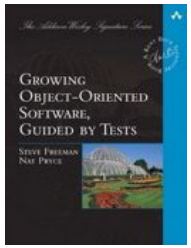
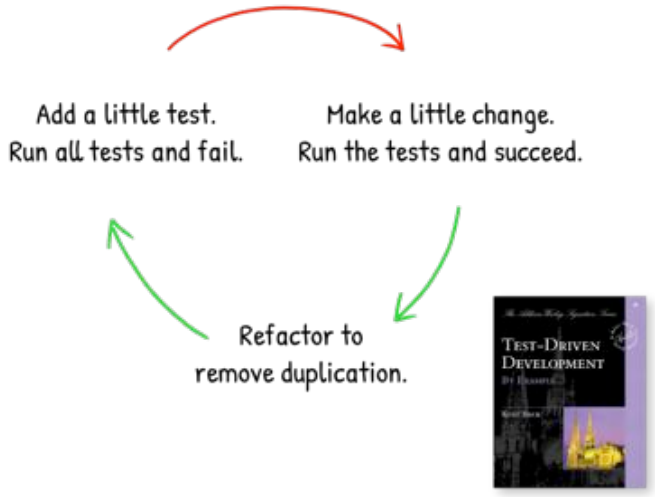
Test-Driven Development

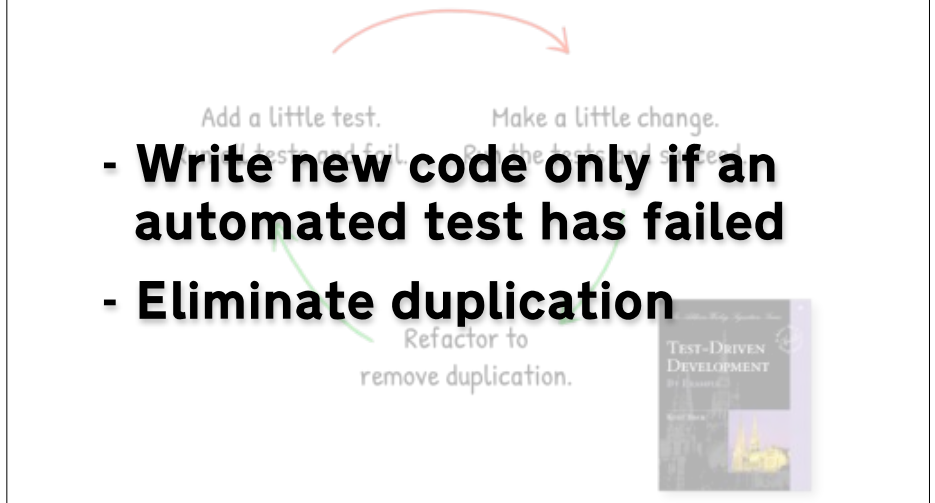
(that's not what we meant)

steve.freeman@higherorderlogic.com
@sf105



HIGHER ORDER
logic





- **Write new code only if an automated test has failed**
- **Eliminate duplication**


```
public class ExchangeRateUploaderTest extends EasyMockTestCase {
    private Logger logger;
    private CurrencyManager mockCurrencyManager;
    private ExchangeRateManager mockExchangeRateManager;
    private PriceManagerFactory mockPriceManagerFactory;
    private PriceManager mockPriceManager;
    private GodObject mockGod;
    private DatabaseFacade mockPersistenceManager;
    private DatabaseFacade mockFrameworkPersistenceManager;
    private CyclicProcessManager mockCyclicProcessManager;
    private SystemVariableManager mockSystemVariableManager;
    private ScreenManager mockScreenManager;
    private Registry registry;
    private User adminUser;
    private Server server;

    private ExchangeRateUploader newExchangeRateUploader(CyclicProcessThread thread) {
        return new ExchangeRateUploader(thread) {
            @Override protected void initializeAction() throws FrameworkException {
                // Does nothing to prevent excessive mocking
            }
            @Override public Logger getLogger() { return logger; }
            @Override protected void setLogMDC() { }
            @Override protected User getUser() { return adminUser; }
            @Override protected CurrencyManager newCurrencyManager() { return mockCurrencyManager; }
            @Override protected PriceManagerFactory newPriceManagerFactory() { return mockPriceManagerFactory; }
            @Override protected CyclicProcessManager newCyclicProcessManager() { return mockCyclicProcessManager; }
            @Override protected DatabaseFacade newPersistenceManager() { return mockPersistenceManager; }
            @Override protected Registry newTaskPerformanceRegistry() { return registry; }
            @Override public PriceDataManager getPriceDataManager() { return null; }
            @Override protected ExchangeRateManager newExchangeRateManager() { return mockExchangeRateManager; }
        };

    }
```



```
public void testInternalAction() throws FrameworkException {
    mockGod = addMock(GodObject.class);
    expect(mockGod.getPriceDataManager()).andStubReturn(null);
    mockPersistenceManager = addMock(DatabaseFacade.class);
    registry = addMock(Registry.class);
    adminUser = new TestUser("Admin", "", "", new TestCompany("company"), "");
    mockCyclicProcessManager();

    registry.finalizeThisThread(isA(String.class), isA(String.class));
    Date now = DateUtils.trimMinutesAndSecondsFromDate(new Date());

    mockSystemVariableManager();
    mockLogger();
    mockContextPersistenceManager();
    mockPriceManager();

    CyclicProcessThread thread = mockUserStateAndGetCyclicProcessThread();

    String primeName = "prime";
    String aName = "a";
    String otherName = "other";

    Currency primeCurrency = new TestCurrency(primeName, true);
    Currency aCurrency = new TestCurrency(aName, true);
    Currency otherCurrency = new TestCurrency(otherName, false);

    FXCurrencyPair aCurrencyPair = new FXCurrencyPair(primeCurrency, aCurrency);
    FXCurrencyPair otherCurrencyPair = new FXCurrencyPair(otherCurrency, primeCurrency);

    setupCurrencyManager(primeCurrency, aCurrency, otherCurrency);
    mockExchangeRateManager = addMock(ExchangeRateManager.class);

    mockGetFXRatesAtDatesForCurrencies(now, aCurrencyPair, otherCurrencyPair);

    FrameworkNumber aCurrencyValue = new FrameworkNumber("5");
    FrameworkNumber otherCurrencyValue = new FrameworkNumber("2");
    ExchangeRate aCurrencyRate = new ExchangeRate(primeCurrency, aCurrency, aCurrencyValue, now);
    ExchangeRate otherCurrencyRate = new ExchangeRate(otherCurrency, primeCurrency, otherCurrencyValue, now);
    expect(mockCurrencyManager.getParentToFractionalCurrencyMapForFractionalCurrency()).andStubReturn(newMap());

    expect(
        mockExchangeRateManager.saveExchangeRate(null, new FrameworkString(primeName), new FrameworkString(aName), aCurrencyValue,
            new FrameworkDate(now))).andReturn(null);
    expect(
        mockExchangeRateManager.saveExchangeRate(null, new FrameworkString(otherName), new FrameworkString(primeName),
            otherCurrencyValue, new FrameworkDate(now))).andReturn(null);

    Map<String, ExchangeRate> out = new HashMap<String, ExchangeRate>();
    out.put("primea", aCurrencyRate);
    out.put("otherprime", otherCurrencyRate);
    expect(mockPriceManager.getLatestExchangeRates(newList(aCurrencyPair, otherCurrencyPair))).andReturn(out);

    mockPMFactoryCleanup();

    replayMocks();

    ExchangeRateUploader uploader = newExchangeRateUploader(thread);
    uploader.initialise();
    uploader.run();
}
```



```
private void mockPMFactoryCleanup() {
    PersistenceFactory mockPersistenceFactory = addMock(PersistenceFactory.class);
    mockPersistenceFactory.purgeAllStateForThisThread();
    expect(mockGod.getPersistenceFactory()).andReturn(mockPersistenceFactory).anyTimes();
    expect(mockPersistenceFactory.getExceptionsInRequest()).andReturn(Collections.<Throwable>emptyList()).times(1);
}

private void mockCyclicProcessManager() throws CyclicProcessException {
    mockCyclicProcessManager = addMock(CyclicProcessManager.class);
    expect(mockGod.getCyclicProcessManager()).andStubReturn(mockCyclicProcessManager);
    mockCyclicProcessManager.updateServerCyclicProcessCurrentRunStatus(isA(String.class),
        isA(LISTENER_STATUS.class), isA(String.class), isA(Double.class), isA(Date.class));
    expectLastCall().anyTimes();
    server = addMock(Server.class);
    expect(mockCyclicProcessManager.getThisServer()).andStubReturn(server);
}

private void setupCurrencyManager(Currency primeCurrency, Currency aCurrency, Currency otherCurrency) {
    mockCurrencyManager = addMock(CurrencyManager.class);
    List<Currency> allCurrencies = new ArrayList<Currency>();
    allCurrencies.add(aCurrency);
    allCurrencies.add(primeCurrency);
    allCurrencies.add(otherCurrency);
    expect(mockCurrencyManager.getPrimeCurrency()).andReturn(primeCurrency).anyTimes();
    expect(mockCurrencyManager.getAllParentCurrencies()).andReturn(allCurrencies).times(2);
}

private void mockGetFXRatesAtDatesForCurrencies(Date now, FXCurrencyPair aCurrencyPair,
    FXCurrencyPair otherCurrencyPair) throws CurrencyException
{
    FrameworkNumber originalACurrencyRate = new FrameworkNumber("1.23");
    Map<FXCurrencyPair, Collection<Date>> currencyPairAndDatesMap = new HashMap<FXCurrencyPair, Collection<Date>>();
    currencyPairAndDatesMap.put(aCurrencyPair, Arrays.asList(now));
    currencyPairAndDatesMap.put(otherCurrencyPair, Arrays.asList(now));
    FXCurrencyPairRates outputObj = addMock(FXCurrencyPairRates.class);
    expect(outputObj.rateMapSize()).andReturn(5).anyTimes();
    expect(outputObj.getActualPriceDateForCurrencyPair(aCurrencyPair, now)).andReturn(null).once();
    expect(outputObj.getRateFromFxRateMap(now, aCurrencyPair)).andReturn(originalACurrencyRate).once();
    expect(outputObj.getActualPriceDateForCurrencyPair(otherCurrencyPair, now)).andReturn(null).once();
    expect(outputObj.getRateFromFxRateMap(now, otherCurrencyPair)).andReturn(originalACurrencyRate);
    expect(mockExchangeRateManager.getFXRatesAtDatesForCurrencies(currencyPairAndDatesMap)).andReturn(outputObj);
}
```

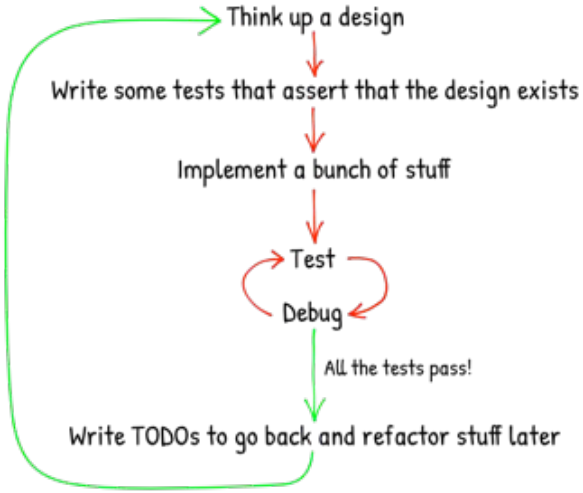
```
    private CyclicProcessThread mockUserStateAndGetCyclicProcessThread() {
        Role mockAdminRole = addMock(Role.class);
        CyclicProcessThread thread = addMock(CyclicProcessThread.class);
        expect(thread.getAdminRole()).andReturn(mockAdminRole).anyTimes();
        expect(thread.getAdminUser()).andReturn(adminUser).anyTimes();
        thread.interrupt();
        expectLastCall();
        mockScreenManager = addMock(ScreenManager.class);
        expect(mockGod.getScreenManager()).andReturn(mockScreenManager).anyTimes();
        mockScreenManager.setThreadSignedOnState(new SignedOnState(adminUser, mockAdminRole, false));
        expectLastCall().anyTimes();
        expect(thread.getGod()).andReturn(mockGod).anyTimes();
        expect(thread.getShutdownInProgress()).andReturn(false).anyTimes();
        return thread;
    }

    private void mockContextPersistenceManager() {
        mockFrameworkPersistenceManager = addMock(DatabaseFacade.class);
        expect(mockGod.getDatabaseFacade()).andReturn(mockFrameworkPersistenceManager).anyTimes();
        mockFrameworkPersistenceManager.beginNewSession();
        expectLastCall().anyTimes();
    }

    private void mockPriceManager() throws PriceException {
        mockPriceManagerFactory = addMock(PriceManagerFactory.class);
        mockPriceManager = addMock(PriceManager.class);
        expect(mockPriceManagerFactory.newPriceManager(mockFrameworkPersistenceManager,
                                                       mockSystemVariableManager, null))
            .andReturn(mockPriceManager).once();
    }

    private void mockSystemVariableManager() {
        mockSystemVariableManager = addMock(SystemVariableManager.class);
        expect(mockGod.getSystemVariableManager()).andReturn(mockSystemVariableManager).anyTimes();
        expect(mockSystemVariableManager.getSystemVariable(CYCLIC_PROCESS_LISTENER_HEART_BEAT_TOLERANCE, "30000"))
            .andReturn("30000").anyTimes();
    }

    private void mockLogger() {
        logger = addMock(Logger.class);
        logger.info(isA(String.class)); expectLastCall().atLeastOnce();
        logger.debug(isA(String.class)); expectLastCall().atLeastOnce();
    }
}
```

Keith Braithwaite

Difficult to understand

Overspecified

Obscure

Brittle

Meaningless failures

"Security Theatre"

Test-Driven Theatre

Should!


```
given_I_have_a_Foo_when_I_call_setBar_with_3_then_getBar_should_return_3()
```


What are the tests for TDD?

Steady, incremental progress

Constant positive reinforcement

I think before I code

Things break when they're supposed to

Surprising designs emerge


```
BasketTest.add_adding_item()
  sut = new Basket()
  sut.add(ITEM)
  assertEquals(
    ITEM,
    backdoor(sut, "itemList")[0])
```


Write readable code

```
is_empty_when_created()
  assertThat( new Basket().itemCount(),
              equals(0) )

returns_items_in_the_order_they_were_added()
  basket = new Basket()
               .add(pen).add(ink).add(paper)
  assertThat( basket,
              hasItems(pen, ink, paper) )

totals_up_the_cost_of_its_items()
fails_when_removing_an_absent_item()
...
```


Interfaces, not internals

HIGHER ORDER logic steve.freeman@higherorderlogic.com ©2013

Protocols, not Interfaces

HIGHER ORDER logic steve.freeman@higherorderlogic.com ©2013

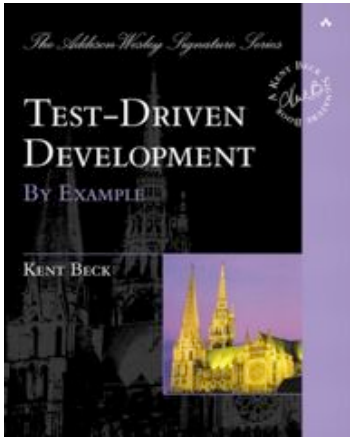
From simple to general

HIGHER ORDER logic steve.freeman@higherorderlogic.com ©2013

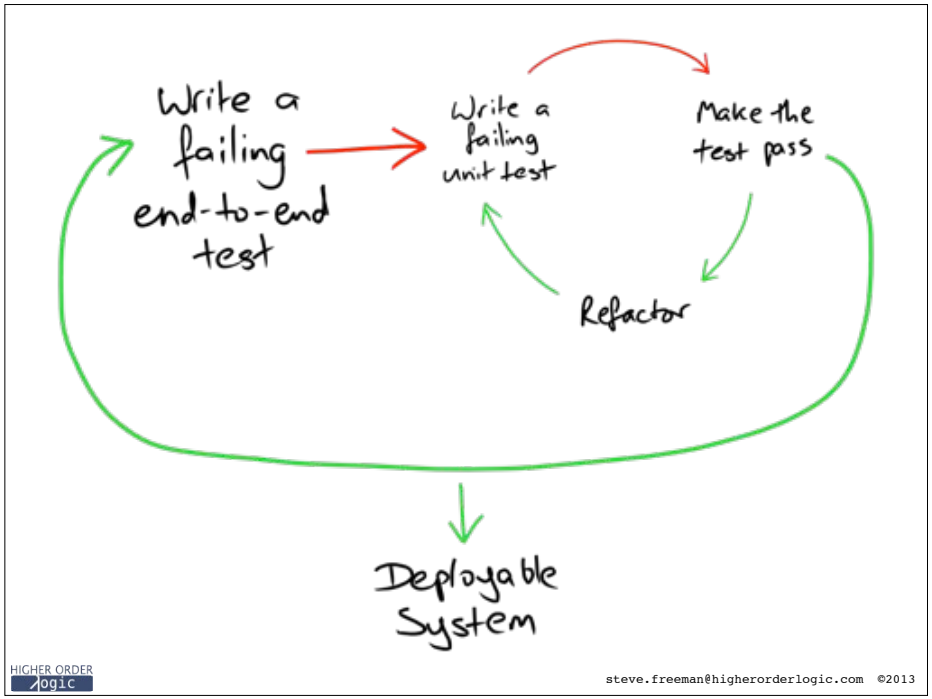
"It's about explaining the domain, not about proving the correctness of the code."

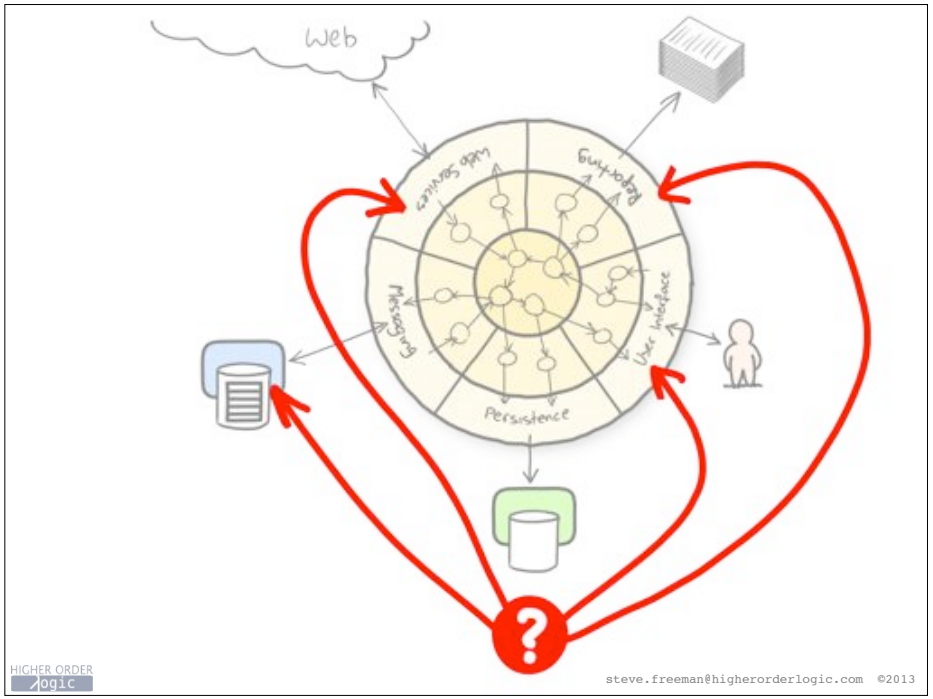
Andrew Parker

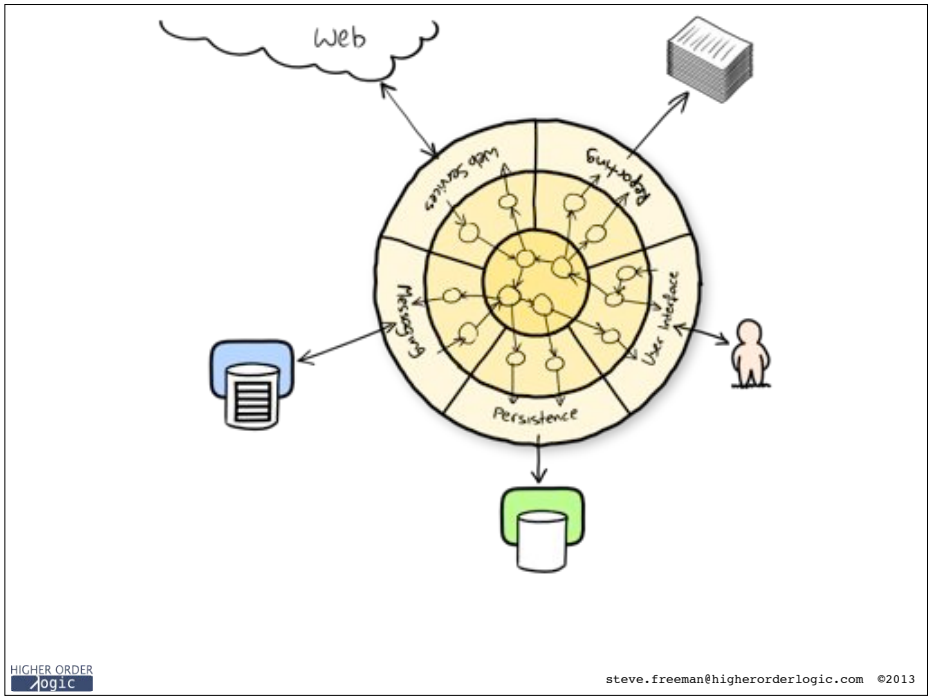
When you're lost, slow down

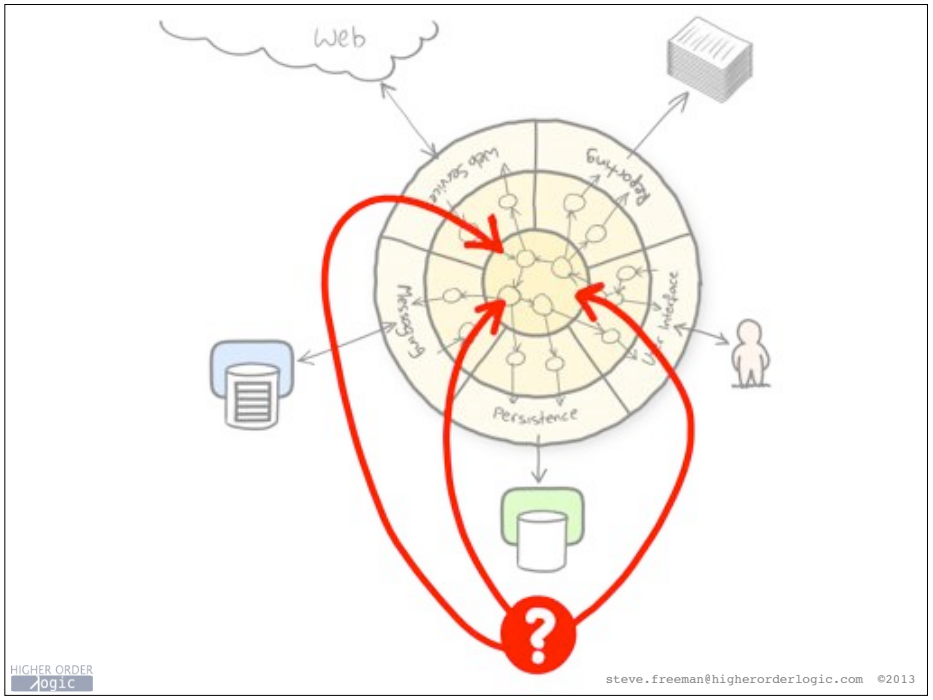


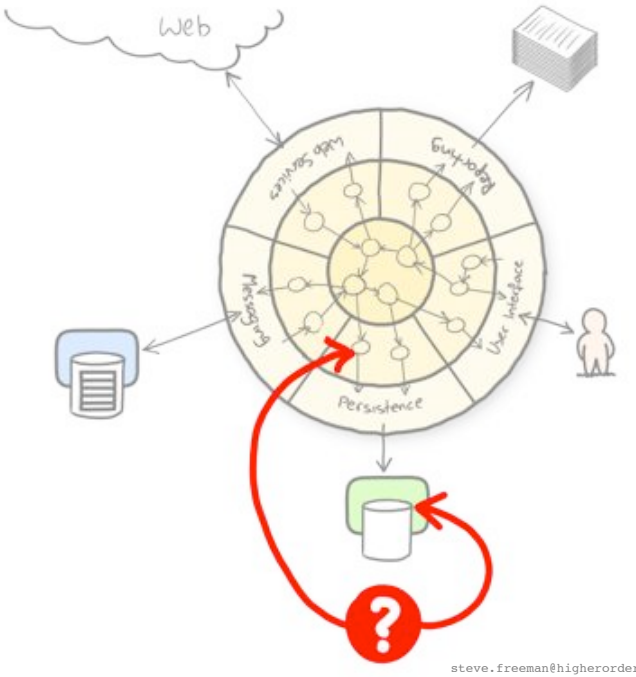
Test at the right level

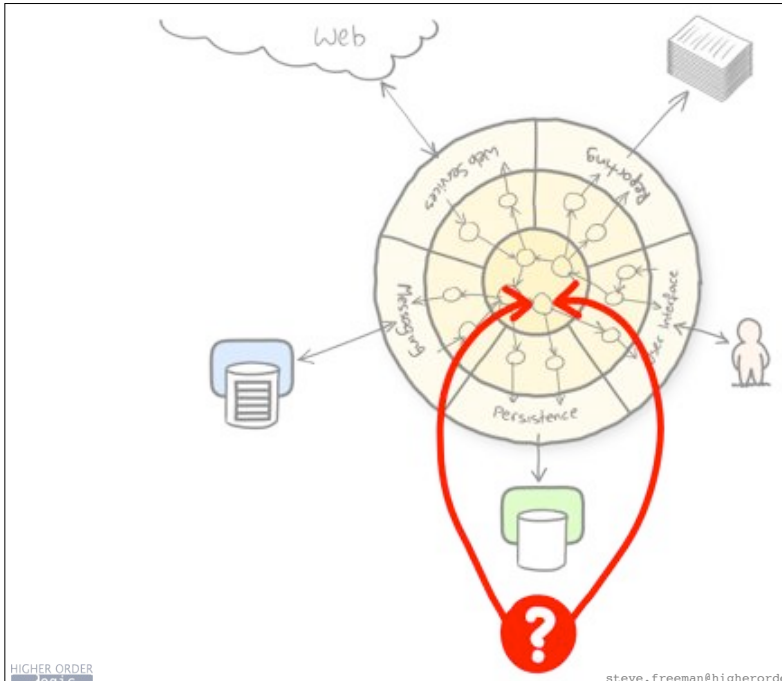

Write a failing end-to-end test
Write a failing unit test
Make the test pass
Refactor
Deployable System


Web
Persistence
?


Web
Persistence


Web
Persistence
?





Traffic

Click-through Rate	PVs with Outbrain	Clicks
10.14%	93,557	9,485
		Organic:8,432 Paid:1,053

View detailed report

Revenue

RPM	PVs with Paid Links	Revenue
$0.39	93,557	$36.75

View detailed report

Traffic Sources

#	Source	Percent
1.		62.86%
2.		9.62%
3.		5.08%
4.		4.03%
5.		18.41%

Why does TDD work?

Focussed

Kent Beck 15:10

anyway, i had been using sunit for a couple of months when i remembered this passage from a programming book i'd read as a kid. it it said the way to program is to look at the input tape ily type and manually type in the output tape you expect. then al and you program until the actual and expected tapes match.

15:10

i thought, what a stupid idea. i want tests that pass, not tests that fail. why would i write a test when i was sure it would fail. well, i'm in the habit of trying stupid things out just to see what happens, so i tried it and it worked great.

15:11

i was finally able to separate logical from physical design. i'd always been told to do that but no one ever explained how.

Concrete

Importer.RulesTest

Japanese Short Code

Sierra JAPANESE_SHORT name is built from the TSE code and always taken from GLOSS.

sierra value for	JAPANESE_SHORT	name	
derives from	gloss	classification	2200
derives from	gloss	identifier	EXCO

examples

Notes	sources	gloss classification	gloss identifier	expectedSierraValue()	expectedWarningMessage()
-	gloss	JBCX	003100076	JFB310	*NONE*
-	bnd2	-	-	*NONE*	*NONE*
-	bnd2, gloss	JBGL	000640064	JNR64	*NONE*
-	gloss, bnd2	JBGS	000070031	JFWI	*NONE*
Empty TSE code	gloss	JBGL	*NONE*	*NONE*	*NONE*
Unsupported MOF code	gloss	JBCO	000017969	*NONE*	*NONE*
Supported MOF code, unsupported TSE code	gloss	JBGL	000360063	*NONE*	Cannot convert TSE code 000360063 into Japanese Short code

When Issued Inflation Linked JGB

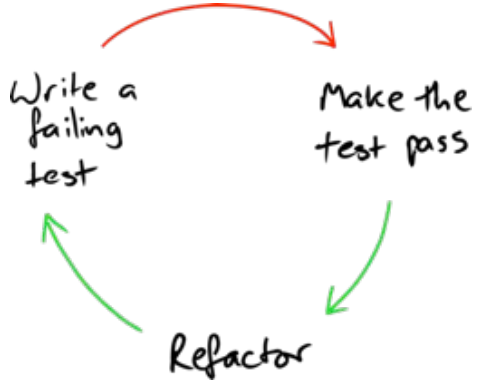
examples

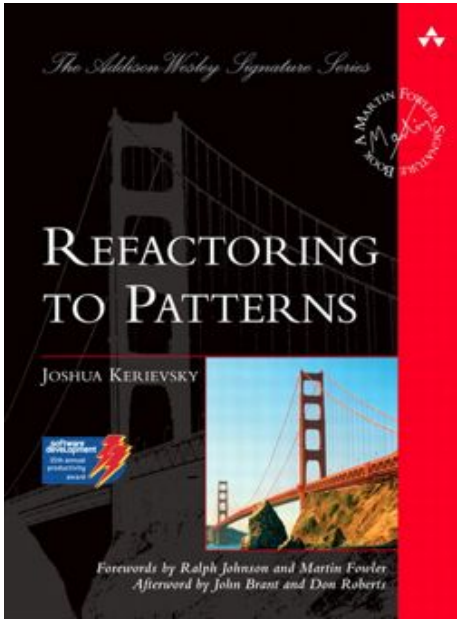
Notes	sources	gloss classification	gloss identifier	expectedSierraValue()	expectedWarningMessage()
-	gloss	JBGL	000090035	JSWI	*NONE*
-	gloss	JBGL	000100037	JBWI	*NONE*
-	gloss	JBGL	000070038	JKWI	*NONE*
-	gloss	JBGL	000090039	JLWI	*NONE*

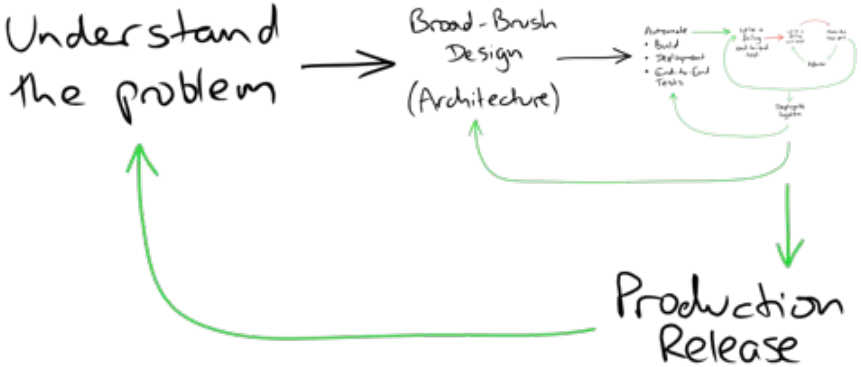
fit.Summary

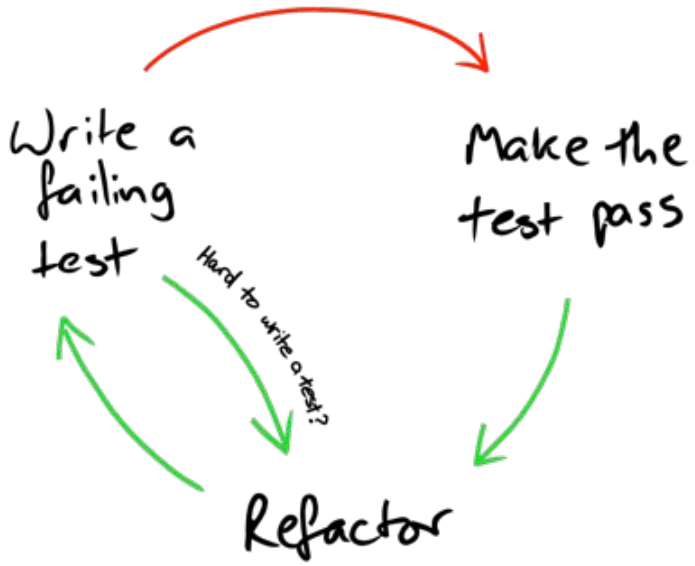
counts	22 right, 0 wrong, 0 ignored, 0 exceptions
input file	C:\Projects\sierra\acceptance-tests\features\Importer\NamesAndIdentifiers\JapaneseShortCode.html
input update	Mon Oct 17 09:45:42 BST 2005
output file	C:\Projects\sierra\build-output\artifacts\tests\fit\features\Importer\NamesAndIdentifiers\JapaneseShortCode.html
run date	Mon Oct 31 11:16:30 GMT 2005
run elapsed time	0:00.40

Empirical





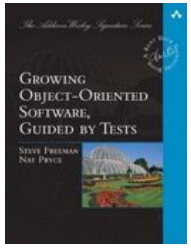




Test-Driven Development

(that's not what we meant)

steve.freeman@higherorderlogic.com
@sf105



HIGHER ORDER logic

BCS Edinburgh 4 December 2013