ay_of_the_web_tester) (/about)

Getting Started (/) Videos (/videos) Courses (/bootcamp)

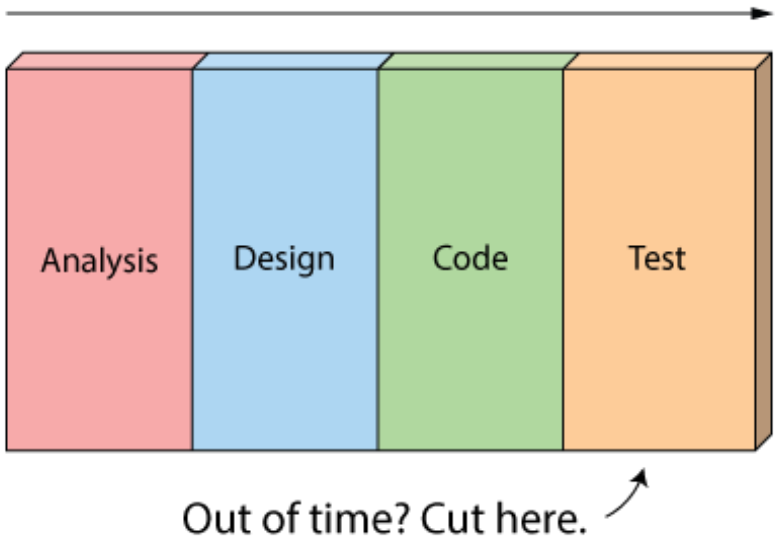
Learn More

Agile vs Waterfall

Waterfall challenges

Traditional Waterfall treats analysis, design, coding, and testing as discrete phases in a software project. This worked OK when the cost of change was high. But now that it's low it hurts us in a couple of ways.

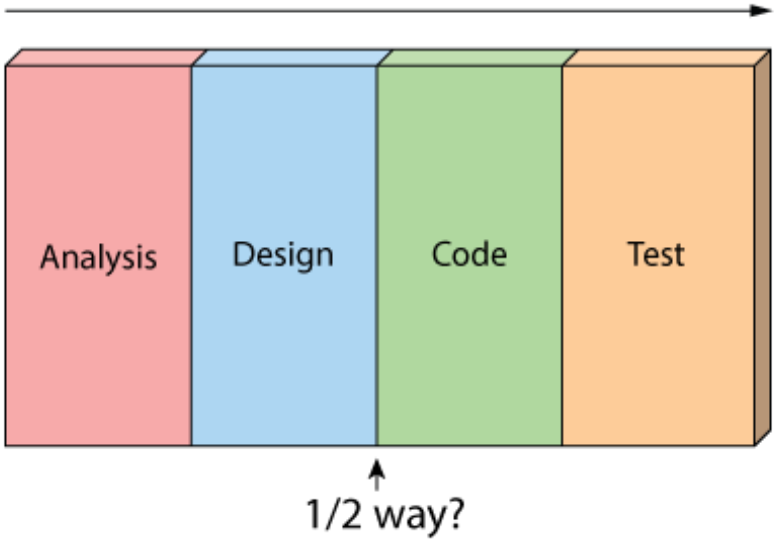
Poor quality



First off, when the project starts to run out of time and money, testing is the only phase left. This means good projects are forced to cut

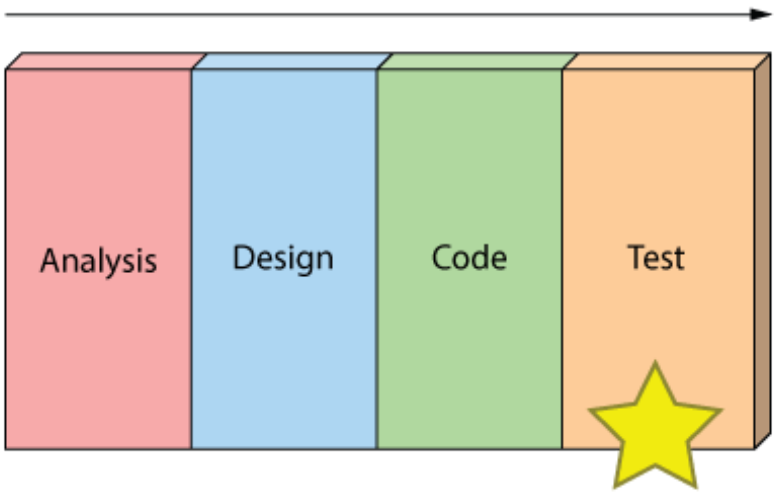
testing short and quality suffers.

Poor visibility



Secondly, because working software isn't produced until the end of the project, you never really know where you are on a Waterfall project. That last 20% of the project always seems to take 80% of the time.

Too risky



Houston we have a problem

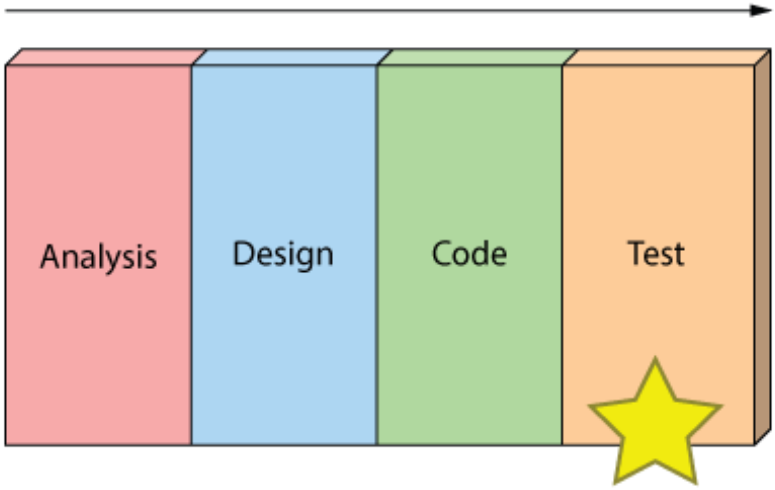
Thirdly you've got schedule risk because you never know if you are going to make it until the end.

You've got technical risk because you don't actually get to test your design or architecture until late in the project.

And you've got product risk because don't even know if you are building the right until it's too late to make any changes.

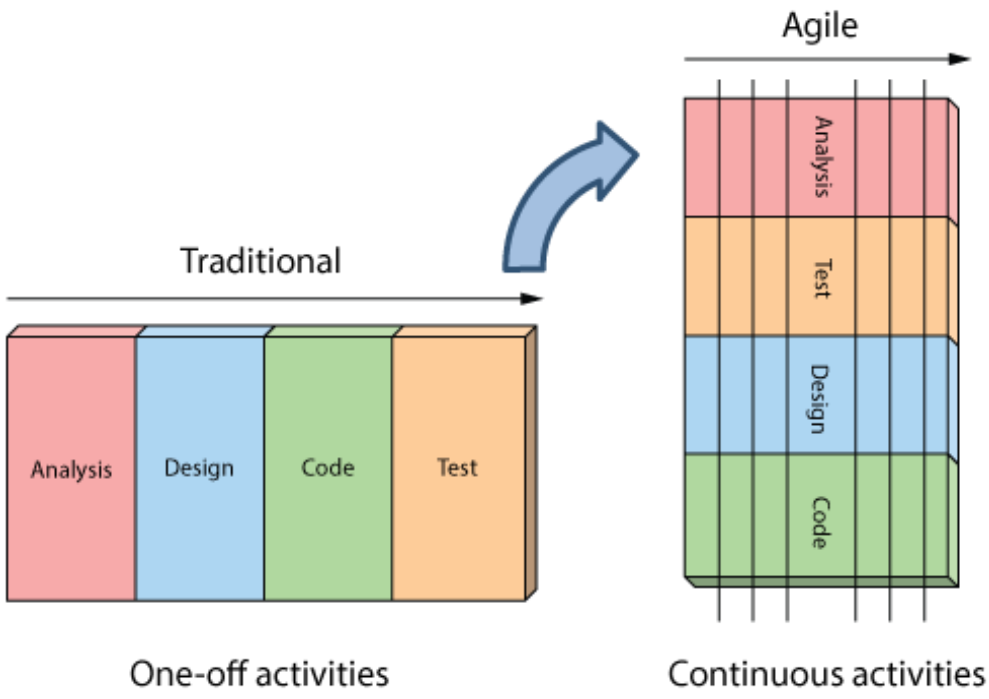
Can't handle change

And finally, most importantly, it's just not a great way for handling change.



'I know what I really want!'

The Agile Approach



Instead of treating these fixed stages Agilists believe these are continuous activities.

By doing them continuously:

- Quality improves because testing starts from day one.
- Visibility improves because you are 1/2 way through the project when you have built 1/2 the features.
- Risk is reduced because you are getting feedback early, and
- Customers are happy because they can make changes without paying exorbitant costs.

← Previous (/agile_myths)

Next → (/user_stories)

IN A NUTSHELL

What is Agile? (/what_is_agile)

How does it work? (/how_does_it_work)

How is it different? (/how_is_it_different)

Agile Myths (/agile_myths)

Agile vs Waterfall (/agile_vs_waterfall)

FUNDAMENTALS

User Stories (/user_stories)

Estimation (/estimation)

Iterations (/iterations)

Planning (/planning)

ENGINEERING

Unit Testing (/unit_testing)

Refactoring (/refactoring)

Continuous Integration (/continuous_integration)

Test Driven Development (/test_driven_development)

CONCEPTS

Burndown Charts (/burndown)

Cone of Uncertainty (/cone_of_uncertainty)

Management by Miracle (/management_by_miracle)

Three Simple Truths (/three_simple_truths)

XPISMS

Bill of Rights (/bill_of_rights)

YAGNI (/yagni)

Yesterday's Weather (/yesterdays_weather)

The Simplest Thing (/simplest_thing)

Testing (/could_possibly_break)

Production Readiness (/production_readiness)