Ministério da Agricultura, Pecuária e Abastecimento

Comunicado Técnico 25

Dezembro, 2002
Campinas, SP

ISSN 1677-8464

Cobertura de Testes Unitários no SIGI

João Francisco Gonçalves Antunes[1]
Moacir Pedroso Júnior[2]
Marcos Cezar Visoli[3]
Marcos Geraldo Tozzi[4]

O desenvolvimento de software envolve uma série de atividades de produção em que a oportunidade de ocorrer falhas humanas são grandes. Enganos podem acontecer tanto no início do processo de desenvolvimento, nas fases de análise e especificação, como no decorrer dele, 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 testes 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 sistemática.

Umas das técnicas de teste sistemática é a estrutural que preconiza o desenvolvimento de casos de teste com o objetivo de testar a implementação do programa. Embora geralmente usado durante a fase de codificação, testes estruturais podem ser usados em todas as fases do ciclo de vida do software nas quais o software é representado formalmente. A intenção desse tipo de teste é encontrar dados de teste que terão cobertura suficiente de todas as estruturas presentes na representação formal.

Neste trabalho é apresentada a experiência de como medir a cobertura de testes unitários do SIGI - Sistema de Informação Gerencial do *Instituto Nacional de Investigaciones Agrícolas de Venezuela* – (Pedroso Júnior et al., 2001).

[1] *Bsc. em Matemática Aplicada e Estatística, Pesquisador da Embrapa Informática Agropecuária Caixa Postal 6041, Barão Geraldo – 13038-970 – Campinas, SP. (e-mail: joaof@cnptia.embrapa.br)*

[2] *Ph.D. em Pesquisa Operacional, Pesquisador da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo –13038-970 Campinas, SP. (e-mail: pedroso@cnptia.embrapa.br)*

[3] *Bsc. em Ciência da Computação, Pesquisador da Embrapa Informática, Caixa Postal 6041, Barão Geraldo – 13038-970 – Campinas, SP. (e-mail: visoli@cnptia.embrapa.br)*

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

Embrapa

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 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 compilação de todo o código fonte do sistema. Para a elaboração e execução automática dos casos de testes unitários do SIGI, mais precisamente da parte cliente, foi utilizada a ferramenta gratuita DUnit (SourceForge, 2002).

O trabalho a seguir mostra como medir a cobertura dos testes unitários no SIGI utilizando a ferramenta Discover (Cyamon Software, 2002).

Técnicas de Teste

As técnicas de teste dividem-se em dois níveis: teste funcional (caixa preta) e teste estrutural (caixa branca).

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 testes 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 iniciação e de finalização. Esse processo não necessita dos códigos fonte do programa, apenas de seu executável.

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

No SIGI, os casos de testes unitários foram desenvolvidos com Delphi 5. Delphi é um ambiente de desenvolvimento de aplicações para Windows que utiliza a linguagem Pascal orientado a objeto. Um programa Delphi é dividido em módulos chamados *Units* compostos por classes que definem a estrutura dos campos, métodos e propriedades. As instâncias de um tipo de classe são chamadas objetos. Cada *Unit* é armazenada num arquivo próprio onde são implementadas as funções e procedimentos que depois será compilada separadamente.

Os testes unitários foram automatizados com a ferramenta DUnit que possibilita criar programas que testam a implementação do sistema. Geralmente, os testes unitários são feitos pelo próprio desenvolvedor do módulo testado.

Ferramenta para Cobertura de Testes Unitários

A Discover é uma ferramenta de análise que permite aos desenvolvedores Delphi medir como os seus programas estão sendo testados. Mostra de forma interativa e direta, dentro do código origem quais os blocos básicos de código que são executados pelo menos uma vez e quais nunca foram executados. Baseado nisso, pode-se preparar ações ou elaborar mais testes unitários específicos para eliminar as áreas descobertas.

A Discover foi escolhida para medir a cobertura dos testes unitários do SIGI porque é uma ferramenta de fácil uso e de baixo custo (U$ 30,00). A versão atual trabalha com Delphi 3 a 7.

Diretrizes para Utilização da Discover

Este item mostra passo-a-passo como integrar o Delphi à Discover e como fazer a medição da cobertura dos testes unitários que foram elaborados com a DUnit.

Para a Discover poder fazer a cobertura de código de um programa, primeiro o *Map file* do Delphi deve ser gerado durante a compilação do teste unitário. Esse arquivo armazena uma lista de segmentos, endereços de inicialização do programa, avisos ou erros produzidos durante a ligação do código executável e uma lista de símbolos públicos ordenada alfabeticamente, incluindo o endereço, tamanho em bytes, nome, grupo do segmento e a informação do módulo do programa.

Para isso, é necessário configurar algumas opções a partir do menu *Project, Options...*, no projeto .dpr no Delphi correspondente ao código fonte do teste unitário.

Na pasta *Linker*, marcar a opção *Detailed*, conforme mostrado na Fig.1.

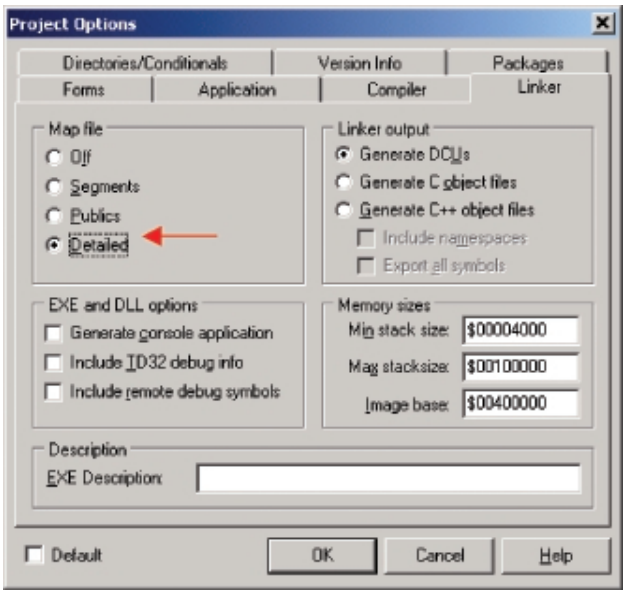

Fig. 1. Pasta *Linker* do Delphi com a opção *Detailed* marcada.

Na pasta *Directories/Conditionals*, no campo *Search path*, especificar o diretório onde se localizam os códigos fonte do programa a ser testado para possibilitar a integração com a Discover, como mostrado na Fig. 2.

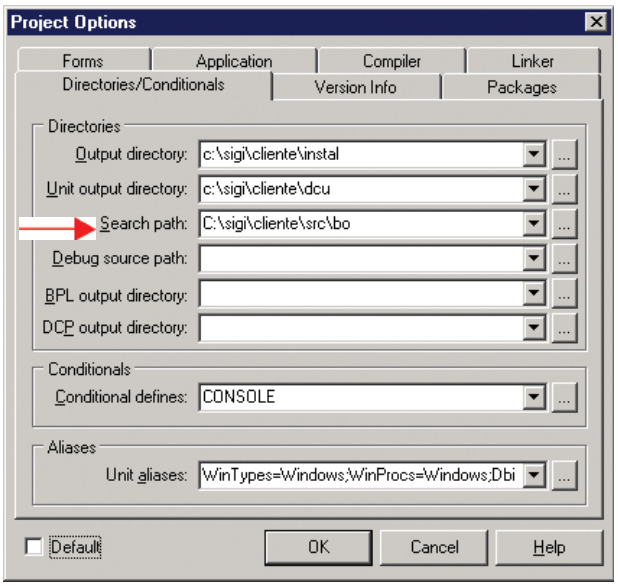

Fig. 2. Configuração do *Search path* no Delphi.

Após isso, o projeto de teste unitário deve ser compilado no Delphi.

Na Discover, o teste unitário compilado anteriormente deve ser carregado através da opção *Load*, no menu *Project*. Em seguida deve ser executado através da opção *Application, Run*. Se o teste tiver processos que dependam da interação do usuário é necessário navegar pela interface.

A Discover lê o *Map file* gerado para o projeto de teste unitário bem como o código fonte do programa a ser testado que localiza-se no *Search path* e, após a finalização do programa, gera as estatísticas de cobertura do código na pasta *Summary*, conforme mostrado na Fig. 3.

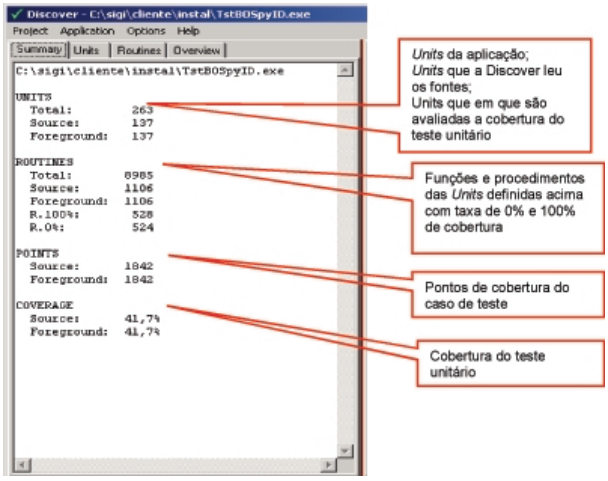

Fig. 3. Estatística de cobertura do caso de teste.

Na pasta *Units*, Fig. 4, são exibidas as *Units* do Delphi que compõem a aplicação e as suas respectivas coberturas, isto é, por onde o teste unitário passou, baseado na seguinte legenda:

Unit avaliada na cobertura (*foreground*), diretamente relacionada ao caso de teste unitário.

'*Unit* não avaliada na cobertura (*background*), isto é, que não entrou no caso de teste unitário.

Unit que a Discover não leu o código fonte, por exemplo, componentes visuais.

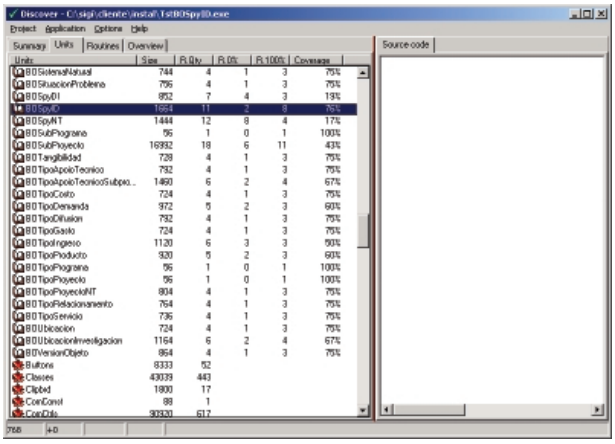

Fig. 4. *Units* Delphi com as respectivas coberturas.

Para cada *Unit*, a Discover mostra o tamanho, o número total de rotinas, o número de rotinas com 0% de cobertura, o número de rotinas com 100% de cobertura e a porcentagem de cobertura da *Unit*.

A pasta *Routines*, Fig. 5, exibe todas as funções e procedimentos da aplicação com seus respectivos métodos e propriedades, o código fonte e os pontos de cobertura do teste unitário.

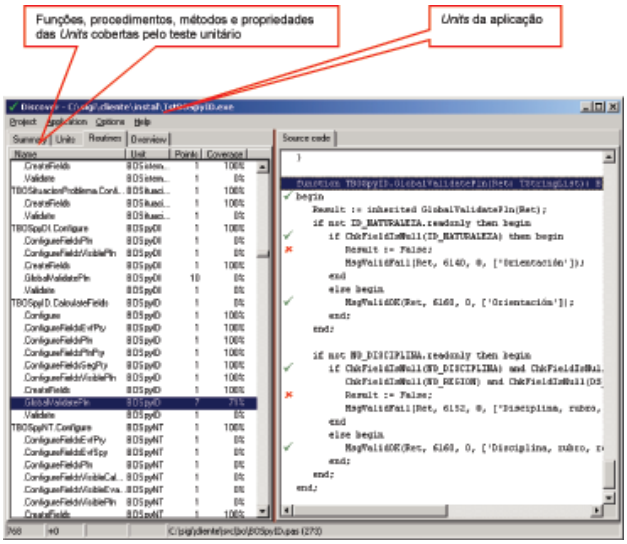

Fig. 5. Código fonte da aplicação e os pontos de cobertura.

Para cada rotina, ou seja, as funções e procedimentos implementados, a Discover mostra a correspondente *Unit*, o número de pontos e a porcentagem de cobertura. O ponto de cobertura é a posição na rotina que representa a parte do código que sempre é executada seqüencialmente pelo teste unitário, marcado com um sinal de checagem no código fonte. Pontos não cobertos, ou seja, não testados, são marcados com um X.

Uma outra funcionalidade da Discover é exportar os dados de cobertura que são mostrados nas páginas *Units* e *Routines* para um arquivo texto. A configuração desse arquivo é apresentada na Fig. 6.

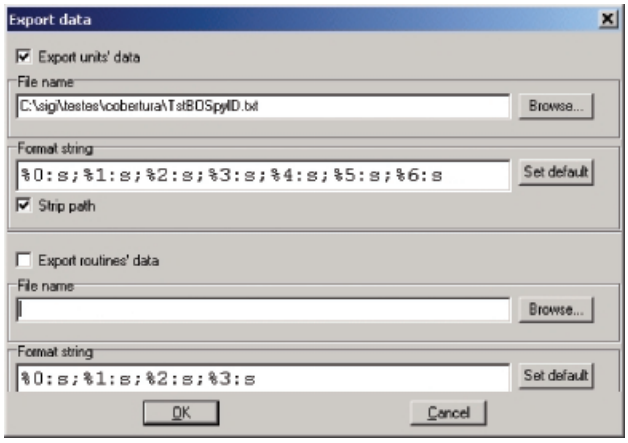

Fig. 6. Exportação dos dados de cobertura para um arquivo texto.

O formato padrão utilizado é %0:s;%1:s;%2:s;%3:s;%4:s;%5:s;%6:s, onde:

- s0: nome do arquivo da Unit.
- s1: nome da Unit.
- s2: tamanho da Unit.
- s3: total de rotinas da Unit
- s4: rotinas da Unit com 0% de cobertura.
- s5: rotinas da Unit com 100% de cobertura.
- s6: cobertura da Unit.

Cobertura dos Testes Unitários do SIGI

Esta tabela mostra a cobertura de alguns testes unitários do SIGI cliente medida em maio de 2002.

Nome do Projeto DUnit (dpr)	Nome da Unit testada (.pas)	Cobertura (%)	Autor
TstBOPdoSegNT	BOPdoSegNT	69	Gustavo
TstBOPlanEstrategico	BOPlanEstrategico	100	Cantu
TstBOPotImp	BOPotImp	100	Gustavo
TstBOPotPmt	BOPotPmt	100	Gustavo
TstBOPotVta	BOPotVta	100	Gustavo
TstBOPresupuestoPartida	BOPresupuestoPartida	91	Banzatto
TstBOProblemaSeguimiento	BOProblemaSeguimiento	100	Gustavo
TstBOProblemaMetaSeguimiento	BOProblemaMetaSeguimiento	100	Gustavo
TstBOProductoMetaSeguimientoPry	ProductoMetaSeguimientoPry	87	Gustavo
TstBOProductoMetaSeguimientoSpy	BOProductoMetaSeguimientoSpy	89	Gustavo
TstBOProductoPlanificacion	BOProductoPlanificacion	78	Gustavo
TstBOProductoSeguimiento	BOProductoSeguimiento	81	Cantu
TstBOProductoSeguimientoPry	BOProductoSeguimientoPry	28	Gustavo
TstBOProductoSeguimientoSpy	BOProductoSeguimientoSpy	42	Gustavo
TstBOPrograma	BOPrograma	100	Gustavo
TstBoProyecto	BOProyecto	60	Cantu
TstBoPryDl	BOPryDl	67	Gustavo

Conclusões e Comentários

A atividade de teste é complexa porque software é complexo 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.

No SIGI, a execução automática dos casos de teste unitários foi realizada com auxílio da ferramenta DUnit. Cada desenvolvedor, após implementar os testes unitários de sua responsabilidade, executa a Discover para saber quanto cada teste está cobrindo. Usando essas informações pode-se tomar ações para elaborar testes mais específicos para eliminar as áreas descobertas até se conseguir uma porcentagem de cobertura aceitável. O ideal seria sempre 100%, mas às vezes isso não é possível.

A cobertura dos testes unitários foi medida freqüentemente para avaliar a consistência dos testes. Porém, não pode ser feita de forma centralizada na máquina que faz a compilação e execução contínua de todos os casos de testes unitários porque não foi possível compilar os programas de testes integrados a Discover, via linha-de-comando, devido a configuração do *Search path* no arquivo de configuração do Delphi (.cfg).

Para o futuro fica a tarefa de automatizar todo o processo de medição da cobertura de testes unitários já que a DUnit e a Discover podem ser executadas via linha-de-comando, gerando o arquivo texto de saída com os dados de cobertura.

Referências Bibliográficas

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

CYAMON SOFTWARE. *Discover*: version 1.50. Disponível em: <http://www.cyamon.com>. Acesso em: 11 nov. 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.

SOURCEFORGE. *SourceForge.net*: Project Info - DUnit: Xtreme Unit Testing for Delphi. Disponível em: <https://sourceforge.net/projects/dunit>. Acesso em: 11 nov. 2002.

Comunicado Técnico, 25

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