Ministério da Agricultura, Pecuária e Abastecimento

Comunicado Técnico 26

Dezembro, 2002
Campinas, SP

ISSN 1677-8464

Utilizando a Vermont para Automação de Testes Funcionais no SIGI

João Francisco Gonçalves Antunes[1]
Moacir Pedroso Júnior[2]
Marcos Cezar Visoli[3]
Flávia Gobet de Aguiar[4]

O desenvolvimento de software envolve uma série de atividades de produção nas quais a chance de ocorrência de falhas humanas é grande. Enganos podem acontecer tanto no início do processo de desenvolvimento, nas fases de análise e especificação como nas fases de projeto e implementação. Devido a esta característica, o desenvolvimento de software é acompanhado por atividades de testes que visam garantir a qualidade do software (Rocha et al., 2001).

Teste é o processo de executar um programa com a intenção de descobrir a presença de defeitos. Casos de testes são então projetados para descobrir sistematicamente diferentes categorias de defeitos em uma quantidade de tempo e esforço mínimos. O teste, em especial, é um elemento usado para fornecer evidências da confiabilidade do software em complemento a outras atividades, sendo relevante para identificação e eliminação de defeitos que persistem no software (Pressman, 1997).

Por isso, os casos de teste desenvolvidos e submetidos ao programa têm o objetivo de produzir uma saída que esteja em desacordo com a especificação. Quando isto ocorre, diz-se que o teste foi bem sucedido (Myers, 1979). Caso defeitos não sejam detectados, o testador tem aumentada a sua confiança quanto à qualidade do software. Porém, para que isto ocorra, é preciso que os testes tenham sido realizados de maneira rigorosa e sistemática.

Uma dessas técnicas é o teste funcional que preconiza o desenvolvimento de casos de teste com o objetivo de testar aspectos importantes da especificação do sistema. Neste trabalho é apresentada a experiência de desenvolvimento e automação de execução de casos de testes no SIGI - Sistema de Informação Gerencial do *Instituto Nacional de Investigaciones Agrícolas de Venezuela* – (Pedroso Júnior et al., 2001).

A arquitetura do SIGI pode ser chamada de cliente/servidor off-line, isto é, o cliente não opera em comunicação direta com o servidor. A parte cliente trata do planejamento, acompanhamento, reprogramação e avaliação final de projetos de pesquisa em uma base de dados local trazida do servidor. A criação de projetos, dentre outras funcionalidades, é feita diretamente no servidor através de uma aplicação Web. As máquinas clientes locais (executando sistema operacional Windows) trazem da base central projetos criados no estado inicial. Esses projetos, por sua vez, são alterados localmente e, ao final de cada etapa do ciclo de vida, reenviados ao servidor para atualização. Dessa maneira, os dados dos projetos na base local estão sempre sincronizados com a base central no servidor

[2] *Bsc. em Matemática Aplicada e Estatística, Pesquisador da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo 13083-970 – Campinas, SP. (e-mail: joaof@cnptia.embrapa.br)*

[2] *Ph.D. em Pesquisa Operacional, Pesquisador da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo – 13083-970 Campinas, SP. (e-mail: pedroso@cnptia.embrapa.br)*

[3] *Bsc. em Ciência da Computação, Pesquisador da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo – 13083-970 – Campinas, SP. (e-mail: visoli@cnptia.embrapa.br)*

[4] *Consultora da Embrapa Informática Agropecuária na área de testes do SIGI. (e-mail: flaviag@cnptia.embrapa.br)*

e disponíveis aos interessados com a devida autorização de acesso.

O processo de desenvolvimento de software adotado no SIGI é baseado no paradigma eXtreme Programming (Beck, 2000). Do ponto de vista de qualidade, neste paradigma, o teste sistemático é uma atividade essencial porque ocorre repetidamente a cada alteração no sistema. Isso envolve altos custos em relação a recursos humanos e tempo dispendido na atividade de teste que, mesmo com o auxílio de ferramentas, já não são pequenos. Para facilitar esse processo, foi utilizada no desenvolvimento do SIGI a ferramenta Vermont (Vermont Creative Software, 2001) que automatiza a execução dos testes disparando scripts de eventos, isto é, grava e reproduz eventos de seleção de menus, movimento de telas, cliques de botões, movimento do mouse e também captura e compara imagens de janelas. Com isso, é possível executar automaticamente *scripts* que testam todo o ciclo de vida de um projeto de pesquisa.

O trabalho a seguir mostra como criar e executar testes funcionais utilizando a ferramenta de automação de testes Vermont.

Técnicas de Teste

As técnicas de teste dividem-se em estrutural (caixa branca) e funcional (caixa preta). Um caso de teste, por sua vez, é composto por um conjunto de entradas, condições de execução e os resultados esperados, tendo como objetivo verificar os requisitos especificados do sistema. A metodologia para desenvolvimento de casos de teste funcionais no SIGI é descrita em Antunes et al., (2001).

Os testes estruturais focalizam a implementação do programa (código, funções, procedimentos e fluxo de dados e controle). Os casos de testes podem ser elaborados, por exemplo, para garantir que a maior abrangência possível das instruções do programa tenham sido exercitadas pelo menos uma vez.

Os testes funcionais estabelecem os requisitos de teste com base na especificação do software, não levando em consideração o funcionamento interno do programa. Os casos de teste derivados com o teste funcional visam detectar defeitos dos seguintes tipos: funções incorretas, defeitos de interface, defeitos em estrutura de dados, defeitos de acesso às bases de dados, problemas de desempenho e defeitos de inicialização e de finalização. Esse processo não necessita dos códigos fonte do programa, apenas de seu executável.

Os testes de regressão têm o papel de verificador das atualizações de uma versão para a outra do software, servindo para validar uma correção ou verificar um possível problema por conseqüência de uma alteração. Nesse caso, os casos de teste (estruturais ou funcionais) devem se repetir para que a nova versão seja adequadamente testada e para garantir que as partes do software que permaneceram inalteradas continuem funcionando dentro das especificações (Crespo et al., 2001).

No SIGI, os casos de teste derivados com a técnica funcional são executados automaticamente com auxílio da ferramenta Vermont. A cada versão liberada pela equipe de desenvolvimento, os *scripts* de eventos são executados para validação das funcionalidades e para a execução dos testes de regressão que devem ser repetidos após as modificações no sistema.

Os defeitos encontrados a partir da execução automática dos teste funcionais são verificados pelos testadores e cadastrados no Bugzilla (Mozilla Organization, 2002). A partir disso, os desenvolvedores procedem a correção dos defeitos e uma nova versão do sofware é liberada para equipe de testes. Os defeitos encontrados na versão são certificados e os scripts de eventos novamente executados. Se os defeitos realmente forem corrigidos, o defeito cadastrado no Bugzilla é fechado, senão, é reaberto. Assim continua o ciclo sempre com os testes de regressão assegurando que as funcionalidades de uma versão para outra permanecem corretas.

Ferramenta para Automação de Testes

A ferramenta utilizada para automação dos casos de teste do SIGI é a Vermont High Test Plus 3.50, da Vermont Creative Software (2002) que auxilia na execução de testes de aplicações Windows e simplifica tarefas repetitivas. É uma ferramenta da classe captura e reproduz (*capture and playback*), isto é, grava os eventos gerados pelo testador através dos dispositivos de entrada e saída e gera um script que pode ser reproduzido inúmeras vezes.

A Vermont oferece as seguintes funcionalidades:

- gravação orientada a objetos e reprodução dos eventos especificados, tais como seleção de menus, movimento de telas, cliques de botões e movimento do mouse;
- captura e compara imagens, funções internas e controles do Windows e conteúdo de arquivos;
- mascara áreas de imagens ou arquivos que se deseja excluir da comparação;
- construção e adaptação dos *scripts* de testes através de uma linguagem própria de programação;
- criação de uma coleção organizada de scripts de teste através de um gerenciador de conjunto de *scripts*.

A Vermont foi escolhida para automação dos testes do SIGI porque é uma ferramenta de fácil uso e de baixo custo (U$195,00). Além disso, não é necessário ser um programador experiente para usá-la efe-

tivamente. A versão atual pode ser executada no Windows 9x, NT, Me e 2000. O suporte pode ser obtido através da *home page* do fabricante (Vermont Creative Software, 2002) onde existe um fórum de discussão, uma base de conhecimento sobre anotações técnicas e uma área de atualizações da ferramenta.

Diretrizes para Utilização da Vermont

A utilização da Vermont é dividida basicamente em duas fases:

- criação dos *scripts*: primeiramente deve-se indicar o nome do programa executável que está sendo testado. Depois disso, o testador cria os *scripts* de eventos através da execução do sistema a partir da Vermont, determinando objetos de interface (menus, janelas, botões, etc.) que devem ser comparados e o caminho a ser seguido através das telas, baseado no caso de teste;
- reprodução dos *scripts*: para repetir o teste, o *script* criado anteriormente deve ser reproduzido. Os dados de entrada durante a gravação servem como referência na fase de reprodução. Nesta fase, os objetos de interface capturados são comparados com os do software ativo e se houver alguma diferença na interface, com relação a posição, fontes, cores, etc., a execução da Vermont é interrompida imediatamente mostrando a linha do *script* onde ocorreu o problema.

Configuração da Vermont

A configuração inicial da Vermont é feita através do menu *Options*, *Playback,* conforme mostrado na Fig. 1. É possível selecionar a velocidade de execução do *script*, as funções que param a execução do *script* quando defeitos são detectados e parâmetros de comparação. Assim, é possível controlar precisamente a execução dos scripts de eventos.

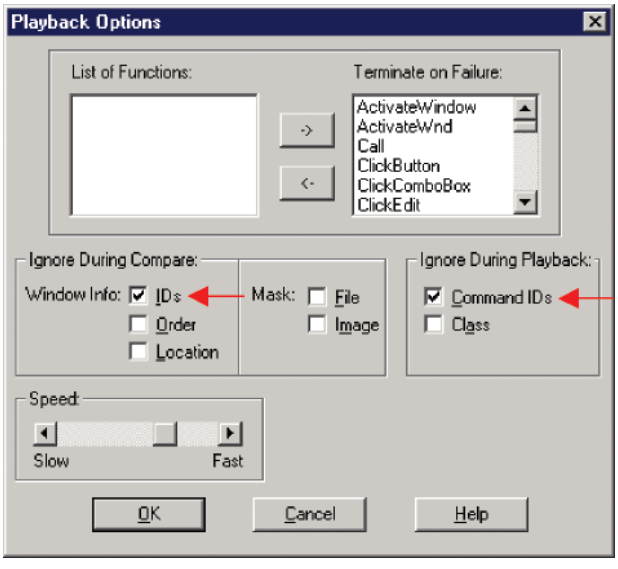

Fig. 1. Opções de configuração da Vermont.

Como a Vermont funciona com a captura de eventos, cada janela capturada tem um identificador. Quando um *script* que utiliza o comando *WaitWindow* é reproduzido, os identificadores das janelas novamente gerados são diferentes dos identificadores das janelas já gravadas. Isso acarreta um problema de conflito na identificação da janela referente a tela que está sendo testada. Para evitar esse problema é necessário ignorar os identificadores durante a comparação, desabilitando a opção *Commands IDs*.

A Vermont não suporta os caracteres acentuados ´, ‘, ^, ç, ~ e os caracteres /, ?, º, ª. Os caracteres “, [ ], (), ; são reservados da linguagem de programação e como os anteriores não podem ser utilizados na elaboração dos scripts.

Procedimento para Criação de um *Script*

Este item apresenta as orientações passo-a-passo para criação de um *script* com a Vermont.

O menu *Record* da Vermont inicializa efetivamente a gravação de um *script*. Um *script* novo é criado através da opção *New w/ StartApp* como mostrado na Fig. 2. A opção New também pode ser utilizada, porém a aplicação que será testada deverá estar aberta em segundo plano.

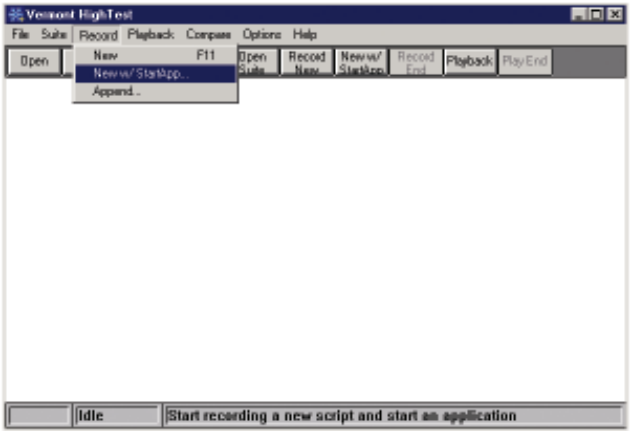

Fig. 2. Início da criação de um *script*.

O *script* deve ser salvo em disco antes de iniciar a gravação dos eventos (Fig. 3).

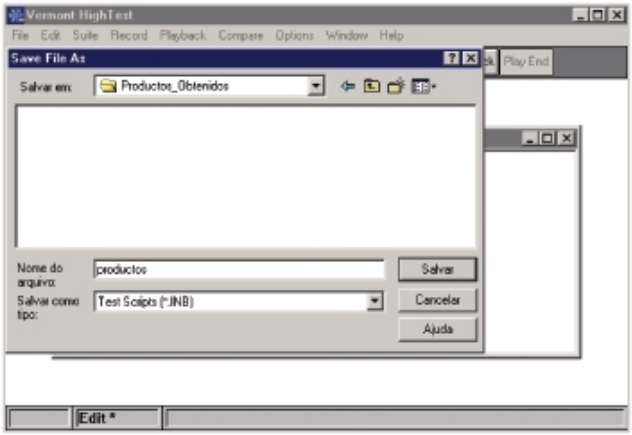

Fig. 3. Salvar o *script* antes de iniciar a gravação.

Nesse momento aparecerá uma janela onde deverá ser especificado o nome com o caminho completo da aplicação, no caso SigiCliente.exe, a ser testada, como mostrado na Fig. 4.

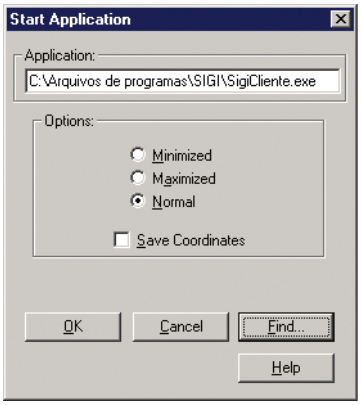

Fig. 4. Nome completo da aplicação a ser testada.

A captura dos eventos de gravação deverá ser feita com auxílio da barra de ferramentas que é mostrada na Fig. 5.

Fig. 5. Barra de ferramentas de gravação da Vermont.

Descrição dos comandos da barra de ferramentas de gravação (*Record bar*) utilizados para a criação de *scripts*:

Full	***Screen button***: captura a tela inteira para comparação, incluindo a barra de tarefas do Windows.
	Application button: captura a tela inteira da aplicação para comparação, sem incluir a barra de tarefas do Windows.
	Capture Window button: captura uma tela da aplicação para comparação
	Capture Region button: captura uma região especifica da tela para comparação.
	Capture Window Info button: captura uma tela da aplicação para comparação com detalhes dos componentes tais como identificador, estilo, posição, etc.
	Capture File button: captura o conteúdo de um arquivo para comparação.
+S	***Comment button***: insere comentários no script.
+S	***Script button***: adiciona o script a uma suíte. Suíte é um conjunto de scripts que devem ser executados numa ordem seqüencial.
+D	***Add Delay button***: insere um comando de tempo *script*.
+P	***Comando Add Pause button***: insere uma pausa no *script playback*, retornando somente com o auxilio do usuário.
	Add WaitWindow button: captura uma tela da aplicação para comparação especificando um tempo em segundos.
	Suspend Recording button: pára o processo de gravação de um script, retornando para a janela principal da Vermont.

A Fig. 6 mostra o SIGI cliente integrado à Vermont pronta para iniciar o processo de gravação dos eventos com a barra de ferramentas de gravação.

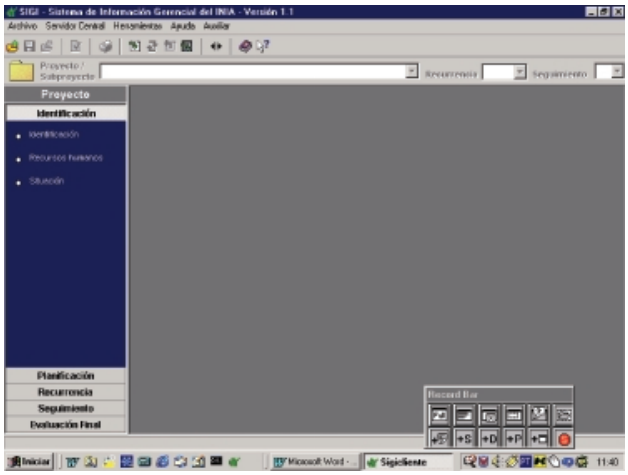

Fig. 6. SIGI cliente iniciado pela Vermont.

Para capturar todo o SIGI cliente é utilzada a opção *Add WaitWindow* conforme mostrado na Fig. 7. Esta opção suspende o processo de gravação até que a janela que foi capturada esteja ativa.

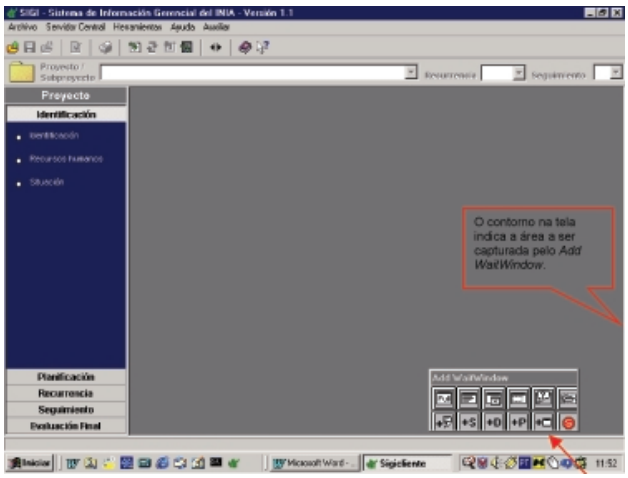

Fig. 7. Captura da aplicação com o comando *WaitWindow*.

Neste ponto o projeto do SIGI que encontra-se no disco local será aberto, conforme mostrado na Fig. 8. Como é uma tela de diálogo com o usuário, deve ser usado o comando *Add WaitWindow* porque a tela leva um determinado tempo até aparecer.

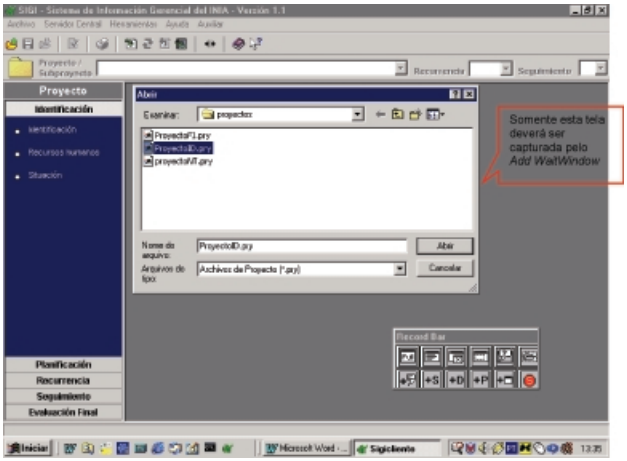

Fig. 8. Tela de diálogo capturada com o comando *Add WaitWindow.*

Assim que o projeto do SIGI for aberto é necessário capturar a tela inteira com o comando *Add WaitWindow*, para que a execução do *script* só continue após o aparecimento total da tela, conforme mostrado na Fig. 9.

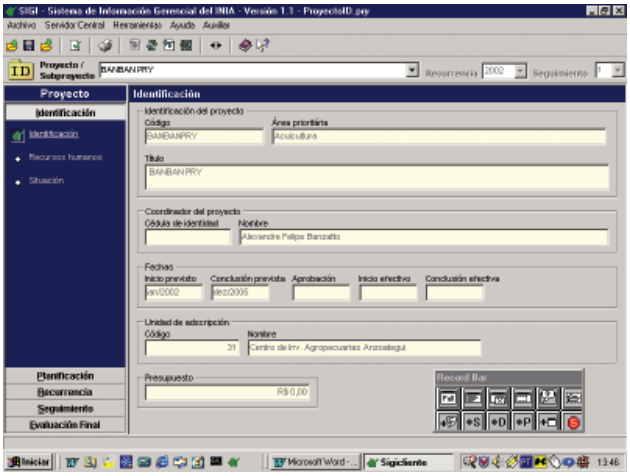

Fig. 9. A tela do projeto aberta captura com o comando *Add WaitWindow*.

O próximo passo é capturar uma tela para comparação através do comando *Capture Window*, como mostrado na Fig. 10. A tela especificada servirá de referência quando o *script* for executado novamente numa nova versão do sistema. Assim, é possível detectar qualquer alteração na interface.

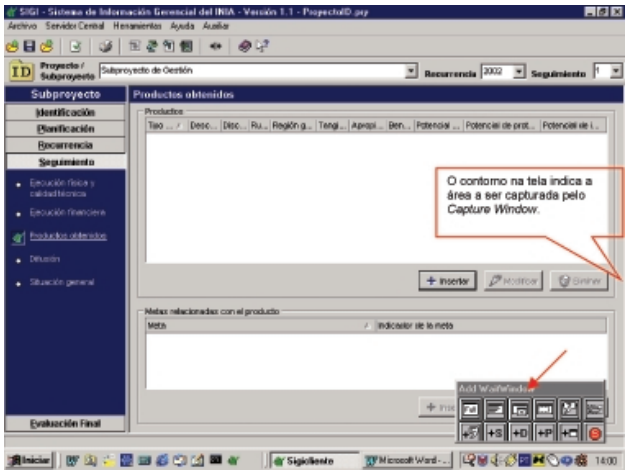

Fig. 10. Tela captura com o comando *Capture Window.*

Para parar o processo de gravação é utilizado o comando *Suspend Recording,* conforme mostrado na Fig.11.

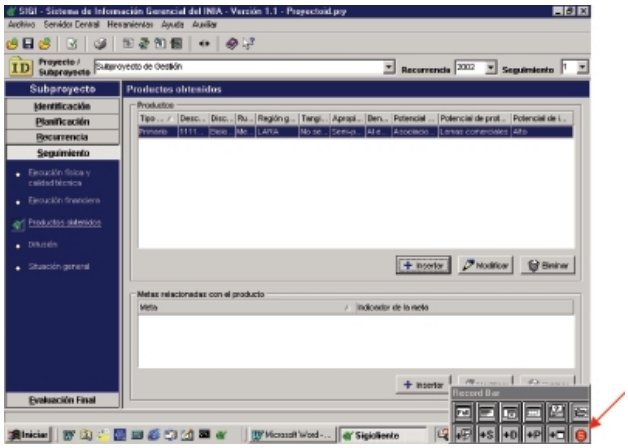

Fig. 11. Finalizando o processo de gravação.

Os testes de comparação de telas do sistema podem ser feitos baseando-se nos passos que foram apresentados até agora, lembrando que o primeiro comando a ser utilizado é o *Add WaitWindow* e para cada tela a ser comparada utiliza-se o comando *CaptureWindow*.

Ao retornar para a janela principal da Vermont é exibido o código do script que foi gerado automaticamente através dos eventos gravados e das telas capturadas durante a execução do sistema, conforme pode ser visto na Fig. 12.

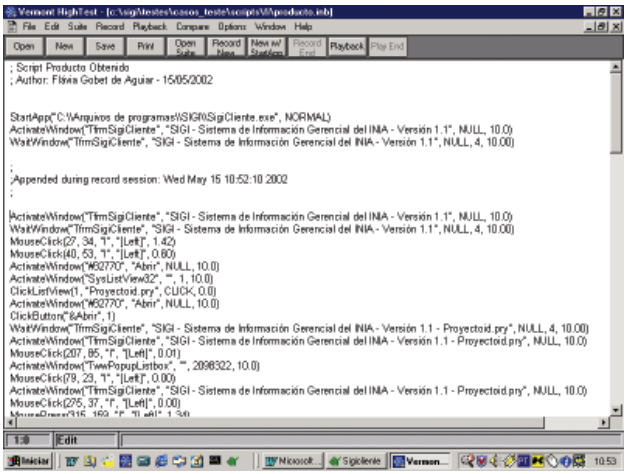

Fig.12. *Script* de eventos gerado automaticamente.

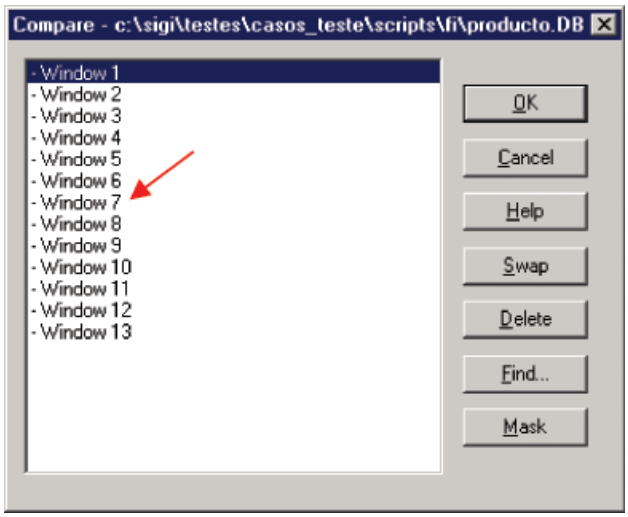

Fig. 14. Lista de telas usadas na comparação.

Quando a reprodução do *script* for executada, a Vermont faz a comparação da tela atual com a tela de referência capturada durante o processo de gravação do *script*. Em alguns casos não é necessário comparar o conteúdo de campos, isto é, é possível mascarar o que não é necessário comparar.

Para fazer isso, no menu *Compare* do *script* que foi gerado anteriormente, selecione a opção *Screenshot*, como mostrado na Fig. 13.

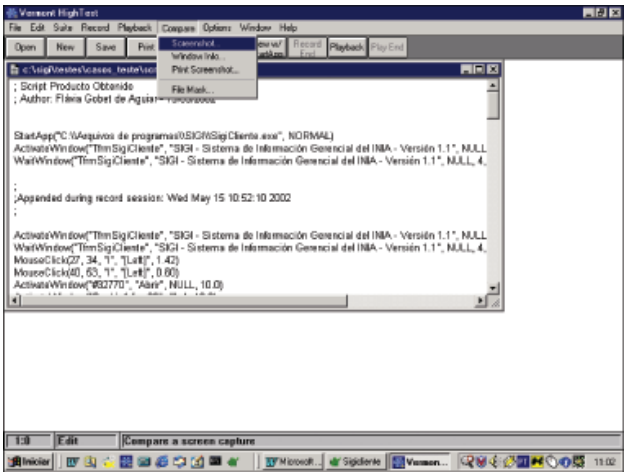

Fig. 13. Opção *Screenshot* do menu *Compare*.

Será exibida uma janela com a identificação de todas as telas que foram capturadas durante o processo de gravação para comparação, como mostrado na Fig. 14. Selecione a tela que se deseja mascarar:

A tela que foi capturada será exibida acompanhada da barra de ferramentas de comparação (*Compare bar),* como mostrado na Fig. 15.

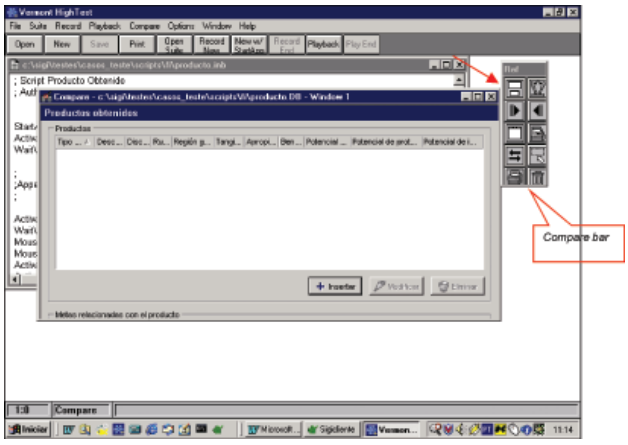

Fig. 15. Tela capturada e a barra de ferramentas de comparação da Vermont.

Descrição dos comandos da barra de comparação (*Compare bar*) utilizados para a criação de *scripts*:

	*Show Differences button*: destaca todos os pixels diferentes entre a tela referência e a do *playback*.
	*Toggle between Reference and Playback button*: alterna entre a tela de referência e a do *playback*.
	*Get the Next Image button*: avança para a tela seguinte armazenada para comparação.
	*Get the Previous Image button*: volta para a tela anterior armazenada para comparação.
	*Toggle Full Screen button*: intercala entre a tela de comparação cheia e a padrão.

	Get a New Image button: abre uma tela de comparação
	Replace the Reference Image button: sobrepõe a tela de referência com a atual do *playback*.
	Start or End Masking button: permite definir máscaras para regiões diferentes de uma tela.
	Print Image button: imprime a tela.
	Delete Image from the Database button: exclui a imagem da base de dados da Vermont.

A Fig. 16 mostra a região onde foi colocada uma máscara. Quando o *script* for novamente executado e caso já exista um registro inserido, essa região será ignorada pela Vermont.

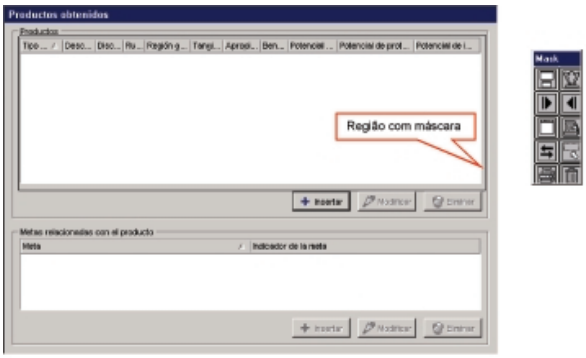

Fig. 16. Região onde foi colocada uma máscara.

Um outro comando bastante utilizado na Vermont durante a gravação de *scripts* é o *Append*. Com ele é possível adicionar eventos novos num *script* já existente devido a incorporação de novas funcionalidades na tela que deve ser testada, como mostrado na Fig. 17.

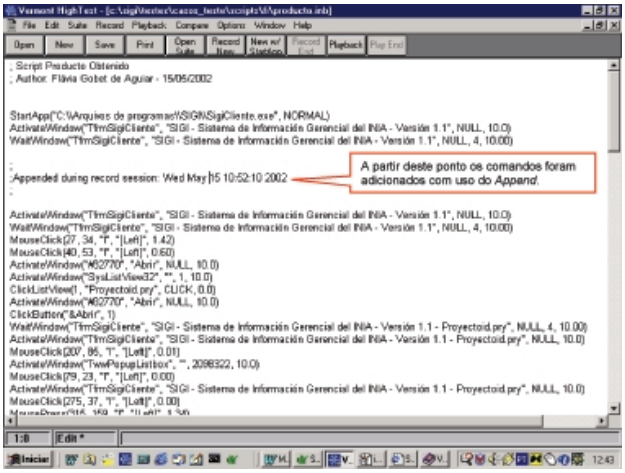

Fig. 17. Uso do comando *Append*.

O procedimento para a utilização do *Append* é análogo ao utilizado para a criação de *scripts* novos como mostrado a partir da Fig. 2. A diferença fundamental é que a opção a ser utilizada deve ser *Append* no menu *Record* da Vermont abrindo um *script* que já foi gravado anteriormente. A partir disso, a utilização dos comandos é a mesma, como foi explicado passo-a-passo.

Procedimento para Execução da Vermont via Linha de Comando

Num ambiente de teste automatizado, após os *scripts* estarem prontos para testes funcionais de uma determinada versão do sistema, é necessário executar a ferramenta de testes via linha de comando, sem a interferência do testador. A Vermont oferece esse recurso, mas é necessário configurar o ambiente do microcomputador para isso.

Primeiro deve-se configurar o *path* do sistema operacional, incluindo o diretório do executável da Vermont. O diretório padrão é C:\Arquivos de programas\Vermont HighTest Plus.

A execução da Vermont via linha de comando é feita através do seguinte comando:

Hightest [nome do *script* | "nome da suíte"] [options], onde suíte é um conjunto de *scripts* que devem ser executados numa ordem lógica.

As opções usadas são as seguintes:

/ t: finaliza a Vermont após a execução do script.

/ s nnn: determina a velocidade onde o nnn é um valor entre 1 e 100.

Exemplo:

Hightest C:\sigi\testes\casos_teste\scripts\FI\Hacer_Planificacion_Subproyecto\suite.inb /t /s 060

Com isso é possível criar um arquivo em lote fazendo a chamada dos *scripts* ou suítes por ordem seqüencial, testando as funcionalidades do sistema conforme necessário.

Os defeitos encontrados a partir da execução automática dos testes funcionais são verificados pelos testadores e cadastrados no Bugzilla para dar ciência aos desenvolvedores.

A partir disso, os desenvolvedores procedem a correção dos defeitos e uma nova versão do sofware é liberada para equipe de testes. Os defeitos encontrados na versão são certificados e testados novamente através da execução de todos os *scripts* de testes funcionais com uma nova versão do software. Esses são os chamados testes de regressão que garantem que as funcionalidades de uma versão para outra permanecem corretas.

Conclusões e Comentários

A atividade de teste é complexa porque software é complexo, intangível e altamente modificável. É uma atividade que exige planejamento, projeto, execução, acompanhamento, integração com outras áreas e recursos como equipe, processo, treinamento e ferramentas adequadas. Por isso, é responsável pela maior porcentagem de esforço técnico no processo de desenvolvimento do software.

No SIGI, a execução automática dos casos de teste foi realizada com auxílio da ferramenta Vermont. Apesar das suas particularidades, a Vermont foi de grande utilidade para realização dos testes da parte cliente automatizando todo o teste do ciclo de vida de um projeto de pesquisa. Através dos recursos fornecidos pela Vermont é possível popular bases de dados sem ter que digitar todo o conteúdo dos campos, testar o tamanho limite de campos, comparar componentes de interface e executar testes de regressão que devem ser repetidos após as modificações no sistema.

A utilização da Vermont não precisa aguardar que todo o desenvolvimento do software esteja finalizado. Para cada módulo liberado em que existem funcionalidades a serem testadas, é possível elaborar um *script* e executá-lo. Hoje, ao final do desenvolvimento do SIGI, é possível executar automaticamente *scripts* que testam todo o ciclo de vida de um projeto de pesquisa, cobrindo a fase de planejamento, acompanhamento, reprogramação e avaliação final.

Uma deficiência da Vermont é o não retorno de um código de erro na execução de um *script*. Quando um *script* quebra é retornada uma mensagem visual, informando qual o comando que falhou. Por esse motivo é necessário que o testador supervisione a execução do *script* periodicamente para que a execução do teste não fique parada por muito tempo.

Referências Bibliográficas

ANTUNES, J. F. G.; PEDROSO JÚNIOR, M.; VISOLI, M. C.; AGUIAR, F. G. de; TOZZI, M. G.; PALMA ABEDRAPO, M. P. **Desenvolvimento e execução automática de testes funcionais do SIGI.** Campinas: Embrapa Informática Agropecuária, 2001. 8 p. (Embrapa Informática Agropecuária. Comunicado Técnico, 29).

BECK, K. **Extreme programming explained:** embrance change. Boston: Addison-Wesley, 2000. 224 p.

CRESPO, A. N.; FANTINATO, M.; MARTINEZ, M. R. M.; JINO, M.; ARGOLLO JÚNIOR, M. de T. e. **Teste de software.** [Campinas: Instituto Nacional de Tecnologia da Informação, 2001]. 58 transparências.

MOZILLA ORGANIZATION. **Bugzilla - mozilla.org - the Mozilla bug database.** Disponível em: <http://bugzilla.mozilla.org>. Acesso em: 10 dez. 2002.

MYERS, G. J. **The art of software testing.** New York: John Wiley, 1979. 192 p. (Business Data Processing, a Wiley Series).

PEDROSO JÚNIOR, M.; ANTUNES, J. F. G.; VISOLI, M. C.; CASTRO, A. M. G.; LIMA, S. M. V. **Sistema de Informação Gerencial de Projetos de Pesquisa Agropecuária para o Instituto Nacional de Investigaciones Agrícolas da Venezuela.** Campinas: Embrapa Informática Agropecuária, 2001. 14 p. (Embrapa. Programa 12 – Automação Agropecuária. Projeto 12.2001.950). Projeto em andamento.

PRESSMAN, R. S. **Software engineering:** a practitioner's approach. 4th ed. New York: McGraw-Hill, 1997. 852 p.

ROCHA, A. R. C.; MALDONADO, J. C.; WEBER, K. C. (Ed.). **Qualidade de software:** teoria e prática. São Paulo: Prentice-Hall, 2001. 303 p.

VERMONT CREATIVE SOFTWARE. **Vermont high test plus:** version 3.50 for Windows – the comprehensive testing tool that´s easy to use. Disponível em: <http://www.vtsoft.com>. Acesso em: 10 dez. 2001.

Comunicado Técnico, 26

Embrapa Informática Agropecuária
Área de Comunicação e Negócios (ACN)
Av. André Tosello, 209
Cidade Universitária - "Zeferino Vaz"
Barão Geraldo - Caixa Postal 6041
13083-970 - Campinas, SP
Telefone (19) 3789-5743 - Fax (19) 3289-9594
e-mail: sac@cnptia.embrapa.br

1ª edição
2002 - on-line