Rapid Application Development

Otavio Rodolfo Piske – angusyoung@gmail.com

Fábio André Seidel – faseidel@yahoo.com.br

Especialização em Software Livre

Centro Universitário Positivo - UnicenP

Resumo

O RAD é uma metodologia de desenvolvimento de grande sucesso em ambientes proprietários. Embora as ferramentas RAD livres ainda sejam desconhecidas por grande parte dos desenvolvedores, a sua utilização está ganhando força pela comunidade de software livre.

A quantidade de ferramentas livres disponíveis para o desenvolvimento RAD aumentou nos últimos anos e mesmo assim uma parcela dos desenvolvedores tem se mostrado cética em relação a maturidade e funcionalidades disponíveis nestas ferramentas devido ao fato de elas não estarem presentes, por padrão, na maioria das distribuições mais utilizadas. Além disso, elas não contam com o suporte de nenhum grande distribuidor e, ainda que sejam bem suportadas pela comunidade, este acaba sendo um empecilho para alguns desenvolvedores.

Outro foco para se utilizar no desenvolvimento RAD é a utilização de frameworks, onde esses estão disponíveis para desenvolvimento em linguagens como C e C++ sendo as mais utilizadas em ambientes baseados em software livre, embora estas linguagens não sejam tão produtivas para o desenvolvimento de aplicações rápidas.

Abstract

RAD is a highly successful software development methodology in proprietary environments. Though free RAD tools is yet unknown for a great range of developers, its usage is growing in the free software community.

The amount of free RAD tools available has increased in the last years, yet a considerable amount of developers is skeptic about the maturity and features available in these tools due to the fact that they are not available by default on the biggest distribution. Additionally, they do not have the support of any big free software supplier and, though they are well supported by the free software community, this represents a major drawback for some developers.

Another focus on RAD development is the use of frameworks, available for languages like C and C++ which are the most used languages on free software based environments, though they are not as productive for RAD development.

1. Introdução

O RAD (*Rapid Application Development*) surgiu na década de 70, anos em que o desenvolvimento de aplicações demorava tanto que se tornava comum os requerimentos mudarem antes que a aplicação estivesse pronta. A formalização do RAD como prática de desenvolvimento se deu em 1991, com a publicação do livro *“Rapid Application Development”*, escrito por James Martin. Este livro era um refinamento dos conceitos de prototipação e desenvolvimento iterativo propostas no livro *“A Spiral Model Of Software Development and Enhancement”.[1]*

Uma vez que o RAD encoraja a

participação do usuário no processo de análise e *design*, o produto final tende a ter custos menores de manutenção e menor tempo de desenvolvimento, em geral compromentendo a escalabilidade e o desempenho do sistema.

As Ferramentas RAD estão há muito tempo inseridas na principal plataforma de sistemas proprietários (sistemas operacionais Windows). Neste sistema operacional se pode fazer a utilização de duas ferramentas bastante conhecidas no mercado de desenvolvimento que são o Microsoft Visual Basic e o Borland Delphi.

O Visual Basic é um produto composto de um ambiente de desenvolvimento e uma linguagem de programação baseada em eventos, o que significa que a organização do fluxo lógico do programa se concentra em torno dos eventos emitidos pela interface gráfica.

Outra ferramenta para desenvolvimento rápido de aplicações e baseado na linguagem Object Pascal é o Delphi. O Delphi também oferece um ganho de produtividade através da utilização de uma linguagem prática e direta e uma curva de aprendizado menor devido as construções simples utilizadas pela linguagem.

Em 2001 a Borland (responsável pelo Delphi) lançou o Kylix, uma versão do Delphi voltada para Linux. Este produto, devido a seus inúmeros bugs, não foi bem recebido pelos desenvolvedores e acabou sendo abandonado após o lançamento da terceira versão. Uma curiosidade é que assim como o Delphi, cujo nome é baseado na cidade grega de Delphi[2], o nome Kylix também deriva da cultura grega, e seu significado é xícara.

Atualmente é natural que desenvolvedores de software procurem por alternativas livres cujas funcionalidades sejam capazes de atender as necessidades encontradas no desenvolvimento de aplicações rápidas.

Ainda que a maioria das ferramentas para desenvolvimento de aplicações rápidas disponíveis atualmente tenha suas funcionalidades voltadas para o desenvolvimento nas linguagens C++ e Java, é importante citar que existem alternativas livres para as linguagens mais utilizadas em ambientes proprietários como o Visual Basic e o Delphi.

2. Ferramentas RAD Livres

Algumas das ferramentas RAD para ambientes livres ainda são desconhecidas por muitos profissionais de TI, levando-os a acreditar que o desenvolvimento de aplicações para Linux e ambiente UNIX ainda se encontra preso ao VI; um editor de textos bastante utilizando em ambientes Unix/Linux e cuja principal característica é a possibilidade de trabalhar em 2 modos visuais diferentes, desta maneira sendo pouco amigável a usuários pouco acostumados a trabalhar sem ambientes de desenvolvimento. Além disso, a imensa gama de ferramentas de desenvolvimento auxiliares como Makefiles (arquivos de script, cuja finalidade é automatizar o processo de construção e instalação de softwares a partir do código fonte) e a linha de comando. Abaixo será abordado em detalhes sobre algumas ferramentas RAD disponíveis para ambientes livres e como os desenvolvedores de Software, podem se beneficiar dessas ferramentas, aumentando o número de plataformas nas quais os seus softwares estão disponíveis.

2.1.Hbasic

O Hbasic é uma IDE (Integrated Development Environment - Ambiente Integrado de Desenvolvimento) livre utilizada para desenvolver programas utilizando uma linguagem semelhante ao Basic ou C#, possibilitando ao desenvolvedor utilizar 3 bibliotecas básicas diferentes: Hbasic Standard (stdgui), Hbasic NET (.Net Library) e QT-C. Esta possibilidade de escolher diferentes “dialetos” permite ao desenvolvedor uma curva de aprendizado menor nos casos aonde o desenvolvedor já conhece uma das linguagens em questão (como no caso do C#).

O Hbasic é desenvolvido utilizando a linguagem C++ e o *framework* de desenvolvimento C++ QT, que será

abordado futuramente em outro tópico, desta maneira tendo todo o “look and feel” de uma aplicação nativa Linux e mantendo
o desempenho dentro de padrões aceitáveis para aplicações desktop. Além disso, sua IDE conta com todas as funcionalidades esperadas em um ambiente de desenvolvimento moderno, como debugger embutido, editor de interfaces gráficas, etc. O compilador utilizado pelo Hbasic ainda possibilita gerar binários estáticos, desta maneira não sendo necessário distribuir outras bibliotecas em conjunto com o executável.

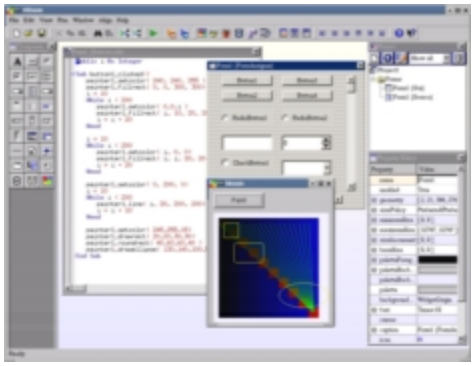

Figura 1. - Ambiente de desenvolvimento do HBasic

A linguagem utilizada pelo Hbasic é uma derivada do Basic, com suporte a orientação a objetos, acesso a múltiplos bancos de dados, criação de componentes em C++, possibilitando extender as funcionalidades inexistentes no Hbasic. Ao utilizar classes baseadas no modelo de hierarquia utilizado pela QT (ex.: criando uma classe derivada da Qobject) é possível utilizar o sistema de sinais e slots – este assunto será discutido em tópicos subseqüentes – fornecido pela QT.

Embora o Hbasic seja um projeto capaz de despertar o interesse dos fãs das linguagens baseadas no Basic, o HBasic parece estar descontinuado, não tendo atualizações já que, segundo a página oficial do projeto, a última atualização ocorreu em abril de 2004.

2.2.Gambas

O Projeto Gambas tem como produto de desenvolvimento um ambiente integrado de desenvolvimento baseado em um interpretador Basic com suporte a Orientação a Objetos. Segundo os desenvolvedores, a linguagem utilizada pelo Gambas lembra, de alguma maneira, o Visual Basic. Entretanto, os desenvolvedores ressaltam que esta não é um clone da linguagem utilizada no Visual Basic e não tenta ser compatível com a mesma ou com a utilizada no Visual Basic.Net.

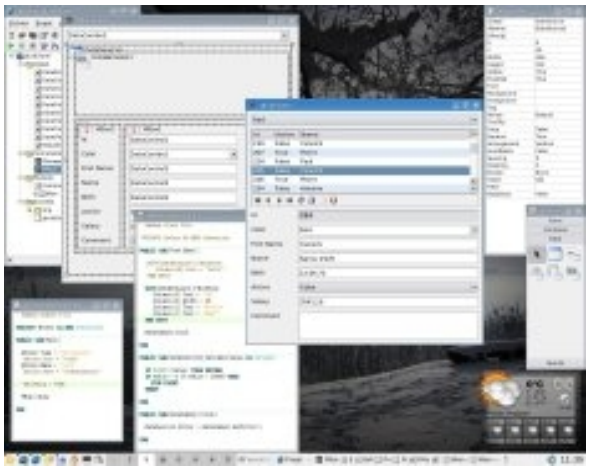

Figura 2. Ambiente de desenvolvimento do Gambas

O Gambas, cujo nome é um acrônimo recursivo cujo significado é “Gambas Almost Means BASic” (Gambas quase significa Basic) é desenvolvido em C++ e, assim como o Hbasic, utiliza os componentes do *framework* QT para desenvolvimento da sua interface gráfica.

O Gambas é multiplataforma, funcionando sob diferentes versões dos sistemas operacionais da família BSD além de funcionar no Windows através do Cygwin. Desenvolvedores interessados em utliizar bancos de dados em suas aplicações tem a possibilidade de utilizar PostgreSQL, MySQL e SQLite. Segundo os desenvolvedores espera-se que a próxima versão do Gambas, atualmente em desenvolvimento, suporte muitos outros bancos de dados [3].

A compilação de projetos do Gambas requer apenas a compilação das classes modificadas, desta maneira diminuindo o tempo de compilação e, por consequência, de desenvolvimento.

O interpretador Basic utilizado pelo Gambas é desenvolvido através de uma arquitetura de componentes, de tal maneira que no futuro poderá permitir escolher qual

toolkit gráfico utilizar.

O Gambas, como informado anteriormente, não é compatível com o Visual Basic (embora seja similar ao Visual Basic em algumas areas), sendo os principais pontos aonde é possível encontrar diferenças em relaçao ao Visual Basic os seguintes:

- Extensões de arquivos dos arquivos utilizados no projeto (classes, formulários, etc)
- Controles de formulários (componentes de um formulário) são “private” por padrão.
- Funções de conversão de expressões, valores, strings e datas do Gambas respeitam as configurações regionais (“locale”).
- O Gambas utiliza passagem por valor para tipos de dados simples (inteiros, strings, etc) e não podem ser passadas por referência, como no Visual Basic (embora o Visual Basic possa passar parâmetros por valor através do uso da palavra-chave ByVal).
- Não existem variáveis globais de escopo de projeto no Gambas. A documentação do Gambas ensina como simular o uso de variáveis globais através do uso de atributos estáticos públicos.
- No gambas o operador “+” somente pode ser utilizado para operações aritiméticas e não para concatenar strings.
- No gambas não é possível utilizar GOTO para capturar excessões e/ou erros.
- Os formularios do Visual Basic utilizam “twips” como unidade de medida de posicionamento nos formulários. O Gambas trabalha diretamente com pixels.
- O Gambas utiliza UTF-8 (UCS Transformation Format), desta maneira gerando aplicativos completamente passíveis de tradução e internacionalização.

O Gambas, ao contrário do Hbasic, é uma ferramenta mais profissional e completa, podendo ser utilizada para desenvolvimento de aplicativos *desktop* bem como aplicações comerciais.

2.3.Lazarus

O Projeto Lazarus desenvolve um conjunto de bibliotecas para o compilador FreePascal cujo objetivo é emular o Delphi. O compilador no qual o projeto é baseado, o Free Pascal, não apenas é capaz de entender a sintaxe utilizada pelo Delphi mas também é capaz de rodar em diversas outras plataformas como OS/2, Mac OS X[4].

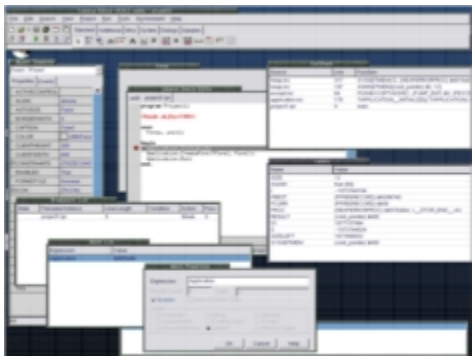

Figura 1. Ambiente de desenvolvimento do Lazarus

Ao contrário das ferramentas demosntradas anteriormente, o Lazarus é desenvolvido utilizando o toolkit gráfico GTK+ 1.x (figura 1.3), embora os desenvolvedores já estejam trabalhando para porta-lo para GTK+ 2.x e QT.

Um dos objetivos do projeto Lazarus é ser compatível com Delphi de tal maneira que seja possível converter projetos do Delphi e do Kylix para que sejam desenvolvidos no Lazarus. Segundo os desenvolvedores já é possível portar projetos do Kylix ou do Delphi, desde que não sejam utilizados componentes específicos do sistema operacional[4]. A compatibilidade com o Delphi é atingida através do LCL (Lazarus Component Library), uma biblioteca de componentes que é altamente compatível com a VCL (Visual Componet Library) utilizada pelo Delphi.

O Lazarus, assim como o Delphi, pode ser utilizado para desenvolver os mais variados tipos de aplicações incluindo:

- Aplicativos console: aplicativos

console são aplicativos que não tem uma interface gráfica. Isso é útil para desenvolvimento de aplicativos como servidores de aplicação, aplicativos de modelagem e processamento de dados.
- Bibliotecas de carregamento dinâmico: o Lazarus também permite criar bibliotecas de funções, passíveis de serem utilizadas por outros programas e linguagens. Essas bibliotecas são os arquivos .dll no windows e .so no Linux/*BSD.
- Aplicativos GUI: aplicativos com interface gráfica de usuário.

O Lazarus é uma ferramenta tão completa quanto o Gambas e com funcionalidades capazes de atender as necessidades dos órfãos do Kylix.

3. Frameworks

Um *Framework* é uma estrutura de suporte no qual um projeto de software pode ser organizado e desenvolvido, desta maneira evitando que um desenvolvedor tenha que criar uma estrutura auxiliar para o desenvolvimento de sua aplicação. [6]

O grande objetivo dos *frameworks* é tornar o desenvolvimento de aplicações menos oneroso ao desenvolvedor, desta maneira fornecendo a fundação necessária para que o desenvolvedor gaste mais tempo trabalhando com os requerimentos do software do que com os detalhes de baixo nível inerentes ao seu desenvolvimento.

Em ambientes livres como o Linux e o FreeBSD, a maioria dos aplicativos são desenvolvidos nas linguagens C, C++ e Java[7]. Sendo as duas primeiras linguagens mais complexas e, no geral, menos produtivas do que as linguagens citadas até o momento neste artigo a utilização de *Frameworks* é um diferencial considerável quando se deseja aumentar a produtividade e ainda contar com os poderosos recursos disponíveis nestas linguagens. Deste modo, neste artigo são abordados dois *Frameworks* que atendem exatamente esses objetivos e estão disponíveis para as linguagens C e C++.

3.1.GTK+

A GtK+ (*The Gimp ToolKit*) não é, exatamente, um *framework* devido as suas inúmeras características de baixo nível. Entretanto seria injusto não cita-lo, devido a sua enorme importância no desenvolvimento de software livres. A GTK+ é mais conhecida por ser a biblioteca utilizada no desenvolvimento do ambiente de desktop Gnome (Figura 2.1), amplamente utilizado no Linux e FreeBSD e variantes.

A GTK+ é composta de um conjuto de bibliotecas cuja função é auxiliar o desenvolvimento de interfaces gráficas e permitir ao desenvolvedor suportar um amplo conjuto de funcionalidades, variando desde internacionalização até um conjuto de interfaces para acessibilidade. Sendo escrita na linguagem C, a GTK+ é constituída de aproximadamente 500 mil linhas de código fonte[8].

Figura 2. Aplicativos usando GTK+

Embora existam ferramentas para desenhar as interfaces gráficas em GTK+, por padrão elas são criadas chamando diretamente as funções existentes na biblioteca, deste modo exigindo do desenvolvedor um profundo conhecimento das funções disponíveis, para que seja possível desenvolver um aplicativo.

A GTK+ é um *toolkit* gráfico baseado em eventos, isto é, a execução do programa fica em espera até que um evento ocorra e a função apropriada seja chamada.

3.2.wxWidgets

O wxWidgets é um conjuto de

bibliotecas que permitem aplicações em C++ compilarem e rodarem em diferentes tipos de computadores e sistemas com mudanças mínimas no código fonte. O desenvolvimento do wxWidgets inciou-se em 1992 na Universidade de Edinburgh por Julian Smart[9].

Uma vez que o wxWidgets provê uma camada de abstração entre as APIs do sistema operacional (incluindo as bibliiotecas responsáveis pelo desenvolvimento de interfaces gráficas), as aplicações desenvolvidas utilizando este *framework* tem um "*look-and-feel*" do sistema operacional em questão. No Windows a wxWidgets utiliza a API nativa do Windows, no Linux é utilizada a GTK+, nos Unices a Motif e no Mac OS X a cocoa é utilizada. O wxWidgets suporta ainda uma vasta gama de outros sistemas operacionais como HP-UX, AIX, Solaris e outros.

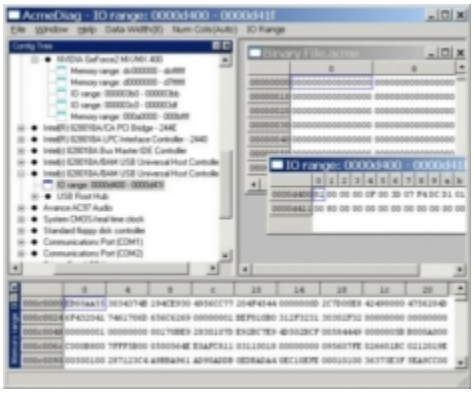

Figura 3. Aplicação desenvolvida com wxWidgets

Além das funcionalidades básicas utilizadas pelas aplicações GUI, o wxWidgets ainda oferece o suporte a:

- Suporte a threads
- Drag and drop
- Suporte a desenvolvimento de aplicações de rede com suporte a sockets, smtp, http e ftp.
- Bancos de dados, através de DAO.
- Visualização e impressão de páginas HTML utilizando diferentes engines de renderização (mozilla e IE).
- Unicode.

O wxWidgets ainda é reconhecida por ser semelhante a MFC o que, a torna atrativa para desenvolvedores acostumados com desenvolvimento em Windows mas que desejam suportar outras plataformas.

O wxWidgets usa uma tabela de eventos para ligar os eventos os métodos responsáveis por trata-las.

3.3.QT

A QT é um *framework* de desenvolvimento criado pela empresa norueguesa Trolltech, e inicialmente disponibilizado em 1995.

A QT, ao contrário da wxWidgets não é uma camada de abstração entre as APIs gráficas do sistema operacional, mas sim uma reimplementação das mesmas utilizando uma API gráfica e uma engine de renderização próprias. Ainda assim é possível fazer, através de temas e/ou estilos com que as aplicações desenvolvidas em QT (Figura 2.3) fiquem parecidas com uma aplicação nativa.

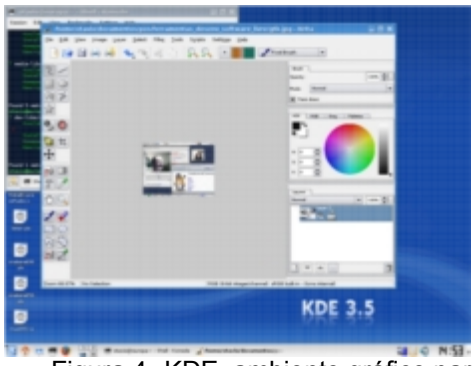

Figura 4. KDE, ambiente gráfico para Unix e Linux, desenvolvido com a QT.

A QT, assim como a GTK+, é baseada em eventos, utilizando um esquema de slots e sinais para efetuar a ligação entre os eventos "emitidos" pelo *framework* e os "*slots*" desenvolvidos pelo programador.

Assim como a maioria dos *frameworks* a QT oferece uma série de funcionalidades, *wrappers* e ferramentas para auxiliar no desenvolvimento de aplicações. Destacando-se:

- Suporte a threads.
- Suporte nativo aos bancos de dados PostgreSQL, MySQL, DB2, InterBase, SQLite, Oracle,

Sybase.
- Suporte a desenvolvimento de aplicações de rede com suporte a sockets, http e ftp.
- Suporte ao desenvolvimento de servidores de rede.
- Possibilidade de integrar controles ActiveX em aplicativos rodando sobre a plataforma Windows.
- Engine HTML embutida.
- Possibilidade de desenvolvimento de aplicações sem a necessidade de embutir uma interface gráfica. Isso torna possível utilizar as classes disponíveis na QT para o desenvolvimento de aplicações console. (Disponível na QT 4.0 ou superior)
- Engine, parser e gerador XML embutido, com suporte a DOM e SAX.

De todos os *frameworks* demonstrados anteriormente a QT é, de longe, o mais completo.

Um dos destaques da QT é o seu conjunto de ferramentas auxiliares ao desenvolvimento de aplicações. É importante citar:

- Qmake: aplicativo responsável por gerenciar e executar, de maneira portável, os scripts de construção e compilação do aplicativo.
- Assistant: um aplicativo que serve como um navegador da imensa documentação disponibilizada pela Trolltech. O Assistant lembra, de certo modo, as ferramentas disponibilizadas na MSDN Library (a documentação de desenvolvimento do Windows, disponibilizada pela Microsoft).
- Designer: ao contrário os outros *frameworks*, a Trolltech fornece como parte do pacote padrão disponibilizado na QT um intuitivo designer de interfaces gráficas.
- Linguist: utilitário utilizado para efetuar a tradução de aplicativos desenvolvidos com a QT.

A Troltech disponibiliza documentação de qualidade sobre a QT, tornando o seu aprendizado mais prático visto as outras alternativas.

4. Conclusão

As ferramentas RAD e os *Frameworks* oferecem um enorme ganho de produtividade para o desenvolvedor, tornando possível dedicar mais tempo à parte do desenvolvimento ligada às regras de negócio. A partir da analise de algumas das ferramentas RAD disponíveis, bem como suas funcionalidades e defeitos é possível identificar as ferramentas mais adequadas para cada trabalho.

O Gambas demonstrou-se a ferramenta RAD mais madura entre todas as ferramentas abordadas, possuindo características capazes de tornar o desenvolvimento mais produtivo, como por exemplo o suporte internacionalização. A ferramenta RAD Lazarus, é similar ao Gambas, mas possui dependência de bibliotecas ultrapassadas, deixando o desenvolver sem opções e tendo que depender de bibliotecas que ficarão sem suporte em poucos anos.

Ainda que as ferramentas RAD livres tenham que evoluir bastante até chegarem ao nível das ferramentas RAD proprietárias, é bastante claro que elas podem ser utilizadas para o desenvolvimento de aplicações simples, protótipos e até mesmo para portar aplicações originalmente desenvolvidas para ambientes proprietários.

Entre os *Frameworks* abordados, todos demonstraram um bom nível de produtividade e diferenciam-se apenas pelas ferramentas disponibilizadas, documentação e licença. O wxWidgets possui similaridade com a MFC sendo uma boa opção aos desenvolvedores C/C++ acostumados com o ambiente Windows.

Os Frameworks, especialmente a GTK+ e a QT, são utilizados em inúmeros grandes projetos (proprietários e livres) sendo possível assegurar sua maturidade, confiabilidade e estabilidade.

5. Referências

[1] Blue Ink.biz. Rapid Application Development. http://www.blueink.biz/RapidApplicationDevelopment.aspx, visitado em 29/05/2006.

[2] Thorpe, Danny. Borland History: Why the name Delphi? http://bdn.borland.com/article/0,1410,20396,00.html, visitado em 19/05/2006.

[3] Gambas Documentation. http://gambasdoc.org/help/doc/faq, visitado em 19/05/2006.

[4] Lazarus Project. http://www.lazarus.freepascal.org/modules.php?op=modload&name=StaticPage&file=index&sURL=about, visitado em 19/05/2006.

[5] Lazarus Project. http://www.lazarus.freepascal.org/modules.php?op=modload&name=FAQ&file=index&myfaq=yes&id_cat=1#q6, visitado em 19/05/2006.

[6] Framework. http://en.wikipedia.org/wiki/Framework, visitado em 19/05/2006.

[7] Freshmeat.net http://freshmeat.net/browse/160/, visitado em 19/05/2006.

[8] Taylor, Owen. Why GTK_MODULES is not a security hole. http://www.gtk.org/setuid.html, visitado em 19/05/2006.

[9] Bolia, Priyank. Introduction to wxWidgets. http://www.codeproject.com/library/wxwidgets.asp, visitado em 19/05/2006.