Mapeamento de Critérios de Aceitação de Transparência de Software para Testes Automatizados

Bruno Luiz Gonçalves, Ernani César Dos Santos, Patrícia Vilain

Universidade Federal de Santa Catarina, Florianópolis SC, Brasil
brunolg9@gmail.com
ernani.santos@posgrad.ufsc.br
patricia.vilain@ufsc.br

Abstract. Quality requirements include software transparency, which defines how information should be presented and how the software should run and be developed. In order to ensure compliance with these requirements in the Brazilian Public Sector, the Law on Access to Information (LAI) was proposed, which establishes rules for the disclosure of information, demonstrating the need for transparency of processes and information in public organizations. In this context, we propose the creation of abstract acceptance test case templates in order to evaluate the fulfillment of requirements on public organizations' websites. To validate the work, automated tests were developed using Cucumber and Selenium tools. These tests were defined from the abstract templates and addressed sites that meet and do not meet the requirements being tested. The results show that the proposed templates can serve as acceptance tests to be used during the design of government websites

Keywords: Software Transparency, Acceptance Tests, Non-Functional Requirements, Test Case Templates.

1 Introdução

O crescimento do armazenamento de dados e o acesso direto às informações de várias fontes, por usuários internos e externos de organizações, têm ampliado a necessidade por informações de alta qualidade [1]. Nas organizações públicas, a sociedade, que é afetada pelas ações do estado, tem o direito de acesso às informações que forem necessárias para monitorar a implementação das políticas públicas e da qualidade do serviço público. Quanto maior a transparência pública, maior o conhecimento sobre o andamento da gestão e maior a eficiência dos gastos públicos [2].

De acordo com [3], para uma informação ser considerada transparente, as seguintes premissas devem ser cumpridas: *ser completa*, *objetiva*, *confiável*, *relevante*, *acessível* e de *fácil compreensão*. No Brasil, para garantir o acesso e a transparência das informações à população, foi sancionada a Lei de Acesso à Informação (LAI - Lei nº 12.527/2011). Esta lei regulamenta o direito constitucional de acesso às informações públicas. A LAI abrange os três Poderes da União, Estados, Distrito Federal e Muni-

cípios. O objetivo é que essas instituições implementem gestões transparentes, disponibilizando as informações para todos através da Internet [4].

Em [6] são apresentados vários requisitos não-funcionais de qualidade, associados à transparência de software, através de um *checklist* denominado *checkTrans*, voltado para organizações públicas. Todavia, apesar do *checkTrans* definir os requisitos não-funcionais associados à transparência de software, essa definição não assegura que um software atenda a esses requisitos.

Para tanto, neste trabalho propõe-se que estes requisitos não-funcionais de qualidade sejam identificados e descritos através de cenários de teste de aceitação. Foram selecionados alguns requisitos definidos no *checkTrans* e para estes requisitos, foram definidos cenários de testes de aceitação abstratos, isto é, *templates* de cenários de teste. Desta forma, outras organizações poderão utilizar estes *templates* para avaliar a transparência de seus produtos de software ou, ainda nos casos de terceirização do desenvolvimento do produto de software, estes *templates* de testes de aceitação poderão compor a descrição do objeto no instrumento convocatório (edital de licitação ou carta convite).

Este trabalho está organizado da seguinte forma, a Seção 2 aborda uma visão geral de alguns conceitos utilizados neste trabalho. Na Seção 3 são apresentados os requisitos de transparência de software que foram selecionados para serem mapeados para *templates* de testes de aceitação. A Seção 4 detalha os *templates* de testes de aceitação e na Seção 5 é discutida a validade dos *templates* definidos. Na Seção 6 são apresentados os trabalhos relacionados. Finalmente, as conclusões e trabalhos futuros são apresentados na Seção 7.

2 Visão Geral

2.1 Transparência de Software e *CheckTrans*

Um software é considerado transparente não apenas se as informações apresentadas aos usuários são transparentes, mas, também, se todo seu processo de engenharia de requisitos é transparente. Neste contexto, entende-se que a transparência de software deve ser considerada como um requisito não-funcional, que, por sua vez é representado por um conjunto de requisitos associados à qualidade do software [5].

A transparência de software é realizada através de 5 atributos: (i) *acessibilidade*, (ii) *usabilidade*, (iii) qualidade da informação – o software é *informativo*? – , (iv) *entendimento*, (v) *auditabilidade*. Com o intuito de definir os requisitos de transparência de software, [3] associou a estes 5 atributos de transparência um conjunto de 33 requisitos não-funcionais.

O *checkTrans* é um conjunto de ações organizado em formato de *checklist*. O objetivo deste *checklist* é apresentar as ações e exemplos para realização dos requisitos não-funcionais de qualidade associados à transparência de software e definidos por [3]. A seleção das ações foi baseada no Catálogo de Transparência do Grupo ER - PUC-RIO [6].

Para cada ação sugerida no *checkTrans*, são associados exemplos com o intuito de auxiliar no entendimento do que deve ser feito para atingi-la [6]. Na Tabela 1 são

apresentadas duas ações e exemplos do requisito de *Publicidade* do atributo de *Acessibilidade*. A tabela completa do *checkTrans* está disponível na URL http://leb.inf.ufsc.br/index.php/home/cibse2020/tabela1.

Tabela 1. Ações e Exemplos para o requisitos de Publicidade do atributo de Acessibilidade

Ações	Exemplos
Fornecer informações sobre o conjunto de dados do sistema.	Catálogo de Dados.
Definir as facilidades de navegação, inclusive, para pessoas com deficiência.	Acesso ao computador sem mouse. Padrões de acessibilidade (e-MAG).

2.2 Testes de Aceitação e BDD

Os testes de aceitação fazem parte do processo de validação do software. Estes testes são realizados pelo cliente para que ele decida se o software é bom o suficiente para ser implantado e usado em seu ambiente operacional [7]. Os testes de aceitação são definidos a partir das necessidades dos usuários identificadas durante o processo de elicitação e documentação dos requisitos. Eles podem ser especificados no início do processo de desenvolvimento de um software ou no final, na etapa de validação. Na entrega do produto, os testes são executados para que o cliente decida se o sistema é aceitável ou não [7]. Existem diversas técnicas para o desenvolvimento e execução de testes de aceitação. O *Behavior-driven development* (BDD) é um exemplo de técnicas de teste de aceitação amplamente utilizadas na indústria e academia [8]. No processo de análise do software, o BDD utiliza *templates* predefinidos para a criação de histórias de usuário e cenários de teste. Entre as ferramentas mais populares que utilizam BDD está o *Cucumber* [9].

2.3 Testes Automatizados

Testes automatizados são trechos de código-fonte que exercitam as funcionalidades de um software e realizam asserções automáticas com o resultado do processamento do sistema em teste (*System Under Test – SUT).* Ao contrário dos testes manuais, a execução dos testes automatizados, por ter pouca ou nenhuma dependência da intervenção humana, é mais efetiva tanto na verificação quanto na validação do software. Entretanto, os testes automatizados requerem implementação, manutenção e gerenciamento da execução [10]. Dentre as diversas ferramentas existentes utilizadas para o desenvolvimento de testes automatizados está o Selenium, uma ferramenta para criação e execução de testes automatizados para aplicações Web.

3 Requisitos de Transparência de Software Selecionados

Nem todos os requisitos modelados no *checkTrans* conseguem ser mapeados para testes de aceitação automatizados. As seguintes restrições foram consideradas na sele-

ção de atributos do *checkTrans*: (1) os testes devem ser executados sob a perspectiva do usuário, pois do contrário, os próprios testes não seriam transparentes. Sendo assim, serão definidos *templates* de teste somente para ações e exemplos de atributos que podem ser testados a partir da máquina do usuário; (2) embora não seja desejável que testes de aceitação sejam realizados sobre a GUI, para evitar o acoplamento e dependência, esta é a interface mais disponível para realização de testes sob a perspectiva do usuário e de forma transparente. Portanto, os testes devem ser executados através de um navegador Web em um software disponível na Internet com interface baseada em elementos DOM; (3) ações e exemplos que buscam validar atributos de transparência relacionados a ações do processo de desenvolvimento de software serão desconsiderados; (4) os *templates* de teste para as ações e exemplos de um atributo devem ser automatizáveis. A partir destas restrições foram extraídos do *checkTrans* os atributos, junto com suas respectivas ações e exemplos, que podem ter seus *templates* de teste de aceitação especificados. A Tabela 2 lista todas as ações dos atributos identificados como adequados para serem mapeados para *templates* de testes de aceitação automatizados.

Tabela 2. Atributos selecionados do *checkTrans*

Atributos (Requisitos)	**Ação sugerida**	**Exemplos**
Portabilidade	Especificar as diferentes tecnologias onde o software poderá ser utilizado.	(i) Navegadores Internet Explorer e Chrome; (ii) Padrões de desenvolvimento W3C.
Publicidade	Definir as facilidades de navegação, inclusive, para pessoas com deficiência.	(i) Acesso ao computador sem mouse; (ii) Padrões de acessibilidade (e-MAG).
	Definir e fornecer alternativas para disponibilização da informação.	Informações em figuras, vídeos e áudios.
Amigabilidade	Definir os elementos de informação que são apresentados em cada contexto.	Ferramenta de busca presente em todas as páginas, mapa do sítio ou sistema, índice de palavras.
Simplicidade	Especificar os elementos que ajudam a simplificar a interação do usuário.	Formulários simplificados, disponibilizar ajuda dentro do próprio sistema.
	Definir como serão destacados os elementos essenciais.	(i) Conteúdos mais importantes, páginas, seções ou serviços mais utilizados no início da página; (ii) Campos obrigatórios ou opcionais indicados.

4 Mapeamento dos Critérios de Aceitação

Para mapear os critérios de aceitação de requisitos não-funcionais de qualidade para testes de aceitação foi utilizado como artefato cenários escritos na linguagem *Gherkin*, linguagem utilizada pelo BDD. A linguagem *Gherkin* foi escolhida por ser muito próxima de uma linguagem natural, o que facilita sua compreensão por até mesmo usuários não-técnicos [8]. Além disso, os cenários de teste escritos em *Gherkin* são

compatíveis com uma séria de *frameworks* que os integram com outras ferramentas de testes automatizados.

4.1 *Templates* dos Cenários de Testes

Nesta seção serão apresentados alguns *templates* de testes de aceitação para validação do atributo de Portabilidade. Os *templates* dos demais atributos selecionados estão disponíveis em http://leb.inf.ufsc.br/index.php/home/cibse2020/templates.

Atributo Portabilidade. Validar se a página Web mostra as mesmas informações em diferentes navegadores e se foi implementada de acordo com os padrões da W3C.

Feature. *Portabilidade*

Eu, como usuário do Website <site governamental>

Quero ter acesso a informações transparentes de qualquer lugar

Então poderei acessar ao Website <site governamental> de diversas plataformas, sistemas operacionais ou navegadores de Internet.

Cenário 1. Acessar <site governamental> através de diferentes navegadores de Internet no sistema operacional <sistema operacional>

Dado que estou utilizando um computador <sistema operacional>

Quando eu acesso a página <página> do Website <site governamental> utilizando o navegador de Internet <navegador A>

E acesso esta mesma página utilizando o navegador de Internet <navegador B>.

Então as informações contidas na página são disponibilizadas da mesma forma tanto com <navegador A> quanto no <navegador B>.

Cenário 2. Acessar Website <site governamental> através de uma mesmo navegador em diferentes plataformas

Dado que estou acessando a página <página> do Website <site governamental> utilizando o navegador de Internet <navegador> em um <dispositivo A> <sistema operacional A>

Quando eu acesso esta mesma página utilizando o mesmo navegador em um <dispositivo B> <sistema operacional B>

Então as informações contidas na página são disponibilizadas da mesma forma tanto no <dispositivo A> quanto no <dispositivo B>.

5 Validação dos Testes de Aceitação

Para avaliar a validade dos *templates* de testes definidos na subseção 4.1, foram executados em sites do governo casos de teste criados a partir destes *templates*. Todavia, antes da execução automatizada de cada um dos casos de teste, foram identificados, através de uma validação manual, sites governamentais que atendem e que não atendem aos atributos de transparência associados aos *templates* criados. Através deste procedimento foi possível avaliar se os *templates* cumprem com o esperado, isto é, não apontam falhas em sites cujo atributo de transparência testado é satisfeito e apontam falhas quando o atributo não é satisfeito.

5.1 Especificação dos Testes de Aceitação Concretos

Os cenários de teste foram adaptados para a linguagem *Gherkin*, substituindo as *tags* do *template* por valores reais, assim como no exemplo a seguir, aonde o cenário é definido a partir do *template*.

Template. *Template* de cenário de teste para validação do atributo de Publicidade. Permitir a navegação no site <site governamental> sem utilizar o mouse

Dado que eu estou utilizando um computador <sistema operacional> sem mouse

Quando eu acesso a página <página> do Website <site governamental> utilizando o navegador de Internet <navegador de internet>

E eu interajo utilizando apenas as teclas TAB e ENTER do teclado

Então eu consigo acessar a página <pagina> do site

Cenário. Navegação no site "https://www.fernao.sp.gov.br" sem utilizar o mouse

Dado que eu estou utilizando um computador Linux sem mouse

Quando eu acesso a página "/" do Website "https://www.fernao.sp.gov.br" utilizando o navegador de Internet Firefox

E eu interajo utilizando apenas as teclas TAB e ENTER do teclado

Então eu consigo acessar a página de contato do site

Com as *tags* dos *templates* devidamente substituídas por valores reais, para executar os testes de aceitação dos atributos de transparência é necessário conectar os testes ao site governamental a ser testado (SUT). Esta conexão é realizada através do código de cola.

O código de cola é uma classe *Java* denominada *Feature* gerada automaticamente pela próprio *Cucumber*. Para cada passo do teste é gerado um método nesta classe. Então, para fazer o teste funcionar, é necessário implementar este método. A seguir é apresentado um exemplo de método escrito em *Java* que liga um passo do teste de aceitação ao SUT.

```
@Quando("^eu acesso a pagina \"([^\"]*)\" do site \"([^\"]*)\" utilizando o navegador web Firefox$")
public void eu_acesso_a_pagina_do_site_utilizado_o_navegador_web_Firefox(String pagina, String site) {
        this.firefoxDriver = DriverFactory.getDriver("firefox");
        this.firefoxDriver.get(site + pagina);}
```

5.2 Resultados

Para os 4 atributos de transparência foram mapeados 8 *templates* de testes de aceitação. Todos estes *templates* foram automatizados para sites que cumpriam os requisitos de transparência de software e também para sites que não cumpriam estes requisitos, totalizando 16 cenários de teste automatizados. É importante ressaltar que cada página possui um DOM diferente e é através deste documento que o *Selenium* encontra os componentes HTML utilizados para executar os testes. Apesar de ser possível a generalização de cenários para diferentes páginas Web, a implementação de cada teste automatizado é diferente. Por este motivo, cada cenário foi avaliado em apenas uma página que atendia ao requisito e uma página que não atendia.

A Tabela 3 apresenta um resumo do resultado da execução dos testes.

Tabela 3. Ações e Exemplos para o requisitos de Publicidade do atributo de Acessibilidade

Atributo	Cenário	Site	Resultado esperado	Resultado obtido
Portabilidade	Acessar site através de diferentes navegadores Web	A	Sucesso	Sucesso
		B	Fracasso	Fracasso
Publicidade	Navegar no *site* sem utilizar o *mouse*	C	Sucesso	Sucesso
		D	Fracasso	Fracasso
	Alternativas para imagens ou vídeos	E	Sucesso	Sucesso
		F	Fracasso	Fracasso
Amigabilidade	Ferramenta de busca presente em todas as páginas	G	Sucesso	Sucesso
		H	Fracasso	Fracasso
	Existência do mapa do site	I	Sucesso	Sucesso
		J	Fracasso	Fracasso
Simplicidade	Informações que orientem no preenchimento de formulários	K	Sucesso	Sucesso
		L	Fracasso	Fracasso
	Elemento essencial destacado na página inicial do site	M	Sucesso	Sucesso
		N	Fracasso	Fracasso
	Indicação de campos obrigatórios ou opcionais	O	Sucesso	Sucesso
		P	Fracasso	Fracasso

(A) estrutura.ufsc.br/servicos-gratuitos/ (B) sectur.ma.gov.br/ (C) http://ufsc.br/mapa-do-site/ (D) http://fernao.sp.gov.br/ (E) http://noticias.ufsc.br/2019/10/pesquisa-da-ufsc-revela-nova-perspectiva-para-diagnostico-e-tratamento-da- http://tuberculose/ (F) desterrodeentrerios.mg.gov.br/noticias/?noticia=25 (G) prae.ufsc.br/ (H) alterosa.mg.gov.br/ (I) consulta.tesouro.fazenda.gov.br/gru_novosite/gru_simples.asp (J) contas.tcu.gov.br/debito/Web/Debito/CalculoDeDebito.faces (K) ufsc.br/ (L) trabiju.sp.gov.br/ (M) tjsc.jus.br/ (N) tjrs.jus.br (O) otrs.setic.ufsc.br/otrs/public.pl?Action=NewTicketWizardPublic (P) contas.tcu.gov.br/debito/Web/Debito/CalculoDeDebito.faces

Os resultados obtidos mostram que os casos de testes automatizados derivados dos *templates* definidos não apresentaram falsos positivos ou falsos negativos. Acredita-se, que as organizações públicas possam fazer uso desses *templates* para validar, de forma automatizada, que os produtos de software, desenvolvidos *in company* ou por empresas terceirizadas, atendem aos requisitos de transparência selecionados.

6 Conclusão

A proposta deste trabalho foi mapear critérios de aceitação de 4 atributos de transparência de software por meio de testes de aceitação abstratos (*templates*). A definição dos critérios de aceitação foi realizada com base nas ações e exemplos propostos no *checkTrans*. Os *templates* definidos foram especificados como cenários de testes de aceitação concretos e executados em sites do governo. Não foram detectados falsos positos ou negativos durante a validação dos *templates*, o que aponta que estão prontos para serem utilizados em ambiente produtivo.

Em [11], os autores adaptaram o *checkTrans* em um novo *checklist*, selecionando ações que podem ser aplicadas em sites existentes, entretanto as ações não são mapeadas para testes de aceitação automatizados. Outros dois trabalhos propuseram o uso de testes de aceitação automatizados para requisitos não-funcionais, porém, o foco não é a transparência de software. Em [12], os autores desenvolveram testes de aceitação para garantir acessibilidade, simulando a navegação em aplicações web utilizando o teclado. Em [13], os autores implementaram testes automatizados para avaliar um software de controle de qualidade através da injeção de código de testes no código em produção, o que a diferencia da proposta do presente trabalho.

O código fonte da implementação dos testes automatizados está disponível no endereço https://github.com/brunolg6/testesTransparencia/.

Referências

1. LEE, Y. W. et al. AIMQ: a methodology for information quality assessment. Information & management, v. 40, n. 2, p. 133-146, 2002.
2. MELO, D. A.. Transparência da informação pública: uma avaliação de sítios eletrônicos de universidades federais brasileiras. Dissertação (Pós-Graduação em Administração Pública) - Universidade Federal de Goiás, Goiânia, 2019.
3. CAPPELLI, C. Uma Abordagem para Transparência em Processos Organizacionais Utilizando Aspectos. 2009. 328p. Tese de doutorado (Pós-Graduação em Informática), Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2009.
4. BRASIL. Acesso federal à informação. Disponível em: http://www.acessoainformacao.gov.br. Último acesso em: 06/12/2019.
5. CAPPELLI, C., LEITE, J. C. S. P. Transparência de Processos Organizacionais. Universidade Federal Fluminense. II Simpósio Internacional de Transparência nos Negócios. 2008.
6. MACEDO, F. F. de.. Transparência de software como apoio à publicidade da administração pública. Dissertação (Pós-Graduação em Ciência da Computação) - Universidade Federal de Santa Catarina, Florianópolis, 2014.
7. SOMMERVILLE, I.. Software Engineering. Pearson, 9ª Edição, São Paulo, 2011.
8. DOS SANTOS, E. C., VILAIN, P. Automated Acceptance Tests as Software Requirements: An Experiment to Compare the Applicability of Fit Tables and Gherkin Language. In International Conference on Agile Software Development (pp. 104-119).
9. SOLIS, C.; WANG, X.. A study of the characteristics of behaviour driven development. In: 2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications. p. 383-387, IEEE, 2011.
10. CHEQUE, P.. Padrões de testes automatizados. Dissertação (Mestrado em Ciência da Computação) - Universidade de São Paulo, São Paulo, 2011.
11. FORTE, F. B., VILAIN, P., MACEDO, F. F. Adaptação de um Checklist para Análise de Transparência de Software em Sites. Anais do XI Simpósio Brasileiro de Sistemas de Informação (SBSI 2015), 2015. p.355-362.
12. WATANABE, W. M.; FORTES, R. P. M.; DIAS, A. L.. Using acceptance tests to validate accessibility requirements in RIA. In: Proceedings of the International Cross-Disciplinary Conference on Web Accessibility. ACM, p. 15, 2012.
13. METSA et al.. Testing Non-Functional Requirements with Aspects: An Industrial Case Study. In: Seventh International Conference on Quality Software (QSIC 2007). 2007.