Documentação Dirigida por Testes

Ismayle de Sousa Santos[1], Pedro de Alcântara dos Santos Neto[1], Raimundo Santos Moura[1], André Castelo Branco Soares[1]

[1]Departamento de Informática e Estatística - Universidade Federal do Piauí (UFPI) – Teresina – PI – Brasil

{ismayle,pasn,raimoura,andre.soares}@ufpi.edu.br,

Resumo. *A geração de documentos durante o desenvolvimento de software é uma tarefa onerosa e não indicada pela maioria dos processos ágeis. Manter documentos atualizados é uma tarefa árdua, uma vez que é necessário refletir cada mudança do código nos artefatos relacionados. Neste artigo é apresentada uma alternativa à criação manual da Documentação de Usuário e da Especificação de Testes, baseada na geração semi-automatizada a partir de scripts de testes funcionais, denominada de Documentação Dirigida por Testes. No trabalho é apresentado um estudo experimental que mostra que essa idéia pode reduzir o esforço na criação dos documentos, mantendo o mesmo nível de qualidade da geração manual.*

Abstract. *The generation of documents during software development is an expensive task and not addressed by several agile process. The task to synchronize the documentation and source code is a hard task. In this paper is presented an alternative to create User Document and Tests Specification by reusing functional tests. The approach is called Test Driven Documentation. In this work is also presented an experiment showing that this idea can decrease the effort spent during the document creation.*

1. Introdução

Durante o desenvolvimento de *software*, vários documentos podem ser criados para registrar informações relacionadas ao acompanhamento do projeto e para servirem de base para as transformações existentes no ciclo de vida do *software*. Dentre tais documentos, há a Documentação de Usuário (DU) e a Especificação de Testes (ET). Esses documentos normalmente auxiliam os usuários finais do produto (DU) e os testadores (ET), além de serem exigidos como forma de evidenciar a prática de alguns processos chaves em modelos de maturidade, como por exemplo, o MPS.Br [SOFTEX 2009].

A Documentação de Usuário de *Software* é um material impresso ou eletrônico que provê informações para os usuários de um *software* [IEEE 2001]. Essa documentação é útil para o usuário entender como manusear o produto e para dirimir dúvidas que podem surgir ao longo do tempo. Como o principal objetivo desse artefato é orientar o usuário final na execução de tarefas, ele é muitas vezes tão necessário para o sucesso do *software* quanto o código executável [Padua 2003]. Já a Especificação de Testes tem por objetivo detalhar como os testes deverão ser implementados [Padua 2003]. A função desse documento é descrever os casos e procedimentos de testes que irão constituir a bateria de testes a ser realizada.

De um modo geral os métodos ágeis não prescrevem a criação de documentos, embora enfatizem a criação de testes. Corroborando com isso, ainda existem muitos profissionais que evitam qualquer tipo de documentação por acharem uma tarefa cansativa e repetitiva. Embora haja divergência sobre o assunto, é certo que a criação de documentos e sua manutenção atualizada acarretam em custo e esforço nos projetos. Muitas das vezes os documentos se tornam obsoletos pouco tempo após sua criação devido à incapacidade de atualização contínua.

O tempo gasto para se gerar a documentação exigida em um projeto varia muito, mas em geral, corresponde a uma porção significativa do tempo total. Jones (2007), por exemplo, sugere que em projetos de sistemas *web*, o custo da Documentação de Usuário é de cerca de 10% do custo total do projeto, o que segundo ele, corresponde a um terço do tempo gasto na codificação desse tipo de sistema.

Dessa forma, é interessante a pesquisa por meios de automatizar essa tarefa, preferencialmente reutilizando artefatos já comumente criados pelos desenvolvedores. Este artigo apresenta um método ("método" na acepção de [Houaiss e Villar 2001]), que corresponde ao conjunto de passos para geração de documentos a partir do *script* de teste funcional, e o protótipo de uma ferramenta para a geração semi-automática da Documentação de Usuário e da Especificação de Testes Funcionais. A abordagem proposta, denominada Documentação Dirigida por Testes (DDT), consiste em utilizar *scripts*[1] de testes funcionais como insumo para a geração de documentos.

Como as Especificações de Testes detalham como devem ser feitos os testes e, portanto, deveriam ser criadas antes dos testes propriamente ditos, parece ser uma incoerência gerar tal documento a partir de testes funcionais. Contudo, geralmente documentar é uma atividade cansativa e com isso os documentos ficam rapidamente desatualizados durante o desenvolvimento de um *software*. Fazer o caminho inverso (testes gerando documentos) pode ser uma saída viável para manter tais documentos sempre atualizados. A documentação atualizada, por sua vez, facilita o ingresso de novos colaboradores a um projeto, pois ela auxilia o entendimento das partes descritas. Além disso, conforme já ressaltado, alguns modelos de maturidade exigem a comprovação de certas práticas recomendadas via documentação associada.

O restante deste artigo está organizado da seguinte forma: a Seção 2 apresenta alguns trabalhos relacionados; a Seção 3 descreve os principais conceitos relacionados ao método; a Seção 4 descreve em linhas gerais a geração da Documentação de Usuário e da Especificação de Testes a partir de testes funcionais; a Seção 5 apresenta a ferramenta protótipo desenvolvida; a Seção 6 contém a descrição do experimento realizado para validar a abordagem proposta; e a Seção 7 apresenta as conclusões e os trabalhos futuros.

2. Trabalhos Relacionados

Como a criação de documentos é uma atividade com custos significativos, já existem na literatura trabalhos relacionados à geração de documentos. Contudo, é difícil encontrar trabalhos direcionados à Documentação de Usuário e a Especificação de Testes. A seguir são apresentados alguns trabalhos relacionados à geração automática de documentos voltados para o usuário final.

[1] Sequencias de ações que serão executadas durante os testes

Jungmayr e Stumpe (1998) propuseram a utilização de modelos de uso na geração automática de Documentação de Usuário. Estes modelos foram escolhidos porque descrevem a interação de usuários com o *software* contendo, portanto, informações relevantes para gerar este documento. Contudo, a criação de modelos de uso exige um esforço significativo [Jungmayr e Stumpe 1998].

Paris e Linden (1996) apresentam o projeto DRAFTER, no qual é apresentado o uso de ferramentas que permitem a construção de modelos de interface utilizáveis para a geração da Documentação de Usuário.

Johnson descreve em [Johnson 1994] uma ferramenta denominada I-Doc que automatiza o processo de geração do *help* de usuário e da documentação do *software*. A ferramenta faz isso extraindo algumas informações diretamente do código e a partir da captura das decisões de *design* definidas com anotações no código.

Reiter, Mellish e Levine (1992) desenvolveram o projeto IDAS (*Intelligent Documentation Advisory System*). O projeto tem como objetivo gerar documentação e mensagens de ajuda a partir do domínio de uma base de conhecimento e utilizando técnicas de geração baseada em linguagem natural.

Moriyon, Szekely e Neches (1994) descrevem um sistema que utiliza o modelo de interface para gerar uma ajuda *on-line*. As mensagens de ajuda geradas indicam as ações que um usuário pode executar em uma situação particular e quais são os resultados esperados.

Conforme pode ser observado, existem diversos trabalhos relacionados à geração de Documentação de Usuário ou *help on-line*. Entretanto, tais trabalhos utilizam-se de modelos ou representações formais do sistema descritos em artefatos normalmente não prescritos pela maioria dos processos de *software*. Neste trabalho é proposto a geração da Documentação de Usuário e da Especificação de Teste de *Software* utilizando um artefato comum em quase todos os processos, que são os testes funcionais. Além disso, nos trabalhos citados anteriormente não é feita uma avaliação do custo e da qualidade dessa geração. Neste trabalho apresentamos uma avaliação preliminar do que foi desenvolvido.

3. Conceitos Relacionados

3.1. Testes Funcionais

O teste de *software* pode ser definido como a verificação dinâmica do funcionamento de um programa, utilizando um conjunto finito de casos de teste, adequadamente escolhido dentro de um domínio de execução, em geral infinito, contra seu comportamento esperado [IEEE 2004]. Assim, um teste, de maneira geral, envolve a execução de um programa com a aplicação de certos dados de entrada, examinando suas saídas, verificando se elas combinam com o esperado e estabelecido nas suas especificações.

Dentre os tipos de testes existentes, há o teste funcional que é um teste para avaliar se o comportamento observado está em conformidade com as especificações [IEEE 2004]. O teste funcional é feito para verificar, por exemplo, se o cadastro de um

novo usuário em um sítio *web* funciona conforme o esperado. A Figura 1 apresenta um exemplo de um *script* de teste funcional feito com a ferramenta Selenium IDE[2].

O *script* de teste é desenvolvido utilizando-se os comandos da ferramenta. No *script* ilustrado temos: "*open*", usado para abrir uma nova página; "*type*" para inserir valores em campos da página, possuindo como atributos o campo e o valor a ser inserido; "*click*" que executa um clique em um botão ou *hiperlink*; e "*assertTextPresent*" para verificar a presença de um texto na tela.

```
<tr>
    <td>open</td>
    <td>http://www.ufpi.br/admin/index.php</td>
    <td></td>
</tr>
<tr>
    <td>type</td>
    <td>login</td>
    <td>admin</td>
</tr>
<tr>
    <td>type</td>
    <td>senha</td>
    <td>admin2009</td>
</tr>
<tr>
    <td>click</td>
    <td>Submit</td>
    <td></td>
</tr>
<tr>
    <td>assertTextPresent</td>
    <td>Sair</td>
    <td></td>
</tr>
```

Figura 1: Exemplo de script de teste funcional.

Dessa forma, o *script* da Figura 1 corresponde a um teste funcional no qual uma página (*www.ufpi.br/admin/index.php*) é acessada, em seguida são inseridos valores nos campos login ("*admin"*) e senha ("*admin2009"*) e após clicar em um botão (*Submit*) é feita a verificação da presença de um determinado texto (*Sair*) na página.

3.2. A Documentação de Usuário e a Especificação de Teste

Como mencionado anteriormente, a Documentação de Usuário é muitas vezes fundamental para o sucesso de um *software*, pois um usuário incapaz de executar as ações desejadas fica frustado e com isso passa a ter uma imagem negativa do *software*. Neste trabalho é utilizada uma estrutura para a DU proposta pelo processo Praxis [Padua 2003] que é composta por diversas partes, incluindo detalhes de como executar as funções do produto, detalhando pré-requisitos, preparação, entradas, saídas, condições de erro etc. Dentre alguns formatos analisados, constatou-se que a organização sugerida pelo Praxis era a mais apropriada por ser bem conhecida e bastante utilizada. Cabe ressaltar que não foi encontrado um padrão mundial com uma estrutura básica pré-definida, tal qual existe para a Especificação de Teste [IEEE 1998].

[2] http://seleniumhq.org/

Como existem informações da Documentação de Usuário que não podem ser obtidas diretamente a partir da análise de *scripts* de testes, como por exemplo, o nome do produto, parte do documento não pode ser gerado de forma totalmente automatizada. Contudo, como a maior parte deste documento está na orientação de execução de tarefas e essas descrições podem ser extraídas dos testes funcionais, pode-se ter uma redução significativa do tempo gasto para gerar esse documento. Em dois projetos reais analisados, constatou-se que essa parte manual corresponde a apenas 15%-20% do total da DU, enquanto que a parte automatizável via testes funcionais corresponde a 80%-85% do documento.

A estrutura do documento de Especificação dos Testes também segue o formato especificado pelo processo Praxis [Padua 2003], que por sua vez é aderente às prescrições do IEEE [IEEE 1998]. Esse documento tem como principal parte a descrição detalhada dos procedimentos e casos de testes.

É importante mencionar que, assim como a Documentação de Usuário, não é possível gerar todos os itens da Especificação de Testes a partir de *scripts* de testes funcionais. Uma pequena parte do documento deve ser definida manualmente. A análise utilizando os mesmos projetos reais revelou que de 5%-10% da especificação não pode ser atualmente automatizável por completo.

4. Documentação Dirigida por Testes

O termo "Documentação Dirigida por Testes" se refere à geração dos documentos presentes durante o desenvolvimento de *software* utilizando como insumo *scripts* de testes de *software*. Inicialmente o foco do trabalho foi em cima da Documentação de Usuário e da Especificação de Testes, porém diante das pesquisas que foram realizadas, acredita-se que é possível gerar outros tipos de documentos a partir de *scripts* de testes de *softwares.* Um relatório de mudanças no *software*, por exemplo, poderia ser gerado através da análise dos *scripts* de testes funcionais feitos antes e depois do *software* ser alterado.

Para se gerar um artefato a partir de outro, o primeiro passo é a extração das informações necessárias do artefato usado como insumo. No nosso contexto, isso equivale a extrair dos *scripts* de testes funcionais as informações úteis para se gerar a Documentação do Usuário e a Especificação de Teste. Para isso, valeu-se bastante de dois conceitos essenciais relacionados ao teste: casos de teste e procedimento de teste.

Um caso de teste pode ser resumido como a especificação das entradas, saídas esperadas e demais restrições relacionadas a um teste. Já um procedimento de teste é o conjunto de ações para a execução de um teste.

Portanto, para se obter as informações necessárias a geração da Documentação do Usuário e da Especificação de Testes, é preciso analisar o *script,* identificar os comandos usados e o que eles representam. No caso da ferramenta Selenium, por exemplo, a partir do comando "*type*" podemos inferir as entradas utilizadas e o comando "assert" indica a saída esperada. Com relação a procedimentos de teste, os comandos "*type*", "*open*" e "*click*" indicam boa parte das ações integrantes na execução de um teste.

O resultado da extração das informações do *script* ilustrado na Figura 1 é exibido na Figura 2. Observa-se que no *script* ilustrado temos um procedimento de teste

cujas ações são: 1 – Acessar a página "*http://www.ufpi.br/admin/index.php*"; 2 – Digitar um valor no campo "*login*"; 3 – Digitar um valor no campo "*senha*"; 4 - Clicar no comando "*Submit*". Além disso, temos no *script* um caso de teste que utiliza este procedimento de teste descrito acima. As entradas deste caso de teste são "*admin"* (no campo "*login"*) e "*admin2009"* (no campo "*senha*"), e como saída esperada tem-se a presença do texto "*Sair*" na página visualizada.

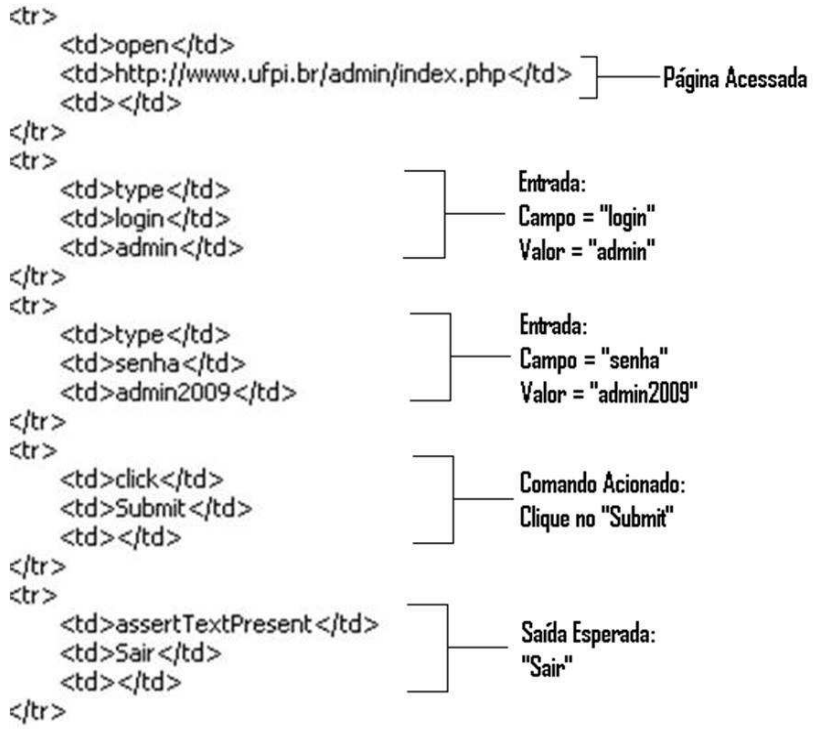

Figura 2: Análise simplificada de um script de teste funcional.

Com as informações extraídas do teste funcional, o próximo passo para a geração de documentos a partir de *scripts* de testes funcionais corresponde na indicação do que exatamente será documentado. Isso porque um teste funcional poderia estar utilizando diferentes funções do sistema sob teste. Assim, é preciso especificar o que será documentado. Com relação a Documentação de Usuário isso corresponde na indicação explícita de quais funções serão documentadas em quais partes do documento. A documentação das funções pode ser feita indicando quais os procedimentos e casos de testes associados a cada funcionalidade do sistema.

Observa-se que os procedimentos de testes são úteis para documentar as ações que devem ser executadas e os casos de testes são úteis para extração da saída esperada ao executar as ações que foram documentadas. Além disso, com base nas saídas esperadas dos casos de testes feitos para os caminhos "infelizes" (relacionados a existência de erros ou condições excepcionais), podem ser retiradas as mensagens de erro que precisam ser documentadas na Documentação de Usuário.

Já com relação a Especificação de Teste, é preciso agrupar os casos e procedimentos de testes de acordo com o caso de uso com o qual eles estão relacionados. Assim, podemos compor as seções deste documento através da descrição desses casos e procedimentos de testes extraídos do *script* de teste funcional.

Por fim, existem informações que não podem ser extraídas de um *script* de teste funcional, como por exemplo, o objetivo da funcionalidade, no caso da Documentação de Usuário, e o objetivo dos procedimentos descritos, no caso da Especificação de Teste. Logo, estas informações devem ser fornecidas para que possam junto com as informações já extraídas dos testes formar o documento que será gerado.

5. A ferramenta Test2Doc

5.1. Arquitetura

Foi desenvolvido um protótipo de ferramenta para validar a idéia proposta neste trabalho. Esse protótipo é denominado Test2Doc, tendo sido construído com a tecnologia J2SE[3]. A partir do uso da ferramenta é possível gerar Documentação de Usuário e Especificação de Testes fornecendo como insumo *scripts* de testes funcionais e algumas informações que não podem ser extraídas diretamente desses *scripts*.

A arquitetura da ferramenta é exibida na Figura 3. Observa-se que a ferramenta é composta de dois módulos: o Extractor e o DocGenerator. O Extractor é a parte responsável por extrair dos *scripts* de testes funcionais as informações necessárias à geração dos documentos. Após extrair essas informações é gerada uma representação abstrata do teste funcional, que é persistida em um arquivo no formato XML[4].

O outro módulo da ferramenta, o DocGenerator, é responsável por utilizar as informações extraídas pelo Extractor para gerar a Documentação de Usuário e a Especificação de Testes. Os documentos são gerados no formato *Rich Text Format* (RTF) a partir do uso da biblioteca iText[5], a qual permite a geração de arquivos nos formatos *Portable Document Format* (PDF) e RTF.

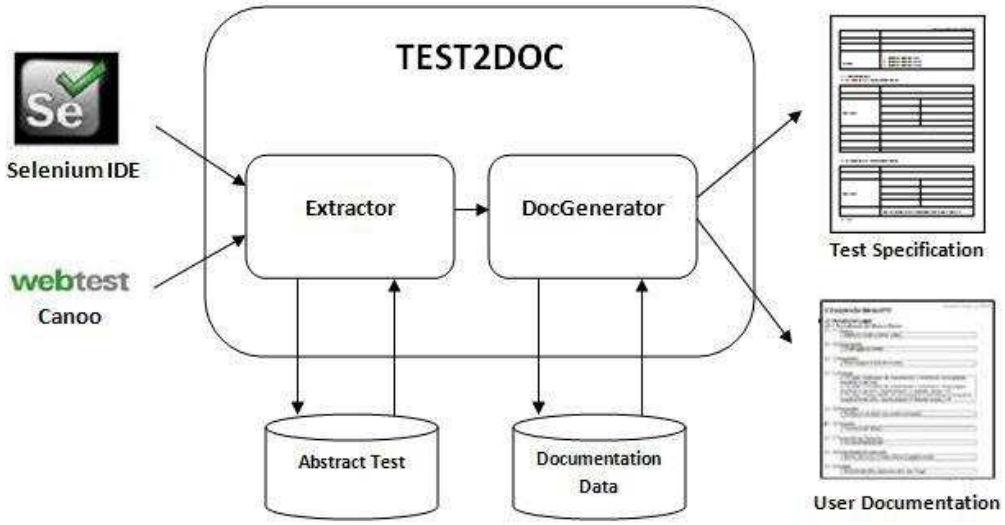

Figura 3: Arquitetura da ferramenta implementada.

[3] Java 2 Standard Edition
[4] Acrônimo de eXtensible Markup Language
[5] http://itextpdf.com/

Até o presente momento, o protótipo construído da Test2Doc trabalha somente com *scripts* feitos com o Selenium IDE e com o Canoo Web Teste[6]. Entretanto, ele foi implementado de forma a facilitar sua extensão para outras ferramentas de testes funcionais.

5.2. Questões de Implementação

Conforme já mencionado anteriormente, o Extractor é a parte da ferramenta responsável por extrair os casos de testes do *script* de teste funcional, gerando assim uma representação abstrata do mesmo. Como cada ferramenta de teste funcional possui seu conjunto próprio de comandos, é necessário construir um Extractor específico para cada uma delas. Com base nessa necessidade, o Test2Doc foi construído de forma a facilitar a inclusão de extratores, a partir da existência de uma classe abstrata genérica, facilmente extensível para a incorporação de novos extratores. Essa classe, denominada BaseExtractor, implementa algumas funções comuns a todos os extratores, como por exemplo, o tratamento do arquivo recebido como entrada e a montagem da bateria de testes. Existem alguns métodos abstratos que devem ser implementados para a construção de outros extratores, mas sua documentação torna simples essa tarefa. O diagrama das classes utilizado para representar o teste funcional é exibido na Figura 4.

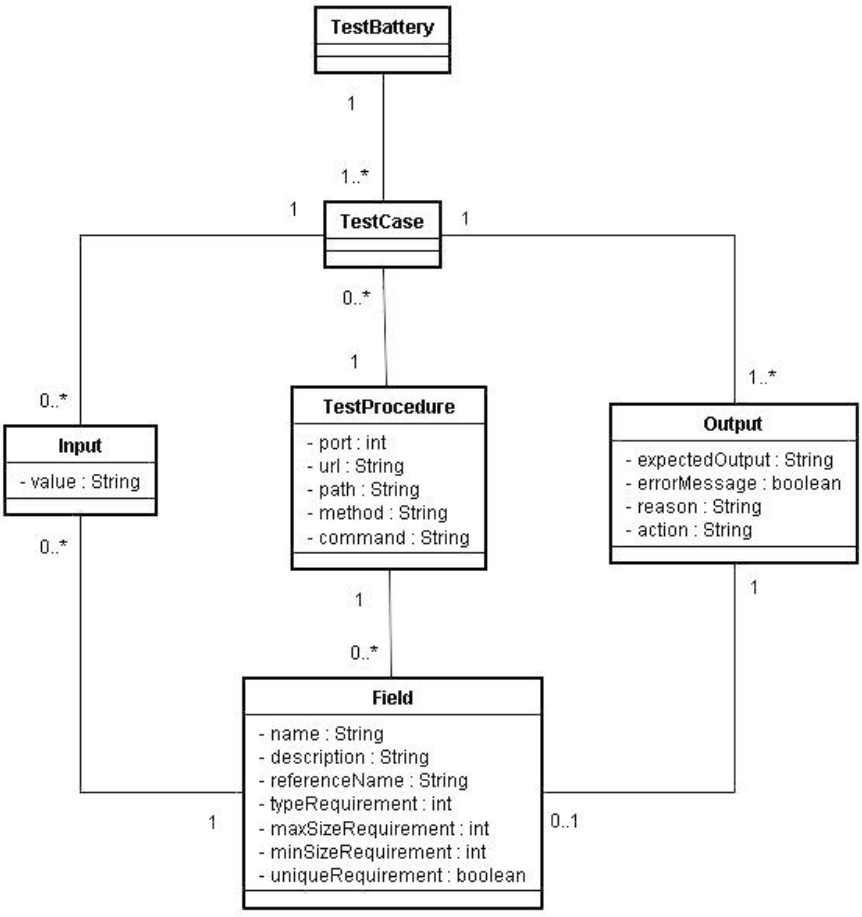

Figura 4: Diagrama de classes para a representação abstrata do teste funcional.

[6] http://webtest.canoo.com/

Uma bateria de testes (TestBattery) contém uma lista de casos de testes (TestCases). Cada caso de teste está ligado a uma entrada (Input), uma saída (Output) e a um procedimento de teste (TestProcedure). O procedimento de teste contém as ações que são executadas nos casos de testes, ou seja, as informações sobre os campos da página e o comando que foi acionado. As entradas e saídas por sua vez, estão ligadas a um campo (Field), que pode conter restrições, como por exemplo, tamanho máximo, tamanho mínimo, tipos de valores aceitáveis.

Para gerar os documentos, além das informações extraídas do teste funcional, é preciso que o usuário informe alguns dados que não podem ser extraídos dos *scripts* de testes. Existem classes que armazenam essas informações gerais a cerca de cada documento. Essas informações são utilizadas para a geração propriamente dita da documentação.

5.3. Funcionamento

O uso da ferramenta desenvolvida é bem simples e pode ser descrito de acordo com os seguintes passos:

- *Extração das informações de um script de teste funcional*. Nesse passo devem ser extraídas do *script* de teste funcional todas as informações necessárias para a criação dos documentos, como por exemplo, o nome dos campos, os comandos acionados e o resultado esperado. Para isso, deve ser submetido ao Extractor adequado um *script* de teste funcional desenvolvido em uma ferramenta de teste funcional.
- *Organização do documento a ser gerado*. A partir do uso da Test2Doc devem ser especificadas as partes que comporão o documento. No caso da Documentação de Usuário, isso corresponde a indicar quais serão as seções e subseções. Já para gerar a Especificação de Teste, devem ser enumerados os casos de uso que serão testados e que comporão o documento, e em seguida, quais casos e procedimentos de testes que estão associados a cada caso de uso.
- *Configuração da documentação*. O próximo passo é o fornecimento de dados gerais sobre o documento, bem como a configuração de cada elemento que estará presente no documento. Nesse passo as informações que não podem ser inferidas diretamente dos testes devem ser informadas.
- *Gerar o documento*. Por fim, basta selecionar qual documento deverá ser gerado e sua localização.

Uma opção existente na ferramenta é a associação de campos identificados nos casos testes a requisitos que devem ser obedecidos durante os testes e que devem ser refletidos na documentação. Essa associação é simplificada a partir do uso de uma das funções da Test2Doc. Exemplos desses requisitos podem ser: tamanho máximo permitido, tamanho mínimo e tipo de valores aceitáveis, assim como obrigatoriedade. Com base nesses requisitos a descrição dos campos é inserida na documentação gerada.

Para uma melhor compreensão de como a Documentação de Usuário e a Especificação de Testes são geradas considere o *script* de teste funcional ilustrado na Figura 5. Este *script* foi feito para verificar se ao efetuar uma busca na página principal

do site testado (http://www.ufpi.br/ufpi2008/) colocando como chave de busca o texto "reitor", aparece na página o texto "Busca pela palavra: reitor".

```
<tr>
    <td>open</td>
    <td>http://www.ufpi.br/ufpi2008/</td>
    <td></td>
</tr>
<tr>
    <td>type</td>
    <td>palavra</td>
    <td>reitor</td>
</tr>
<tr>
    <td>assertTextPresent</td>
    <td>Busca na UFPI</td>
    <td></td>
</tr>
<tr>
    <td>clickAndWait</td>
    <td>submit</td>
    <td>Ok</td>
</tr>
<tr>
    <td>assertTextPresent</td>
    <td>Busca pela palavra: reitor</td>
    <td></td>
</tr>
```

Figura 5: *Script* simples de teste funcional.

Um trecho da Documentação de Usuário gerado pela Test2Doc a partir do *script* acima é ilustrado na Figura 6. As entradas foram extraídas a partir do comando "type", que possui como alvo o nome do campo (palavra) e como valor o texto que foi colocado neste campo (reitor). Uma vez que imediatamente após o "type" existe o comando "assertTextPresent", a ferramenta interpreta o valor deste *assert* como sendo o nome referencial do campo "palavra". Assim, ao invés de ficar "O campo palavra deve ser preenchido", temos "O campo Busca na UFPI deve ser preenchido". Isso se faz necessário porque "palavra" é o nome do campo e está presente somente no código-fonte da página e como a Documentação de Usuário é voltada para o usuário final é preciso utilizar um termo que esteja visível na página ("Busca na UFPI").

2.1.1.4 Entradas

• O campo *Busca na UFPI* deve ser preenchido.

2.1.1.5 Chamada

Acionar o botão *Ok*

2.1.1.6 Saídas

Deve aparecer na página o texto "Busca pela palavra: reitor"

2.1.1.10 Condições de Erro

Número de Ordem	Mensagem	Explicação	Ação Recomendada

Figura 6: Trecho da Documentação de Usuário gerada a partir de um teste funcional.

A Figura 7 apresenta um trecho da Especificação de Teste gerada a partir do *script* da Figura 5. As entradas, o fluxo do procedimento de teste e a saída esperada são extraídas diretamente do *script*. As demais informações são fornecidas manualmente pelo usuário da Teste2Doc.

2.1.6.1 Procedimento de teste Busca na Página Principal

Identificação	UFPI-PT-BU
Objetivo	Realizar a busca de um texto no site da UFPI
Requisitos especiais	Estar na página principal da UFPI
Fluxo	1. Preencher o campo *palavra* 2. Acionar o comando *Ok*

2.1.6 Casos de teste
2.1.6.1 Caso de teste Busca Simples na Página Principal

Identificação	UFPI-CT-BUS	
Itens a testar	A busca através da página principal da UFPI	
Entradas	**Campo**	**Valor**
	palavra	reitor
Saída Esperada	Deve aparecer na página o texto "Busca pela palavra: reitor"	
Ambiente	Ambiente de teste	
Procedimento	UFPI-PT-BU	
Dependências	-	

Figura 7: Trecho da Especificação de Teste gerada a partir de um teste funcional.

A ferramenta ainda possui algumas limitações. Um exemplo é a documentação de funcionalidades que estão associadas a diversas telas diferentes, uma vez que até o momento a Test2Doc somente está preparada para documentar ações contidas por completo em um único caso de teste. Além disso, é necessário seguir algumas convenções para a criação dos testes, para facilitar a identificação de elementos a serem gerados nos documentos. A idéia associada ao trabalho é introduzir cada vez mais inteligência nesse processo, reduzindo bastante a necessidade de intervenção manual.

5. Experimento Realizado

Foi realizado um estudo experimental com o intuito de comparar o tempo necessário para a geração da Documentação de Usuário com e sem o uso do método e da ferramenta desenvolvida. O experimento também procurou avaliar a qualidade do trabalho produzido. As principais questões relacionadas ao experimento foram: o uso do método e da Test2Doc gera uma redução no esforço necessário para a criação da DU comparado com a forma manual? Da mesma forma, avaliou-se a qualidade dos documentos gerados, analisando se a qualidade com e sem o uso da Test2Doc foi a mesma.

Foram utilizados como participantes alunos das disciplinas Tópicos em Programação do curso de Ciência da Computação da Universidade Federal do Piauí (UFPI). A disciplina foi ofertada no período complementar de 2010, tendo iniciado em

janeiro de 2010. Os participantes eram, na maioria, alunos com até 4 períodos de curso e não tinham experiência alguma com a geração de Documentação de Usuário. Por conta disso não foi feito nenhum tipo de agrupamento baseado no perfil dos mesmos. Antes do início do estudo foi realizada uma breve apresentação com relação aos documentos e as atividades associadas. Também foi garantido que seria preservado o anonimato dos estudantes explicando ainda como esses dados seriam utilizados. Todos os voluntários concordaram com o experimento e permitiram sua utilização para fins acadêmicos.

O desenho experimental utilizado levou em consideração as possíveis ameaças à validade do mesmo. Os alunos foram divididos em dois grupos. Os participantes criaram a DU de forma manual e com auxílio da ferramenta. O Grupo 1 iniciava pelo uso da ferramenta enquanto que o Grupo 2 iniciava pela geração manual. Isso foi feito para que seja possível medir eventuais interferências geradas pela ordem de atribuição dos participantes aos tratamentos.

A Tabela 1 apresenta os dados colhidos no estudo. O tempo gasto pelos participantes para criar a DU de forma manual foi, em média, 26 minutos, enquanto o tempo médio para fazer a mesma atividade com o apoio da ferramenta foi 16 minutos. Ao utilizar o teste *t de student* (95% de significância) para comparar os dados, verificou-se que a diferença é estatisticamente significante. A explicação para isso está no fato da T2Doc automatizar parte da geração da DU, utilizando os testes funcionais como base para essa criação. Com isso, o usuário da ferramenta precisa apenas especificar os dados gerais da documentação.

Foi analisado também se a ordem de execução das tarefas influenciou de alguma forma os resultados. Foi comparado o tempo para criar a DU de forma manual pelo Grupo 1 e Grupo 2, uma vez que isso foi feito depois de usar a ferramenta no Grupo 1 e antes no Grupo 2. Não houve diferença significativa nos dados. O mesmo aconteceu para o uso da ferramenta: a ordem de uso não influenciou de forma significativa nos tempos registrados.

Tabela 1. Dados colhidos durante o experimento.

Ordem	Documentação de Usuário					
	Voluntário	Manual	Avaliação	Ferramenta	Avaliação	Diferença
Grupo 1: 1.ferramenta 2.manual	1	23	ruim	10	ruim	13
	2	21	regular	16	ruim	5
	3	28	bom	20	ruim	8
	4	27	ruim	14	regular	13
	5	31	ruim	20	regular	11
	6	36	bom	28	regular	8
	7	26	regular	17	bom	9
	8	19	bom	14	bom	5
	9	30	bom	16	bom	14
	10	42	bom	15	bom	27
Grupo 2: 1.manual 2.ferramenta	11	32	bom	19	bom	13
	12	23	regular	10	bom	13
	13	24	regular	12	bom	12
	14	22	ruim	15	bom	7
	15	20	bom	13	regular	7
	16	19	regular	15	ruim	4
-	Média	26	-	16	-	11

Outra análise realizada diz respeito à qualidade dos documentos gerados. Uma equipe composta por 3 avaliadores analisou a qualidade dos documentos gerados. Para isso, foi apresentado para essa equipe um documento modelo para a comparação. Este modelo continha as informações que deveriam estar presentes nos documentos que foram gerados durante o experimento. Os resultados da avaliação também estão contidos na Tabela 1. Não houve diferença significativa nos dados referentes à qualidade. Assim, conclui-se que o uso da ferramenta reduz o tempo para criação dos documentos, mantendo o mesmo nível de qualidade, pelo menos nos experimentos realizados neste trabalho.

Como houve quatro documentos com qualidade considerada ruim tanto na geração manual como na geração baseada no uso da ferramenta, foi feito uma análise para verificar se o tempo associado à criação desses documentos poderia influenciar no resultado geral obtido. Para isso, foi analisado se os tempos para gerar os documentos considerados regular ou bom diferem do tempo gasto para gerar documentos com avaliação ruim. Isso foi feito tanto para analisar os documentos manuais ruins em relação ao restante dos documentos manuais, assim como em relação aos documentos gerados com o apoio da ferramenta (ruins contra regular e bom). Em ambos os casos não foi registrado uma diferença significante. Isso pode indicar que o tempo não foi o fator diretamente associado à qualidade do documento, ou seja, um pouco mais de atenção, ou de intimidade com a ferramenta, talvez pudesse ser decisivo em relação à qualidade.

Os principais problemas registrados nos documentos gerados a partir do uso da ferramenta estão associados a problemas de configuração, como os dados básicos do documento (nome da equipe, data...), além de erros na especificação dos testes a serem usados para gerar a descrição de algumas entradas e mensagens associadas. Acredita-se que um pouco mais de treinamento na ferramenta reduza a taxa dos erros. Os principais erros com relação à geração dos documentos manuais foram a falta de preenchimento de diversas seções e copiar/colar indevidos. Esses erros também poderiam ser eliminados com um pouco mais de atenção e treinamento para os participantes.

Ao realizar uma análise na validade interna do estudo, acredita-se que o resultado obtido é causal e não influência de outros fatores não controlados. O uso da ferramenta gera uma redução no esforço necessário para a criação da DU mantendo o mesmo nível de qualidade. Como os participantes tiveram que gerar a documentação para a mesma parte do sistema, tanto de forma manual quanto usando a ferramenta, não foi notado o problema da instrumentação, uma vez que não há diferença no problema utilizado no estudo. No entanto, afirmamos que a avaliação da qualidade do documento foi baseada mais em regras empíricas que em critérios bem definidos. Isso pode influenciar o resultado.

Outro problema que poderia influenciar o resultado é a ameaça conhecida como testagem. Ela se refere ao fato da pessoa ficar mais produtiva entre as tarefas, por conhecer mais o sistema e o próprio problema a ser tratado. Conforme foi enfatizado anteriormente, a testagem não influenciou as atividades uma vez que a execução do teste t mostrou que os tempos para executar as tarefas em ordem diferente não obtiveram diferença significativa.

Com relação à maturação, acredita-se que os participantes melhoraram com o tempo, uma vez que são alunos e estão submetidos a diversos conteúdos novos, mas

nada que pudesse influenciar diretamente o estudo desenvolvido. O desenho experimental escolhido, em que todos os participantes utilizaram as duas ferramentas, permitindo que um grupo atuasse como controle do outro, validam a conclusão obtida ao mesmo tempo em que minimizam qualquer ameaça relacionada ao comportamento competitivo e ao comportamento compensatório [Wholin et al. 2000].

Ao realizar uma análise da validade externa nota-se que não é possível generalizar os resultados obtidos para a prática industrial. Os participantes do estudo, alunos do 2º ao 4º período do curso em sua maioria, ainda estão em fase inicial de formação. É possível que a exposição desses alunos aos diversos trabalhos e conceitos no restante do curso influenciem a realização do mesmo experimento em outro momento. Os treinamentos ministrados não se mostraram suficientes a ponto de impedir certos erros básicos na realização das atividades. Esses são pontos de melhoria para estudos futuros. O exemplo utilizado, embora pequeno, possui as características comumente existentes nos sistemas de informação para *Web*. Assim, acredita-se que as conclusões obtidas no estudo possam ser estendidas para outros sistemas, utilizados por estudantes com pouca experiência, sem prejuízos nos resultados observados. Já para a prática industrial isso não pode ser afirmado.

7. Conclusões e Trabalhos Futuros

Neste artigo foi apresentada uma nova abordagem para criação de documentos no desenvolvimento de *software* denominada Documentação Dirigida por Testes. Embora esse termo seja utilizado em alguns sítios pelo mundo, este é o primeiro trabalho acadêmico que foi encontrado definindo o termo, um método associado e apresentando uma avaliação do seu uso.

A Documentação de Usuário é um artefato crítico em muitos projetos, sendo importante para o sucesso de diversos produtos. Por outro lado, sua criação demanda tempo e esforço. Conforme discutido na Seção 6, a abordagem aqui descrita e implementada pela ferramenta Test2Doc pode reduzir os custos associados à geração desses documentos, uma vez que os testes funcionais são reaproveitados com essa intenção. Essa abordagem pode ser integrada em boa parte dos processos de *software*, incluindo os processos ágeis, que praticamente não prescrevem a criação de nenhum modelo ou documento, mas que são largamente utilizados e fortemente baseados na prática de teste.

Como trabalhos futuros pode-se destacar a implementação de novos Extratores, aumentando assim a abrangência da ferramenta, além da introdução de mais inteligência no gerador, reduzindo ainda mais a necessidade de intervenção humana, embora seja sabido que essa intervenção não possa ser completamente eliminada a um custo razoável. Para verificar isso, novos estudos experimentais serão realizados, levando-se em consideração as fraquezas identificadas no estudo apresentado neste trabalho.

Por fim, a utilização dos testes funcionais como insumo para gerar a Documentação de Usuário abre caminhos para gerar e manter atualizado outros documentos como, por exemplo, o Cadastro de Requisitos e Relatórios de Acompanhamento.

Agradecimentos

Este trabalho recebeu apoio financeiro do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e da empresa Infoway Consultoria e Soluções em Tecnologia para Gestão de Saúde.

Referências

Capers, J. (2007). *Estimating Software Costs: Bringing Realism to Estimating*. McGraw-Hill, NewYork, NY, USA, 2ª edição.

IEEE (1998). *IEEE Standard for Software Test Documentation – IEEE Std 829-1998*. IEEE Computer Society.

IEEE (2001). *IEEE Standard for Software User Documentation*. IEEE Computer Society.

IEEE (2004). *Guide to the Software Engineering Body of Knowledge*. IEEE Computer Society, Los Alamitos, California.

Johnson, W. L. (1994). Dynamic (re)generation of software documentation. In: RMI-94-003, A.R.C.R., editor, *Proceedings of the Fourth Systems Reengineering Technology Workshop*, pages57–66, Monterey, California, USA. Johns Hopkins University Applied Physics Laboratory.

Jungmayr, S. e Stumpe, J. (1998). Another motivation for usage models: Generation of user documentation. In *Proceedings of CONQUEST'98*, Nüremberg, Alemanha.

Houaiss, A. e Villar, M. (2001). *Dicionário Houaiss de língua portuguesa*

Moriyon, R., Szekely, P. e Neches, R. (1994). Automatic generation of help from interface design models. In: *CHI'94: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 225–231, NewYork, NY, USA. ACM.

Padua,W. (2003). *Engenharia de Software: Fundamentos, Métodos e Padrões*. LTC, 2ª edição.

Paris, C. e Linden, K. V. (1996). Building knowledge bases for the generation of software documentation. In: *Proceedings of the 16th Conference on Computational Linguistics*, pages734–739, Morristown, NJ, USA. Association for Computational Linguistics.

Reiter, E., Mellish, C. e Levine, J. (1992). Automatic generation of on-line documentation in the idas project. In: *Proceedings of the Third Conference on Applied Natural Language Processing*, pages 64–71, Morristown, NJ, USA. Association for Computational Linguistics.

SOFTEX (2009). "MPS.BR - Guia Geral", http://www.softex.br, Abril.

Wohlin, C., Runeson, P., Host, M., Ohlsson, M., Regnell, B. e Wesslen, A. (2000). *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers.