

# Correlating Unit Tests and Methods under Test

Markus Gälli

Software Composition Group  
University Bern  
gaelli@iam.unibe.ch

<sup>1</sup> **Keywords:** unit tests, methods under test, method examples, test scenarios, traits

**Research Questions:** What are the relationships between unit tests and between unit tests and methods under test? What can be gained by making this relationships explicit? How does the concept of *method examples* compare with other possible techniques to relate this entities?

**Significant problems and current solutions:** (1.) Missing explicit relationships between unit tests and methods under test make it difficult to trace which features are thoroughly tested and hinder navigability between unit tests and their methods under test. xUnit uses a naming convention which is brittle when it comes to renaming the methods and classes under test. (2.) Schuh *et al.* [1] introduce the concept of *ObjectMother* to compose complex test scenarios. (3.) Failing unit tests are presented randomly and not in a meaningful order. [2]

**Definition:** A *method example* tests a single method **and returns** the resulting receiver, parameters and potential return value of its method under test.

**Approach:** Show which kind of relations between unit tests and between unit tests and method under tests exist. Correlate the unit tests of the base system of Squeak by decomposing them into *method examples*. Show, that the single concept of *method examples* enables navigation and traceability between unit tests and methods under test, provides concrete types for the methods under test, fits well together with traits [3], and allows the composition of complex unit tests. Compare with other techniques to make this relationships explicit.

**Achieved Results:** Case studies show that a significant amount of the relationships between unit tests cover each other when one compares the sets of signatures of their called messages [2], and that the Squeak base unit tests can be successfully refactored to *method examples*.

## Acknowledgments

We gratefully acknowledge the financial support of the Swiss National Science Foundation for the project “Tools and Techniques for Decomposing and Composing Software” (SNF Project No. 2000-067855.02, Oct. 2002 - Sept. 2004).

## References

1. Schuh, P.: Recovery, redemption and Extreme Programming. *IEEE Computer* **18** (2001) 34–41
2. Gälli, M., Nierstrasz, O., Wuyts, R.: Partial ordering tests by coverage sets. Technical Report IAM-03-013, Institut für Informatik, Universität Bern, Switzerland (2003) Technical Report.
3. Schärli, N., Ducasse, S., Nierstrasz, O., Black, A.: Traits: Composable units of behavior. In: Proceedings ECOOP 2003. Volume 2743 of LNCS., Springer Verlag (2003) 248–274

---

<sup>1</sup> 5th International Conference on Extreme Programming and Agile Processes in Software Engineering (XP 2004) LNCS page 317