

Metadata-based framework to verify external effects in unit tests

Marcus V. C. Floriano

Aeronautical Institute of Technology
Praça Marechal Eduardo Gomes, 50
VI Acácias - SJ Campos – SP, Brazil
+55 12 39475899

Email: marcus.floriano@gmail.com

Debora A. L. Chama

Aeronautical Institute of Technology
Praça Marechal Eduardo Gomes, 50
VI Acácias - SJ Campos – SP, Brazil
+55 12 39475899

Email: deborachama@gmail.com

Eduardo M. Guerra

Aeronautical Institute of Technology
Praça Marechal Eduardo Gomes, 50
VI Acácias - SJ Campos – SP, Brazil
+55 12 39475899

Email: guerraem@gmail.com

Resumo—Metadata-based frameworks are those that use metadata to process their logic. Many tests are dependent on external resources such as database, web services, sockets, files and so on, but it is difficult to check these external effects on unit tests. As a result, this paper presents a framework based on metadata in order to help developers to verify the external effects in their unit tests.

I. INTRODUCTION

Software usually rely on external resources such as databases, web services, sockets, files and so on.

Aiming to improve software quality, unit tests are created. During the verifications of software using these unit tests, the features that work with external resources are verified, but it is not easy to know the effect of the test on the external resource.

When the verifications of these external effects are not done, we can not say that the software is fully tested and that all features that depend on external resources are functioning properly, since the result could not be verified.

This paper aims to provide a solution in a simplified way that allows verification of external effects of the developed software.

II. TESTS WITH EXTERNAL DEPENDENCIES

To ensure the quality of software, during its development mechanisms are used to verify the quality of what is being developed. One way is through the creation of unit tests.

Unit tests are one way to ensure, programmatically, the verification of a particular feature, always focusing on the software being developed.

Analyzing the external effects that cause these tests, for example, a feature that depends on a web service, in the unit test is complicated to verify the result of tests on the service called.

Another example is when you have a feature that saves files in a particular place or format. Verify the file format created or if the content is correct are more labor intensive tasks.

Our purpose is to build a metadata-based tool that allows the creation of annotations to check the external effects caused by the unit tests, in a much simpler way and giving opportunity

for reuse.

(falem dos problemas, pq é difícil de testar e etc..) ver se aquele artigo do google escolar tem algo q dá pra colocar aqui

III. EXISTING SOLUTIONS

(o que elas fazem e porque não resolvem o problema)

Originalidade e Relevância

Algumas soluções parciais foram encontradas, como por exemplo a associação da anotação a uma classe de execução utilizando reflexão.

Falar daquela solucao que nao me lembro o nome. procurar a referencia bibliografica dela para referenciar aqui.

A solução proposta, como extensão do JUnit a partir da funcionalidade “runners” e a criação de um framework para facilitar a criação de anotações e classes de execução, não foi encontrada até o presente momento.

IV. CONFIGURED TESTS THROUGH METADATA

(falem da idéia de vocês de forma mais abstrata, mostrem soluções parecidas e digam que elas são específicas)

Com o objetivo de criar um framework simples de ser utilizado pelos desenvolvedores, focou-se no uso das anotações, pois elas permitem inclusão de meta-informação em atributos, métodos e classes.

Para processar essas anotações é necessário que os métodos de testes possam ser interceptados. Um proxy dinâmico consegue interceptar os métodos sem a necessidade de implementar a mesma interface da classe original, assumindo esse comportamento de forma dinâmica. Esse artigo apresenta como combinar reflexão e anotações para criar componentes mais flexíveis. [guerra06]

São apresentados nesse artigo os conceitos de proxys estáticos e dinâmicos e as várias formas de criá-los. O artigo serviu como guia no início da construção da ferramenta desse trabalho, pois utilizamos proxy dinâmico para interceptar os

métodos de teste e adicionar funcionalidades. [guerra08]

Frameworks baseados em metadados são aqueles que processam sua lógica baseando-se nas informações (metadados) das classes que estão trabalhando.

A ferramenta proposta nesse trabalho é um framework desse tipo. Como nesse artigo são apresentados padrões de linguagem para desenvolvimento de frameworks baseado em metadados, ele será o ponto de partida para a refatoração a ser realizada no núcleo da ferramenta. O propósito desse artigo é ajudar no entendimento e desenvolvimento desse tipo de framework. [guerra-pattern]

V. FRAMEWORK "MAKE A TEST"

A arquitetura xUnit contém as definições genéricas para qualquer estrutura de testes unitários automatizados. O JUnit é uma instância dessa arquitetura na linguagem Java.

O JUnit é um framework simples e open source para a criação e execução de testes de unidades, atualmente é bem utilizada por desenvolvedores. Dessa forma a solução tem como objetivo principal manter todas as funcionalidades existentes no JUnit sendo assertions, fixtures, anotações, @Test, @Before e @After.

A solução proposta estende o JUnit usando uma funcionalidade denominada "runners" (@RunWith) do JUnit 4, que basicamente permite interceptar o contexto do teste executado e adicionar novas funcionalidades. Dessa forma todas as funcionalidades são preservadas apenas adicionando as novas funcionalidades da solução.

A outra parte da solução é a criação de um framework que permite o desenvolvimento de anotações, como por exemplo "@FileExists(filepath)". Essa anotação é criada juntamente com uma classe que é responsável por executar a inteligência da anotação.

A anotação pode ser criada para ser adicionada no setup do teste ou em um método de teste. A ferramenta entende que existe uma anotação e executa a classe associada a essa anotação que contém a lógica implementada, ou seja, para a anotação criada "@FileExists(filepath)", é verifica a existência ou não do arquivo.

falem da solução de vocês como uma validação da idéia

VI. CASE STUDY

Mostrem o uso da implementação de vocês para pelo menos duas situações diferentes.

Validação

Será apresentada a ferramenta para uma equipe de desenvolvimento, que localizará nos testes de unidade existentes um ponto para aplicar a ferramenta. A ferramenta proposta será aplicada e os testes serão executados com a nova e a velha versão.

O objetivo é validar os resultados, com uma visão sobre a cobertura dos testes, a complexidade de desenvolvimento e a reutilização.

Outra validação é a construção de soluções para verificar os efeitos externos, como a validação de criação e formatos de conteúdos de arquivos e a verificação de efeitos externos em Web Services.

VII. CONCLUSION

The conclusion goes here.

Conclusão (resumir o artigo, ressaltar contribuições, trabalhos futuros)

Contribuições Uma contribuição desse trabalho é a facilidade na criação de anotações para verificar os efeitos externos ocasionados pelos testes de unidade.

Essa facilidade ocorre pois a solução proposta é criar um framework que permita com que a criação da anotação e a associação à classe de execução seja de feita de forma transparente para o desenvolvedor. Assim o único trabalho do desenvolvedor é implementar a lógica da verificação.

Outra contribuição está no aspecto de reuso. Como a verificação de efeitos externos pode se repetir em métodos, classes e até aplicações diferentes, as anotações e as classes de execução podem ser empacotadas em um JAR e serem reutilizadas em qualquer projeto, uma vez que a anotação desenvolvida possa ser reutilizada.

ACKNOWLEDGMENT

The authors would like to thank...

REFERÊNCIAS

- [1] Guerra, Eduardo; Souza, Jerffeson; Fernandes, Clovis, "A Pattern Language for Metadata-based Frameworks", 16th Conference on Pattern Languages of Programming, Chicago, August, 2009.
- [2] Guerra, Eduardo, "Reflexão + Anotações - Uma Combinação Explosiva", Revista Mundo Java, Ed. 0019, p.15-26, 2006.
- [3] Guerra, Eduardo, "Proxys Estáticos e Dinâmicos", Revista Mundo Java, Ed. 0032, p. 51-56, 2008.
- [4] "JUnit". Available at <http://junit.org/>, 2010.