AUTOMATIZAÇÃO DO PROCESSO DE TESTES

Anabela Marques
Teresa Santos
Ana Azevedo

OBLOG Software S.A.
Av. Álvares Cabral, 41 - 7º - 1200 Lisboa, Portugal
Telef. 00-351-1-387 16 69; Fax. 00-351-1-387 21 73

RESUMO

Este artigo descreve a influência do uso de ferramentas de teste na melhoria de qualidade do software desenvolvido na OBLOG SOFTWARE, S.A. Descreve-se aqui todo o processo, desde a selecção das ferramentas necessárias até aos resultados da sua utilização, passando pela montagem do ambiente, cuidados a ter e vantagens da sua utilização. Durante todo este processo foram registados alguns valores comparativos relativamente à execução de testes com recurso ou não à ferramenta.

1. INTRODUÇÃO

O projecto OBLOG tem como objectivo o desenvolvimento de uma ferramenta CASE baseada no paradigma dos objectos. Esta ferramenta é composta por quatro componentes básicos: um **Editor** que permite especificar os requisitos da aplicação, um **Gerador de Código** que gera o código correspondente à especificação, um **Verificador** que testa a completude e correcção da especificação, e um **Gestor de Sessões** que permite gerir toda a informação e aceder a qualquer um dos outros componentes.

O projecto é de grande dimensão (cerca de 4000 ficheiros de codigo source) e integra, desde o inicio, os resultados da investigação, em Sistemas de Informação orientados por objectos, realizada por grupos universitários.

A ferramenta possui versões comerciais para VMS, SOLARIS e HP-UX. Estando neste momento a decorrer o desenvolvimento da versão para Windows.

Dada a dimensão e complexidade do projecto, o modelo de ciclo de vido adoptado foi o incremental [SES94], sendo portanto muito importante a definição do processo de testes e a sua automatização, visto que, quer a inclusão de novas funcionalidades, quer a prevista adaptação para outras plataformas exigia um grande esforço em testes.

O relatório aborda os seguintes pontos:

- Ferramentas de teste - onde se descreve as motivações para o seu uso, os tipos de ferramentas de teste existentes e a identificação de quais as necessárias ao projecto;
- Utilização das ferramentas - onde se descrevem os cuidados a ter na montagem do ambiente e na gravação dos testes;
- Impactos da utilização das ferramentas - onde se descrevem os impactos que a utilização das ferramentas teve sobre a maneira de trabalhar, sobre a documentação dos testes e sobre os tempos de execução dos testes;
- Vantagens e desvantagens no uso da ferramenta - onde se apresentam os problemas sentidos e o valor acrescentado, resultantes da utilização da ferramenta.
- Perspectivas para o futuro - onde se apresentam algumas perspectivas de trabalho futuro.
- Conclusões

2. FERRAMENTAS DE TESTE

2.1 MOTIVAÇÃO

O procedimento de teste tem as seguintes fases[Myers79]:

- Análise, identificação **do que é que** se pretende testar;
- Desenho, definição dos casos de teste e de **como** os utilizar;
- Execução, onde dada uma versão do sistema se aplicam todos os casos de teste.

Os testes a uma determinada funcionalidade têm de ser executados quando ela é concluída e têm de ser repetidos sempre que ela é alterada, é corrigido um problema com ela relaccionado ou é feita qualquer outra alteração, da aplicação, que possa ter implicações no seu funcionamento.

Considerando ainda que qualquer alteração no código pode ter efeitos colaterais imprevisíveis, devem-se executar periodicamente testes de regressão. Estes consistem na re-execução de um conjunto de testes para assegurar que tudo o que funcionava bem anteriormente, continua a funcionar.

A utilização de uma ferramenta de teste tem como principal objectivo, a repetição da execução dos casos de teste, libertando assim os elementos da equipa de teste para actividades mais interessantes e criativas, como a criação de novos casos de teste.

Com a utilização de ferramentas de teste, o ciclo de vida das aplicações/projectos decresce consideravelmente e aumenta a qualidade do produto final, devido fundamentalmente aos seguintes factores:

- A re-execução dos casos de teste torna-se muito mais rápida, pois é automática,e pode inclusivamente ser desencadeada em periodos em que os recursos computacionais não forem tão necessários (e.g durante a noite);

- A possibilidade de erros humanos é eliminada, pois, deixam de existir casos de teste desenhados e não executados, introdução de valores de entrada errados, utilização de variáveis de ambiente e imagens incorrectas.

- Detecção possível de zonas de código não testadas;

- Detecção prévia de problemas de memória;

2.2 TIPOS DE FERRAMENTAS DE TESTE

As ferramentas de teste podem dividir-se segundo as seguintes categorias [Kit 95]:

- **Revisão e Inspecção** - apoiam a execução de revisões, verificações e inspecções dos requisitos, desenho funcional, documentação, testes e codigo.
    - Baseadas no Código - ajudam o utilizador na:
        - Análise da complexidade do código - em geral utilizam as metricas de complexidade e ciclomática de McCabe.
        - Comprensão do código - através da utilização de gráficos de estrutura, identificação de dependências, identificação de áreas de código que devem merecer especial atenção e detecção de código morto.
        - Análise sintática e semântica - permitindo a detecção de erros que o compilador não detecta, verificação do código, podendo detectar erros semânticos e inferir o que está sintaticamente incorrecto.São normalmente dependentes da linguagem utilizada.
- **Planeamento de testes** - apoiam na definição do âmbito, dos recursos e das datas previstas para as actividades relacionadas com os testes. Usualmente oferecem formatos pré-definidos baseadas nas normas ANSI/IEEE.
- **Desenho e Desenvolvimento** - automatizam a geração de dados de teste baseados num formato definido pelo utilizador. Oferecem formatos pré-definidos baseados nas normas ANSI/IEEE.
- **Execução e Avaliação** - Permitem a execução de casos de teste e avaliação dos resultados. Podem ser classificadas em
    - Captura e re-execução - permitem gravar todas as actividades do utilizador durante o teste, incluindo carregamento em teclas, movimentos do rato, captura de imagens e valores ASCII . As imagens e caracteres capturados servem de valores de comparação na re-execução do teste gravado;
    - Análise de cobertura - permitem avaliar a completude de um conjunto de testes, mostrando as partes de código não testadas. São utilizadas com as ferramentas de captura re-execução e permitem criar casos de teste que cobrem 100% do código;

- Análise de erros de memória - permitem a detecção de problemas de memória no código;

- Gestão de casos de teste - permitem organizar e gerir os casos de teste;

- Simuladores e medidores de desempenho - Permitem identificar partes da aplicação que dominam o tempo de execução da mesma.

- **Ferramentas de suporte aos testes** - Fornecem todo o suporte necessário às actividades relacionadas com os testes.

    - Gestão de problemas - são utilizadas para gravar, acompanhar e na generalidade assistir na gestão de problemas detectados quer pelos utilizadores finais quer pela equipa de testes durante o ciclo de vida dos produtos de software.

    - Gestão de configurações - controlam e coordenam as alterações aos documentos, testes e código durante o ciclo de vida do projecto.

2.3 IDENTIFICAÇÃO DAS FERRAMENTAS DE TESTE NECESSÁRIAS AO PROJECTO

Tornou-se evidente que, para evitar o esforço dispendido em testes de regressão, que se repetiam sucessivamente durante o ciclo de vida do projecto, só a automatização da execução dos testes pouparia tempo e motivaria a equipa de testes.

A automatização dos testes das componentes interactivas constituiu o problema principal, pois, exigia a aquisição de uma ferramenta de captura e re-execução de eventos de X windows. No momento, e visto estar-se a trabalhar numa única plataforma (VMS), as experiências efectuadas para a obtenção de uma ferramenta deste tipo não foram bem sucedidas pelo que se optou por centrar esforços em:

- Documentação e gestão dos problemas detectados;

- Documentação dos testes efectuados às componentes interactivas utilizando uma norma interna baseada na norma [IEEE829];

- Automatização dos testes efectuados a componentes não interactivas, ou a partes das quais se pudesse eliminar facilmente a interacção com o utilizador.

Ao iniciar o trabalho sobre mais plataformas (SOLARIS, HP-UX) a quantidade de ferramentas disponíveis foi aumentado, decidindo-se então a selecção e aquisição de uma **ferramenta de execução e avaliação** com componentes de:

- Captura e re-execução - que permitia gravar os testes e re-executa-los sempre que necessário, fornecendo também relatórios sobre o sucesso ou insucesso verificado durante a re-execução;

- Gestão de casos de teste - que permitia organizar os casos de teste e criar baterias com os mesmos, fornecendo também relatórios com métricas e informação histórica relativa à execução dos testes;

- Análise de cobertura - que permitia detectar as zonas de código não testadas e consequentemente a criação de novos casos de teste.

Os critérios para a selecção da ferramenta foram definidos antecipadamente e incluiram os seguintes aspectos:

- Plataformas e linguagens suportadas - A ferramenta teria de trabalhar sobre SOLARIS, HP-UX , WINDOWS e WINDOWS-NT e teria de fazer análise de cobertura a código escrito em C e C++.

- Portabilidade dos ficheiros gerados pela ferramenta, correspondentes aos testes gravados - Dada a necessidade de execução dos mesmos testes em plataformas diferentes, era importante a independência do ficheiro gerado em relação à plataforma.

- Facilidade de gravação dos testes - qualquer pessoa envolvida no projecto deveria poder usar a ferramenta para gravar testes, geralmente desenhados por alguém da equipa de testes.

- Existência de outras funcionalidades complementares às de captura/re-execução, nomeadamente análise de cobertura e gestão de casos de teste.

Além desta foram posteriormente adquiridas ferramentas de

- Análise de erros de memória - Permitem detectar zonas de código com problemas de memória tais como memória alocada e nao libertada, tentativas de utilizacao de memória não alocada;

- Análise de desempenho - Permitem identificar zonas de código da aplicação que determinam o tempo de execução.

A utilização destas ferramentas durante a re-execução dos testes gravados permitia detectar as zonas de código com problemas, cobertas pelos casos de teste gravados.

3. UTILIZAÇÃO DAS FERRAMENTAS

Estando perfeitamente convictos das vantagens da utilização de ferramentas de testes, também estávamos conscientes que teríamos de enfrentar algumas dificuldades, tais como:

- Espaço em disco ocupado pelos testes gravados;
- Manutenção dos casos de teste;
- Aprendizagem e adaptação a uma nova ferramenta.

Para minorar estes problemas houve então grande cuidado na definição de:

- Ambiente de gravação e re-execução dos testes;
- Cuidados a ter durante a gravação dos testes.

3.1 AMBIENTE DE GRAVAÇÃO E RE-EXECUÇÃO

3.1.1 DISPOSIÇÃO DAS JANELAS NO ÉCRAN

Tendo em conta que todas as posições relativas a *clicks* e movimentos do rato são capturadas pela ferramenta é obrigatório que a disposição das janelas e ícones usados num determinado teste sejam as mesmas no início da gravação/captura e re-execução do mesmo teste. Qualquer pequena diferença nas condições iniciais poderá levar a falhas ou a resultados enganadores na re-execução. Para evitar que cada pessoa tivesse que documentar cuidadosamente a disposição das janelas, antes de iniciar a gravação do seu teste optou-se por uma disposição comum para todos os testes.

3.1.2 DEFINIÇÃO DE VARIÁVEIS, SÍMBOLOS E *LINKS*

De modo a permitir que um teste gravado numa determinada plataforma possa ser re-executado nas diferentes plataformas é necessário existirem guiões com a definição das mesmas variáveis símbolos e *links* para os diferentes plataformas de modo a garantir que a interpretação seja a mesma. Estes guiões são sempre executados antes da re-execução de cada teste gravado.

Exemplo:

Para lançar um determinado componente da OBLOG CASE é necessário lançar um executável que dependendo da plataforma está em directorios

diferentes. Para permitir que um teste a este componente seja gravado numa única plataforma e re-executado nas outras, tem-se o seguinte nos guiões de ambiente:

SOLARIS: alias oblog=/apps/smg/smg

HP-UX: alias oblog=/apps/smg/Release/smg

VMS:oblog="@oblog_com:setings.com"

A gravação do teste passa a ser independente da plataforma e bastando executar o comando:

$oblog

3.1.3 REPOSITÓRIOS

Dada toda a informação, inserida durante um teste às componentes interactivas da OBLOG CASE, ser guardada em repositório é necessário assegurar que o seu estado inicial seja guardado. Para facilitar esta tarefa é necessário ter ferramentas que automatizem a importação e exportação do repositório.

3.2 CUIDADOS A TER DURANTE A GRAVAÇÃO

- O estado inicial da aplicação em teste deve ser facilmente reproduzível, tanto em relação ao ambiente como ao conteúdo do repositório.

- Devem utilizar-se os botões do rato o menos possível, de modo a evitar que pequenas alterações de posição conduzam à repetição da gravação do teste;

- Sempre que possível devem utilizar-se mnemónicas e *keyboard accelerators* (conjunções de teclas que permitem aceder directamente a uma opção de um menu sem ter de o abrir) , de modo a evitar que simples trocas de posição nos menus levem à repetição da gravação do teste;

- Sempre que possível deve-se capturar só a área necessária para comparação de modo a economizar o espaço em disco.

- Devem utilizar-se mecanismos de sincronização de modo a garantir que a re-execução do teste não falhe por alterações no tempo de resposta do sistema. Em geral as ferramentas de captura e re-execução disponibilizam mecanismos de sincronização com janelas, com areas do ecran ou com strings.

4. IMPACTOS DA UTILIZAÇÃO DAS FERRAMENTAS

4.1 IMPACTOS NA NOSSA MANEIRA DE TRABALHAR

Selecionadas as ferramentas de teste iniciou-se o processo de automatização de testes ao editor gráfico. A primeira fase foi a de gravação de todos os testes já desenhados.

De modo a aproveitar, ao máximo, a utilização da ferramenta, foi necessário proceder a algumas alterações na maneira de trabalhar:

- Redesenhar os testes, de modo a permitir a reutilização de certos ficheiros gerados pela ferramenta em vários testes. Antes da gravação de qualquer teste, é necessário, identificar o conjunto de casos de teste, que possam ser reutilizáveis noutros testes e colocá-los em bibliotecas de ficheiros reutilizáveis. Minimizando-se assim, o trabalho necessário na gravação de testes, o espaço em disco ocupado pelos ficheiros gerados pela ferramenta e o esforço de manutenção de casos de teste;

  Exemplo:

  Com a bancada OBLOG é possivel criar objectos de 3 categorias diferentes (OBL, TBX, DBX). Para todas estas categorias é possivel criar acções passivas e activas de vários tipos de acordo com determinadas restrições.

  O teste de criação de acções tinha o seguinte desenho

  TC1 - Criar acções passivas para um objecto tipo OBL

  TC2 - Criar acções activas para um objecto tipo OBL

  TC3 - Criar acções passivas para um objecto tipo TBL

  TC4 - Criar acções activas para um objecto tipo DBX

  TC5 - ...

  Para evitar a gravação repetida da criação de acções passivas e activas para cada tipo de objecto estas foram gravadas isoladamente num ficheiro reutilizadas em todos os casos de teste.

- Passar a utilizar *keyboard accelerators* e mnemónicas sempre que possível de modo a minimizar o esforço de manutenção dos testes, decorrente das alterações efectuadas na aplicação;

Exemplo:

Supondo que num determinado menu tem-se as opções

- SAVE Ctrl+S
- PRINT Ctrl+P

Ao gravar um teste à funcionalidade de *print,* se se usa o rato para a sua selecionar directamente a opção correspondente, corre-se o risco de mais tarde este teste vir a falhar por alteração da posição da opção.

Se numa posterior versão se decidir trocar as posições do *save* e *print* no menu. O teste que foi gravado para o *print* vai falhar pois irá em primeiro lugar escolher a opção de *save* e o diálogo que irá aparecer será completamente diferente do diálogo que pareceu quando se escolheu *print*. A solução para resolver este tipo de problemas é usar sempre que estejam disponíveis os *keyboard accelerators* (Ctrl+P e Ctrl+S), pois estes se forem alterados, facilmente se alteram nos ficheiros gerados pela ferramenta de gravação enquanto que as posições de *click* são muito mais difíces de determinar e como tal de alterar.

- Adaptação das ferramentas de importação e exportação do repositório OBLOG de forma a serem utilizados a partir da ferramenta de teste;
- Adaptação dos guiões de definição de variáveis, símbolos e *links*;

4.2 IMPACTOS NA DOCUMENTAÇÃO

Todos os testes desenhados até ao momento, estavam documentados segundo uma norma interna baseada na norma [IEEE829]

Uma vez que o componente de Gestão de casos de teste permitia documentar os casos de teste e respectivas baterias de teste, muita da informação que estava registada em papel passou a estar embutida na ferramenta.

Por outro lado, passou a ser necessário ter informação adicional relativamente a:

- Casos de teste e ficheiros, gerados pela ferramenta, relativos a cada teste;
- Bibliotecas de ficheiros reutilizáveis;
- Ficheiros correspondentes à baterias de teste executados pelo Gestor de casos de teste.

4.3 IMPACTOS NOS TEMPOS DE EXECUÇÃO

Durante todo o processo de automatização, foram registadas alguns valores que nos indicam uma ordem de grandeza relativamente a:

- Tempo dispendido na gravação de um teste *versus* execução do teste sem recurso ao uso da ferramenta;

  Verificou-se que o tempo gasto na gravação de cada teste era em média três vezes mais que o tempo gasto pela pessoa a executá-lo sem recurso à ferramenta. Neste tempo de gravação está incluído o tempo necessário para redesenhar os testes, o que aconteceu na maioria dos casos e o tempo de adaptação à nova ferramenta.

- Tempo dispendido na execução da baterias de testes gravados *versus* tempo que seria necessário para repetir a execução dos testes sem recurso à ferramenta.

  Esta relação é muito variável pois, o tempo de execução das baterias de teste é proporcional à sobrecarga da máquina e velocidade de execução definida para os testes na ferramenta. O tempo de execução de um teste sem recurso à ferramenta depende do tipo de pessoa que o vai executar.

  Contudo a título de exemplo conseguimos obter os seguintes valores:

  Um determinado teste planeado para 1 homem/dia supondo que iria ser executado por uma pessoa rápida e eficaz, passou a ser re-executado em 5 minutos, numa situação em que a máquina não apresentava grande sobrecarga. A este tempo é preciso adicionar a análise dos resultados que no caso de sucesso do teste é irrelevante, e no caso de insucesso leva a uma análise mais cuidada.

5. VANTAGENS E DESVANTAGENS NO USO DA FERRAMENTA

A maioria dos problemas sentidos na utilização da ferramenta foram consequência da:

- Inexistência de formação para a utilização da mesma, o que levou a um período de aprendizagem com recurso a manuais e ao grupo de suporte da empresa fabricante da mesma;

- Alteração no modo de trabalhar, com vista a evitar a necessidade de ter que se redesenhar alguns dos casos de teste levou a uso de bibliotecas e a um maior

cuidado, por parte da equipa de desenvolvimento nas alterações ao interface gráfico;

- Necessidade de espaço em disco para a gravação dos testes;
- Falta de recursos humanos totalmente disponíveis para esta tarefa, o que levou a um acréscimo do tempo inicialmente previsto para a gravação dos testes;
- Dificuldades em fazer as pessoas respeitar as regras definidas para a gravação dos teste de modo a garantir o sucesso na re-execução;

6. PERSPECTIVAS PARA O FUTURO

Num futuro próximo, pensamos melhorar significativamente este processo, adquirindo ferramentas que nos permitam fazer:

- Utilização da ferramenta em ambiente windows e windows-NT para os quais se estão a desenvolver versões da OBLOG CASE.
- Integração com a ferramenta de gestão de configurações no novo ambiente de desenvolvimento de modo a permitir o controle de alterações aos testes gravados em cada versão da OBLOG CASE;
- Geração automática de casos de teste;
- Análise estática de código de modo a de obter informação que nos permita detectar código morto, variáveis e funções declaradas e não usadas, variáveis não inicializadas, comentários dentro de comentários entre outras possíveis problemas.
- Obtenção de mais métricas relacionadas com o código fonte relativamente a linhas de código, linhas de comentários, número de funções, complexidade ciclomática.
- Obtenção de métricas mais rigorosas relativamente ao desempenho na gravacão de testes, de modo a permitir avaliar com mais rigor o tempo ganho com automatização do processo de teste.

7. CONCLUSÕES

Este artigo apresentou uma descrição da automatização do processo de testes usado na OBLOG Software, com recurso ao uso de ferramentas de teste.

O valor acrescentado, resultante da utilização das mesma, fez-se sentir em vários campos, dos quais importa salientar:

- Redução do tempo necessário para as diferentes fases de teste, com consequente redução do ciclo de vida de cada de desenvolvimento de *software*;
- Redução de erros humanos, isto é, casos de teste que ficam por executar, variáveis de ambiente mal definidas;
- Detecção prévia, de áreas de código não testadas com o uso da ferramenta de análise de cobertura;
- Detecção de problemas de memória, re-executando os testes com imagens linkadas com a ferramenta de análise de memória;
- Maior facilidade na obtenção de métricas;
- Maior grau de motivação da equipa de trabalho;
- Aumento da qualidade do processo de desenvolvimento e do produto final, como consequência óbvia dos pontos referido acima.

BIBLIOGRAFIA

[Myers79] Glenford J. Myers , *The Art of Software Testing* . Wiley ,1979.

[Yourdon95] Ed Yourdon, "New Trends in Software Testing", *Application Development Strategies*, March 1995.

[Beizer90] Beizer, B. *Software Testing Tecniques,* Van Nostrand Reinhold, 1984

[kit95] Edward Kit, "Software Testing in Real World", Factor Equilíbrio, Junho 1995.

[SES94] Mazza C.,Fairclough J. Melton, Pablo D. de, Scheffer A., Stevens R., *Software Engineering Standards*. Prentice Hall Internacional (UK) Limited, 1994

[IEEE829] ANSI/IEEE Std 829-1983 -IEEE Standard for Software Test Documentation

[IEEE1008] ANSI/IEEE Std 1008-1987 -IEEE Standard for Software Unit Testing.

[SR95] - *Software TestWorks User's Guide*, Software Research , Inc, 1995