# aUnit

Initial Concept Exploration

Russ Miles, 2004

## Objective

The objective of this document is to provide a basis and rationale for the initial approaches incorporated into the production of a unit testing tool for cross-cutting concerns implemented using AspectJ.

## Rationale

Similar to how JUnit provides unit testing for Java, aUnit will provide mechanisms by which developers can test their aspects in isolation in support of Test Driven Development [2].

## A Definition of "Unit Test"

A unit test can be thought of as being:

*"a means by which a discrete software component, of varying scale, can be asserted as working correctly according to the constraints of the test."*

Therefore:

*"An aUnit test provides a means by which a discrete aspect, of varying scale, can be asserted as working correctly according to the constraints of the test."*

## Advice, the building blocks of AO unit tests

Advice blocks, in a similar manner to traditional Java methods, are seen as the smallest element that can be tested and therefore should be the focus of the initial implementation of aUnit.

Many options have been considered, including pointcut declaration testing, aspect lifecycle testing but these were seen as an inaccurate fit to testing the aspect's behavior itself. Since the only behavior that an aspect exhibits is in the form of its advice then it is this construct that is the best candidate for testing during the first iteration of aUnit development.

## Initial Implementation Goals for Iteration 1 of aUnit

The goals for this first iteration of aUnit development are:

- Provision of the basic classes that conform to the xUnit architecture and meet the demands of the aforementioned use cases.
- Provision of rudimentary user interfaces, based on Swing, Text, AWT and potentially SWT if deemed necessary.
- Provision of a plug-in that supports aUnit testing within the Eclipse IDE and working alongside the existing AspectJ Developer Tools (AJDT).

Achieving these goals would lead to the provision of a stable aUnit 1.0.

## Proposals for Future Iterations of aUnit

The following extensions are intended only to inform the initial development of aUnit as to the potential future demands on the tool. At this point there is not distinct roadmap as to when and why these extensions may be provided.

Extension 1: AJML

It is proposed that one extension to the initial aUnit offering would be, either by means of using the current project participants or by forming another project, to offer support for a modelling language that could be used for defining and informing aUnit test cases, similarly to how the Java Modeling Language (JML) [1] can currently translate to JUnit test cases.

## Summary and Conclusion

Advice appears to be the core focus of aspect testing. Although there exists certain high level desires to test aspect lifecycles these work on the periphery of the testing requirements for advice, be they statically or dynamically determinable.

## Bibliography

[1] The Java Modeling Language, http://www.cs.iastate.edu/~leavens/JML/

[2] ObjectMentor - Test Driven Development, http://www.objectmentor.com/writeUps/TestDrivenDevelopment