# Configuring Behavior of Mocks

**Nicolae Caprarescu**
FULL-STACK SOFTWARE DEVELOPMENT CONSULTANT

www.properjava.com

# Overview

- Configure the mocks we created in the previous module
- Write tests using different types of mocks
- Use argument matchers
- Configure exception throwing in mocks

# The Three Stages of Using Mocks in EasyMock

**Create mocks: instantiate the actual mock objects**

**Record mocks: register expected behavior (expected method calls)**

**Replay mocks: check that the expected ‘recordings’ actually happened during the test’s execution**

# Demo

This module is mostly demo

We'll kick off with configuring some mocks, so that we can finalize the test we started in the previous module

# Strict Mock vs. Nice Mock

## Strict Mock

Throws an exception for any unexpected method calls it experiences

## Nice Mock

Allows all method calls by default, without throwing an exception

Returns appropriate empty values when its methods are called, such as 0, null, or false

# Demo

Let’s write an annotation-based test!

# Demo

We'll now test the scenario where PensionReady approves an account opening

# Demo

We'll add a new collaborator to the AccountOpeningService and handle its effects on the existing tests

# Argument Matchers

Increased flexibility for configuring mock expectations

EasyMock compares arguments of actual method calls with the ones from mock expectations using equals() (and Arrays.equals())

Argument matchers allow you to perform that argument matching in a more flexible manner. Predefined matchers include *same*, *isA*, *isNull*, or, you can write your own custom matcher.

# Demo

Let’s see argument matchers in action!

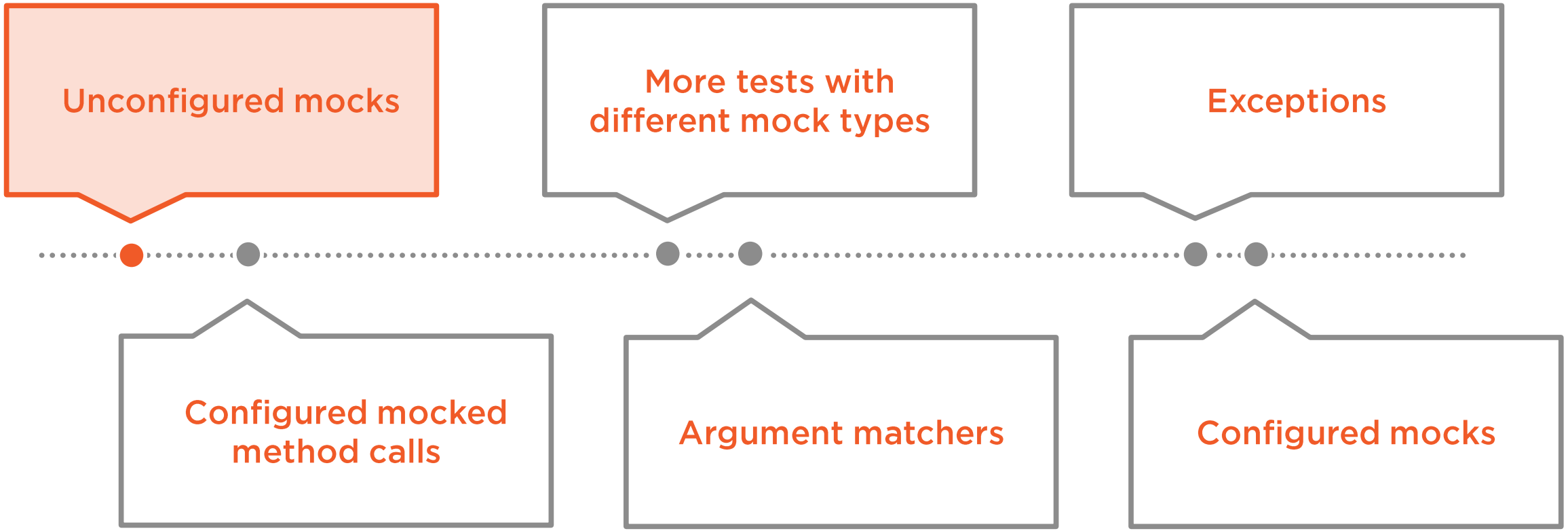
# Demo

This demo focuses on configuring exception throwing on mocks

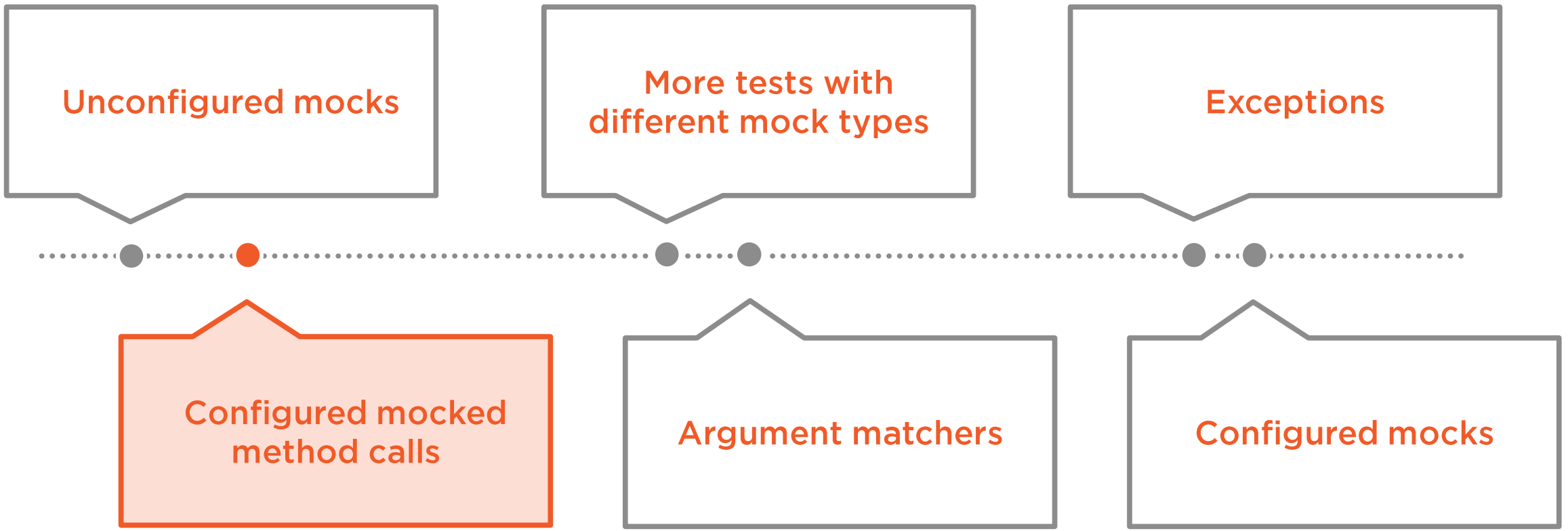
# Demo

This final demo is about configuring expectations and exception throwing on void methods

# Demo Concepts Recap



# Demo Concepts Recap



# Recap: Configuring Mocked Methods

expect...andReturn

EasyMock supports different types of mocks

One nice mock type and two strict mock types are available

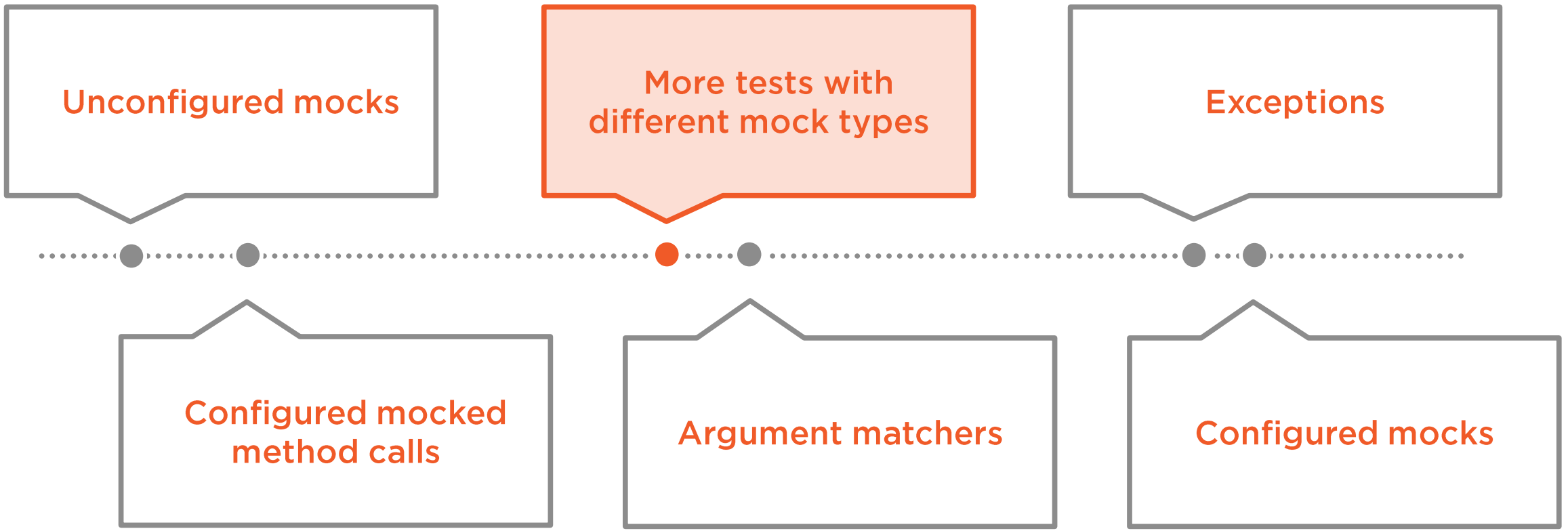
# Recap: EasyMock's Mock Types

MockType.NICE is a nice mock: *niceMock(...)*

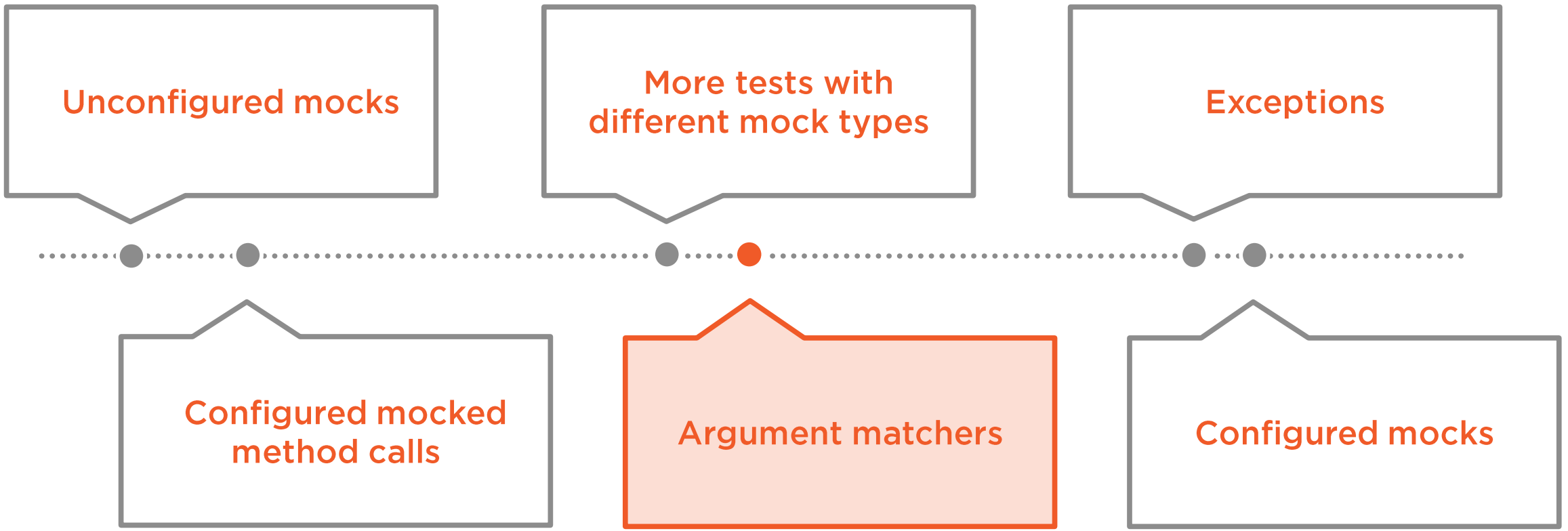
MockType.DEFAULT is a strict mock: *mock(...)*

MockType.STRICT is a strict mock, with added order-checking enabled: *strictMock(...).* We'll demonstrate this in upcoming modules.

# Demo Concepts Recap



# Demo Concepts Recap



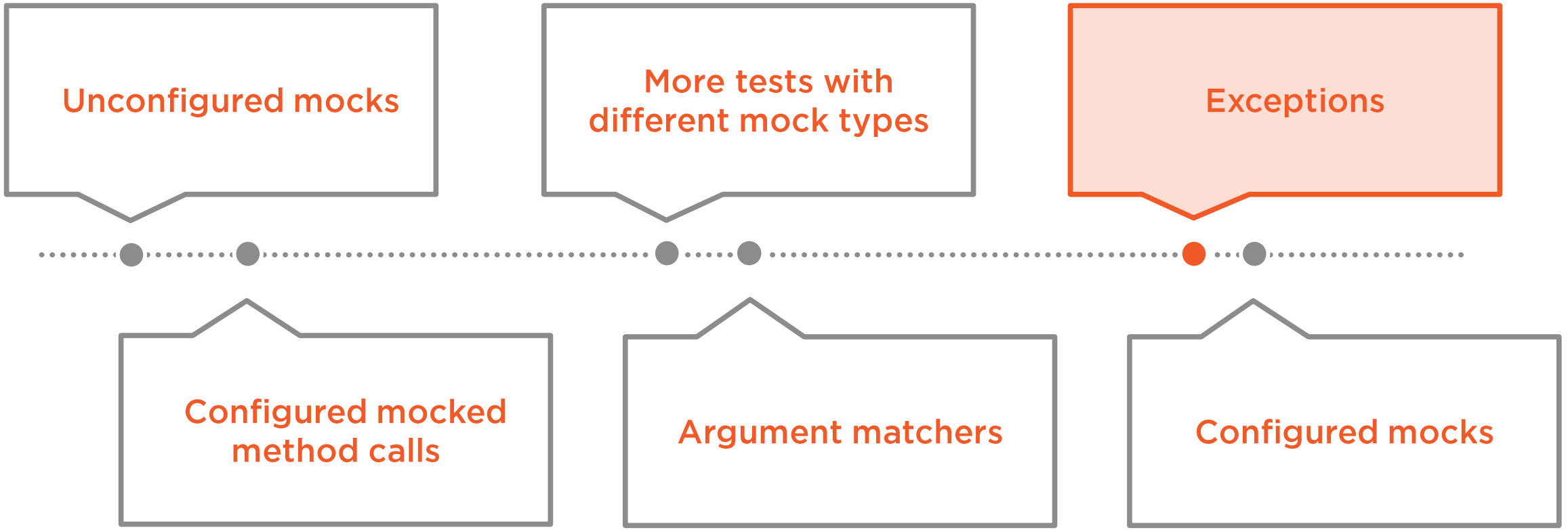
# Recap: Argument Matchers

They add flexibility to how you configure mocked methods

isA(Class clazz), same(), notNull(Class clazz) and other predefined matchers are available. Or write your own custom matcher.

If you use a matcher in a method call, all arguments of that method call must use matchers

# Demo Concepts Recap



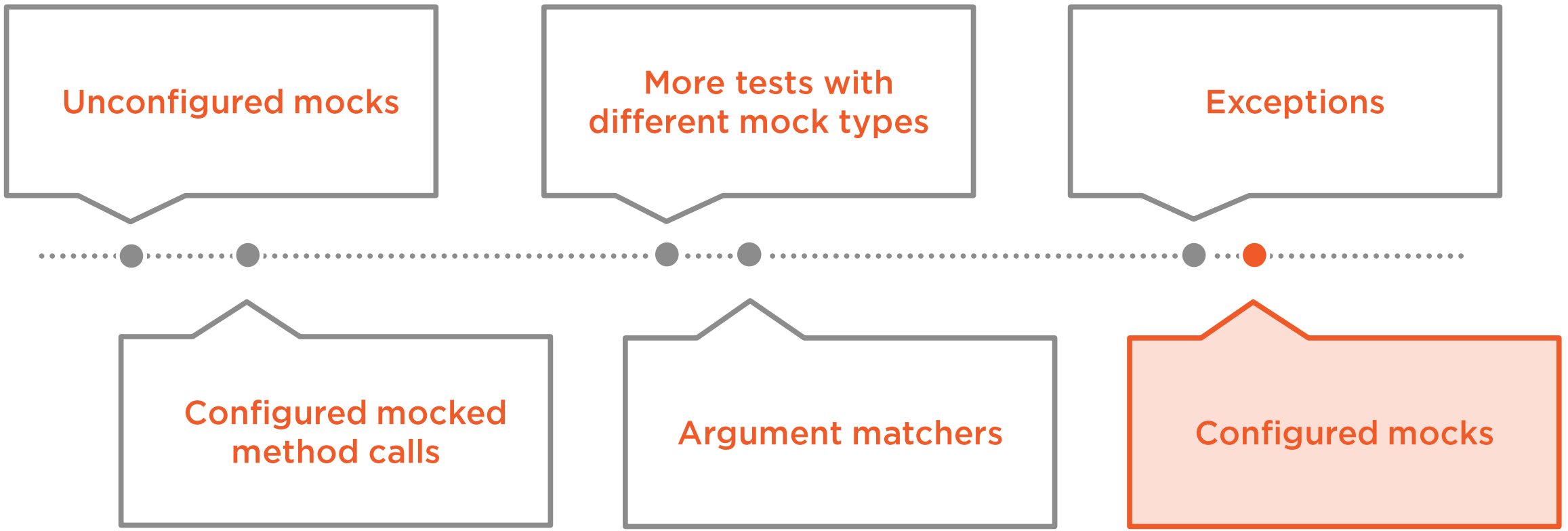
# Recap: Configuring Exceptions

expect...andThrow and expectLastCall...andThrow for void methods

EasyMock offers full support by allowing exception throwing from both void and non-void methods

Can stub methods with both checked and unchecked exceptions (note that checked exceptions can only be thrown from methods that do actually throw them)

# Demo Concepts Recap



# Summary

Started this module with the unconfigured mocks we created in the previous module

Configured mocked methods of both nice and strict mocks

Used argument matchers

Configured exception throwing on mocks

Covered both void and non-void methods

In the next module, we'll focus on how to verify that the expectations we set up actually happened during test execution