Interfaces Gráficas

Os produtos de software atuais são muito mais complexos do que eram há apenas alguns anos atrás.

Paralelamente, nos últimos 10 anos o computador passou a ser utilizado por usuários leigos em computação, e a experiência desses usuários com produtos como pl anilhas, editores de texto, e programas de apresentação os convenceu que um programa de computador pode ser fácil de usar.

Parte dessa facilidade foi proveniente da utilização de interfaces gráficas homem-computador com elementos do tipo janelas, menus, m ouse e ícones, que representam um grande avanço em tornar os computadores pessoais mais amigáveis.

Embora fáceis de usar, aplicativos com interfaces gráficas são difíceis de desenvolver, envolvendo paradigmas não familiares para grande parte dos projetistas de software tais como orientação a objetos e processamento dirigido por eventos.

Estão disponíveis no mercado várias ferramentas para desenvolvimento de interfaces gráficas que oferecem uma gama de funcionalidade bastante variável, e a escolha da ferramenta para um projeto pode ter consequências não somente em sua fase de desenvolvimento, mas também na sua evolução, uma vez que algumas ferramentas permitem o desenvolvimento de código fonte portátil, isto é, que pode ser transportado para vários ambientes sem necessidade de modificações.

Por sua vez, o teste de aplicativos com interfaces gráficas também não é uma tarefa simples, e realizar tais testes manualmente é uma tarefa tediosa e pouco confiável. Encontram-se disponíveis no mercado algumas ferramentas que permitem automatizar a fase de testes, mas que apresentam preço e funcionalidade bastante díspares, não sendo fácil a escolha de uma delas.

Embora o uso do ambiente ms-windows já esteja consolidado no brasil, a indústria nacional de software somente agora começa a produzir produtos de modo sistemático para esse ambiente.

É dentro desse contexto que o cti, através do programa de qualidade e produtividade em software, decidiu montar um grupo com o objetivo de absorver tecnologia avançada para a produção de interfaces gráficas portáteis de alta qualidade e repassar essa tecnologia para a indústria nacional de software.

Neste documento procuramos, de forma simplificada, apresentar a experiência adquirida e as perspectivas atuais do nosso grupo, abordando o trabalho realizado nas áreas de portabilidade, testes de aplicativos e metodologia para desenvolvimento de interface s gráficas.

Atualmente a microsoft detém a maior parcela da base de sistemas operacionais instalados com a dupla ms-dos e ms-windows. Este fato faz com que as plataformas da microsoft sejam o principal alvo dos desenvolvedores de software, acarretando uma grande concorrência. Embora uma pequena fração da base instalada de sistemas operacionais não seja microsoft, o tamanho do mercado mundial de software (aproximadamente 200 bilhões de dólares/ano) torna não desprezível o oferecimento de aplicativos para outras plataformas. Uma empresa que domine uma tecnologia que permita desenvolver produtos para o ambiente microsoft sem abandonar as outras fatias do mercado, passa a ter uma importante vantagem competitiva em suas mãos e, portanto, portabilidade torna-se uma palavra -chave nesse contexto.

A questão portabilidade encontra-se razoavelmente equacionada nas áreas de linguagens de programação, acesso a dados e comunicações através dos padrões ANSIC, SQL e TCP/IP ou OSI/ISO, entre outros.

O mesmo não se pode afirmar no caso de interfaces gráficas pois cada ambiente (ms-windows ou os/2, por exemplo) possui um conjunto de recursos básicos, o toolkit, para a implementação de interfaces gráficas, todos desenvolvidos de forma independente e não padronizada. A falta de padronização acarreta a necessidade de se reprogramar totalmente os módulos de controle da interface gráfica quando

se transporta um aplicativo para um ambiente distinto de seu ambiente alvo inicial. Essa reprogramação pode ser ex tremamente cara pelo fato de, em geral, a maior parte do código de um aplicativo tratar sua interface gráfica.

Quando se consideram os diversos ambientes gráficos percebe-se que a funcionalidade básica oferecida é, em grande parte, idêntica, sendo que as principais diferenças estão relacionadas com a forma pela qual elementos da interface como barras de rolagem, botões e janelas são representados graficamente.

Uma vez que a maior parte dos ambientes gráficos apresenta um conjunto comum de objetos com funcionalidades idênticas, uma das soluções para o problema de portabilidade de interfaces gráficas é a definição e implementação de uma interface de programação (api) padronizada que suporte esse conjunto comum e que o mapeie nos toolkits originais (figura 1). Dessa forma, um aplicativo desenvolvido com a interface de programação padronizada pode ser facilmente transportado para diversas plataformas, preservando s empre a aparência e funcionalidade, o look-and-feel, da plataforma na qual é executado. Várias empresas desenvolveram ferramentas baseadas em toolkits portáteis tais como zinc, zapp e xvt.

Após ter realizado um levantamento das ferramentas disponíveis no mercado, o grupo escolheu o xvt para ganhar experiência no uso de tecnologia para a produção de interfaces gráficas portáteis. Essa ferramenta fornece uma api padrão que permite a implement ação de tais interfaces e seu transporte entre os principais ambientes gráficos atuais (ms-windows, ms-windows nt, presentation manager OS/2, OSF/MOTIF e macintosh).

A ferramenta apresenta características de prototipação rápida, permitindo que seus usuários simulem, na fase inicial do desenvolvimento, a execução da interface do aplicativo, possibilitando o reconhecimento de possíveis problemas de projeto [argollo 1995].

O XVT é composto por dois módulos principais: um editor de interfaces gráficas e uma biblioteca de funções multi-plataforma. O editor de interfaces gráficas permite a criação dos objetos da interface de modo interativo e gera um arcabouço da aplicação na linguagem c/c++ com o código necessário para o controle da interface [xvt 1994]. A biblioteca multi-plataforma dispõe de primitivas para a manipulação dos objetos básicos de interfaces gráficas: janelas, menus, controles (botões de rádio, campos de edição , tabelas, planilhas, etc.), e funções para operações gráficas, de impressão, manipulação do clipboard e de fontes de caracteres, etc.

Após ter utilizado o xvt para desenvolver várias aplicações e de tê-las transportado da plataforma ms-windows para a plataforma osf/motif e vice-versa podemos afirmar o seguinte:

é possível desenvolver interfaces gráficas portáteis com a tecnologia existente desde que seja utilizada uma metodologia voltada para a obtenção de portabilidade;

Normalmente as ferramentas não são orientadas para domínios específicos e, por isso, funcionalidades do tipo acesso a base de dados ou geração automática de relatórios devem ser implementadas através de pacotes específicos;

O código gerado pelas ferramentas não impacta o desempenho da aplicação final;

As ferramentas não permitem abstrair todos os detalhes de cada plataforma; por exemplo, o ambiente osf/motif direciona saídas impressas para um arquivo postscript, enquanto o ms-windows as envia para o gerente de impressão.

Embora as afirmações feitas acima tenham sido observadas a partir do uso de uma única ferramenta, o xvt, em apenas dois ambientes distintos (ms-windows e OSF/MOTIF), não existe razão técnica que

as invalide quando consideradas outras ferramentas com as me smas características ou outros ambientes.

O teste de aplicativos com interfaces gráficas homem-computador é uma atividade difícil de realizar porque a complexidade desse componente de software, notada através de características como processamento multi-tarefa, manipulacão direta de objetos com resposta ao usuário em tempo-real e processamento dirigido por eventos, é responsável por uma explosão combinatória de cenários com as possíveis interações homem-computador que devem ser testadas.

Embora existam poucas referências sobre métodos e estratégias para teste de interfaces gráficas na literatura técnica, diversas ferramentas cujo objetivo é a automação desses testes estão disponíveis no mercado.

Essas ferramentas utilizam o método "gravar e repetir" para capturar tanto as ações do usuário sobre a interface quanto os resultados dessas ações. O conjunto de ações e resultados capturados são mapeados numa linguagem de programação e gravados em arquivos denominados scripts de teste. As ferramentas permitem a repetição das ações do usuário e a comparação dos resultados obtidos através da execução automática dos scripts de teste.

Essa abordagem do problema exige que a interface gráfica a ser testada esteja operacional e executando de modo confiável para que o usuário possa gravar suas ações nos diversos cenários de interação. Essa é uma limitação importante porque a fase de testes fica relegada ao final do desenvolvimento, quando talvez já não haja tempo disponível para realizá-la de modo adequado. Apesar disso, o fato das ferramentas repetirem os scripts de teste já gravados permite que testes regressivos sejam realizados sempre que o software tenha sido modificado, o que é um passo importante para a obtenção de produtos robustos.

Com o objetivo de permitir o desenvolvimento de scripts de teste sem que a interface esteja implementada, foram criados mecanismos que permitem a especificação das ações do usuário em termos dos objetos presentes na interface gráfica. Esse novo enfoque do problema tornou possível a geração de casos de teste a partir da especificação da interface gráfica, permitindo que a fase de testes seja realizada em paralelo com a fase de implementação.

O grupo de interfaces gráficas selecionou e adquiriu a ferramenta qa partner que incorpora uma tecnologia moderna para testes automáticos de aplicativos com interface gráfica [segue 1994].

O primeiro passo para o desenvolvimento de casos de teste com essa ferramenta é a declaração dos elementos da interface. Esse passo permite que os objetos presentes na interface sejam referenciados por nomes mnemônicos e facilita o desenvolvimento de scri pts de teste robustos, uma vez que mudanças de posição, tamanho e rótulo dos objetos na interface não irão impactar de modo significativo os testes já desenvolvidos.

O passo seguinte é o desenvolvimento de casos de teste na linguagem 4test que é orientada a objetos e possui uma sintaxe bastante semelhante à da linguagem c. A linguagem 4test possui comandos que referenciam os objetos da interface através de seus nomes mnemônicos anteriormente declarados. Por exemplo, o seguinte comando

gotoline.ok.click () ;

especifica um clique no botão "ok" da janela "gotoline" do aplicativo sendo testado.

Com a ferramenta, os scripts de teste podem ser desenvolvidos de forma manual, sem a necessidade de o aplicativo estar operacional, ou automaticamente, com ajuda de um módulo que captura as ações do usuário, caso o aplicativo tenha uma versão executável disponível. Os scripts de teste desenvolvidos controlam a execução do aplicativo em análise, simulando a interação com o usuário. A ferramenta se

encarrega de comparar os resultados do programa testado com os resultados esperados. Tal comparação pode verificar o conteúdo de mensagens de erro apresentadas em janelas de diálogo, conteúdo de arquivos ou de áreas gráficas, etc.

O uso efetivo das ferramentas para teste não prescinde do emprego de métodos viáveis e bem definidos que permitam criar casos de teste que cubram todos os objetos da interface gráfica de uma aplicação e todos os possíveis cenários de interação homem-computador. O grupo de interfaces gráficas tem um projeto em andamento cujo objetivo é investigar técnicas e desenvolver um método para teste de interfaces gráficas [villaça 1995].

O conceito de sistema computacional modificou-se substancialmente com o surgimento das interfaces gráficas. Sua definição e seu processo de desenvolvimento tornaram-se mais abrangentes, a partir da intensa proliferação e uso dos computadores pessoais. Hoj e, com os computadores constituindo-se em bens de consumo há um redirecionamento das indústrias de software para atender novos nichos de mercado, compostos por faixas mais amplas e diversificadas de usuários.

As preocupações atuais da indústria de software são diferentes daquelas de anos atrás. Basicamente, hoje há uma preocupação crescente tanto em relação à apresentação do produto quanto às necessidades dos usuários, que passam dessa maneira a ter um papel central no desenvolvimento de interfaces gráficas [norman 1986, laurel 1990, heckel 1991].

A abordagem para desenvolvimento centrada no usuário provocou uma ampliação do conceito tradicional de desenvolvimento de sistemas para além dos requisitos funcionais do aplicativo, incluindo, agora, fatores subjetivos relacionados com o conjunto de inter ações do usuário, essenciais para o design de interfaces com qualidade [winograd 1985]. Em linhas gerais, esta abordagem destaca a inadequação dos métodos tradicionais para desenvolvimento de sistemas com interfaces gráficas, explicitando uma questão acer ca da visão errônea atribuída ao design e ao desenvolvimento de interfaces [kay 1990, norman 1990, hix 1993].

"a postura de que o processo de design da interface com o usuário não é necessário, podendo ser realizado simplesmente durante a finalização total do desenvolvimento, é ingênua. Infelizmente essa atitude tem prevalecido e tem contribuído para a imagem de que as indústrias de software desenvolvem sistemas não usáveis." [browne, 1994, p. 20]

Vários estudos mostram o impacto financeiro e a importância das interfaces gráficas. Em [browne 1994], por exemplo, é apresentado um quadro quantitativo onde cerca de 55% do código total pertence aos elementos de interface, aproximadamente 30% do custo to tal de desenvolvimento do aplicativo está concentrado na interface e 60% do custo na fase de manutenção (responsável por 60% do custo total do aplicativo) está relacionado com alterações na interface.

O resultado deste e de outros estudos quantitativos oferecem argumentos objetivos sobre o papel e o lugar das interfaces gráficas no processo de desenvolvimento e justificam de forma indiscutível a necessidade de utilização de novos métodos de desenvolvimento, para diminuição do custo total do sistema e consequente aumento da produtividade da indústria de software. Diante da importância financeira e da complexidade do tema, muito tem sido discutido e vários métodos de desenvolvimento para aplicativos com interfaces gráficas foram propostos [nielsen 1993, hix 1993, browne 1994, carlshamre 1994].

Nos diversos métodos para desenvolvimento identificamos certas semelhanças, que formam uma base comum. A primeira semelhança está relacionada com a utilização de considerações originárias da abordagem centrada no usuário [norman 1986].

A segunda corresponde ao consenso de que uma boa interface deve ser transparente e permitir o acesso fácil à funcionalidade da aplicação. E a terceira está relacionada com a utilização do conceito

de usabilidade como parâmetro central para desenvolvimento do aplicativo e pa ra avaliações em relação à qualidade da interface. Mas, como elaborar e desenvolver uma interface?

Para tratar esta questão, segundo [nielsen 1993], é necessário diminuir o caráter subjetivo da avaliação de uma interface, que normalmente gira em torno da usabilidade do aplicativo. Com este objetivo, nielsen associa cinco atributos concretos e mensuráveis ao conceito de usabilidade, os quais auxiliam na quantificação de questões relacionadas às condições oferecidas aos usuários na utilização da funcionalidade do aplicativo. Os atributos de usabilidade são:

- facilidade de aprendizado

- Eficiência de uso

- Facilidade de memorização

- Baixa taxa de erros

- Satisfação subjetiva

Nielsen, a partir dessa associação, concentra-se no processo de desenvolvimento e estabelece o que denominou engenharia de usabilidade, onde são propostas atividades e heurísticas necessárias para a obtenção de um produto com alto grau de usabilidade.

Basicamente, na engenharia de usabilidade é apresentado um roteiro de onze fases que, independente do tipo de interface considerada, podem ser realizadas para o aumento da usabilidade. No roteiro, apresentado na tabela 1, há uma tentativa de se definir fa tores críticos para a usabilidade do aplicativo, procurando-se, através de um desenvolvimento interativo, balancear os cinco atributos de usabilidade para obter produtos de software com um grau de usabilidade satisfatório.

Nem sempre é possível realizar todas as fases do roteiro. Afinal, a pressão do tempo e dos custos são dependentes do projeto e da possibilidade econômica das empresas. Assim, nos parece necessário efetuar uma análise e possível redução do ciclo mostrado n a tabela 1 adaptando-o ao contexto brasileiro.

O uso de interfaces gráficas como meio de comunicação com programas de computador é mandatório nos produtos de software atuais; longe de um simples modismo, esse mecanismo permite que usuários de programas de computador encontrem uma facilidade muito gran de para desempenharem suas tarefas.

Dois fatores fundamentais para a implementação de uma boa interface gráfica são a tecnologia e a metodologia de trabalho empregadas.

A tecnologia existente no mercado para implementação de interfaces gráficas já se encontra bastante estável, como demonstra a experiência narrada neste trabalho no uso de ferramentas que tratam dos aspectos de portabilidade e de teste de aplicativos com interfaces gráficas.

Metodologias específicas para desenvolvimento de software com interfaces gráficas foram propostas recentemente e, embora importantes, ainda são pouco empregadas pela indústria de software. A importância dada aos futuros usuários dos produtos de software durante a fase de definição da interface surge como um importante ponto comum entre todas as metodologias que estão sendo propostas.

De forma semelhante ao que acontece em outras áreas, entre o surgimento de uma tecnologia e seu emprego industrial decorre um período de tempo ocasionado pelas dificuldades que as empresas encontram para dominar e incorporar a nova tecnologia. Neste ponto , a atuação de centros difusores de tecnologia é de extrema importância pois permite que empresas possam incorparar mais rapidamente novas tecnologias em seus processos produtivos.

O cti vem trabalhando no processo de absorção e transferência de tecnologia para a produção de software através do programa de qualidade e produtividade em software (pqps), um de seus três programas tecnológicos, procurando os mecanismos mais adequados à s necessidades dos interessados. O trabalho apresentado neste artigo é um dos realizados no âmbito do PQPS, e os resultados obtidos até aqui indicam o acerto do rumo adotado.