Lazy Systematic Unit Testing: JWalk versus JUnit

JWalk is a lazy systematic unit testing tool [1]. This is based on lazy specification, inferring a continuously changing specification from rapidly evolving code, by dynamic code analysis and programmer interaction, and systematic testing, generating complete test-sets that exercise and validate the state-space of the class-under-test (CUT) exhaustively to bounded depths [2].

A challenge was set up to contrast the effectiveness of semi-automated testing with JWalk against expert manual testing using JUnit [3], the most widely used testing tool in the agile community.

Two related pairs of CUTs were tested, including a simple LinkedStack, later modified as a BoundedStack (a code evolution); and a LibraryBook, later extended as a ReservableBook (by inheritance). The competing testers were asked to develop "complete tests" for each initial class. Later, JWalk was allowed to propose further tests for the modified or extended versions.

The table below shows how interactive oracle confirmation in JWalk covered more unique cases (in less time) than the manual assertions thought up by the expert for JUnit. JWalk then automatically tested all state-transition paths to depth 3, compared against slightly less than the transition cover for JUnit (nullops were not tested; two assertions were non-unique).

When retesting the modified or extended versions, JWalk found all additional observations on novel method interleavings, confirmed in under 18 minutes, and then tested up to 1732 paths automatically. JWalk makes better use of test automation, proposing all key test cases for rapid review by the tester, and has much higher coverage than traditional regression testing.

Class-under-test	API size	JUnit asserts	JWalk oracles	JWalk total
LinkedStack	6	9	24	220
BoundedStack	7	N/A	+35	645
LibraryBook	5	11	20	138
ReservableBook	9	N/A	+167	1732

The University Of Sheffield.

Anthony Simons and Christopher Thomson
http://www.dcs.shef.ac.uk/~ajhs/jwalk/

[1] A. J. H. Simons, JWalk: lazy systematic unit testing. http://www.dcs.shef.ac.uk/~ajhs/jwalk/, 2007.
[2] A. J. H. Simons, JWalk: a tool for lazy systematic testing of Java classes by introspection and user interaction, Automated Software Engineering, 14 (4), December, ed. B. Nuseibeh, (Springer, USA, 2007), in press.
[3] K. Beck, The JUnit Pocket Guide, 1st edn., O'Reilly, Beijing, 2004.