pillar BUSINESS AGILITY

# Craftsmanship Workshop

## Developer Edition

October 30, 2015

pillar BUSINESS AGILITY

# Agenda

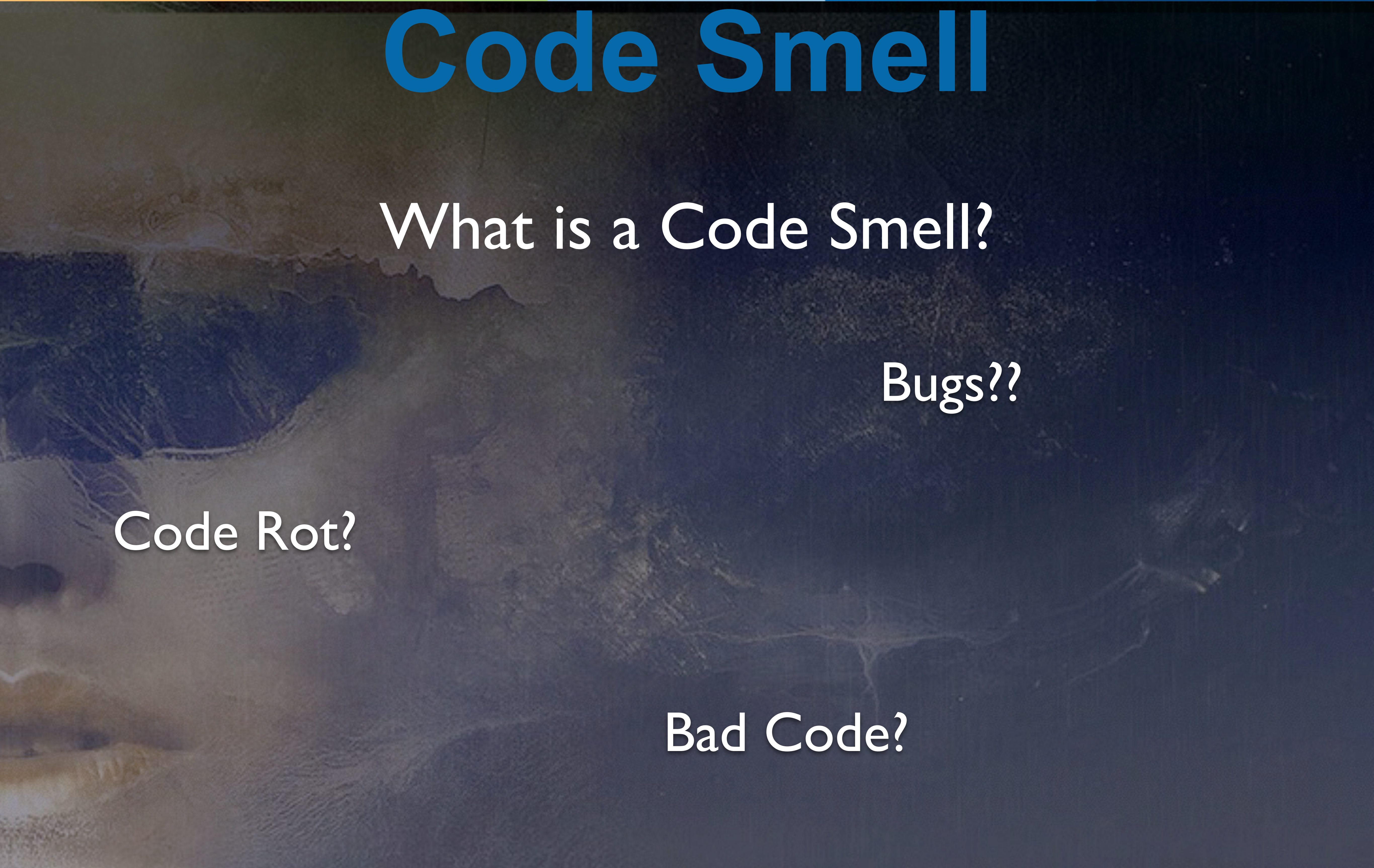
- Introductions
- Code Smells
- Guided walk-through
- Questions

## Introductions

What is your experience with craftsmanship?

pillar BUSINESS AGILITY

## Code Smell



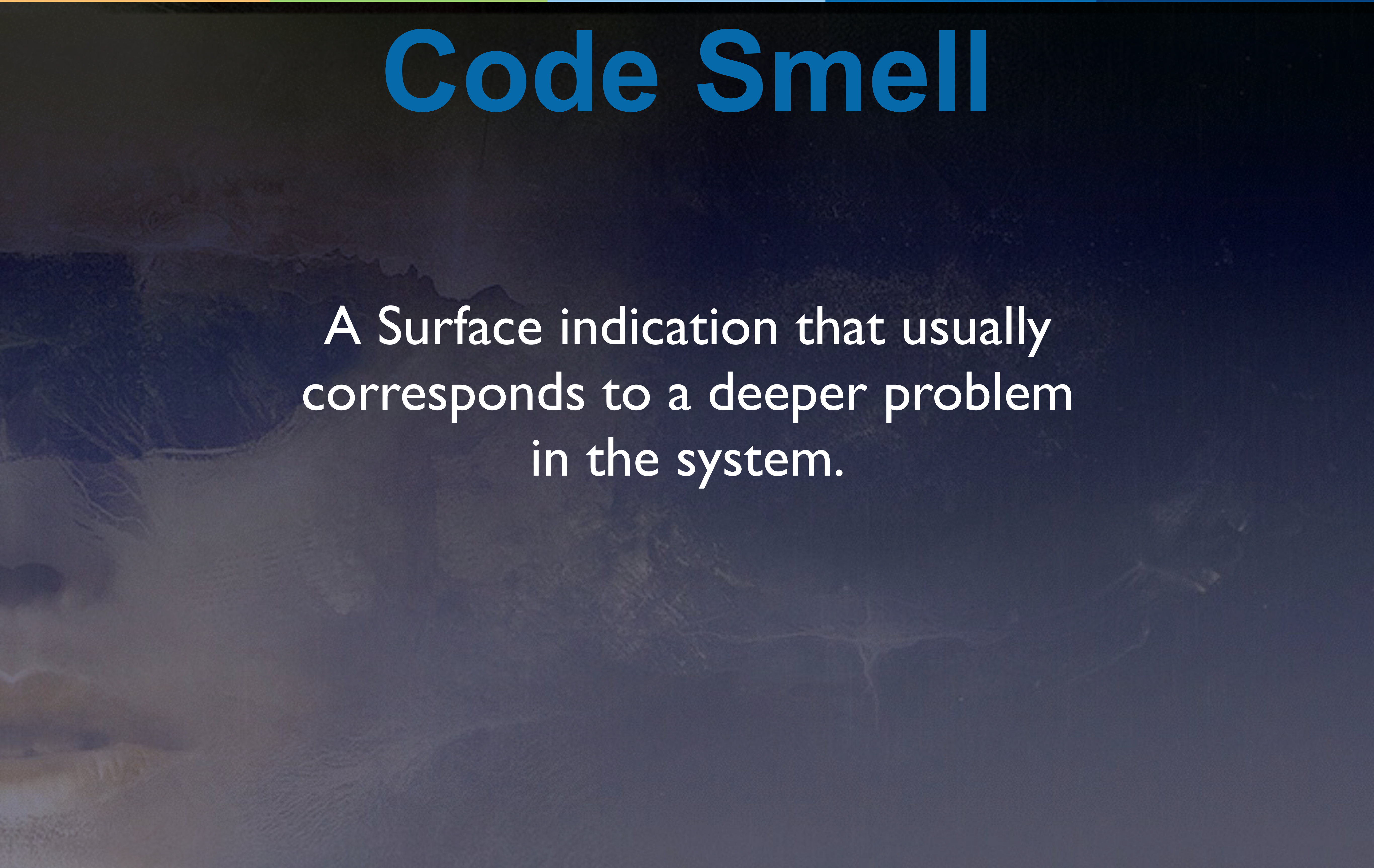
What is a Code Smell?

Bugs??

Code Rot?

Bad Code?

# Code Smell



A Surface indication that usually corresponds to a deeper problem in the system.

pillar BUSINESS AGILITY

# Let’s take a look

pillar BUSINESS AGILITY

## Duplication

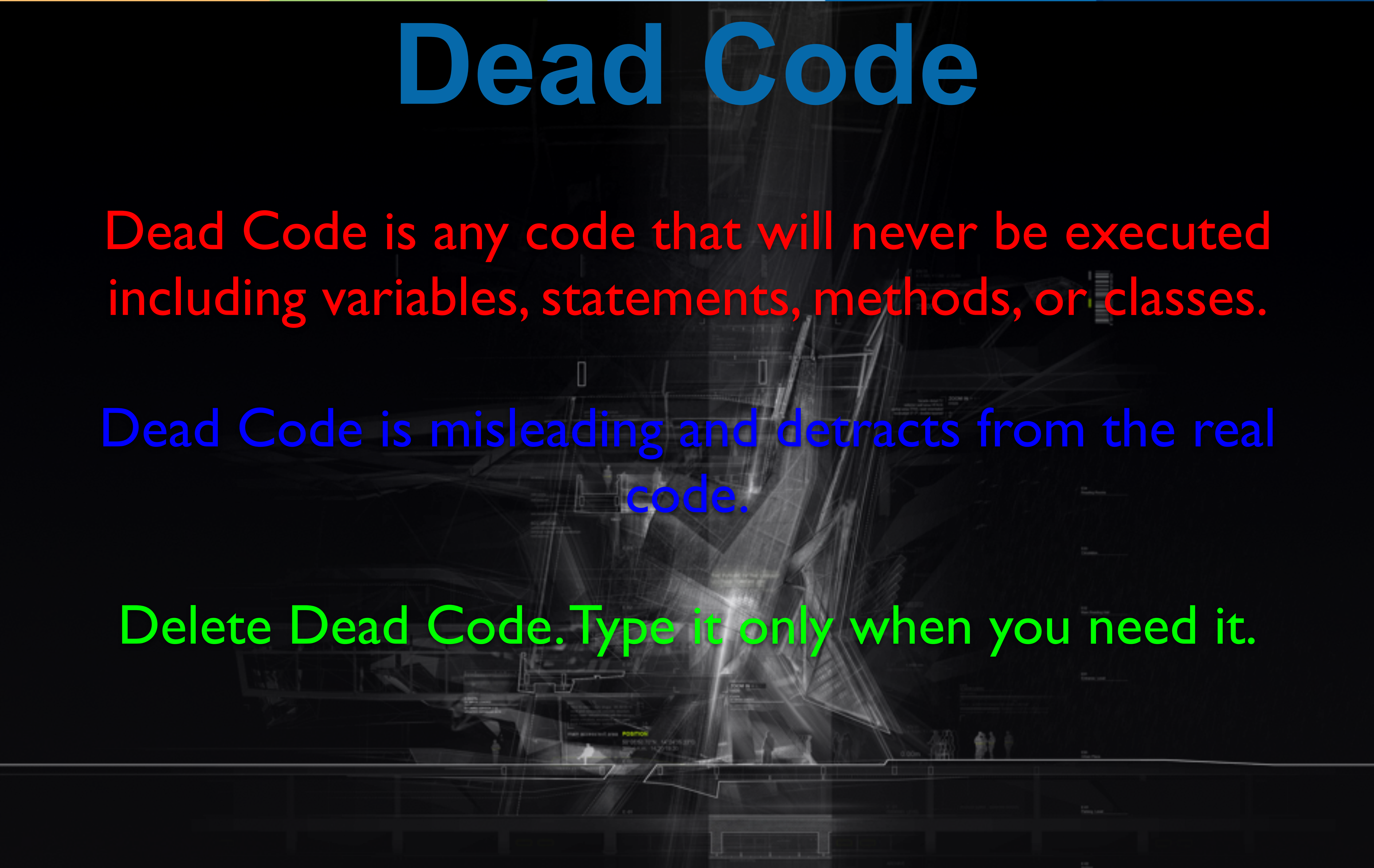
The presence of duplicate code and/or functionality, indicative of the use of copy and paste.

When code is duplicated, it requires more than one place to be updated when changes occur, increasing the chance of bugs with each duplication

Follow the DRY Principle (Don't Repeat Yourself)

pillar BUSINESS AGILITY

## Dead Code



Dead Code is any code that will never be executed including variables, statements, methods, or classes.

Dead Code is misleading and detracts from the real code.

Delete Dead Code. Type it only when you need it.

pillar BUSINESS AGILITY

## Poor Naming

Names of variables, methods, or classes that do not convey their purpose.

Poorly named variables, methods, and classes cause confusion and can lead to accidental bugs and defects in the code.

When creating names, use words that describe the purpose or behaviour. When a name does not make sense, change it to something better.

pillar BUSINESS AGILITY

# Code Comments

Using Comments to explain code instead of letting the code do the talking.

Comments become out of synch with the code leading to misunderstanding the codes intended purpose.

Write code that is self-explanatory. It is far more likely to be updated when it is changed.

pillar BUSINESS AGILITY

# Long Method/Class

Methods and Classes that take a long time to comprehend.

Methods (and Classes) which take a while to comprehend are more difficult to understand leading to an increased chance of bugs.

Refactor large methods and classes into smaller ones. When writing new code, stick to the SRP (Single Responsibility Principle)

pillar BUSINESS AGILITY

# Primitive Obsession

Using primitive data types to represent domain objects.

Grouping logical pieces together into objects helps to segregate functionality and convey additional meaning and clarity to the code.

Create classes to represent domain objects instead of using groups of primitive data types.

pillar BUSINESS AGILITY

# Questions?

pillar BUSINESS AGILITY

# Fred Estabrook

festabrook@pillartechnology.com

@morodomo