

Nesta seção você encontra artigos sobre técnicas que poderão aumentar a qualidade do desenvolvimento de software

## Automação de testes

Automatize os testes de seus sistemas usando o software TestComplete



### Resumo DevMan

Um assunto pouco explorado hoje em dia ainda está relacionado ao teste de software. Muitas empresas ainda não possuem práticas eficazes para aplicar testes em seus aplicativos ou muitas vezes nem o fazem, deixando somente a cargo do desenvolvedor do sistema. Essa falta de testes acarreta em mau funcionamento do aplicativo quando distribuído.

O que pouca gente sabe é que o mercado está cada vez mais exigente e a qualidade de software cada vez mais é quesito para aquisição de um novo sistema. O grande problema é que uma boa parte das software houses hoje em dia efetuam seus testes manualmente, comprometendo todo o software.

Automatizar o processo de homologação de um novo sistema ou funcionalidade do produto, faz com que o nível de qualidade se eleve e fortaleça o produto como um todo. Nesse artigo veremos como utilizar a ferramenta TestComplete para efetuar testes automatizados em nossas aplicações.

### Nesse artigo veremos

- Automação de testes;
- Como utilizar o software TestComplete para automatizar testes;
- A importância de se efetuar testes em aplicações reais.

### Qual a finalidade

- Entender a necessidade de se automatizar os testes e apresentar o software TestComplete como uma ferramenta de auxílio nessa automatização;

### Quais situações utilizam esses recursos?

- Qualquer software necessita de homologação para se comercializar e distribuir sem maiores transtornos, por isso os testes devem ser realizados em todo tipo de software;



### Cristiano Caetano

(cristiano.caetano@testanywhere.com.br)

é certificado CBTS pela ALATS. Diretor da TestAnywhere, consultoria de teste de software. Possui mais de 10 anos de experiência, já trabalhou na área de qualidade e teste de software para grandes empresas como Zero G, DELL e HP Invent. Autor dos livros "CVS: Controle de Versões e Desenvolvimento Colaborativo de Software" e "Automação e Gerenciamento de Testes: Aumentando a Produtividade com as Principais Soluções Open Source e Gratuitas". Criador e mantenedor do portal TestExpert

É muito comum encontrar desenvolvedores e empresas que ainda testam seus softwares apenas de forma manual. E com o aumento de alterações e novas funcionalidades que são incluídas nesses softwares fica difícil manter o ritmo e a qualidade dos testes que são aplicados, comprometendo assim a qualidade final do projeto.

Uma abordagem baseada puramente em testes manuais, normalmente não consegue acompanhar as demandas e o volume de homologações do sistema ao longo do ciclo de vida de desenvolvimento de software. Frequentemente o sistema é liberado sem que tenha sido completamente testado em virtude de falta de tempo e recursos.

A automação de testes, quando utilizada corretamente, permite a execução ininterrupta de testes a qualquer hora do dia ou da noite. A automatização desse processo é sempre mais rápida do que um processo manual e menos suscetível a erros, afinal, a máquina nunca erra. Neste artigo aprenderemos um pouco sobre os principais paradigmas de automação de testes e as funcionalidades básicas do *TestComplete*, uma ferramenta de automação de testes que tem um ótimo suporte para sistemas desenvolvidos em Delphi.

## Paradigmas da automação de testes

Os testes automatizados interagem com o sistema a ser testado. Alguns podem testar um sistema simulando o uso do mesmo por um usuário, como se os controles de um determinado formulário fossem clicados, enquanto que outros aplicam testes em uma parte mais interna do sistema, nas rotinas que formam as regras de negócio. Os tipos de automação são normalmente agrupados de acordo com a forma como os testes automatizados interagem com o sistema. Em geral temos:

- **Baseados na Interface Gráfica:** Nessa abordagem os testes automatizados interagem diretamente com a interface gráfica do sistema simulando um usuário. Normalmente as ações dos usuários são gravadas (*Capture*) por meio da ferramenta. A ferramenta transforma as ações dos usuários em um *script* que pode ser reproduzido (*Playback*) posteriormente.

- o **Vantagens:** Não requer modificações no sistema para a automatização. Não é necessário tornar o sistema mais fácil de testar (*testabilidade*) porque eles se baseiam na mesma interface utilizada pelos usuários;

- o **Desvantagens:** Existe uma forte dependência da interface gráfica. Se a interface gráfica mudar, os testes falham. Por exemplo, um formulário é criado contendo um *CheckBox* e um teste é criado para verificar seu estado. Se amanhã esse *CheckBox* for substituído por um *RadioButton* o teste irá falhar, caso não seja reconstruído. Baixo desempenho para

Item de projeto	Descrição
ActiveX Objects	Este item de projeto permite o reconhecimento de objetos <i>ActiveX</i> .
Events	Este item de projeto permite a sobreposição e personalização da ação que é disparada nos eventos ocorridos durante a execução do teste.
HTTP Load Testing	Permite a criação e execução de testes de desempenho de aplicações <i>WEB</i> .
Low-Level Procedures Collection	Permite a criação de testes baseados nas coordenadas (X, Y) da tela.
Manual Tests	Criação e execução de casos de testes manuais.
Name Mapping	Este item de projeto permite renomear os objetos para nomes personalizados e mais curtos para facilitar a leitura e criação de <i>scripts</i> .
Network Suite	Execução de testes distribuídos em vários computadores.
ODT	Criação de <i>scripts</i> de testes dirigidos a objetos ( <i>Object-driven testing</i> ).
Script	Permite a criação e armazenamento dos <i>scripts</i> de testes (que podem ser criados nas seguintes linguagens: <i>VBScript</i> , <i>JScript</i> , <i>DelphiScript</i> , <i>C++ Script</i> ou <i>C# Script</i> ).
Stores	Este item de projeto armazena arquivos que são usados para posterior comparação durante a execução dos testes.
Unit Testing	Este item de projeto permite a criação de testes unitários que podem ser criados nas seguintes tecnologias: <i>MSTest</i> , <i>JUnit</i> , <i>NUnit</i> , <i>DUnit</i> ou <i>TestComplete unit tests</i> .
User Forms	Este item de projeto permite a criação de formulários personalizados que podem ser usados para obter informações antes da execução dos testes ou apresentar informações ao longo ou ao final da execução.
Web Services	Este item de projeto permite a criação de testes de <i>Web Services</i> .
Win32 Tested Applications	Este item de projeto permite um controle da execução das aplicações que são testadas pelo <i>TestComplete</i> .

Tabela 1. Itens de projeto do TestComplete

testes automatizados que exigem centenas de milhares de repetições, testes de funcionalidades que realizam cálculos complexos, integração entre sistemas diferentes e assim por diante.

- **Baseados na Lógica de Negócio:** Nessa abordagem os testes automatizados exercitam as funcionalidades do sistema sem interagir com a interface gráfica. A interface gráfica é apenas uma casca (*camada*) que tem o objetivo de fornecer um meio para a entrada dos dados e apresentação dos resultados. A camada que abriga a funcionalidade e o comportamento do sistema é a camada de lógica de negócio. Essa abordagem de testes é baseada no entendimento que 80% das falhas estão associados a erros na lógica de negócio, ou seja, existem poucos erros críticos na camada de interface com o usuário. Normalmente é necessário realizar modificações no sistema para torná-lo compatível com essa abordagem. Essas modificações resultam em mecanismos para expor ao mundo exterior as funcionalidades internas do sistema.

- o **Vantagens:** Foco na camada onde



## Nota do DevMan

O *DUnit* é uma ferramenta de código aberto utilizada para efetuar testes unitários em programas desenvolvidos em Pascal ou Delphi. Ele é baseado no *JUnit*, para Java. Porém, a principal premissa para utilização do *DUnit* é que o sistema seja orientado a objetos, diferentemente do *TestComplete* que pode efetuar testes em qualquer tipo de sistema.

Há outros projetos que prometem o mesmo recurso, como o *nUnit*. Uma boa dica é acessar o site *dunit.sourceforge.net/* e entender melhor como funciona a ferramenta como um todo.

ClubeDelphi PLUS

[www.devmedia.com.br/clubedelphi/portal.asp](http://www.devmedia.com.br/clubedelphi/portal.asp)

Acesse agora mesmo o portal do assinante ClubeDelphi e assista uma vídeo aula de Guinther Pauli que apresenta uma introdução a testes unitários utilizando *DUnit*.

<http://www.devmedia.com.br/articles/viewcomp.asp?comp=540>





Figura 1. Sistema Demo usada neste artigo

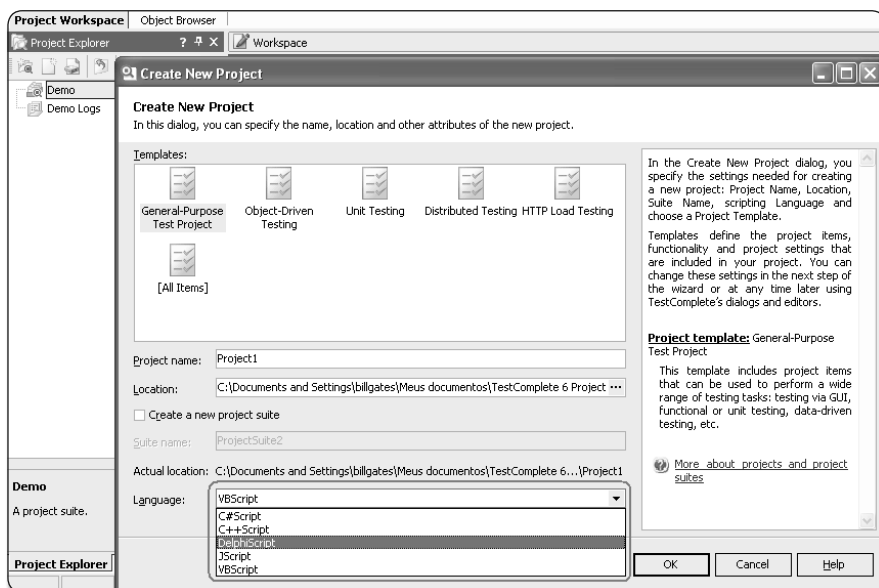


Figura 2. Diálogo de criação de um projeto

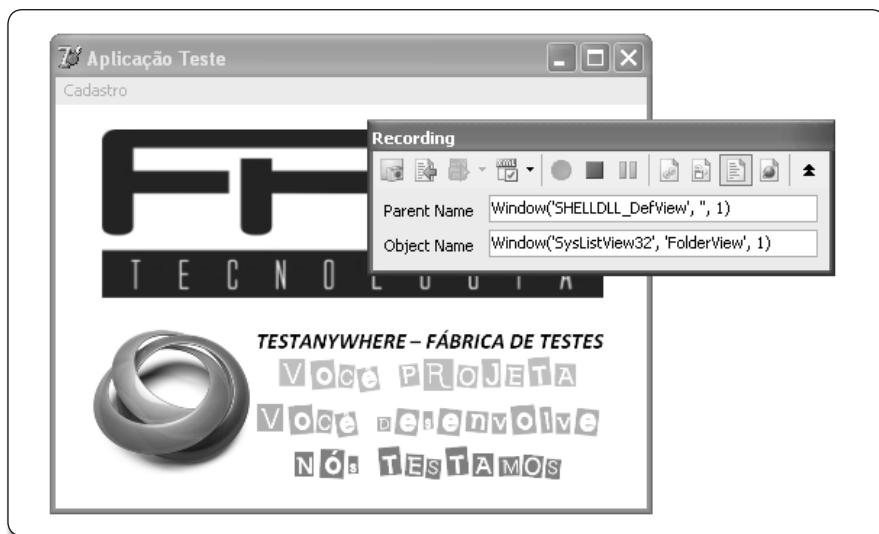


Figura 3. Gravando um script de teste

existe maior probabilidade de existir erros. Independência das mudanças da interface gráfica. Alto desempenho para testes automatizados que exigem centenas de milhares de repetições, testes de funcionalidades que realizam cálculos complexos, integração entre sistemas diferentes e assim por diante.

o *Desvantagens*: Requer grandes modificações no sistema para expor as funcionalidades ao mundo exterior. Exige profissionais especializados em programação para criar os testes automatizados. Existem poucas ferramentas/*frameworks* que suportam essa abordagem (normalmente é necessário criar soluções caseiras).

## TestComplete

O *TestComplete* é uma ferramenta de automação de testes de uso comercial. As versões *Enterprise* e *Standard* custam US\$ 1,999.00 e US\$ 999.00, respectivamente e são desenvolvidas pela empresa *AutomatedQA*. Uma versão de avaliação pode ser encontrada no site da empresa no link [www.automatedqa.com/downloads/testcomplete/index.asp](http://www.automatedqa.com/downloads/testcomplete/index.asp). Um dos destaques do *TestComplete* é o seu excelente suporte para a automação de testes de sistemas desenvolvidos em Delphi.

A ferramenta oferece recursos para automatizar os testes, transformando as ações dos usuários em um *script* que pode ser reproduzido (*Playback*) posteriormente.

A arquitetura é baseada em *plug-ins*. Estes *plug-ins* são chamados de *Itens de Projeto*. Por exemplo, se você precisar testar um *Web Service*, será necessário adicionar o item de projeto chamado *Web Services* para poder usar essa funcionalidade, e assim por diante. Na **Tabela 1** são apresentados os itens de projeto oferecidos pelo *TestComplete* atualmente (a versão *Enterprise* vem com todos os itens de projeto existentes).

## Automação de testes na prática

Nessa parte, apresentaremos as funcionalidades básicas do *TestComplete*. Para fins didáticos, construímos uma aplicação simples no Delphi para demonstrar a utilização da ferramenta em questão. O sistema tem uma tela principal e um

menu que dá acesso a tela de cadastro de clientes com alguns campos básicos, como pode ser observado na **Figura 1**.

Para automatizar os testes é necessário primeiro criar um projeto. Para realizar tal tarefa, devemos abrir o menu *File>New>New Project*. O diálogo de criação de um novo projeto deverá ser exibido e nesse diálogo devemos escolher o *template General-Purpose Test Project*. Observe na **Figura 2**, que é permitido a escolha da linguagem de *script* em que os testes serão criados. Para o nosso exemplo, devemos escolher a opção *DelphiScript*.

Tão logo o projeto seja criado, podemos iniciar a gravação de um *script* de teste automatizado. Para realizar esta tarefa, devemos abrir o menu *Script>Record*. O diálogo *Recording* é exibido (**Figura 3**). A partir desse ponto todas as ações serão gravadas. Em nosso exemplo, vamos abrir a janela de cadastro de cliente por meio do menu *Cadastro>Clientes*, cadas-

trar um novo cliente, gravar os dados digitados e fechar a janela. Após essas ações devemos clicar no botão *Stop* do diálogo *Recording*.

Quando a gravação é finalizada, o *TestComplete* cria um novo procedimento na *Unit1* e converte todas ações realizadas em instruções na linguagem de *script* alvo, no nosso caso é a *DelphiScript*, como pode ser visto na **Figura 4**.

**Nota:** O *script* original foi modificado para aumentar a legibilidade desse artigo.

Devemos ressaltar que, durante a gravação, o *TestComplete* não grava os movimentos do mouse, ou cliques nas coordenadas X e Y. O motor (*engine*) de gravação, consegue entrar no sistema durante a gravação e identificar os nomes e classes das janelas e objetos, bem como as suas propriedades. Além disso, todas as ações são convertidas em instruções fáceis de entender para quem tem familiaridade com programa-

ção, como por exemplo: *ClickButton*, *Click*, *ClickItem*, *Keys*, etc. Dessa forma, a criação e manutenção dos *scripts* de teste fica mais fácil e intuitiva.

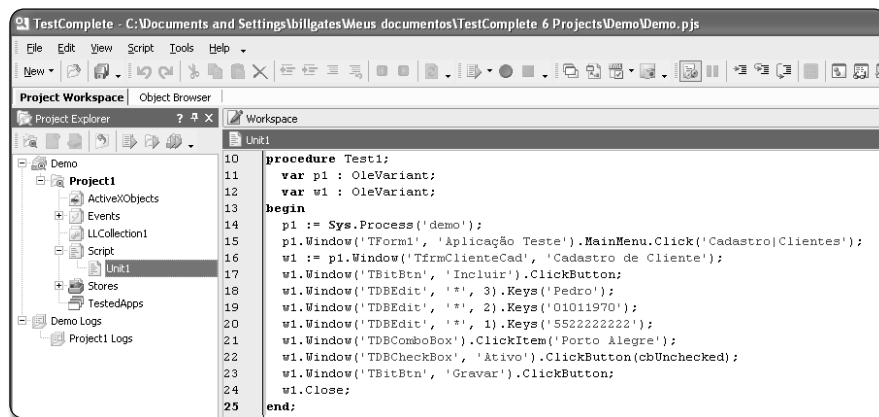
Adicionalmente, devemos reforçar que o *TestComplete* oferece um ambiente completo de desenvolvimento para a criação e manutenção dos *scripts* de testes. Dentre os recursos oferecidos, devemos destacar a integração com sistemas de controle de versões e recursos para a depuração (*debug*) dos *scripts* de testes. O recurso de depuração oferece funcionalidades semelhantes ao Delphi, como por exemplo, *breakpoints*, *watch list*, *call stack*, entre outros (**Figura 5**).

Com o código do *script* gravado podemos executar o teste por meio do menu *popup Run current routine* (**Figura 6**). O programa executará todas as ações da mesma forma e ordem em que foram gravadas. Ao final da execução ele exibe um relatório (*Test Log*). O relatório de execução apresenta informações sobre os *scripts* que foram executados, as ações que foram realizadas, os erros ocorridos e demais informações sobre a execução dos testes (**Figura 6**).

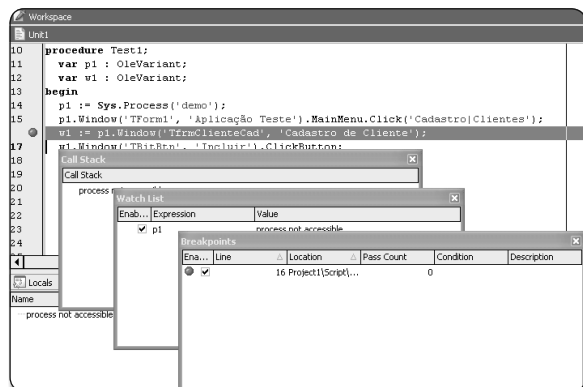
Até agora vimos como gravar e executar um *script* de testes. No entanto, não fizemos nenhum teste de verdade, apenas execução automática de ações.

## Criando um caso de teste

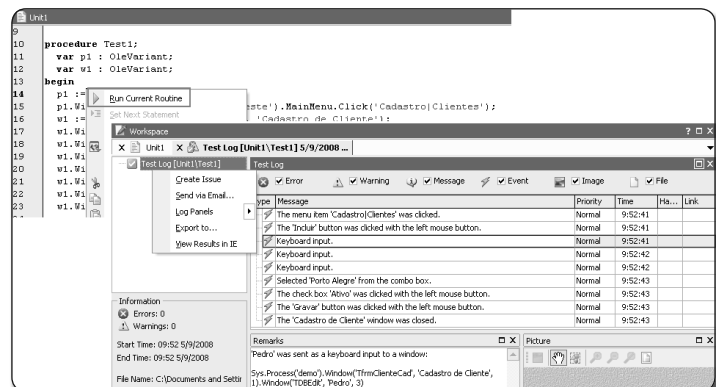
Um caso de teste é, em princípio, a execução de uma ação no sistema e a avaliação/comparação de um resultado esperado. Para realizar a avaliação do resultado esperado, o *TestComplete* oferece um recurso chamado *Checkpoint*.



**Figura 4.** Script de teste criado na linguagem DelphiScript



**Figura 5.** Recursos para depuração de scripts



**Figura 6.** Relatório com o log de execução do script

Um *Checkpoint* é um recurso usado para realizar um ponto de verificação (ou comparação) durante a execução dos testes. Por exemplo, durante a execução de um teste hipotético você usará *Checkpoints* para verificar se o campo *valor total da venda* apresenta o valor esperado, se o botão *Cancelar Venda* está desabilitado e os dados foram inseridos nas tabelas do banco de dados corretamente. Na **Tabela 2** são apresentados os *Checkpoints* suportados pelo *TestComplete*:

Com base no que foi exposto, para criar um *Checkpoint* durante a gravação do teste, devemos clicar no botão *Add Checkpoint from the list* do diálogo *Recording*, como pode ser observado na **Figura 7**. Para fins de análise e entendimento, vamos criar

um *Property Checkpoint*. Este *Checkpoint* cria um ponto de verificação onde uma propriedade de um objeto na tela é comparada com um valor pré-definido. Clique na opção *Create Property Checkpoint*. O diálogo *Create Property Checkpoint* deverá ser exibido. Nesse diálogo, devemos escolher o objeto e a propriedade que será comparada com o valor pré-definido. No nosso exemplo o objetivo desse *Checkpoint* é avaliar se o *ComboBox Cidade* apresenta o valor *Porto Alegre* após gravarmos os dados do cadastro. Para isso vamos escolher esse *ComboBox* da janela *Cadastro de Cliente* e a propriedade *wText*. Essa propriedade contém o nome da cidade que é exibido no *ComboBox* (**Figura 8**). Na **Figura 9** podemos ver o *script* gerado.

## Organizando os Testes

Por fim, após a criação de dezenas ou centenas de *scripts* de testes percebemos que é necessário executá-los numa ordem específica, em grupos ou hierarquicamente. O *TestComplete* oferece um recurso bastante avançado para gerenciar a execução de grupos de *scripts* de testes. Para utilizar esse recurso, você deverá clicar duas vezes sobre o projeto *Project1*.

Nesta janela, selecione a aba *Test Items* e em seguida você poderá incluir os testes que você deseja executar na ordem que quiser, agrupá-los hierarquicamente e configurar o comportamento da execução, tais como: se o teste deverá parar se houver um erro ou exceção, o tempo de espera máximo antes de cancelar a execução de um teste que ficou travado por alguma razão, a quantidade de vezes que um teste deverá ser executado e assim por diante, como pode ser visto na **Figura 10**.

## Conclusão

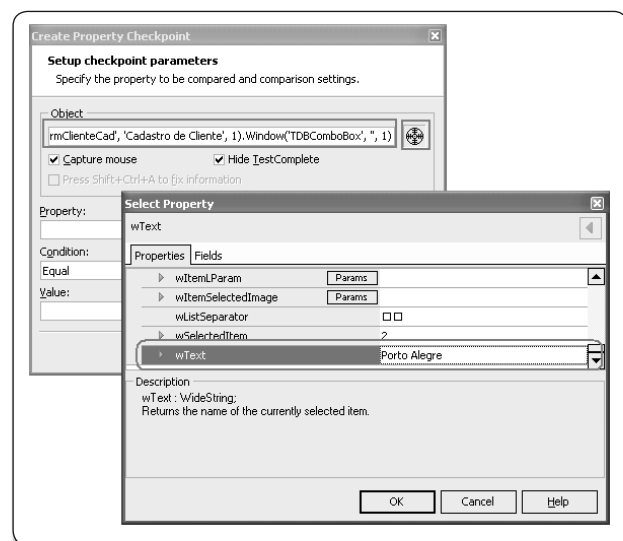
Como observamos neste artigo, a automação de testes não é uma atividade fácil. Além disso, automatizar os testes é uma atividade muito parecida com o desenvolvimento. A automação de testes deve ser encarada como um projeto de desenvolvimento com características e infra-estrutura próprias, ou seja, exige um planejamento detalhado, assim como atividades de projeto e desenvolvi-

Checkpoint	Descrição
Property Checkpoints	Permite a criação de pontos de verificação com base nas propriedades dos objetos.
Object Checkpoints	Permite a criação de pontos de verificação de objetos.
Table Checkpoints	Permite a criação de pontos de verificação de <i>grids</i> e objetos que apresentem as informações em formato tabular.
File Checkpoints	Permite a criação de pontos de verificação para a comparação de arquivos.
XML Checkpoints	Permite a criação de pontos de verificação para a comparação de arquivos no formato <i>XML</i> .
Database Table Checkpoints	Permite a criação de pontos de verificação que realizam a comparação de dados armazenados em tabelas no banco de dados.
Region Checkpoints	Permite a criação de pontos de verificação que realizam a comparação de imagens de regiões da tela.
Web Service Checkpoints	Permite a criação de pontos de verificação dos resultados de uma resposta de um <i>Web Service</i> .
Web Accessibility Checkpoints	Permite a criação de pontos de verificação de alguns aspectos de acessibilidade de páginas <i>WEB</i> .
Web Comparison Checkpoints	Permite a criação de pontos de verificação para comparar alguns tipos de atributos de páginas <i>WEB</i> .

**Tabela 2.** Tipos de checkpoints



**Figura 7.** Listagem dos Checkpoints existentes



**Figura 8.** Adicionando um Property Checkpoint

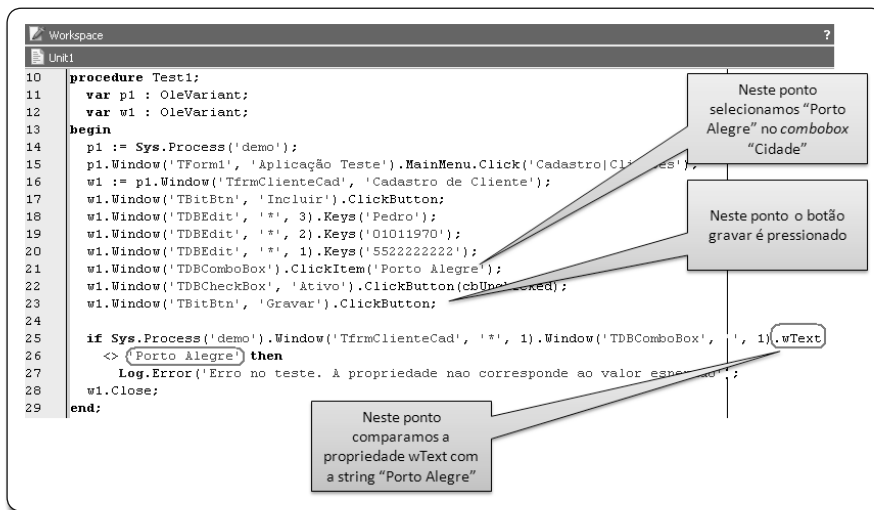


Figura 9. Script de teste contendo um Property Checkpoint

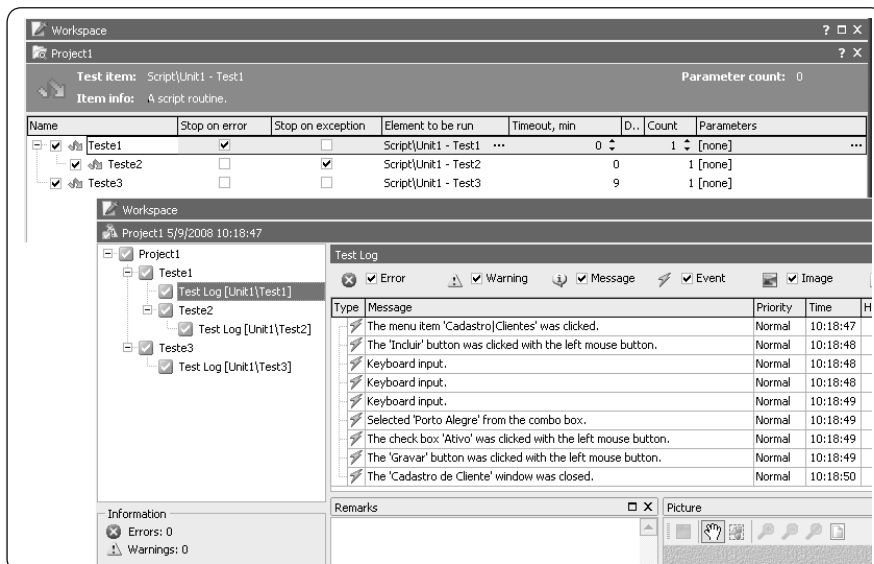


Figura 10. Organização da execução de um conjunto de scripts e o relatório

to/manutenção dos *scripts*, tal qual o desenvolvimento de um software convencional. O investimento em automação de testes não se limita apenas aos custos de aquisição da ferramenta de automação de testes, devemos também considerar a capacitação dos profissionais em desenvolvimento de software.

Segundo Cem Kaner, autor do livro *Lessons Learned in Software Testing*, o propósito da automação de testes pode ser resumidamente descrito como a aplicação de estratégias e ferramentas tendo em vista a redução do envolvimento humano em atividades manuais repetitivas. Dessa forma, devemos reconhecer que a automação de testes não deve ser empregada como um substituto do teste manual. Ela deve ser introduzida como uma técnica adicional cujo objetivo principal é agregar valor, sem, no entanto, invalidar o teste manual existente. Afinal, testes manuais e automatizados são abordagens de testes diferentes que se reforçam mutuamente. ●

#### Links

TestAnywhere – TestComplete no Brasil  
[www.testanywhere.com.br](http://www.testanywhere.com.br)

AutomatedQA – Fabricante do TestComplete  
[www.automatedqa.com](http://www.automatedqa.com)

TestExpert – Comunidade Brasileira de Teste de Software  
[www.testexpert.com.br](http://www.testexpert.com.br)

#### Dê seu feedback sobre esta edição!

A Clubedelphi tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

[www.devmedia.com.br/clubedelphi/feedback](http://www.devmedia.com.br/clubedelphi/feedback)

