Agility Technical Practices

Agility technical practices enable the production of high quality, sustainable code that can be changed or evolved easily and quickly

Agility Technical Practices

Short development cycles require that quality, compliance and security is built into all code. If not, those concerns will increase the time to realize value.

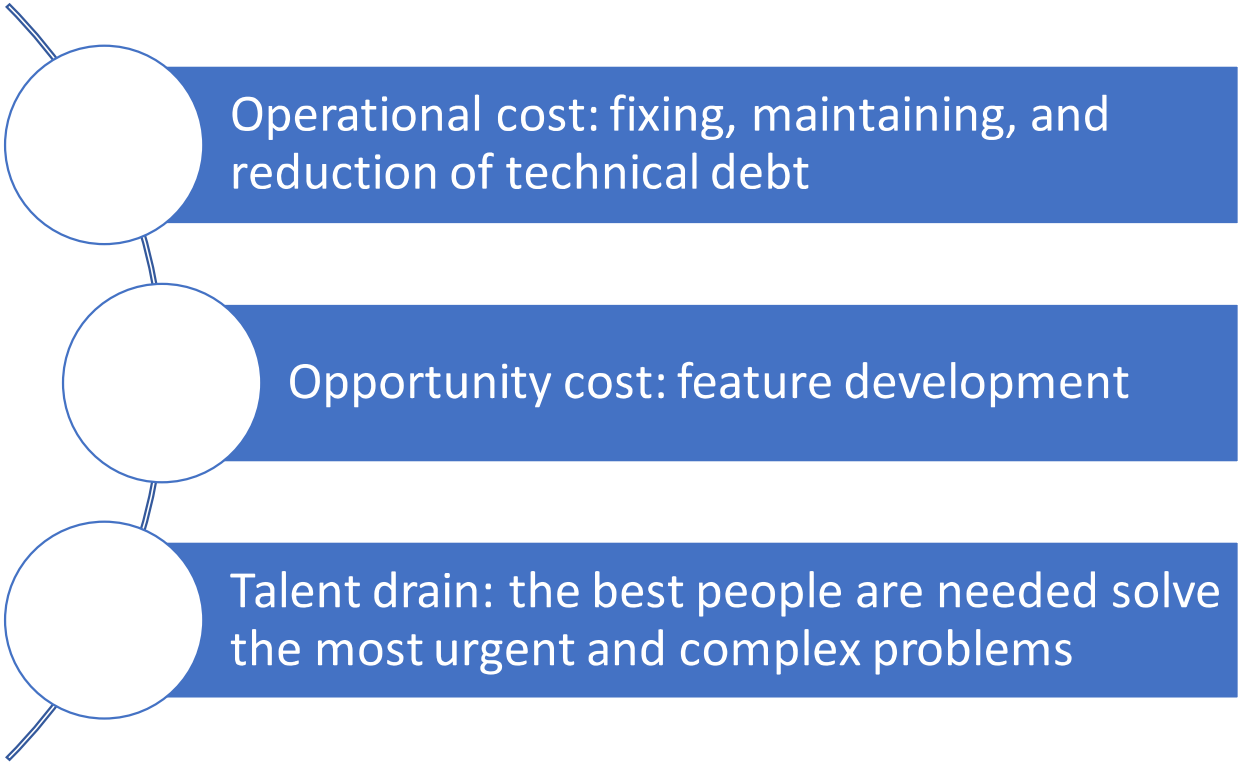
What's changed?

- nature of organizations who rely on software development teams to enable their business has shifted to a project driven to a product driven model
- time to realize value in a marketplace can make or break an organization
- organizations must be able to quickly respond to changes based on the needs of their customers and the market

In 2002, the National Institute of Software Technology estimated that software error cost the US economy $60 billion per year

The Hidden Cost of Code

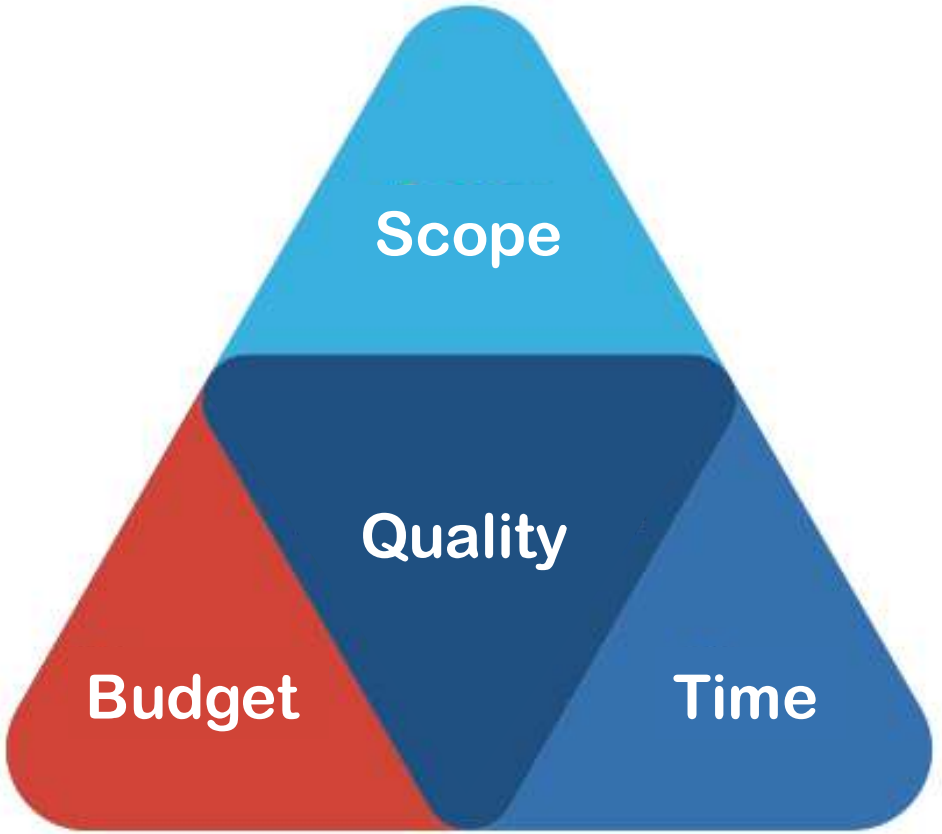
Developers spend 80% of their time reading, troubleshooting, and fixing existing code



Low Quality Code Increases Scope

With time, scope and budget, if one changes the others will also change in proportion

If there is inflexibility on any leg of the triangle when another changes, the only thing that can suffer is quality



The Iron Triangle

Behaviors Required for Technical Agility

Collaboration	The knowledge and experience of a team is more than the sum of its parts
CLEAN code	"We can infer virtually all good software development skills by understanding just a handful of code qualities."
Integrate continuously	Continuous integration provides immediate feedback on code quality
Test first, design last	Write tests to validate behaviors before writing code, then organize the code to be logical, concise, and readable
Automate tests	Automated testing provides developers with instant feedback on quality and enables behaviors visible to users to be regression tested with every build
Manage technical debt	Strive for 100% test coverage of behaviors, though it isn't always achievable. Writing tests before code translates to a higher percentage of code coverage

7

Tactics to Enable Technical Agility

Use version control for everything	Version Control code and non-code files that are critical to a build (config files, scripts, stored procedures, etc.) contribute to releases being unstable and make roll-backs risky. Version every build.
One-click build end-to-end	Automate an end-to-end build process, where developers can compile code and run all tests in their local environment. When all tests pass, the code is automatically checked in and built on the server, where all tests are run again.
Integrate continuously	Continuous integration provides immediate feedback on code quality
Define acceptance criteria	Acceptance criteria define behaviors that can be tested within the software itself, and prevent overbuilding
Write testable code	Automated testing provides developers with instant feedback on whether the code works. Testable code ensures quality and enables automatic regression testing
Keep test coverage where it is needed	Write tests for each behavior before code is written to maximize code coverage with automated tests
Fix broken builds immediately	When the build breaks, no other developers can commit code. Everyone who checks in code is responsible for either fixing or backing out changes immediately.