Automatização de testes de software na norma ISO/IEC 25051

Software Testing Automation for the ISO/IEC 25051 Standard

Joana Vilas Boas, Instituto Politécnico do Cávado e Ave, Portugal, edite.joanavb@gmail.com
Nuno Lopes, Instituto Politécnico do Cávado e Ave, Portugal, nlopes@ipca.pt
João Carlos Silva, Instituto Politécnico do Cávado e Ave, Portugal, jcsilva@ipca.pt

Resumo

A complexidade subjacente ao processo de testes de software tem potenciado o desenvolvimento de ferramentas de automatização de testes de software permitindo reduzir os custos para as organizações. A automatização deve ser capaz de comparar os resultados obtidos com os resultados esperados, evidenciando o resultado do teste. Deve também permitir a realização de testes sistemáticos e paralelos em diferentes ambientes de teste, sem o aumento do tempo da atividade de testes e de recursos humanos.

Este trabalho apresenta a automatização de testes de software para a certificação de software na norma ISO/IEC 25051. A ferramenta de automatização usada baseia-se na automatização dos eventos na interface gráfica, usando a ferramenta Sikuli. Como caso de estudo é apresentada uma aplicação comercial na área do planeamento e construção civil. Os resultados evidenciam que é possível automatizar os testes de software necessários para cumprir os requisitos da norma ISO/IEC 25051.

Palavras-chave: Certificação; Norma ISO/IEC 25051; Testes; Automatização de testes; Sikuli

Abstract

The complexity of testing processes induces the development of tools for automated testing software allowing to reduce time, resources and consequently costs for the organization. Automation should be able to compare the obtained results with the expected results, evidencing the test result. It should also permit systematic and parallel testing in different test environments without the increase of time and human resources.

This paper presents automated software testing for certification in ISO / IEC 25051. The automation tool used is based on the automation of events in the GUI using the Sikuli tool. As a case study, a commercial application on building planning is presented. The results show that it is possible to automate software testing necessary to meet the requirements of ISO / IEC 25051.

Keywords: Certification; Standard ISO/IEC 25051; Test; Automated testing; Sikuli

1. INTRODUÇÃO

Testar Software engloba um conjunto de atividades com o objetivo de encontrar erros no software antes da entrega ao cliente. Estas atividades devem ser planeadas antecipadamente e conduzidas sistematicamente. Por esta razão é necessário definir um modelo incorporando uma sequência de passos com os desenhos de caso de testes e métodos a executar. Esta atividade consome uma parte significativa do esforço de um projeto de desenvolvimento de software, sempre com o objetivo de encontrar erros antes da fase de manutenção, pois o custo de correção nesta fase pode ser até 100 vezes maior [Pressman, 2010].

ISSN 2183-489X
DOI http://dx.doi.org/10.18803/capsi.v16.231-236

Os testes "têm por objetivo identificar o maior número possível de erros tanto nos componentes isolados quanto na solução tecnológica como um todo" [Bartié, 2002], podendo optar-se por dois tipos de abordagem diferentes, o "*black*-box *testing*" e o "*white-box testing*". Normalmente o *white-box testing* tende a ser utilizado no inicio dos testes e o *black-box testing* mais no final dos testes, devido às suas especificações [Javanovic & Irena, 2008]. Por não existir a possibilidade de acesso ao código, os testes a considerar no caso de estudo apresentado são do tipo "*black-box testing*". Esta abordagem é utilizada quando se conhecem as especificações do produto, de modo que os testes possam ser conduzidos para demonstrar que cada função está totalmente operacional ao mesmo tempo que se procuram erros [Sirohi & Parashar, 2013].

Para este processo é fundamental seguir normas de qualidade para criar métodos coerentes, rigorosos e uniformes. A norma que será utilizada neste artigo será a ISO/IEC 25051, que abrange os requisitos de qualidade para produtos de software e avaliação (SQuaRE), requisitos para a qualidade de produtos de software pronto a utilizar (Ready to Use Software Product - RUSP) e instruções para testes.

O software que será usado como caso de estudo é o software CYPE, constituído por diversas aplicações vocacionadas para o projeto de arquitetura, engenharia e construção em Portugal. [TOP Informática, 2008].

Para diminuir o esforço da atividade de testes e aumentar a qualidade do software, a automatização de testes de software ajuda na redução de tempo, recursos e consequentemente os custos para a organização. O software utilizado para a automatização foi o Sikuli (Sikuli, 2015) permitindo identificar e controlar os componentes GUI utilizando reconhecimento de imagens.

A estrutura do presente artigo incorpora na secção 2 uma apresentação do estado da arte, na secção 3 uma descrição dos requisitos de teste para a norma ISO/IEC 25051, na secção 4 a descrição de um caso de estudo, e uma conclusão na secção 5.

2. ESTADO DA ARTE

A estratégia de testes é normalmente criada pelo gestor do projeto, engenheiros de software e especialistas em testes. Uma estratégia de testes de software tem como objetivo criar um plano que descreva os passos necessários para testar o software e quantificar o esforço ao nível do tempo e recursos, que são necessários. Para a obtenção destes objetivos é necessário a criação de plano de testes, desenho de casos de teste, execução dos testes, recolha e avaliação dos dados obtidos (Pressman, 2010).

Definir uma estratégia de testes sistemática é muito importante. A sua condução "ao acaso" induz um esforço desnecessário e não potencia a deteção dos erros. Na estratégia de testes, deve-se definir quais os testes a realizar, para isso podemos integrar os testes ao longo do processo de desenvolvimento de software.

O objetivo da automatização de testes é utilizar ferramentas para automatizar tarefas manuais. Os testes manuais realizados do início ao fim podem ser propensos a erros e é um processo tedioso. Por estes motivos a procura pela automatização dos testes é grande. Ao longo dos anos vários estudos revelaram que testes

totalmente automatizados não são possíveis, mas várias atividades podem ser automatizadas para ajudar no processo. Esta automatização pode ser suportada por ferramentas comerciais ou ferramentas desenvolvidas pelas organizações. A automatização de testes permitirá aliviar as pessoas de tarefas tediosas e repetitivas, melhorando a produtividade da tarefa de testes em geral (Tian, 2005).

Nas diferentes atividades de testes, a execução é a primeira candidata para a automatização, sendo a atividade realizada com maior sucesso através de ferramentas de automatização. Ao automatizar várias tarefas, é possível que as mesmas sejam testadas pela mesma sequência simultaneamente em sistemas operativos diferentes (Tian, 2005).

Quando se efetuam testes de software é necessário criar evidências de teste. Por exemplo, uma das funcionalidades que uma ferramenta para automatização de testes deve permitir é a gravação dos testes para evidenciar o resultado do teste.

O processo automático de testes permite a redução dos custos, sendo uma grande mais valia para o processo de desenvolvimento de software (Mariani, Pezzè, Riganelli, & Santoro, 2011).

2.1. Ferramentas de automatização

As ferramentas para automatização de testes permitem criar processos mais fáceis, permitindo auxiliar ou substituir os testes manuais. Existem no mercado diversas aplicações que ajudam na criação de testes automáticos para as distintas fases de testes. Existem aplicações específicas para automatizar testes de software, e aplicações que permitem automatizar tarefas no computador e que podem ser usadas para automatizar testes.

Várias aplicações que podem ser utilizadas para a automatização de testes: SeleniumHQ, TestMaster, TestMaker, TestComplete, Watir são algumas aplicações específicas para automatização de testes de software. GhostMouse, WinScheduler, Mimer, Automize, Sikuli são algumas aplicações que permitem automatizar tarefas e processos no computador.

O Sikuli [Sikuli 2015] é um software que automatiza tudo o que é visualizado no ecrã. Para identificar e controlar os componentes GUI, a ferramenta Sikuli utiliza reconhecimento de imagens. Uma das aplicações práticas do software Sikuli é a possibilidade de automatizar testes de interfaces gráficas.

3. REQUISITOS DA NORMA ISO/IEC 25051

A norma ISO/IEC 25051 abrange os requisitos de qualidade para produtos de software e avaliação (SQuaRE), requisitos para a qualidade de produtos de software pronto a utilizar (RUSP) e instruções para testes.

Um produto de software pronto a utilizar (RUSP) deve conter, conforme descrito na Cláusula 5 da norma ISO/IEC 25051, uma descrição do produto com toda a informação necessária para potenciais adquirentes e utilizadores do produto assim como documentação de utilizador que apresente a informação necessária para

a utilização do software. A descrição do produto deve dar uma visão geral das funções do produto que podem ser chamadas pelo utilizador final. Um software tem qualidade quando cumpre uma série de requisitos, nomeadamente: funcionalidade, fiabilidade, usabilidade, eficiência, manutenibilidade, portabilidade e qualidade na utilização (ISO/IEC25051, 2006).

Na norma ISO/IEC 25051 não são recomendados quaisquer métodos ou técnicas de teste específicos. A Cláusula 6 da norma ISO/IEC 25051 refere que a documentação de teste deve conter o plano de teste, a descrição do teste e os resultados do teste. Antes de se proceder aos testes é necessário descrever todas as funções do programa.

Cada função descrita deve ser objeto de pelo menos um caso de teste. Adicionalmente todos os procedimentos de instalação devem ser submetidos a casos de teste.

A descrição de cada caso de teste deve incluir um identificador único, o seu objetivo, dados de entrada e limites, os passos detalhados para a realização, o comportamento esperado do sistema, a saída esperada do caso de teste, os critérios para a interpretação do resultado e os critérios utilizados para decidir sobre um resultado positivo ou negativo do caso de teste.

Depois de identificado o objetivo do teste, é necessário identificar os critérios para a decisão sobre o resultado. Estes critérios devem estar em conformidade com a documentação do produto e a documentação do utilizador. No caso de resultados não conformes, deve existir um procedimento para repetir o ensaio das funções envolvidas e de quaisquer funções relacionadas, após a sua correção.

O procedimento de teste deve incluir a preparação do teste, as ações necessárias para iniciar e executar o teste, as ações necessárias para registar os resultados do teste, as condições e ações para parar e eventualmente reiniciar os testes. Estes procedimentos devem ser suficientemente detalhados para proporcionar repetibilidade e reprodutibilidade dos testes.

4. CASO DE ESTUDO

Como caso de estudo apresentamos um conjunto de testes necessários para a certificação do software CYPE e apresentamos uma proposta de automatização usando como base a aplicação Sikuli.

Antes de começar o teste ao software, é necessário analisar todas as funções da aplicação, para criar um plano de testes.

O plano de testes define para todas as funções da aplicação os testes a realizar. Os testes que atualmente compõem o plano de testes podem ser agrupados em três categorias distintas:

- Grupo A – Instalação do software: todos os procedimentos de instalação devem ser submetidos a casos de teste; no software CYPE existem inúmeras formas de instalação do programa, como Versão Profissional, Versão de avaliação, Versão Temporária, Versão After Hours, Versão Campus.

- Grupo B – Opções dos menus do software: identificar todos os menus, dentro de cada menu deve-se descrever todas as opções para que sejam contempladas em casos de teste e posteriormente sujeitas a teste.
- Grupo C – Opções de cálculo do software: deve-se analisar todos os cálculos e opções de cálculo que o programa considera e criar uma função para cada um, para posteriormente testar e verificar se os mesmos são realizados corretamente.

4.1. Automatização de Testes

Todos os testes da aplicação enquadram-se numa das três categorias: A, B, ou C. No exemplo do teste automático do tipo A foi criado um teste para instalar a versão profissional do software CYPE. Para a validação da versão profissional instalada é necessário no final da instalação abrir o programa CYPECAD e calcular a obra exemplo que está dentro do programa. Para evidenciar a realização do teste, foi criado e gravado um *screenshot* do teste. Em caso de erro do teste no ficheiro com a realização dos testes este estará a vermelho com o erro e a posição do mesmo.

Neste caso está-se a automatizar o teste para instalar a versão profissional do software. Na figura 1 pode-se visualizar em exemplo retirado do script da automatização do teste do grupo A.

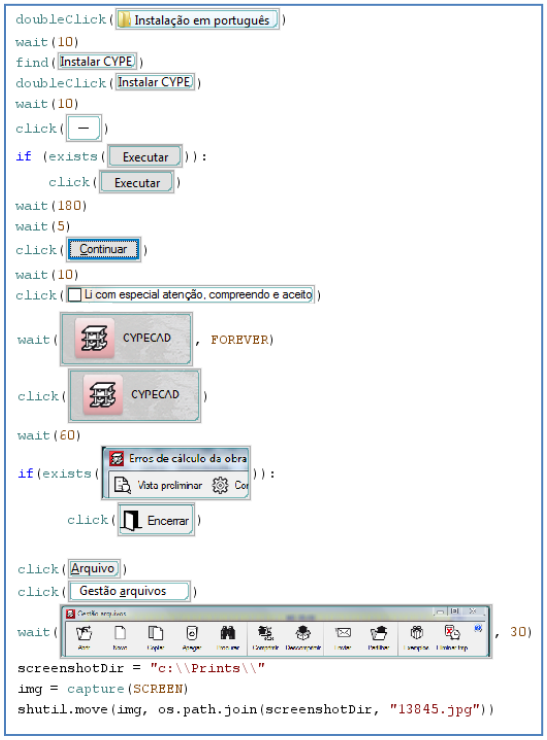

Fig. 1 - Exemplo de script de teste grupo A

Os scripts de testes dos tipos B e C seguem o mesmo tipo de ações descrito na Figura 1.

4.2. Validação de Testes

Neste modelo de automatização de testes a validação dos testes é realizada pelo técnico que anteriormente efetuava os testes manuais. A criação das evidências de teste é realizada através de *screenshots* retirados ao longo do teste, conforme a necessidade de evidenciar o teste. Para a criação da imagem é utilizada uma função que captura a imagem do ecrã e a grava numa determinada pasta.

Com este processo podemos garantir os vários passos realizados pelo teste e o validador poderá validar a "Conformidade" ou "Não conformidade do teste".

5. CONCLUSÃO

O caso de estudo apresentado permite concluir que é possível automatizar os testes de software necessários para a certificação de um software no âmbito da norma ISO/IEC 25051. A automatização de testes ao software CYPE veio ajudar a realização dos testes, em concreto na diminuição do tempo de realização dos testes, permitindo lançar mais atualizações certificadas ao longo do ano. Os testes automatizados são testes repetitivos ao longo da vida do software e terão poucas alterações ao longo desta, o que permite um baixo custo de manutenção dos mesmos.

O custo inicial da criação de testes automatizados é ainda significativo, pelo que o ganho é obtido pela reutilização do mesmo teste ao longo das várias versões do software. Quando se automatiza funções é necessário perceber se as mesmas terão alterações constantes, pois estas alterações poderão ser uma barreira para o sucesso da automatização das mesmas.

Em suma, a automatização de testes é necessária, mas deverá sempre ser medido o impacto, a reação pelos testadores e o que se deve ou não automatizar.

6. REFERÊNCIAS

Andrade, A. P., & Viana, P. (03 de Dezembro de 2012). Criação e Geração de Planos de Teste de Software. Obtido em 09 de Setembro de 2015, de IBM: http://www.ibm.com/developerworks/br/local/rational/criacao_geracao_planos_testes_software/

António Miguel, P. (2006). Gestão de Projectos de Software. (FCA - Editora Informática, Ed.).

Bartié, A. (2002). Garantia da qualidade de software. CAMPUS - RJ, 2002.

Dasso, A., & Funes, A. (2007). Verification, validation and testing in software engineering. Idea Group.

ISO. (2014). ISO/IEC 25051. Retrieved from http://www.iso.org/iso/catalogue_detail.htm?csnumber=61579

Jovanovic, I. (n.d.). Software Testing Methods and Techniques. The IPSI BgD Transactions on Internet Research (pp. 30–41).

Mariani, L., Pezzè, M., Riganelli, O., & Santoro, M. (2011). AutoBlackTest: a tool for automatic black-box testing. 2011 33rd International Conference on Software Engineering (ICSE), 1013–1015. doi:10.1145/1985793.1985979

Pressman, R. S. (2010). Software Engineering A Practitioner's Approach.

Sikuli. (15 de Novembro de 2015). Obtido de Sikuli : http://www.Sikuli.org/

Sirohi, N., & Parashar, A. (2013). Component Based System and Testing Techniques, 2(6), 2378–2383.

Tian, J. (2005). Software Quality Engineering Testing, Quality Assurance And Quantifiable Improvement.

TOP Informática, L. (2008). TOP Informática, Lda. Obtido de TOP Informática, Lda: http://www.topinformatica.pt/