Agility with Legacy Code

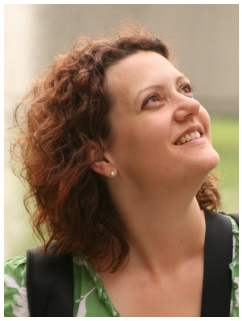
Many teams starting with Agile methods struggle with code quality and getting a sufficient number of automated tests in place. It is not usually all that easy to start writing classic unit tests for existing code that was not designed with testability in mind. JUnit et al require the code to be divided into isolatable units that can be tested individually. Much legacy code just isn't designed this way, making it difficult to improve test coverage without doing risky refactorings first.

In this situation I'd often recommend starting with Text-Based tests in order to get basic automated test coverage while you refactor the design to enable finer granularity unit tests. It's a technique suited to almost any language and platform, for example C/C++ and Java. All that is required is that you can drive your system via the command line, (or certain rich client GUIs with "StoryText"), and convert the output to some form of plain text.

About the seminar
This session is for software developers struggling to use agile methods at the same time as getting legacy code under control. We'll look at these topics:

- Why is Legacy code a problem in agile methods?
- Test Driven Development for confidence and quality
- Text-Based Testing: Managing Behaviour Change
- Rich Client GUIs and StoryText
- Examples & Case Study

About your trainer, Emily Bache
Emily is an experienced programmer with a focus on the engineering practices which make agile methods work. Working as an independent consultant for Bache Consulting, she helps teams to improve the way they build software. Emily has worked for many years using practices like Test Driven Development and Text-Based Testing, and speaks regularly at international events such as Agile Testing Days, XP2012 and Scandinavian Developer Conference. Emily is fluent in both Swedish and English.



Text-Based Testing
Text-Based Testing is a simple idea, which is to scale a common technique from unit testing to a whole program. You're probably familiar with tests containing statements like AssertEquals(String expected, String actual). Text-Based tests are similar, but for larger strings produced by whole programs. You save the "expected" string from the actual output of the program as a "golden copy", which you compare against the "actual" output when you run the program again. This Text-Based test will pass so long as the program output doesn't differ from the golden copy. If it does, the test fails, and you view the differences. You can then either fix the program, or update the golden copy.

StoryText
This tool is designed to help you write readable, robust, agile automated tests for rich-client applications built with Eclipse Rich-Client Platform or Java Swing. For more information see the TextTest website I'm one of the early-adpoter users of this tool, and I'd be more than happy to demo it for you if it would be appropriate for your context.