Automated Testing

Gleb Bahmutov, VP of Engineering at Cypress
Gary Schultz, Chief Marketing Officer at BrieBug

What is Automated Testing?

Software that automatically checks whether actual results match expected results to ensure that an application is bug-free.

Automated testing tools use scripts to execute examinations of software, report outcomes, and compare results with earlier tests.

Types of Tests

- Unit Tests - Testing a single unit of code, component, or module in isolation.
- Component Tests - Testing related functionality in isolation.
- Integration Tests - Testing individually developed components in combination.
- API Tests - Verifying a server interface meets the expectations for functionality, performance and security.
- End-to-End Tests - Testing of a complete application environment that mimics real-world use.
- Visual Snapshots - Comparing rendered application snapshots to find visual differences and regressions.

Why Automated Testing

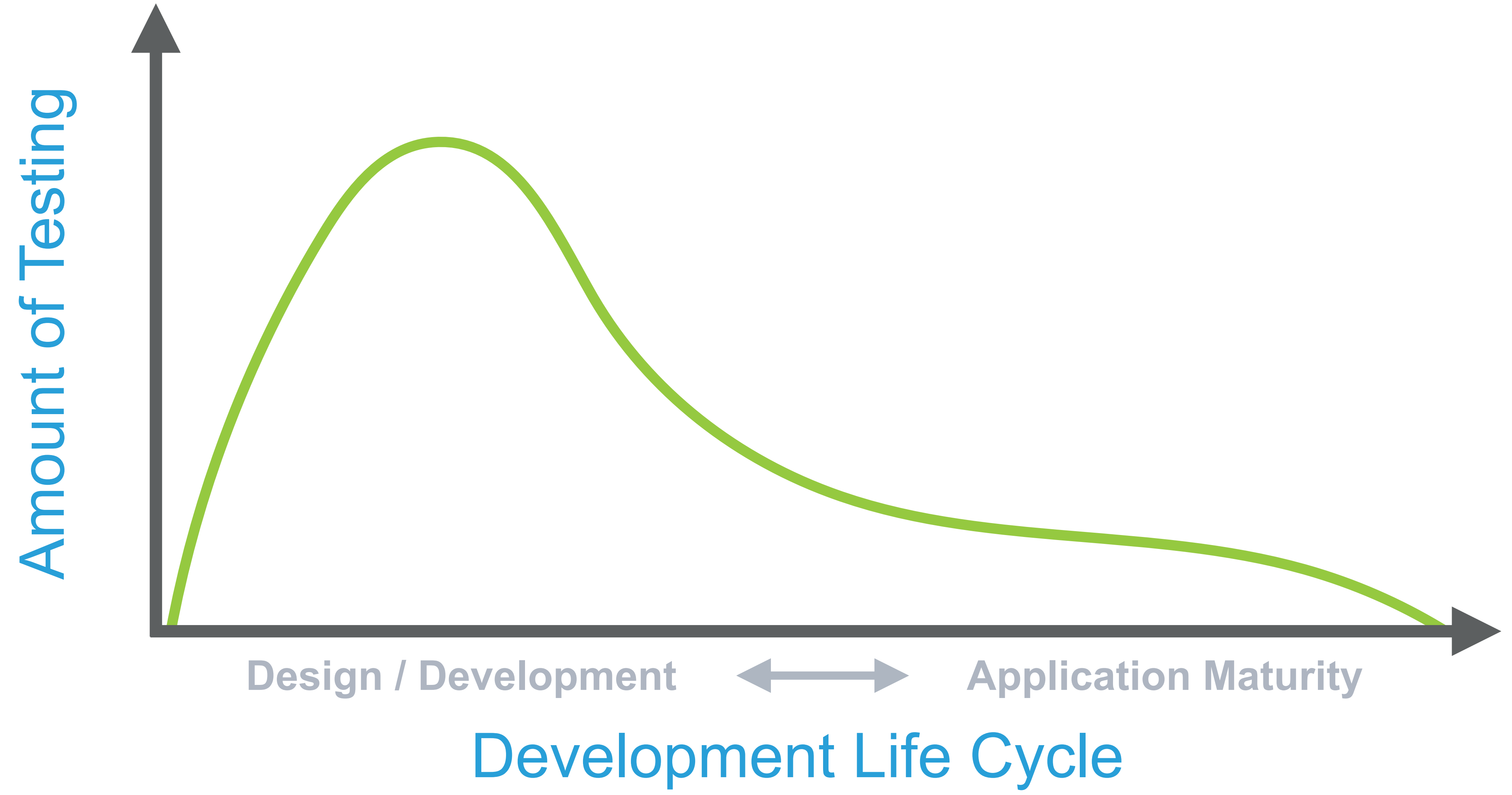
- Ensures predictable user experience - Automated testing constantly validates that your application is working as expected.

- Maximizes efficiency - Less time spent performing manual tests. Since bugs are caught early, developers can solve problems faster with the application design fresh in their minds.

- Prevents regressions- Ensures other functions of application are not broken when adding / modifying new functionality.

What is Shift Left?

Testing earlier in the software development process to find and prevent defects early.



Who is Responsible for Testing?

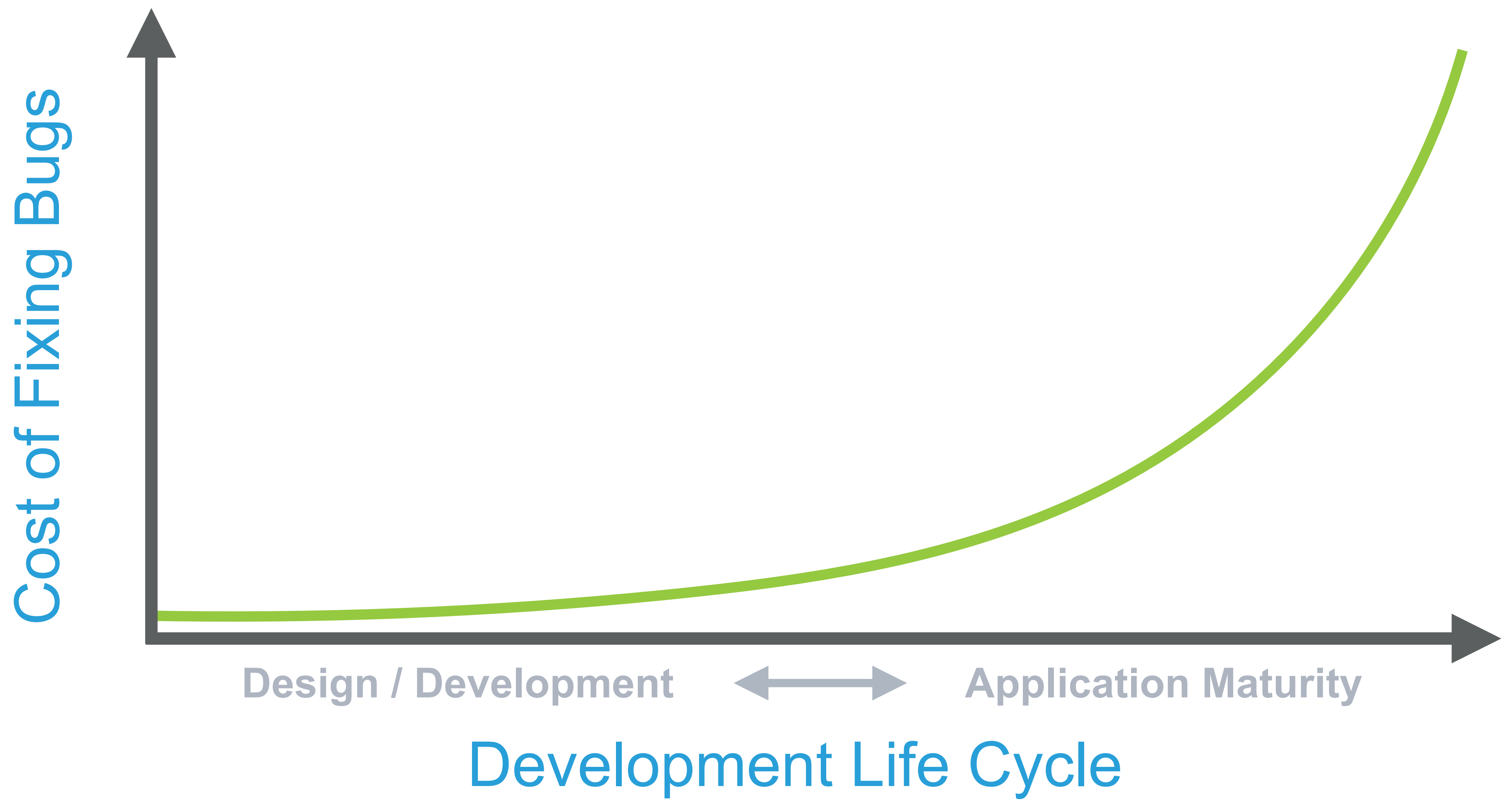
Developers should write tests during their involvement in the development of an application.

Traditional QA-based testing slows down the development process and decreases coding productivity.

When developers write their own tests, they are less reliant on QA and feel more accountable for the quality of their work.

The Cost of Fixing Bugs

Automated testing early in the development life cycle greatly reduces the cost of fixing defects in code.



How to Get Started

- Start with small end-to-end tests
- Set up continuous integration to build and run tests on every commit
- Expand automated tests to include more features of the application
- Establish a long-term testing plan with development team

Common Pitfalls

- Organizations often make the mistake of building intricate automated tests for every part of an application from the beginning. Instead, organizations should start simple and then expand testing piece by piece.
- Using traditional QA-based testing is also a common pitfall. Rather, organizations should have a testing expert (often a outside consultant/partner) to initially guide developers in writing their own automated tests using current best practices.
- Developing homegrown testing tools. Alternatively, use open-source or commercial testing tools will guarantee organizations receive new features, bug fixes, documentation, and support.

TRANSLATING
TECHNOBABBLE

TRANSLATINGTECHNOBABBLE.COM