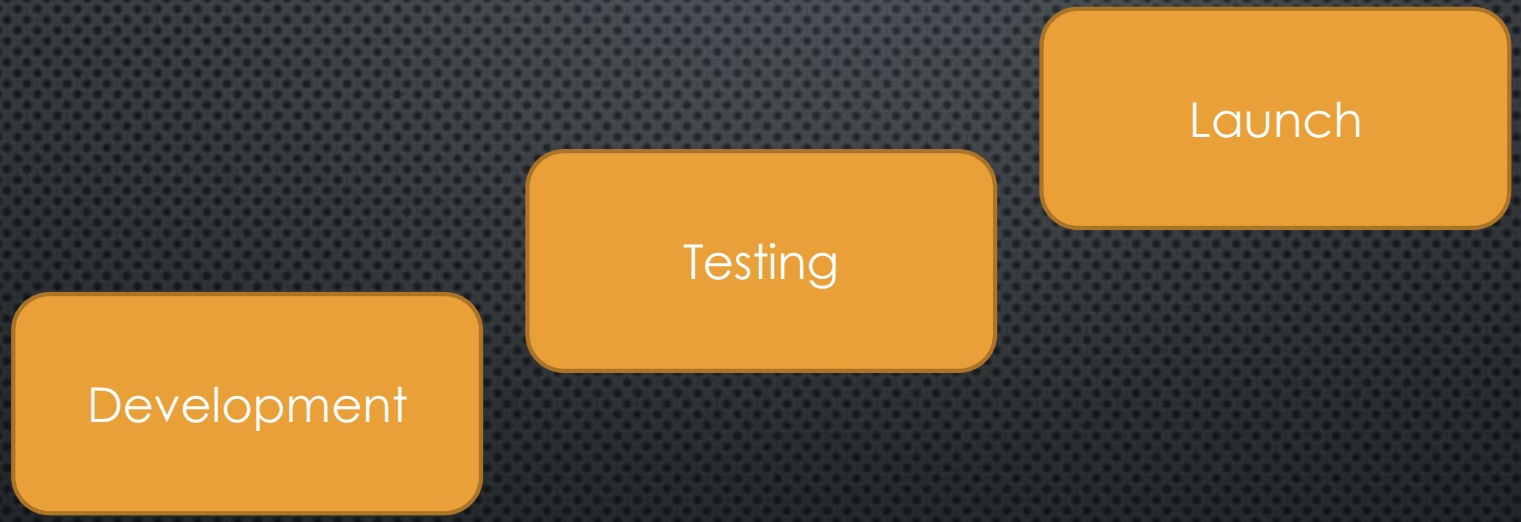
# AGILE PROJECT MANAGEMENT

## Implement agile programs

Traditional project management styles, like waterfall, build in phases



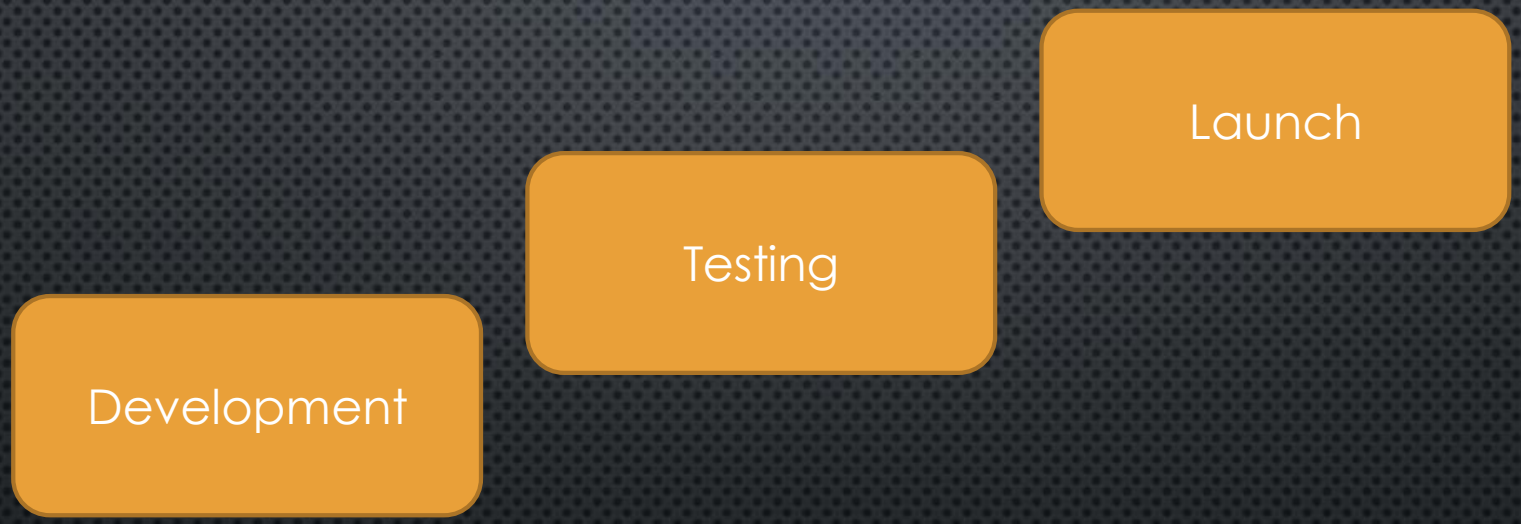

In this case we have a single high-risk release
Once a project passes one phase, it's painful to revisit it because teams are always pressing forward to the next stage

Traditional project management styles often create "critical paths", where the project can't move forward until a blocking issue is resolved

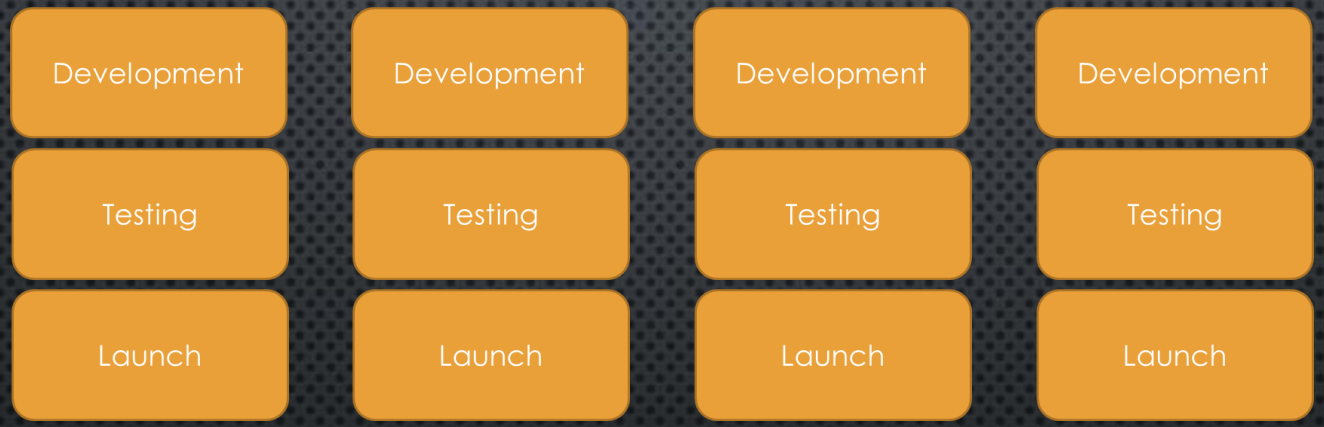
To add insult to injury, the end customer can't interact with the product until it's fully complete

Thus, important issues in the product's design, and code, go undiscovered until release

With an agile project management style, the approach is iterative with regular feedback intervals



With an agile project management style, the approach is iterative with regular feedback intervals




These iterations allow for the team to be diverted to (and productive in) another area of the project while a blocking issue is resolved

This approach removes critical paths
Beside that, iterations let you interact with the product during development
The team has continuous opportunities to build, deliver, learn, and adjust
Teams are prepared to adapt quickly to new requirements

An even greater benefit is shared skill sets among the software team

The team's overlapping skill sets add flexibility to the work in all parts of the team's code base

This way, work and time isn't wasted if the project direction changes

# HOW TO BUILD A GREAT AGILE PROGRAM

The focus of the product owner is to optimize the value of the output of the development team

The product owner defines priorities which the development team needs to take into account when planning and executing their work

The development team can only accept work if there is enough capacity for it

# HOW TO BUILD A GREAT AGILE PROGRAM

The product owner doesn't push work to the team or commit them to arbitrary deadlines

The development team pulls work from the program's backlog when is ready to accept new work

# MECHANISMS IN AN AGILE PROGRAM TO RUN IN AN ITERATIVE WAY

The following are mechanisms which agile programs use to organize, run, and structure work in an iterative way

# Roadmaps

A roadmap outlines how a product or solution develops over time
Roadmaps are composed of initiatives, which are large areas of functionality, and include timelines that communicate when a feature will be available
As the program develops, it's accepted that the roadmap will change–sometimes subtly, sometimes broadly
The goal is to keep the roadmap focused on current market conditions and long-term goals

# Roadmaps Requirements

A roadmap is composed of initiatives which break down into a set of requirements
Agile requirements are short and simple descriptions of required functionality
They evolve over time and finalized based on the team's shared understanding of the customer and the desired product
The definition of each requirements derive from a common understanding achieved by the developer team members via ongoing conversation and collaboration
Full details are defined when the implementation is about to begin

# Roadmaps
# Requirements
# Backlog

The backlog sets the priorities for the agile program
A backlog include all work items: new features, bugs, enhancements, technical or architectural tasks, etc.
The product owner prioritizes the work on the backlog
The development team then uses the prioritized backlog as its single source of truth for what work needs to be done

# Roadmaps
# Requirements
# Backlog
# Agile delivery vehicles

Agile  frameworks use epics and versions (called **delivery vehicles**) to structure development and releases

# Roadmaps
# Requirements
# Backlog
# Agile delivery vehicles
# Agile metrics

Agile teams thrive on metrics:

- Work in progress (WIP) limits: to focus on delivering the highest priority work
- Graphs like burndown and control charts diagrams: to plan and identify bottlenecks

**Roadmaps**
**Requirements**
**Backlog**
**Agile delivery vehicles**
**Agile metrics**
**Trust**

An agile program cannot function without a high level of trust amongst team members