::clickedSave(W_Button *button, W_TextPad *textpad){

    connect(button, SIGNAL(buttonClicked()), textpad, SLOT(saveTextpadContentByDate()));
}
```

Fig. 15 – Excerto de código do *Event Manager*

Se olharmos atentamente para a primeira função, verificamos que o controlo que vai originar a acção a aplicar não está a ser passado como parâmetro de entrada. Tal situação ocorre porque esse controlo é a própria aplicação sendo esta um parâmetro geral e transversal ao sistema.

Por outro lado, a acção que está a ser aplicada a esse controlo (função *quit()*), que como vimos é a própria aplicação, não é nenhuma função do WAACT sendo esta uma funcionalidade do próprio QT o que permite uma interacção directa entre os eventos *click* aplicados a um determinado controlo e a própria *Mainwindow*.

Ao contrário do que acontece na primeira função, na segunda estamos a ligar dois controlos do WAACT, sendo que neste caso os eventos aplicados ao botão irão despoletar uma função que aplicará uma acção no controlo de destino.

Para além do facto deste tipo de funcionalidades permitir uma grande flexibilidade no desenvolvimento, estas também permitem o isolamento dos *widgets* entre si, o que permitirá um desenvolvimento modular e sem compromissos sobre os *widgets* já desenvolvidos.

Capítulo 4

4. Avaliação e Trabalho Relacionado

Após a codificação dos primeiros objectos, foi necessário validar a real mais-valia que poderia ser proporcionada por um *toolkit* deste tipo no desenvolvimento de *software* específico.

Assim sendo, foi importante avaliar e escolhas e caminhos percorridos neste projecto, tendo este, sido outro dos grandes desafios deste trabalho.

Como a área do desenvolvimento de *software* de CAA é uma área bastante restrita, e como não é conhecido actualmente nenhum projecto com os objectivos propostos pelo WAACT, não era possível a realização de uma avaliação comparativa nos moldes usualmente praticados, quer por comparação com softwares do mesmo tipo ou com o mesmo propósito já disponibilizados no mercado ou em fase de investigação, quer por avaliação de programadores da área, externos a envolvente do projecto.

Tendo em contas estas situações foi assim necessária a adequação dos métodos de avaliação para que se pudesse avaliar o sistema de uma forma concreta e quantificável.

4.1. Metodologia de avaliação

Devido aos problemas já abordados houve a necessidade de rever e adequar a metodologia de avaliação que poderia ser vista como a mais provável num projecto desta natureza.

A forma encontrada de fazer tal avaliação, foi a de implementar alguns protótipos funcionais previamente identificados e com características de softwares de CAA, e com base nesse desenvolvimento, serem aplicadas algumas métricas.

As métricas adoptadas neste caso forma as seguintes:

- Portabilidade;
- Tempo de implementação;
- Linhas de código;
- Esforço dispensado na programação.

Contudo, não podemos dizer que estas métricas nos esclarecem de uma forma inequívoca todos os factores de relevo em avaliações deste tipo, sendo por vezes também necessária uma análise, mesmo que cuidada, um pouco mais abstracta, por forma a poder interpretar os valores obtidos nos testes dos protótipos.

4.2. Protótipos e avaliação

A quando da implementação dos protótipos, foi importante definir que tipo de *software* nos traria mais vantagens para avaliação. Assim, verificou-se que mesmo não podendo comparar o WAACT com outra plataforma do mesmo tipo, poderiam ser comparados os seus resultados. Ou seja, o objectivo do WAACT é facilitar o desenvolvimento de aplicações de CAA, então porque não comparar algumas dessas ferramentas já existentes com outras desenvolvidas pelo WAACT?

Uma dessas ferramentas foi o CE-e, da qual se implementou um protótipo bastante simples que apenas dispunha das principais funcionalidades de *notetaking*.

Para além, da portabilidade entre vários sistemas operativos, tendo este protótipo sido compilado, executado e testado em Sistemas Microsoft, Linux e Mac; este teste

demonstrou-nos que estas aplicações poderiam ser implementadas com um relativo baixo esforço de programação, desde que os *widgets* pretendidos para o desenvolvimento da aplicação estejam disponíveis no WAACT. Verificamos assim, que com apenas 30 linhas de código (Fig. 16.); se pode obter uma aplicação gráfica, mesmo que rudimentar, mas funcional.

```
W_TextPad *t = new W_TextPad();

t->getTextpadContent();

W_Button *b = new W_Button();
b->buttonStandard("Sair!!!");

W_Button *save = new W_Button();
save->buttonStandard("Guardar!!!");

W_Button *clear = new W_Button();
clear->buttonStandard("Nova aula");

W_Button *next = new W_Button();
next->buttonStandard("Proxima >>");

W_Button *previous = new W_Button();
previous->buttonStandard("<< Anterior");

EventManager *e = new EventManager();
e->clickedExit(b);
e->clickedSave(save, t);
e->clickedNewTextpadContent(clear,t);
e->clickedNextTextpadContent(next,t);
e->clickedPreviousTextpadContent(previous,t);

QGridLayout *navigate = new QGridLayout;
navigate->addWidget(previous->getButton(),0,0,0,1);
navigate->addWidget(clear->getButton(), 0,1,0,6);
navigate->addWidget(next->getButton(),0,8);

QHBoxLayout *end = new QHBoxLayout;
end->addWidget(b->getButton());
end->addWidget(save->getButton());

QVBoxLayout *layout = new QVBoxLayout;
layout->addWidget(t->textPad());
layout->addSpacing(15);
layout->addLayout(navigate);
layout->addLayout(end);
```

Fig. 16 – Código de implementação de um sistema de *notetaking*

Como a extensão de esta aplicação abrange um número de linhas de código bastante reduzido e se está a trabalhar numa linguagem de programação de âmbito geral, podemos assim também afirmar que o tempo de implementação acaba por sua vez

por também ser muito reduzido, dependendo apenas da prática do utilizador na utilização da plataforma e da disponibilidade, ou não, dos *widgets* pretendidos na aplicação.

No entanto, mesmo que os referidos *widgets* não existam, estes podem sempre ser criados no padrão da modularidade imposta pelo WAACT, e assim ser reutilizados noutras aplicações.

Como já foi referido anteriormente, este projecto assentou no pressuposto da reutilização de componentes existentes noutras aplicações, bem como na experiencia de novas abordagens a sistemas existentes com a integração de novos componentes.

Assim, com vista a demonstrar a flexibilidade da plataforma neste âmbito foi introduzido neste protótipo um componente inexistente no Caderno Escolar Electrónico original, mas que poderia ser da grande utilidade neste sistema, um teclado de ecrã semelhante ao utilizado no Eugénio – Génio das Palavras (Fig. 17).

Neste caso fizemos a integração de um teclado QWERTY (disposição adoptada pelos países ocidentais), no entanto, poderão ser utilizados ou criados outros, sendo que, o WAACT está preparado para qualquer disposição de teclas, podendo inclusivamente ser criado um teclado de raiz.

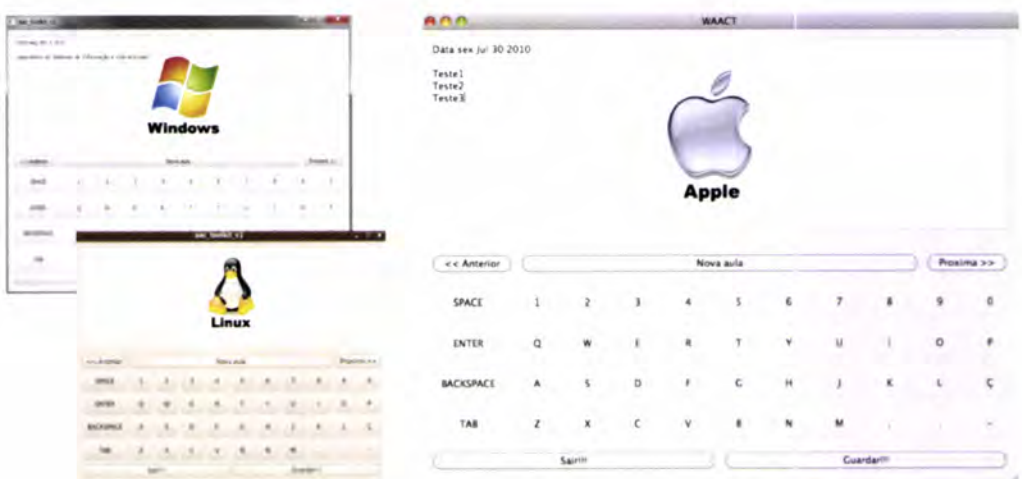

Fig. 17 – Protótipo de um sistema de *notetaking* com teclado de ecrã desenvolvido em WAACT e compilado em Windows e Linux e Mac

Baseado no protótipo anterior, a integração de um teclado virtual, que é um componente comum e bastante utilizado, necessitou apenas do registo de mais duas linhas, no código já implementado.

Com base nestes dados, podemos verificar que desde que alguns dos componentes necessários ao desenvolvimento de determinada aplicação estejam implementados no WAACT, o aumento de produtividade e eficiência é visível no desenvolvimento de soluções deste tipo.

Outra das ferramentas das quais se decidiu fazer um protótipo foi da aplicação de desenho *Paint*, muito utilizada por um grande número de utilizadores.

Á semelhança do protótipo anterior, a portabilidade nesta aplicação também foi verificada, tendo esta sido compilada, executada e testada com sucesso em Sistemas Microsoft, Linux e Mac.

```
161     W_DrawArea *drawArea = new W_DrawArea;                  //Set a drawArea Object
162     W_ColorPalette *colorPalette = new W_ColorPalette();    //Set a colorPalette Object
163     W_PencilPalette *pencilPalette = new W_PencilPalette(); //Set a pencilPaltte Object
164     W_ShapePalette *shapePaltte = new W_ShapePalette();     //Set a shapePaltte Object
165
166
167     W_Button *out = new W_Button();
168     out->buttonStandard("Exit");
169
170     W_Button *clear = new W_Button();
171     clear->buttonStandard("Clear");
172
173     /* Events for the clear and exit button */
174     EventManager *e = new EventManager();
175     e->clickedExit(out);
176     e->clickedClearDrawArea(clear, drawArea);
177
178     /* Layout for the clear and exit button */
179     QHBoxLayout *end = new QHBoxLayout;
180     end->addWidget(out->getButton());
181     end->addWidget(clear->getButton());
182
183     QVBoxLayout *layout = new QVBoxLayout;
184     layout->addWidget(shapePaltte->paletteStandard(drawArea));
185     /* Standard Color Palette */
186     layout->addWidget(pencilPalette->paletteStandard(drawArea));
187     /* Standard Pencil Palette */
188     layout->addWidget(colorPalette->paletteStandard(drawArea));
189     /* Draw Area */
190     layout->addWidget(drawArea);
191     /* Clear and Exit buttons */
192     layout->addLayout(end);
```

Fig. 18 – Codigo de implementação de uma aplicação de desenho

Para além, da já verificada portabilidade, o desenvolvimento deste protótipo demonstrou-nos mais uma vez que desde que os componentes que se pretendem incorporar numa aplicação façam parte do WAACT a implementação é feita com um relativo baixo esforço de desenvolvimento. Neste novo caso, com apenas 20 linhas de código (Fig. 18.) obtemos um protótipo perfeitamente funcional (Fig. 19.) com as principais funcionalidades de desenho possíveis de encontrar num sistema deste tipo.

Assim, este protótipo permite o desenho de formas geométricas básicas, bem como a definição da sua cor e espessura com a qual são desenhadas, sendo esta definição feita pela selecção do atributo desejado no desenho, nas respectivas paletes.

No caso deste protótipo, as paletes utilizadas foram as que estão definidas por defeito, no entanto, todos estes componentes são configuráveis, podendo ser criadas novas paletes apenas pela passagem de um vector que contenha o conjunto que se pretenda implementar, quer cores, quer espessuras.

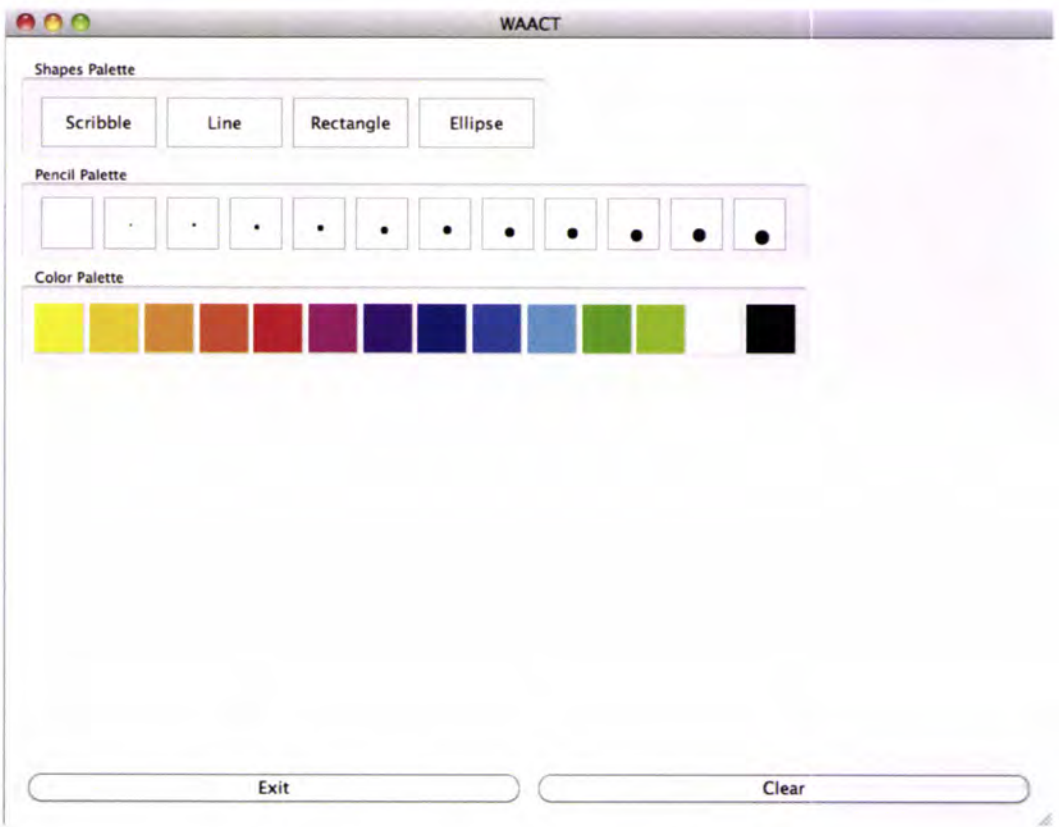

Fig. 19 – Protótipo de um sistema de desenho

Mais uma vez, como a extensão do código deste protótipo abrange um número bastante reduzido de linhas, e a metodologia de implementação é muito similar o que normalmente é praticado na programação por objectos, podemos novamente afirmar que o tempo de implementação acaba também por ser bastante reduzido desde que se cumpram os pressupostos já referidos.

Capítulo 5

5. Conclusões e Trabalho Futuro

Nesta tese propusemos e apresentamos uma plataforma de desenvolvimento de sistemas de Comunicação Aumentativa e Alternativa para programadores, com o objectivo de melhorar a produtividade e diminuir os tempos despendidos no desenvolvimento de soluções deste tipo.

Outro dos objectivos deste trabalho passava pela uniformização da estrutura e nas tecnologias utilizadas na implementação de *softwares* desta área muito específica, tendo como particular preocupação o facto de que a grande maioria dos programadores deste tipo de soluções trabalham sobre sistemas proprietários.

Actualmente existe um protótipo funcional do WAACT, onde já se encontram implementados variados *widgets*, tais como, teclados de ecrã e seus componentes, caixas de *rich text*, botões gerais e específicos, podendo estes ser isolados ou em paletes e ainda áreas de desenho, entre outros.

Neste momento podem ser criados projectos utilizando os componentes da plataforma ou ainda ser criados novos *widgets* para complementar o leque dos já existentes, podendo ainda estes ser independentes ou interligados com algum componente já existente.

Para além dos *widgets* propostos dispomos ainda de um módulo chamado de *Event Manager* que gere os eventos de *input*, processando-os e despoletando um evento de *output*, caso seja esse o objectivo, sendo este módulo a espinha dorsal da modularidade do sistema que permite o desenvolvimento de novos componentes dependentes ou independentes os já existentes.

Assim, tirando partido das propriedades dos *SIGNALs* e *SLOTs* do QT, o *Event Manager* recebe os eventos emitidos pelos componentes integrados numa determinada aplicação reencaminhando-os para outros componentes ou até para a própria *Mainwindow*, sem que o objecto de destino da acção despoletada pelo evento tenha conhecimento do próprio objecto de origem.

Como já foi referido, actualmente dispomos da estrutura do sistema, bem como de variados *widgets* bastante comuns em *softwares* de CAA. Por outro lado, ainda dispomos da possibilidade de proporcionar aos programadores deste tipo de soluções a criação de componentes que se adeqúem as suas necessidades.

Contudo, este é um trabalho que deve ser contínuo e atento às necessidades na área da CAA, devendo ainda ser implementados nesta solução os vários componentes que permitam completar, pelo menos para já, o leque das necessidades actuais.

Numa fase mais avançada, pretende-se disponibilizar o WAACT em regime de "*Open Source*", bem como uma documentação adequada que permita uma fácil utilização e expansão por parte dos programadores.

No entanto, nesta fase, podemos desde já validar o QT como uma escolha bem-sucedida no que diz respeito as questões de portabilidade, bem como, no que diz respeito a qualidade gráfica fornecida no desenvolvimento de aplicações.

Por outro lado, podemos também validar a escolha do C++ para linguagem de suporte como outra escolha bem-sucedida, mesmo que esta acabe por estar estreitamente ligada ao QT, sendo que, devida também à sua portabilidade poderemos estar aptos a trocar de *middleware* se tal situação se verificar necessária.

Por fim, gostaríamos de voltar a referir, que o trabalho aqui descrito e desenvolvido foi um dos eleitos entre os mais de 140 propostos na conferência Inforum 2010 [18], tendo sido a sua aceitação muito positiva.

Bibliografia

1. Universidade de Aveiro – Biblioteca digital, http://portal.ua.pt/bibliotecad, diponível em 05/05/2010
2. Scvcik, R., Romski. M.: Aac: More than three decades of growth and development. The ASHA Leader (2000)
3. Garcia, L.: Concepção, Implementação e Teste de um Sistema de Apoio à Comunicação Aumentativa e Alternativa para o Português Europeu. Master Thesis, Instituto Superior Técnico (2003)
4. Brown, P. F., Pietra, V. : Class-Based n-gram Models of Natural Language. Association for Computational Linguistics. Volume 18, Number 4 (1992)
5. Alexandre, L., Garcia, L., Bruno, L.: Development of an Electronic Scholar Notebook for Students with Special Needs. DSAI2007 - Vila Real (2007)
6. AAC Institute, http://www.aacinstitute.org, diponível em 13/05/2010
7. Sensory Software International, Lda., http://www.sensorysoftware.com/ Talkative.html, disponível em 19/05/2010
8. Vanderheiden, G. C., Kelso, D. P.: Comparative Analysis of Fixed-Vocabulary Communication Acceleration Techniques. AAC Augmentative and Alternative Communication, 3, 192--206 (1987)
9. Besio, S., Ferlino L.: Blissymbolics Software Worldwide: from Prototypes towards Future Optimized Products. In Proceedings of the 2nd ECART Conference (1993)
10. Schlosser, R. W., Koul, R., Raghavendra, P., Lloyd, L. L.: State-of-the-Art in Blissymbolics Research: Implications for Practice. In Proceedings of the 6th ISAAC Conference, 121--123 (1994)
11. Lundälv, M.: Comspec - The Advent of an Integrated Modular Communication System. Proceedings of the Future Integrated Solutions Conference, ACE Centre, Oxford (1994)
12. Sleeter Melissa E., OpenDoc—Building Online Help for a Component-Oriented Architecture, SIGDOC 96. (1996)
13. Lundalv Mats, Doeko Hekstra, Erlend Stav. Comspec, a Java Based Development Environment for Communication Aids. TIDE 98. Helsinki (1998)
14. Kouroupetroglou, G., Pino, A.: A New Generation of Communication Aids under the ULYSSES Component-Based Framework. The 5th International ACM SIGCAPH Conference on Assistive Technologies (2002)

15. Johnson, R. E.: Components, Frameworks, Patterns. Communications of the ACM (1997)

16. Gamma, E., Helm, R., Johnson, R., Vlissides, J. M.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley (1994)

17. Nokia QT – Cross Platform Application and UI Framework, http://qt.nokia.com, disponível em 02/07/2010

18. Fontes, G., Abreu, S.: WAACT - Widget Augmentative and Alternative Communication Toolkit, Inforum 2010, Universidade do Minho, Braga (2010)