Note to the reader: this pattern really belongs as part of a larger group of patterns, some of which are given in "thumbnail" form at the end. Standing alone, it seems to have little to do with testing. As part of a larger group, the connections would be clearer.

Pair Testing

Alexanderian patterns begin with a motivating picture. It says to the reader, "Isn't this nice? Want something like it? Read on..." I'll begin with two word-pictures that have the same purpose.

Imagine two men in a room. The first is elegantly yet casually dressed, sitting at a grand piano. He sings snatches of verse and plays variants of melody. The other man, a bit scruffier, sits nearby with a notepad in his lap. He scribbles as the first man sings. When the first man stops, the second sings out variant verses in a markedly less musical voice. They are a famous composer/lyricist pair.

Imagine yourself in row one of a small commuter airplane. The cockpit door is open, so you can see the pilot and copilot sitting there. One is looking at a checklist and calling out cryptic phrases. The other acts after each phrase – perhaps by looking at an instrument, perhaps by flipping a switch – then responds tersely. They are running through the pre-flight checklist.

... your testing team has at least a preliminary idea of its different tasks. Some of them may have already begun or even have been completed. Some of the tasks require considerable creativity or discipline or close attention to detail, such as EXPLORATORY TESTING or FAILURE IMPROVEMENT or watching program output carefully to notice possible failures. You are concerned with how people should be allocated to those particular tasks.

* * *

Working alone is working without a safety net. When you work alone, no one is there to help you cope with your human imperfections.

What might those imperfections be? Consider, first, creative tasks. Any creation builds on past experience, yet any one person will have only limited experience. Adding people would add experience. It might also add creativity in that wonderful way when one person's ideas build on another's so quickly that their words overlap.

Moreover, any creative person knows the feeling of "getting stuck", of falling into a rut. A well-chosen comment by someone else can unstick you. So, too, can a poorly-chosen comment that just happens to resonate. "Unsticking" is one of the strengths of brainstorming, yet brainstorming is hard to do alone.

Additionally, creative tasks are difficult to evaluate after the fact. How well was the task done? Were the right alternatives considered? Were rejected alternatives rejected for the right reason? Is this person good at the task, or does she need help? Evaluation is easier if the evaluator is participating in the give-and-take of the task. Similarly, novices learning a creative task will gain much greater insight if they watch it happen than if they observe the end product.

Tasks that require discipline are subject to a different sort of failure. It's easy to let discipline slide when no one is watching. Contrarily, if you're working with someone else, you would have to admit to each other that you're sneaking around the rules. That's harder to do.

Meticulous tasks often require discipline, because they usually feel at least faintly unpleasant and unnatural. And they have problems beyond discipline. Meticulous and repetitive tasks dull the brain. Two people working together can miss fewer things, either by duplicating observation or (as with the airline pilots) by dividing work so that attention is more effectively used.

But meticulous work is still draining. When more than one person is working on a task, laughter and other changes of pace arise naturally. Or they can be artificial: an observant partner can say, "I need a break" when it's really the other person who does. (This can deftly avoid *both* partners' need to push beyond their endurance.) Attention, because interrupted occasionally, is kept high.

So there seem to be many reasons to use more than one person on creative, disciplined, or meticulous tasks. Is there an alternative? Reviews and inspections are often used to compensate for an individual's human frailty. But we can see that they do not resolve all the forces. They can help catch mistakes due to lapses of attention or to inexperience, but they cannot provide a creative boost – they are too distant from the creative moment.

It seems better to add more people to the task as it is being done. How many? One? Three? Twenty? Surely more is better... No: it appears that two people per task is a special number, for several reasons.

The first is physical. It is common for tasks to require attention to an *object* of some sort: text on a computer screen, a document laid out on a table, and so forth. Tasks in which people observe work from afar (as in a room looking at a whiteboard) have an essentially different character from ones in which participants are huddled around an object, observing it in detail. Two people can comfortably look at a single object. Three are uncomfortable, and more than three is essentially impossible. (At this writing, computer-assisted collaboration doesn't work well enough.)

The second reason is one of diminishing returns. We know that adding an additional person to suitable tasks provides benefits. Adding yet another provides less. Perhaps that third person should be paired with someone else on some other task.

The final reason has to do with group dynamics. Working in a group of three is quite different than working in a pair. Just think of the different connotations of the words “group” and “pair”. In a group, it’s harder to keep everyone fully involved. <*Not sure how to put this point – help me.* >

Therefore:

Begin by having people work in pairs on high-creativity, high-discipline, or high-concentration tasks where two people can focus on the same object of work. Pair work is not appropriate for all such tasks. For example, you’d be surprised to find a great painting produced by a pair of artists. But it’s worthwhile to begin with pairs, then back away to individual work if the extra person doesn’t seem to contribute enough. <*A crisper description of the situations in which pairing does and doesn’t work would help make this solution stronger, but I don’t have the knowledge to do that.*>

Some people embrace the idea of pair work, but many don’t. Introduce it with care. Ask people to try out the idea before judging it.

Some people truly do not like being part of a pair, at least for some tasks. Don’t force them if they’ve given it an honest try. That will do more harm than good.

Some people find pair work exhausting because they are introverts (gain energy from being alone, rather than from being with other people). Don’t force them to overdo it. Give them some solitary tasks.

* * *

Pair work is emotional. People who could work together well – at a distance – will descend into conflict when they pair. For this reason, it’s wise to have a COACH involved. Coaches can also spot cases where one member of a pair is being overbearing and overcontrolling, which greatly reduces the benefit from the other member.

Pairs can find it difficult to agree on tactics. They drift into an escalating discussion of issues faced and solved in the past. Although both are honestly trying to contribute, the discussion goes on too long. The solution is short-duration, narrowly focused tasks - SMALL TASKS.

<Other patterns that flesh out this one are needed>

Examples

I once paired with a tester who was averse to preplanning tests and writing checklists. He preferred to explore the program, and he was quite effective at finding bugs. I like checklists. So he sat at the keyboard and explored. In parallel, I created a checklist of tests to try, based on my observation of what he was doing, my own ideas about testing, and inspirations gotten by looking for gaps in the checklist. We talked constantly. When he slowed down, I'd suggest new avenues from the checklist.

<More examples are needed.>

Other Sources

Software design and programming requires high creativity. In eXtreme Programming (*Extreme Programming Explained*, by Kent Beck), all programming (and, in effect, design) work is done in pairs. Also, XP requires all programmers to write tests before writing code; this requires discipline that is harder to achieve without the effect of pairs. Laurie Williams of NCSU does research in pair programming. Her publications can be found here: <http://collaboration.csc.ncsu.edu/laurie/>.

Other patterns mentioned

Problem	Solution	Pattern Name
There's much about the program that's unknown until you start using it, no matter how well it's documented.	Devote some time to exploring the program's behavior, both to find bugs and help in planning further testing.	EXPLORATORY TESTING
The first sign of a bug is usually not the best one to report.	Take time exploring variants or consequences of that first failure. Find a better one to report.	FAILURE IMPROVEMENT
It's easy for people enmeshed in daily work to drift into inappropriate behavior.	Assign one person to observe what other people are doing, explain it to them, and help them change.	COACH
Pairs can find it difficult to agree on tactics.	Keep pairs focused on small tasks with close deadlines.	SMALL TASKS

Acknowledgements

This pattern was originally inspired by one written by James Bach and Sam Guckenheimer. It was intended to be an experiment in Alexanderian form. I thank these people for their comments: Robert Binder, David Delano, Noel Nyman, and Linda Rising. I heard SMALL TASKS from Ward Cunningham; I hope I've captured it well enough.