ThoughtWorks®
The art of heavy lifting.℠

Towards a testable Application Architecture

Erik Doernenburg
John Horgan

Mock Objects

Dependency Injection

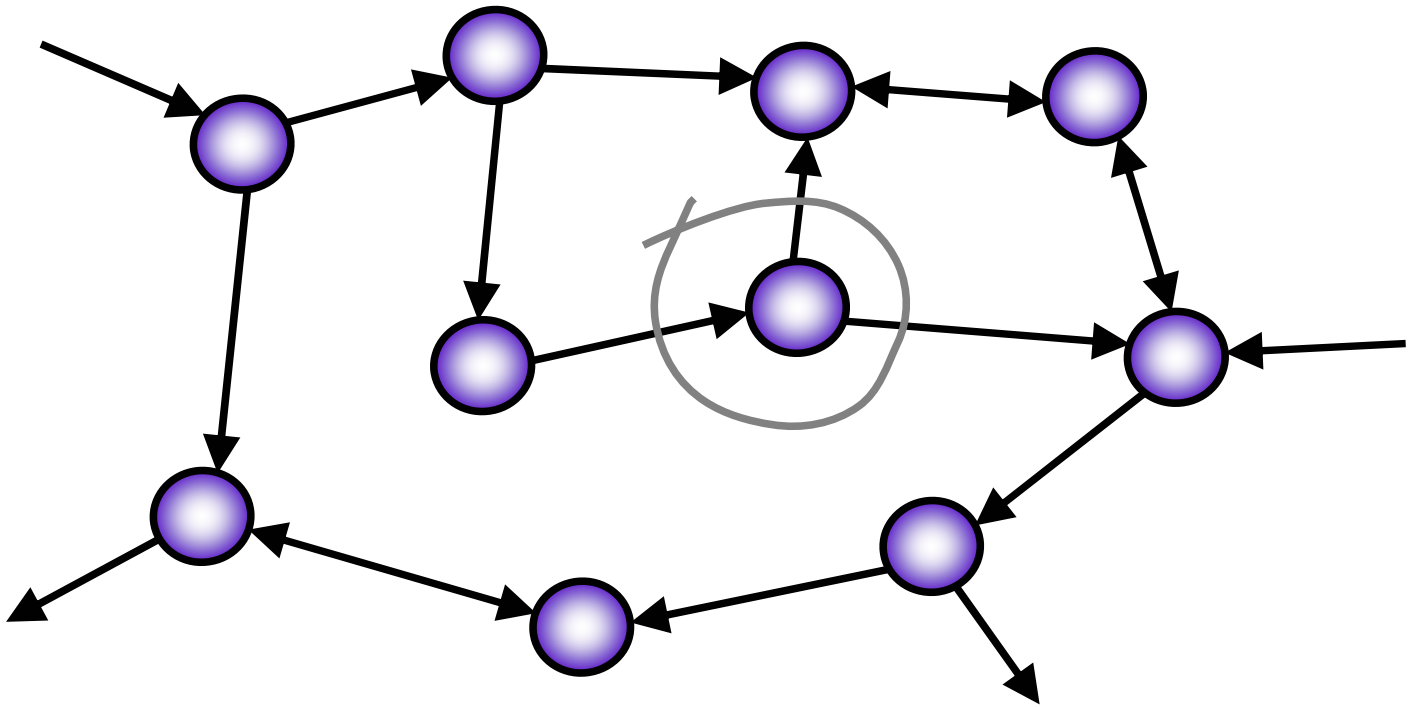
Containers

PicoContainer

Case Study

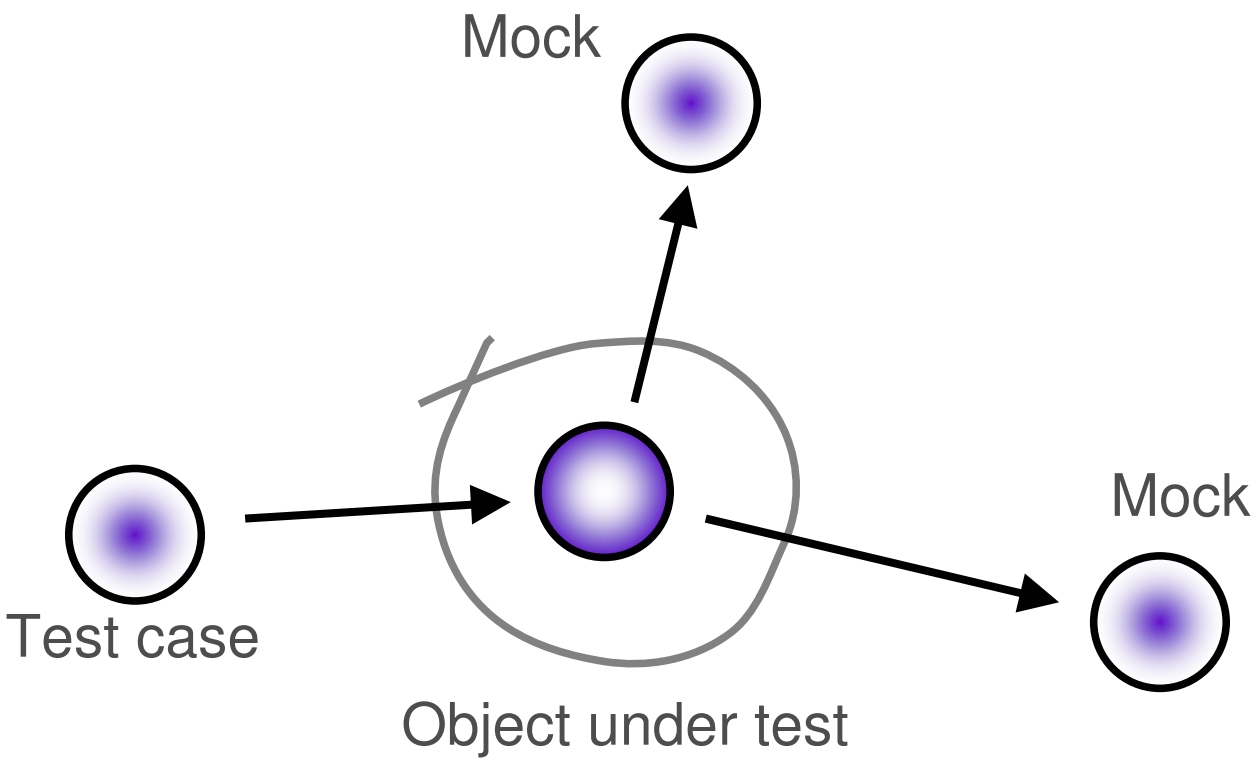
Collaborating Objects

- An object-oriented program is organised as a web of collaborating objects
- To test an object, we test its interactions with its neighbours



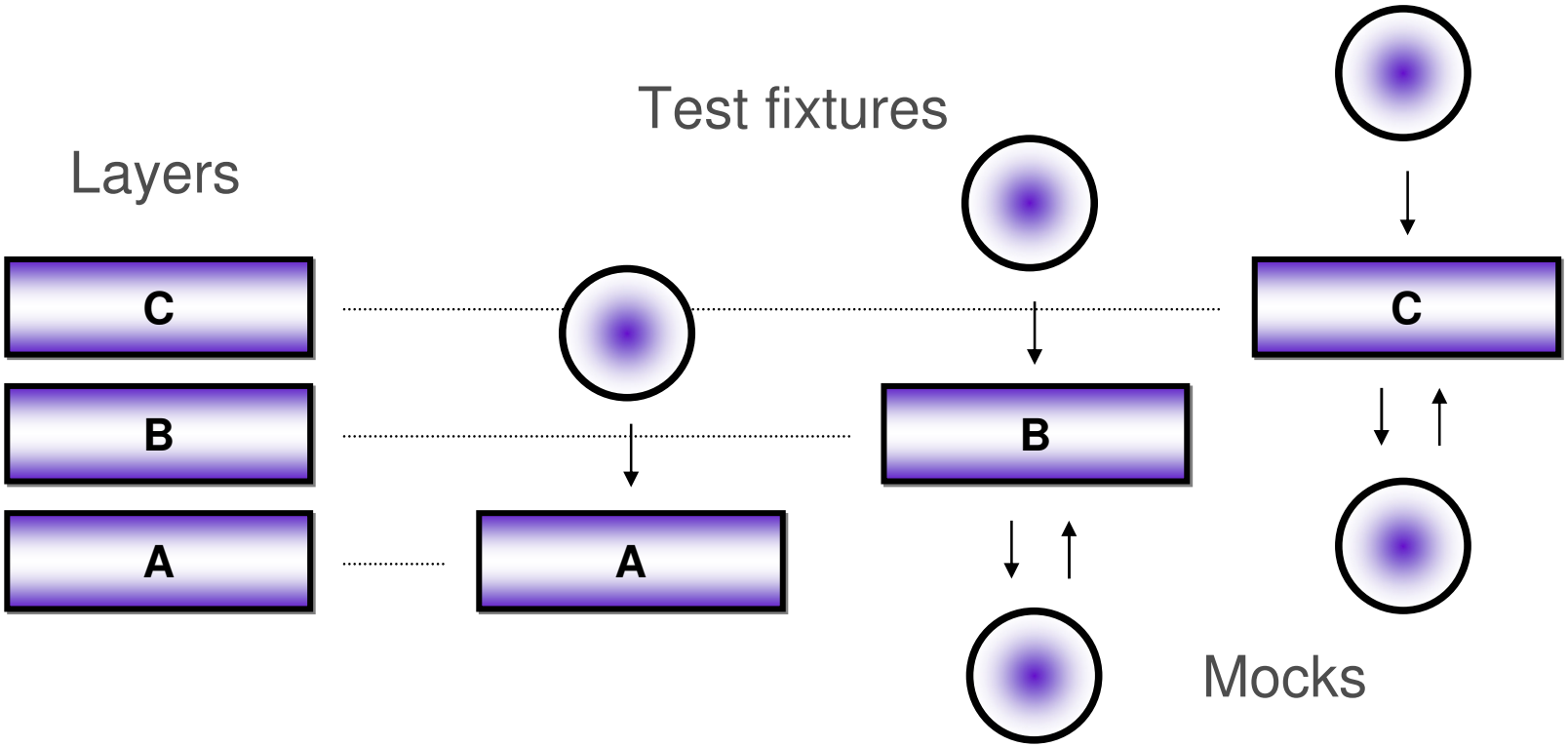
Using Mocks

- To isolate the tested object we replace its neighbours with mock objects



Mocks and layers

- Minimize uncertainty by testing each layer/component in isolation before integration it
- Rely only on already tested components → manage dependency



Mock Objects

Dependency Injection

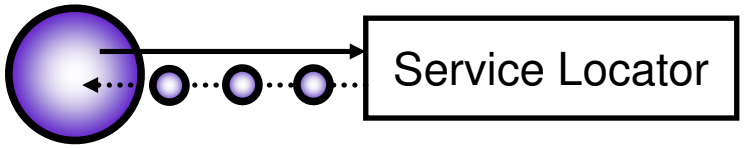
Containers

PicoContainer

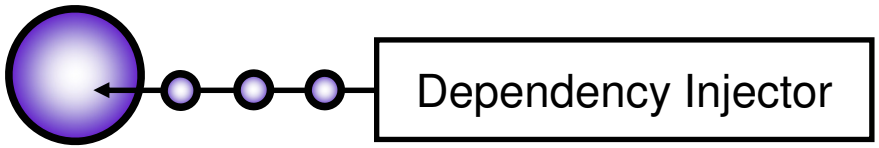
Case Study

Dependency resolution

- Service Locator



- Dependency Injection



Show me the code

ThoughtWorks® The art of heavy lifting.℠

```
public class Foo {
  public Foo() {
  }
  public void doSomething() {
    MyService service = ServiceLocator.getService();
    service().doIt();
  }
```

```
public class Foo {
  protected MyService service;

  public Foo(MyService aService) {
    service = aService;
  }
  public void doSomething() {
    service().doIt();
  }
```

Mock Objects

Dependency Injection

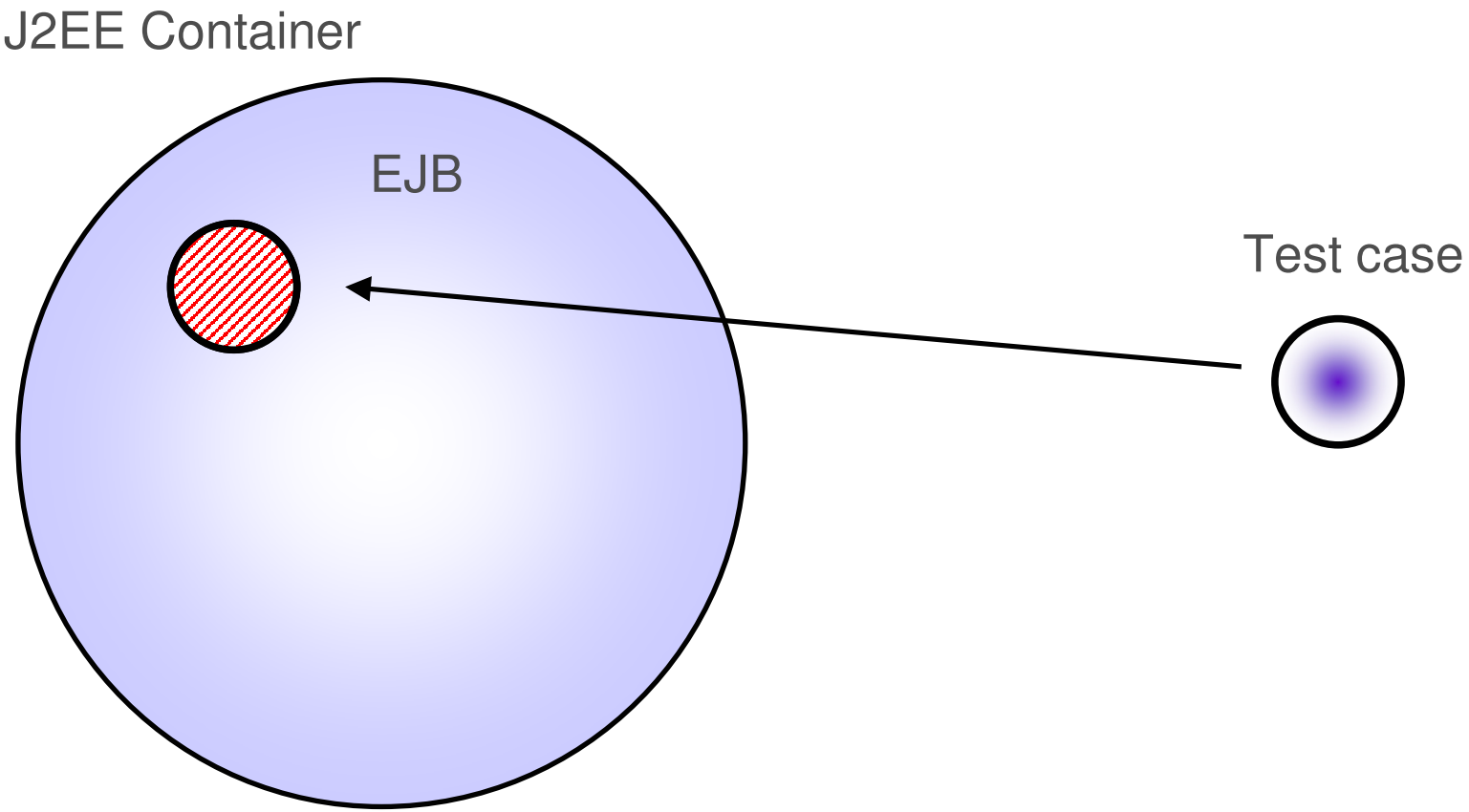
Containers

PicoContainer

Case Study

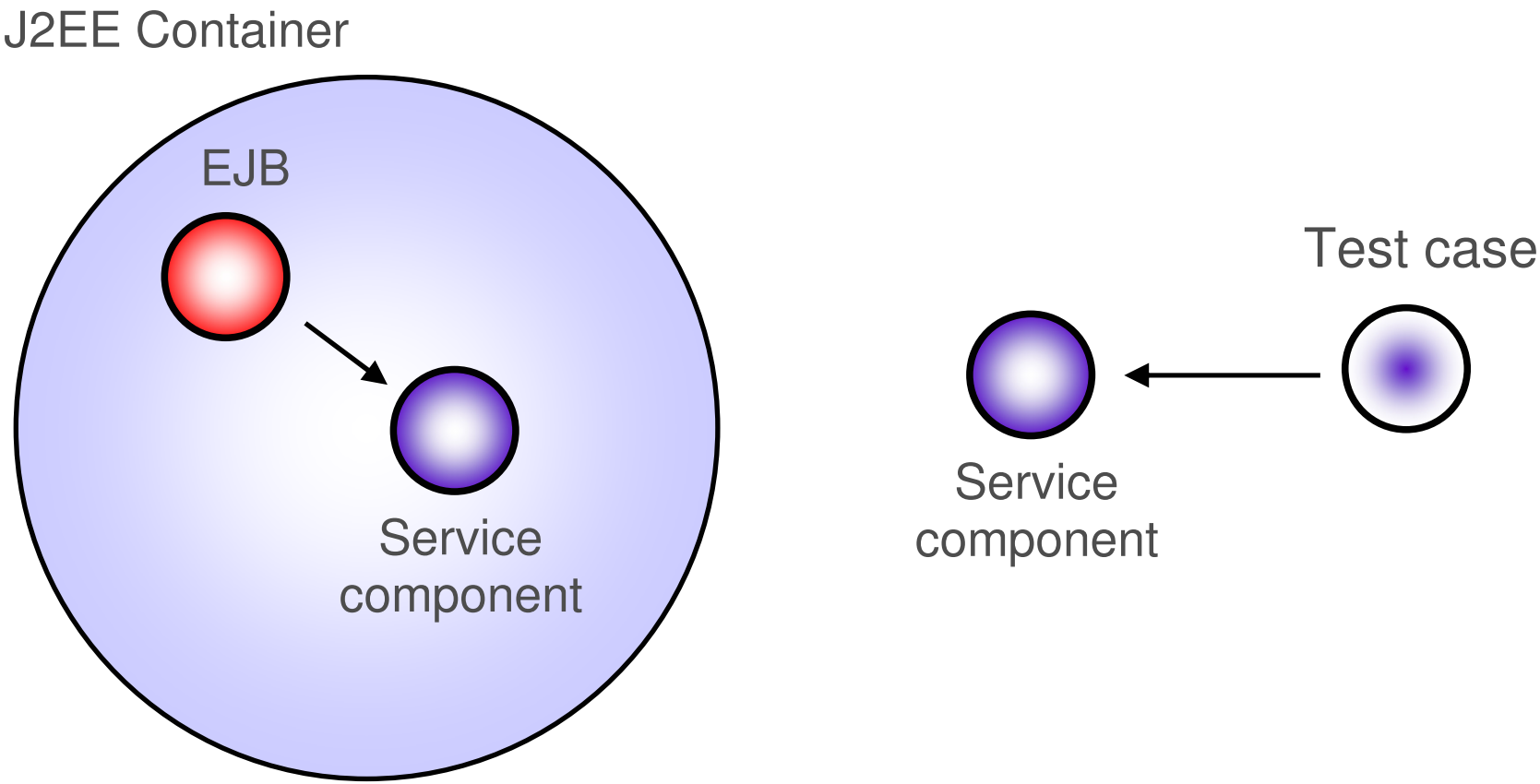
In-container testing

- Containers provide useful infrastructure, but testing a service component inside a container adds complexity.



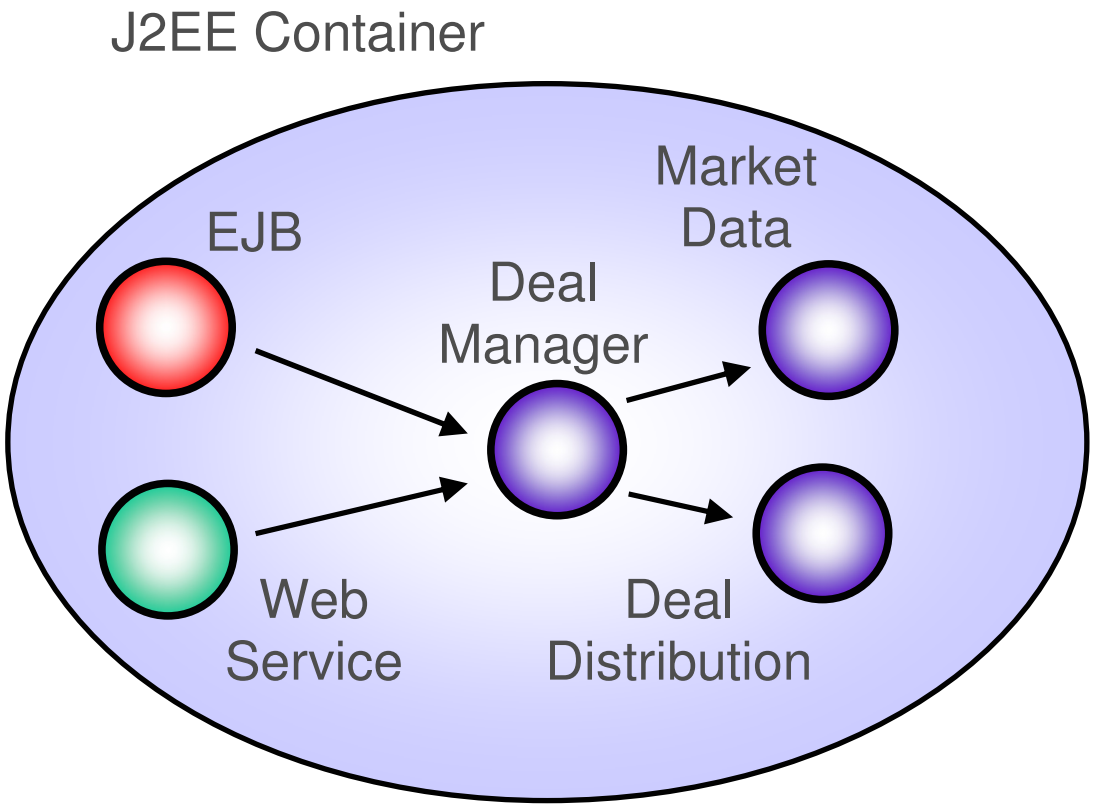
Technology wrappers

- Separating application logic and infrastructure code makes testing of the service component easier.



Choosing the infrastructure

- Can access dependent components with best interface.
- Can add other technology wrappers.



Mock Objects

Dependency Injection

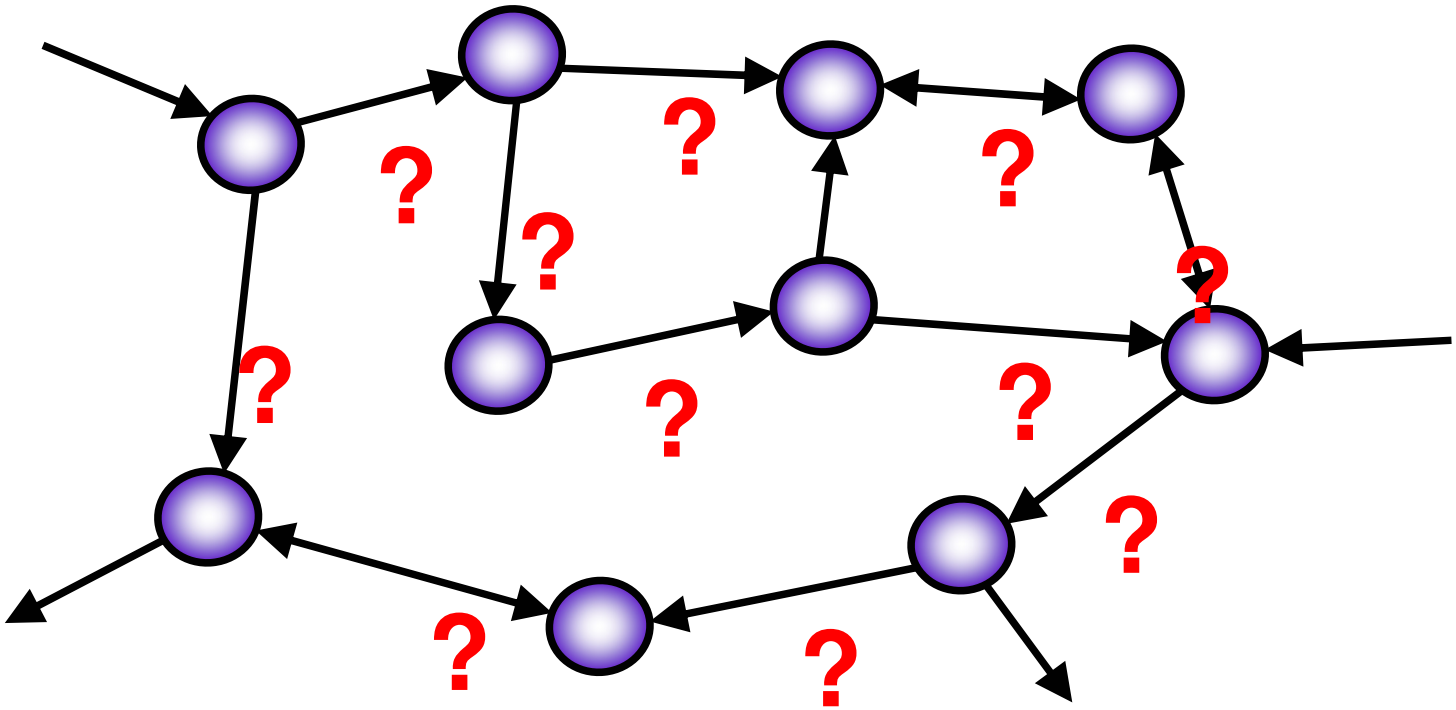
Containers

PicoContainer

Case Study

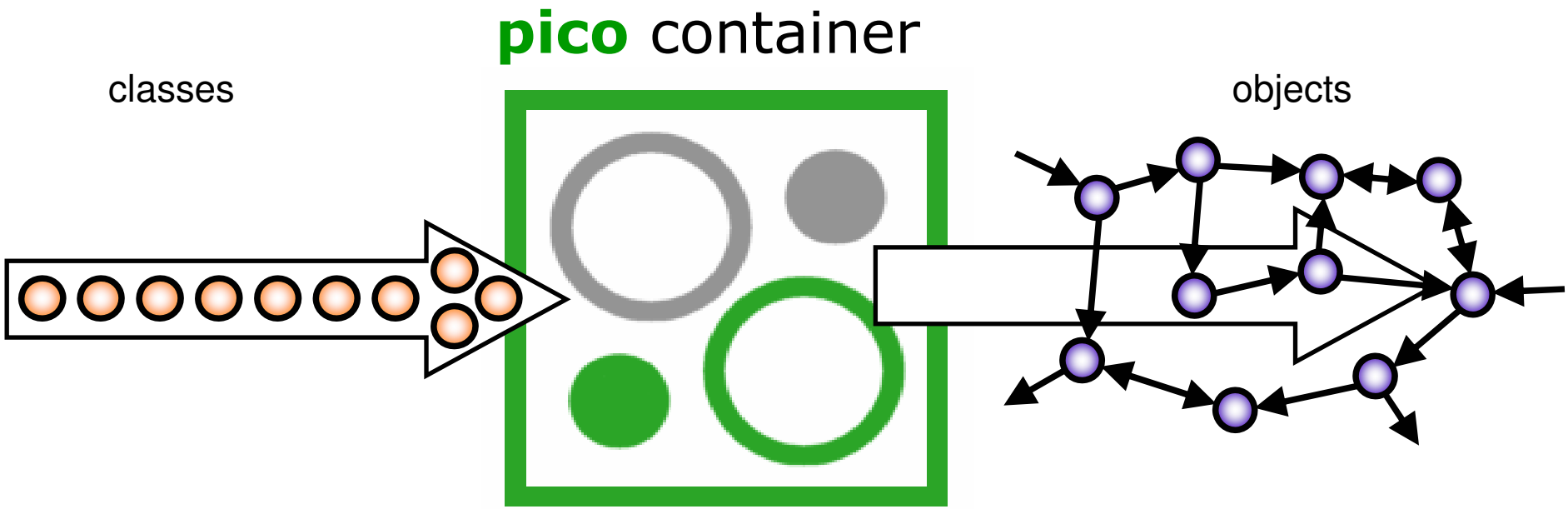
Component dependencies

- How do these relationships get established?



PicoContainer

- A simple Constructor Injection container



Show me the code

ThoughtWorks® The art of heavy lifting.℠

```
public class DealManagerServiceImpl {

  public DealManagerImpl(MarketData aMarketDataService) {
    …
  }
```

```
public class DealEntryMessageBean {

  public void doSomething() {
    PicoContainer pico = getPicoContainer();
    pico.getComponentInstance(DealManager.class).doIt();
  }

  public PicoContainer getPicoContainer() {
    MutablePicoContainer pico = new DefaultPicoContainer();
    pico.registerComponentImplementation(DealManagerServiceImpl.class);
    pico.registerComponentImplementation(MarketDataServiceImpl.class);
    return pico;
  }
```

Mock Objects

Dependency Injection

Containers

PicoContainer

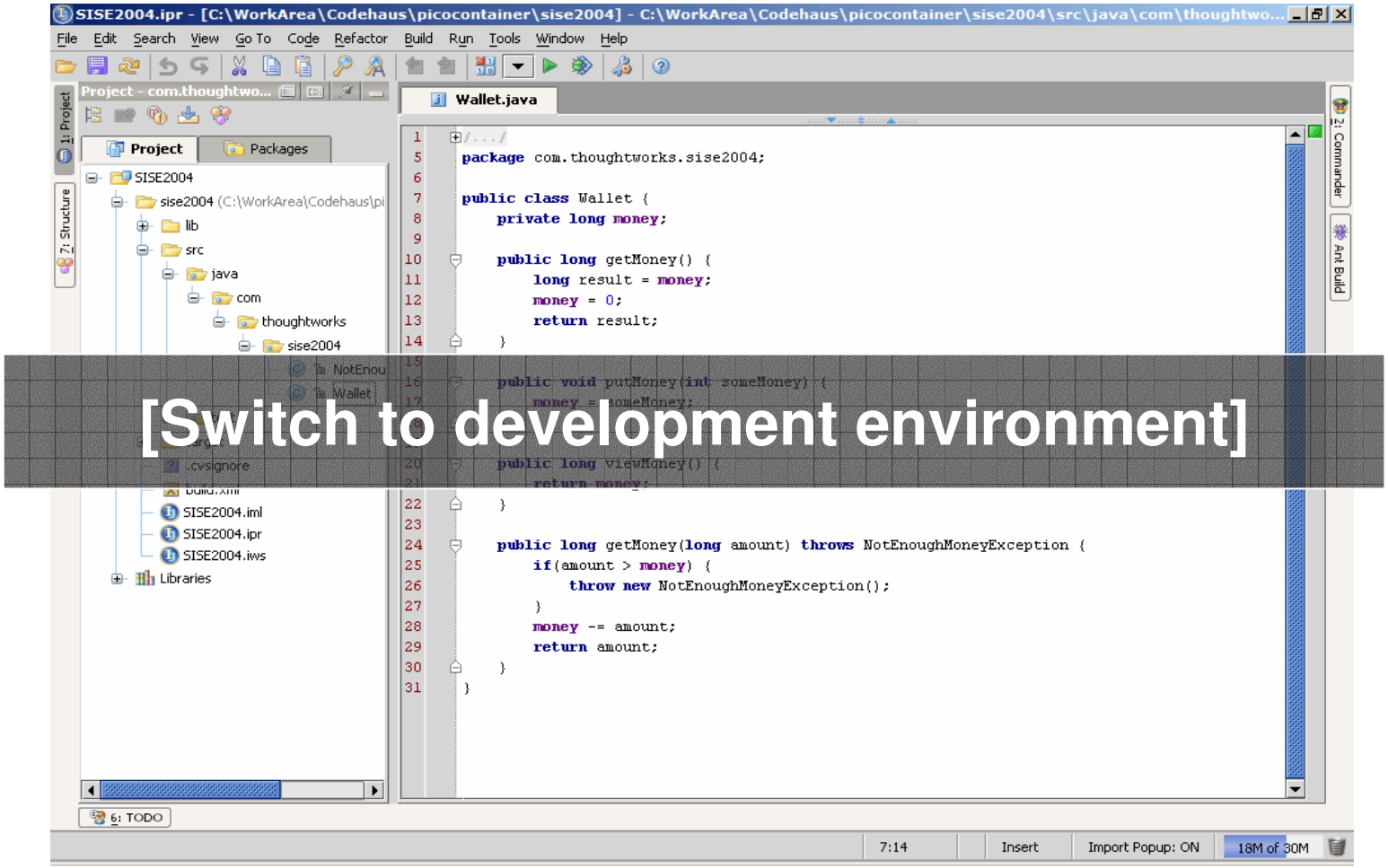
Case Study

Refactoring for testability

ThoughtWorks® The art of heavy lifting.℠

1. From EJBs with embedded logic and service locators
   towards
   an EJB wrapper using PicoContainer and service objects.

2. From Struts Actions with Service Locators
   towards
   Struts Actions with Dependency Injection.



[Switch to development environment]

CustomActionProcessor.java

```
package com.thoughtworks.dealmanager.util;

import java.io.IOException;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.action.RequestProcessor;
import org.apache.struts.util.RequestUtils;
import org.picocontainer.MutablePicoContainer;
import org.picocontainer.defaults.DefaultPicoContainer;

import com.thoughtworks.dealmanager.services.DealDistributionService;
import com.thoughtworks.dealmanager.services.DealEntryService;


public class CustomRequestProcessor extends RequestProcessor
{
    public Action processActionCreate(HttpServletRequest request,
      HttpServletResponse response, ActionMapping mapping)
      throws IOException
    {
        Action action = null;

        try
        {
            String className = mapping.getType();
            MutablePicoContainer pico = getPicoContainer();
            Class actionClass = RequestUtils.applicationClass(className);
            pico.registerComponentImplementation(actionClass);
            action = (Action)pico.getComponentInstance(actionClass);
            action.setServlet(servlet);
        }
        catch(Exception e)
        {
            response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR
              getInternal().getMessage("actionCreate", mapping.getPath()));
        }
        return action;
    }


    private MutablePicoContainer getPicoContainer()
    {
        MutablePicoContainer pico = new DefaultPicoContainer();
        pico.registerComponentImplementation(DealDistributionService.class);
        pico.registerComponentImplementation(DealEntryService.class);
        return pico;
    }

}
```

Thank you

erik@thoughtworks.com