

Storytesting - ScanDevConf '10

Bill Wake, Industrial Logic, Inc.

Storytesting - the practice of using concrete examples to clarify the meaning of a story.

Why? Shift the conversation. Force decisions. Remove ambiguity. Option to automate.

Who? Anybody on the team. "Business side" people - product owners, customers, subject-matter experts can often provide and/or evaluate critical examples. Testers are often good at exploring corner cases as well. Programmers and others may design or encode tests as well.

Stories

Two long-standing tools are particularly useful: *context diagrams* and the notion of *perfect technology*. These help us think about stories in terms of their desired effect on the world outside the system.

We like the story headline template **Role-Action-Context**, e.g., *User Logs In With Expired License*; details can be added incrementally.

The words in your stories create a *domain language*: roles, verbs, nouns. This domain language can be carried into your implementation and into your tests.

Tests: Many types are possible; there are three common forms.

1. **Business rule tests:** Given these inputs, what outputs are expected? (These work very nicely with a table format, one row per test, with some columns the inputs and others the expected outputs.)
2. **Data tests:** Does a particular set of data exist in the system? (This form also used to establish data for a set of tests.) (These also work nicely with a table format.)
3. **Workflow tests:** Does the system behave properly in a series of interactions? ("Given-when-then" is a handy form.) (Works nicely with a textual or code-based style, though can be seen in table form.)

Strive for expressiveness in tests. Avoid low-level descriptions. Use domain language where you can.

Tools

- Tools act as an intermediary. The driver (fixture) is what determines the test's meaning.
- Tests can work at different levels. For example, you can test through the user interface or through an application programming interface.
- There are many acceptance tools out there. They tend to use either text form (e.g., some sort of domain-driven language), table form, or code.

Trends: a personal view, not a scientific one.

- Scripted manual tests moving to automated or exploratory tests.
- Behaviour-driven development (BDD) influence.
- Proliferation of tools.
- Struggles defining the role of testers. (Are they programmers, analysts, "just" testers?)
- How much to automate?