Test-Driven Development

Luigi Libero Lucio Starace
luigiliberolucio.starace@unina.it

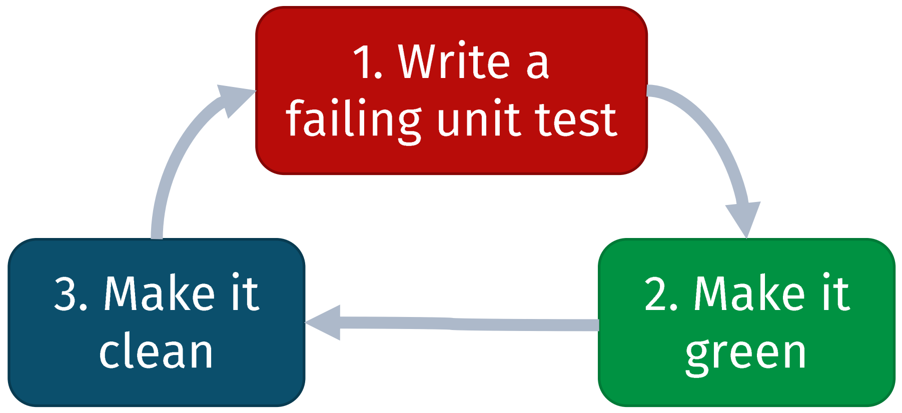
May 4, 2020
University of Naples, Federico II

Test-Driven Development (TDD)

TDD is a way of managing fear during programming.

It's *deceptively* simple:

- Write the tests for your code *before* writing the code itself!



The Three Laws of TDD[1]

1. Don’t write production code until you have a failing test.
2. Don’t write more of a test than is sufficient to fail.
3. Don’t write more code than it is sufficient to make the failing test pass.

[1] Martin, Robert C. "Professionalism and test-driven development.“ *Ieee Software* 24.3 (2007): 32-36.

Benefits of TDD

Better design:

- Makes us write testable (loosely-coupled) units;
- Forces us to be the first consumers of our APIs.

Better implementation:

- Everything is tested;
- Build a suite of regression tests as you go;
- Faults are discovered early.

TDD in practice

Many studies (e.g.: [1, 2]) showed that, in industrial settings, TDD generally leads to:

- Much lower defect density
- Significant reduction of post-release maintenance costs
- More time needed to deliver

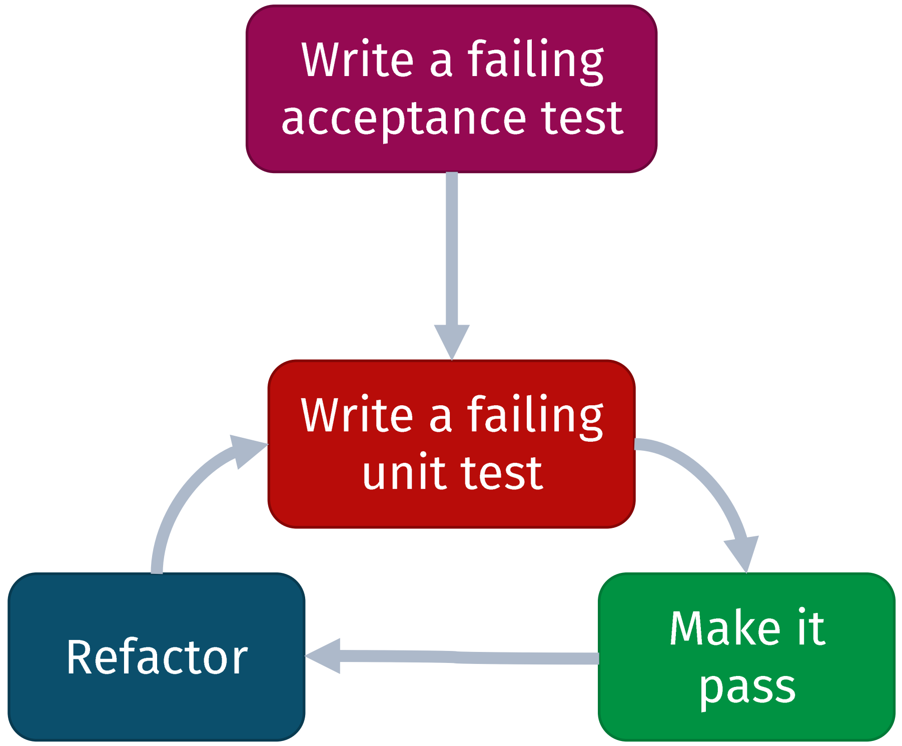
[1] Nagappan, Nachiappan, et al. "Realizing quality improvement through test driven development: results and experiences of four industrial teams." *Empirical Software Engineering* 13.3 (2008): 289-302.
[2] George, Boby, and Laurie Williams. "A structured experiment of test-driven development." *Information and software Technology* 46.5 (2004): 337-342.

Getting started with the TDD process

How do we start with TDD?

And how do we know when to stop?

- Start by writing a failing acceptance test
- Go on with the inner loop
- When the acceptance test goes green, you know you can stop.



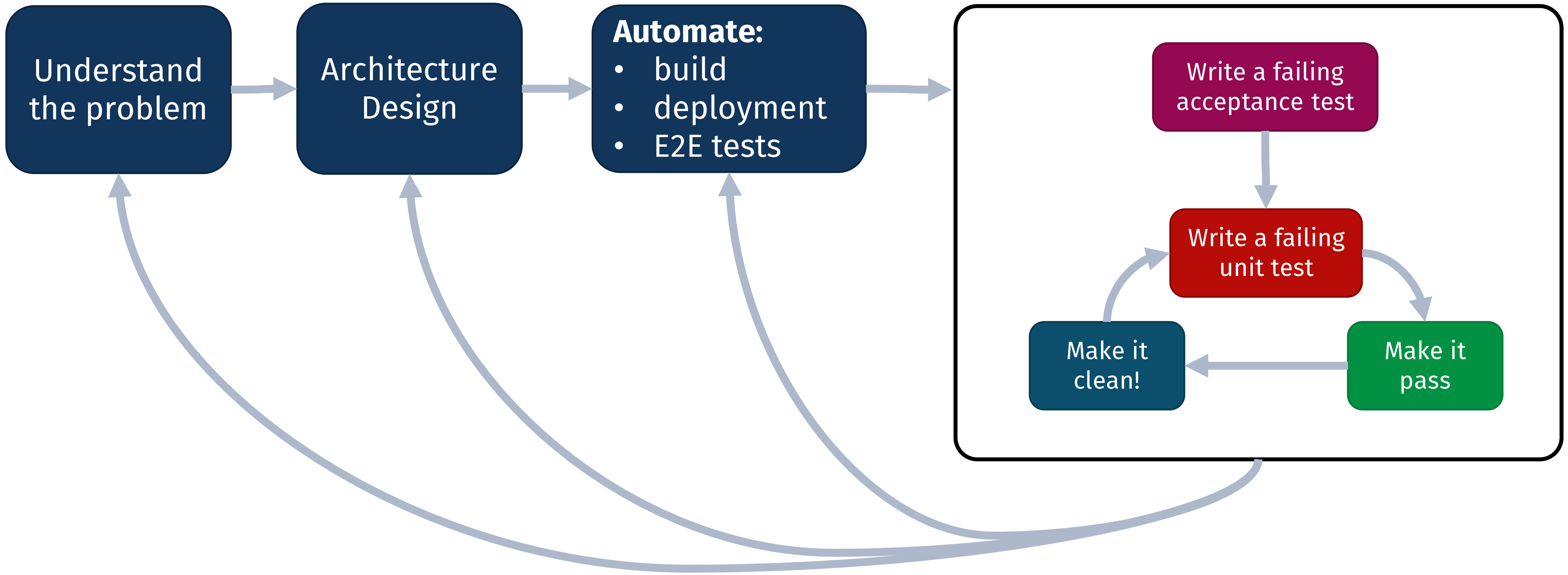
The first-feature paradox

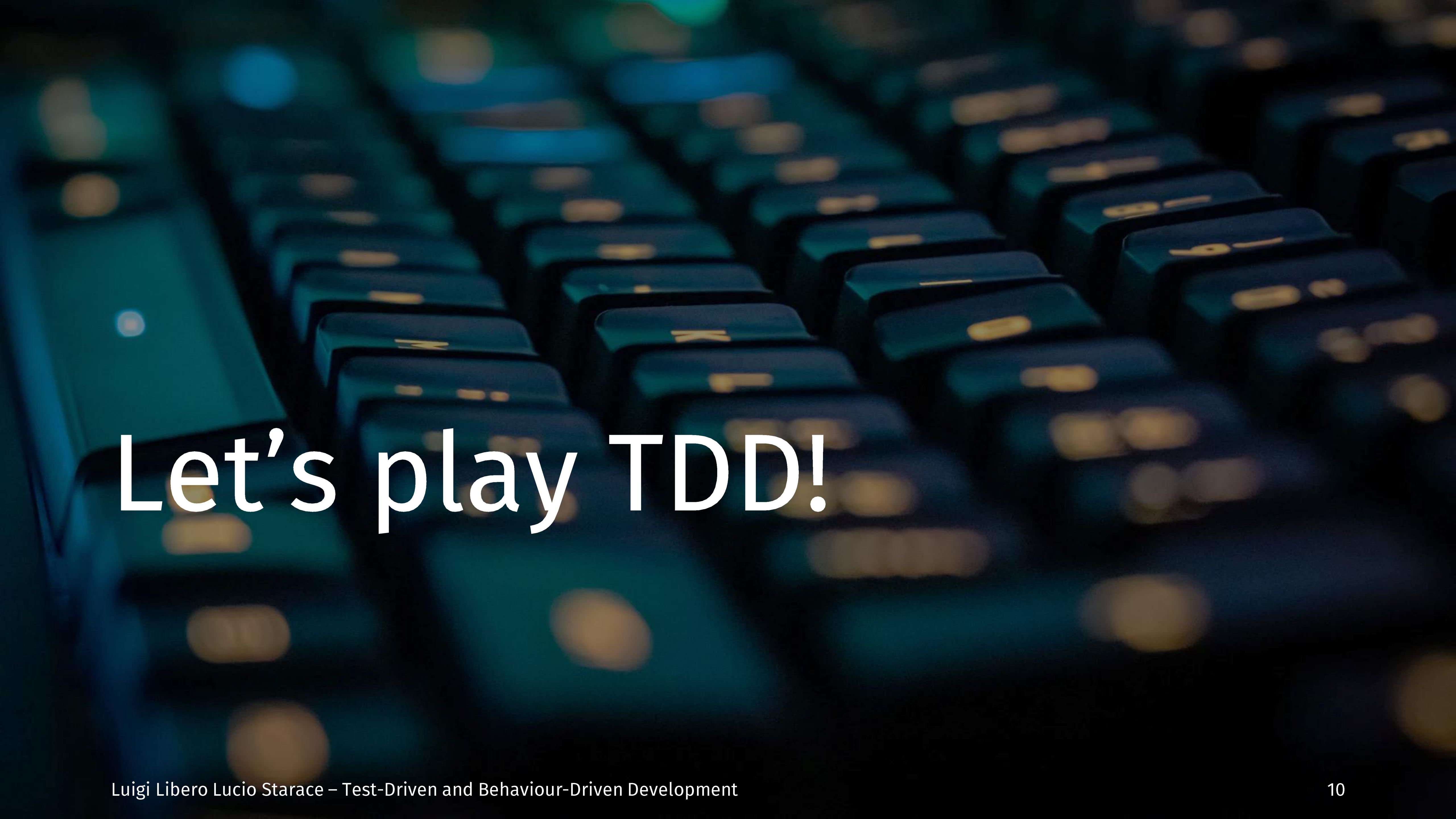
- Running our *very first* acceptance test won’t be easy.
- Acceptance tests should run E2E, so we should have a whole automated build-deploy-test cycle, of a system which has no feature yet!
- That’s quite a lot of work, just to see our first test fail!
- A **“Walking Skeleton”** is a tiny implementation of the system that performs a small E2E function.

Taking the Skeletons out of the closet

- We can’t automate build-deploy-test without an overall idea of the underlying architecture.
- When we decide the shape of our skeleton, we start making high-level design (architectural) choices for our software.
- We don’t need much detail, just the bare minimum to automate E2E testing.
- The design for the walking skeleton should be doable in a few minutes on a whiteboard.

TDD Process – the whole picture





Let’s play TDD!

Let’s play TDD!

We’re the developers of **KeepMovies**, an utility that allows users to keep a list of movies they wish to watch.

So far, we have a basic `Movie` POJO with title, year, and genre properties, and an empty `KeepMovies` class. You can check out our GitHub repository here: https://github.com/luistar/keepmovies-tdd-demo-starter

Let’s get down to business!

Let’s play TDD – First feature

As a user, I want to save movies so I won't forget about them.

Let's play TDD – Second feature

As a user, I want to sort the movies I saved either lexicographically or by release date.

Let’s play TDD – Third feature

As a user, I want to mark movies as already watched.

Let’s play TDD – Fourth feature

As a user, I want to be able to remove movies I marked as "already watched" from my KeepMovies.

Let’s play TDD – Fifth feature

As a user, I want to group my movies by genre and year.

Let’s play TDD – Sixth feature

As a user, I want to group my movies by custom criteria.

References

- Freeman, Steve, and Nat Pryce. "*Growing object-oriented software, guided by tests*". Pearson Education, 2009.
- Martin, Robert C. "*Professionalism and test-driven development.*" *Ieee Software* 24.3 (2007): 32-36.
- Beck, Kent. "*Test-Driven Development: By Example*". Addison-Wesley Professional, 2002.