Metadata-based Test Framework to Verify External Effects of the Target Classes

Marcus V. C. Floriano
Aeronautical Institute of Technology
Praça Marechal Eduardo Gomes, 50
Vl 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
Vl Acácias - SJ Campos – SP, Brazil
+55 12 39475899
Email: deborachama@gmail.com

Eduardo M. Guerra
Aeronautical Institute of Technology
Praça Marechal Eduardo Gomes, 50
Vl Acácias - SJ Campos – SP, Brazil
+55 12 39475899
Email: guerraem@gmail.com

***Resumo*—Metadata-based frameworks are those that use information(metadata) of the classes that are working 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.**

*****Categories and Subject Descriptors

*****General Terms

*****Keywords

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

The proposed solution, as an extension of JUnit functionality "runners"and the creation of a framework to facilitate the creation of annotations and execution classes, has not been found until now.
But some partial solutions were found. Some of them do the combination of annotation to a class of implementation using reflection.

IV. Configured tests through metadata

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

Aiming to create a simple framework to be used by developers, this work focused on the use of annotations, since they allow the inclusion of metadata in attributes, methods and classes.
To process these annotations is necessary that the test methods can be intercepted. A dynamic proxy can intercept methods without the need to implement the same interface as the original class, assuming that behavior dynamically.
This framework was developed combining reflection and annotations to create more flexible components. [Guerra06]

V. Framework "Make A Test"

XUnit architecture contains generic settings for any automated unit testing framework. JUnit is an instance of this architecture in Java. [JUNIT]

JUnit is a simple open source framework for creating and running unit tests, and is currently used by developers. Thus the solution's main objective is to keep all the existing features in JUnit and assertions, fixtures, annotations, @ Test, @ Before and @ After.

The proposed solution extends the JUnit using a feature called "runners"(@ RunWith) of the JUnit 4, which basically allows you to intercept the context of the executed test and add new features. Thus all the features are preserved only by adding new features of the solution.

The other part of the solution is to create a framework that allows the development of notations such as "@ FileExists (filepath). This annotation is created along with a class that is responsible for executing the intelligence of the annotation.

The annotation can be created to be added in the setup of the test or a test method. The tool understands that there is an annotation and runs the class associated with this annotation that contains the logic implemented, that is created for the annotation "@ FileExists (filepath)", it checks whether or not the file.

VI. Case Study

*****Mostrar a solução de leitura de arquivos

VII. Conclusion

The conclusion goes here.

A contribution of this work is the ease in creating annotations to verify the external effects caused by unit tests. This feature occurs because the proposed solution is to create a framework that enable that the creation of the annotation and association with the implementation class can be done transparently to the developer. So the only developer's work is to implement the verification logic.

Another contribution is the aspect of reuse. How to check external effects could be repeated in methods, classes and even different applications, the annotations and implementing classes can be packaged in a jar and reuse in any project, since the annotations developed can be reused.

As future works ..... *****falar do desenvolvimento da verificacao de efeitos externos em webservices

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.