

A Journey in Software Development

An overview of methods and tools

Mathieu Acher

Maître de Conférences

mathieu.acher@irisa.fr

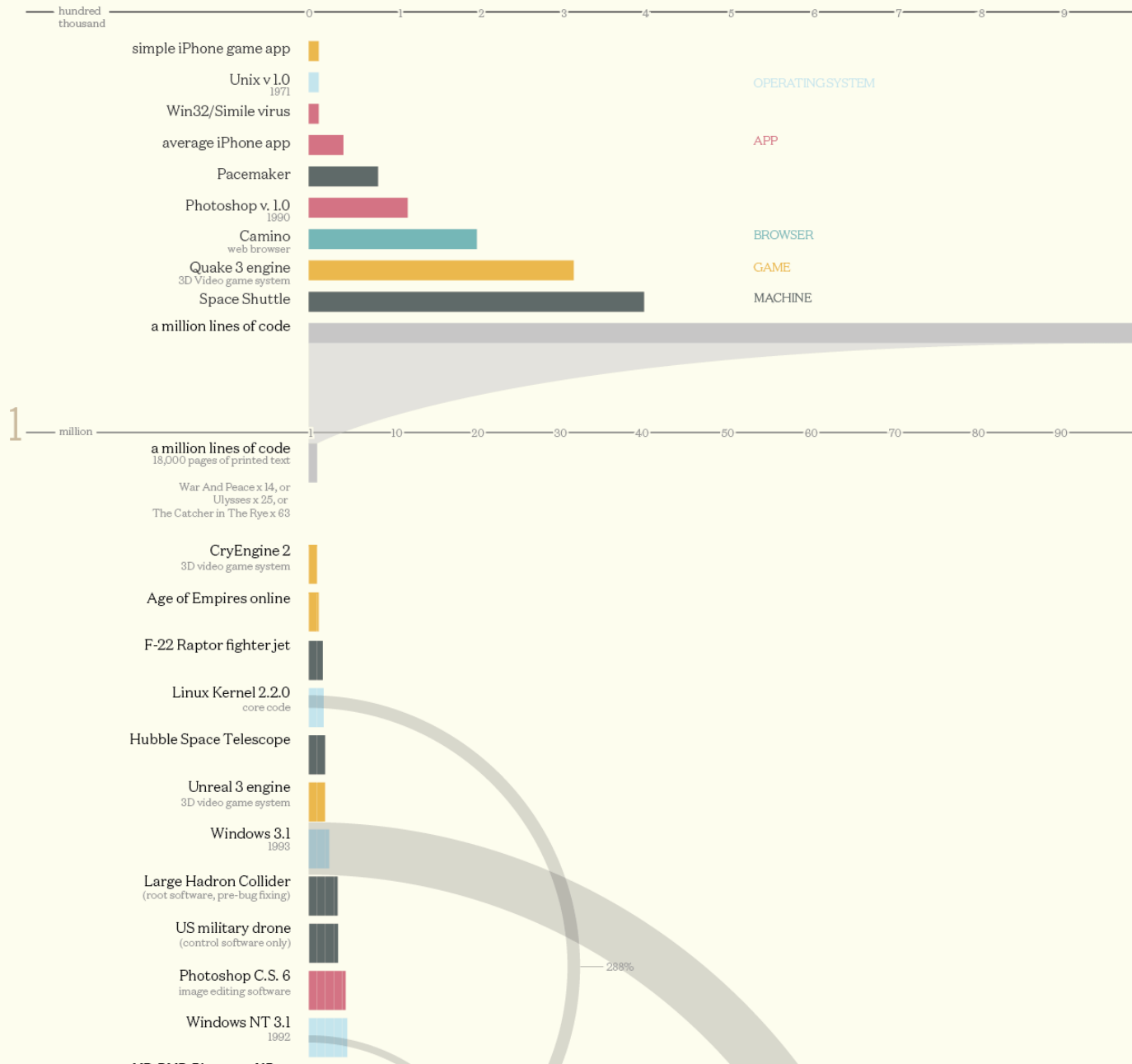
Material

<http://mathieuacher.com/teaching/PDL/>

Motivation

Codebases

Millions of lines of code



5

needed to repair HealthCare.gov
apparently

Mars Curiosity Rover
Martian ground vehicle probe

Linux kernel 2.6.0
2003

Google Chrome
latest

World of WarCraft
server only

Boeing 787
avionics & online support systems only

Windows NT 3.5
1993

Firefox
latest version

10

Chevy Volt
electric car

Intuit Quickbooks
accounting software

Windows NT 4.0
1996

Android
mobile device operating system

Mozilla Core
core code at heart of all Mozilla's software

MySQL
database language

Boeing 787
total flight software

Linux 3.1
latest version

Apache Open Office
open-source office software

F-35 Fighter jet
2013

25

Microsoft Office 2001

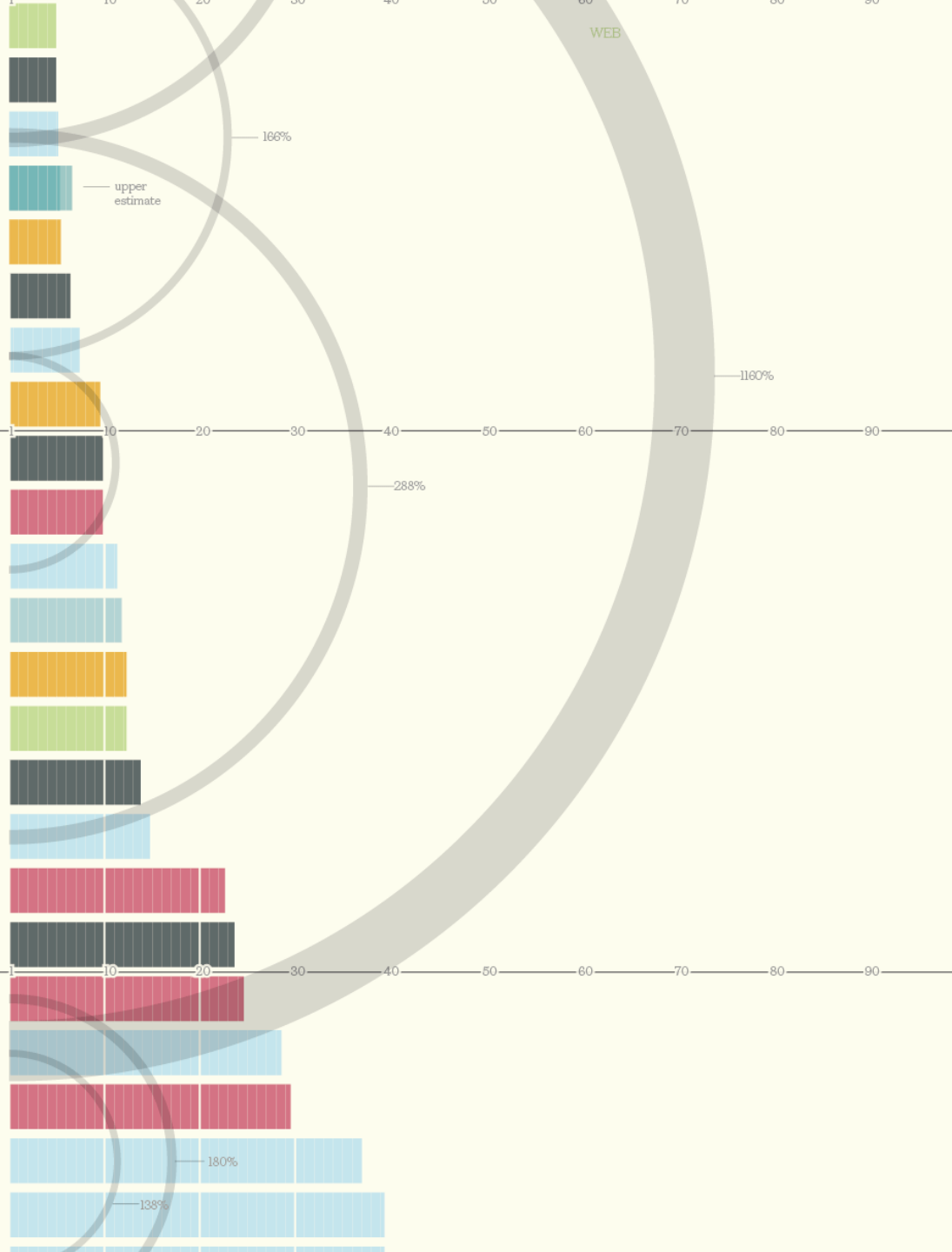
Windows 2000

Microsoft Office for Mac
2006

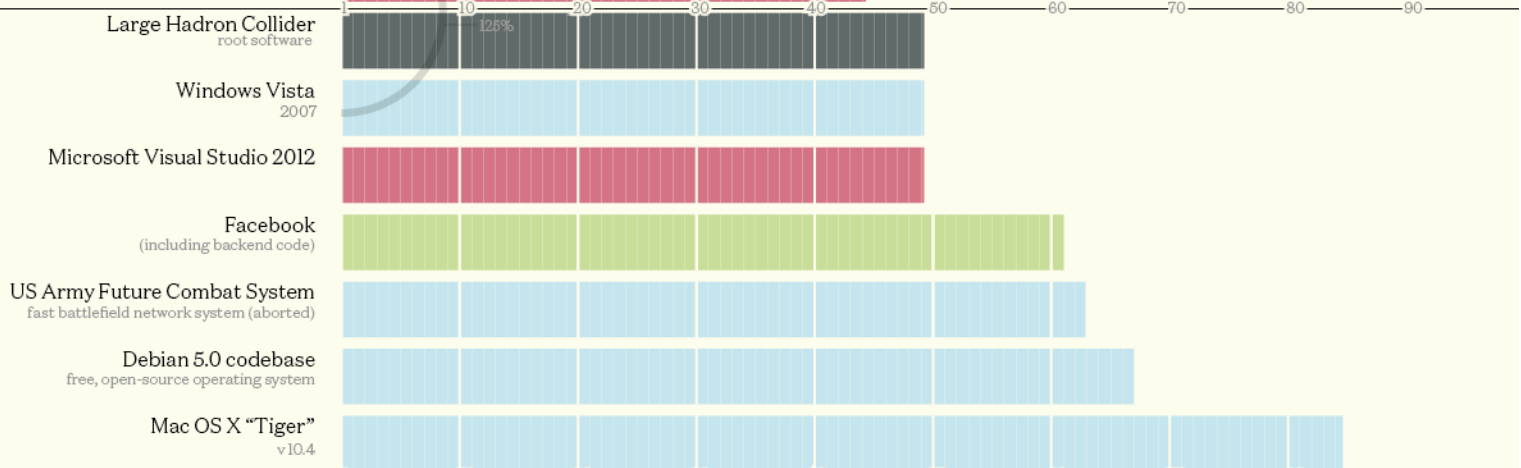
Symbian
mobile operating system

Windows 7
2009

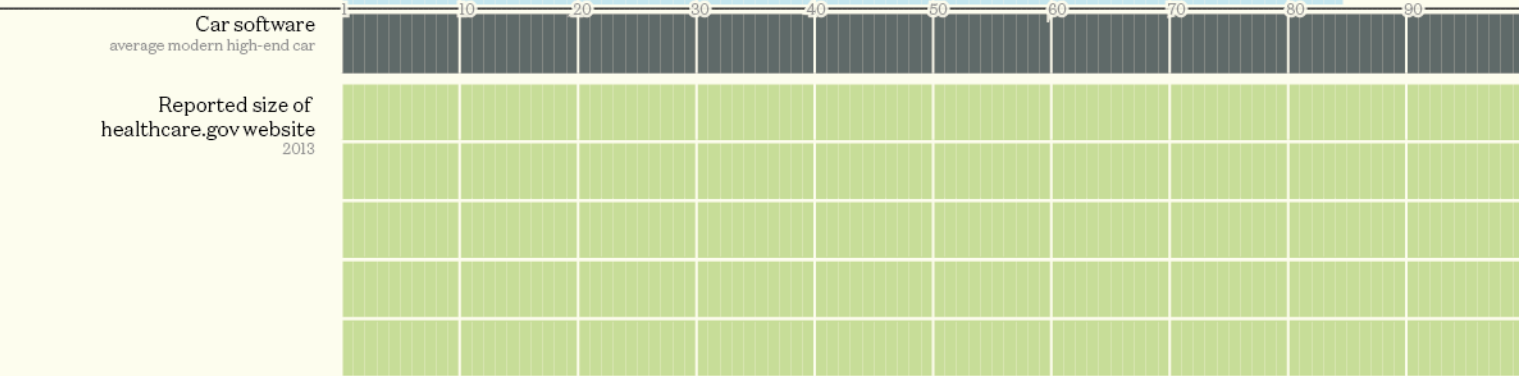
Windows XP



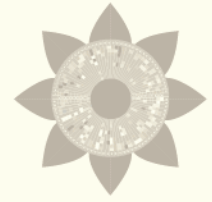
50



100



concept & design: David McCandless
informationisbeautiful.net
 research: Pearl Doughty-White, Miriam Quick



work in progress
 v0.62 // Oct 2013

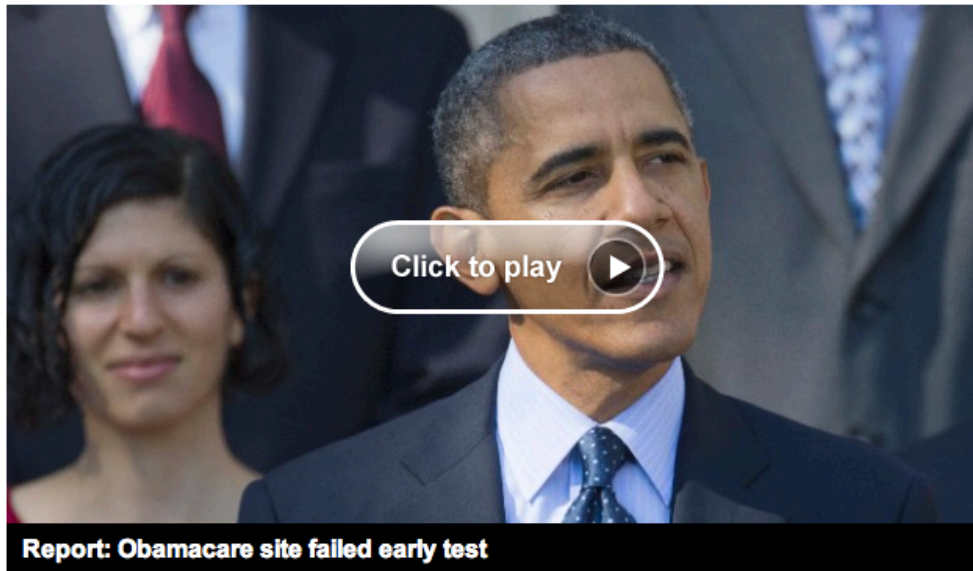
sources NASA, Quora, Ohloh, Wired & press reports
 note some guess work, rumours & estimates
 data bit.ly/KIB_linescode



Report: Healthcare website failed test ahead of rollout

By **Ed Payne**, **Matt Smith** and **Tom Cohen**, CNN

October 23, 2013 -- Updated 0103 GMT (0903 HKT)



Report: Obamacare site failed early test

STORY HIGHLIGHTS

- **NEW:** Top White House official part of "tech surge" on Obamacare
- Obamacare "is not failing" despite website woes, White House spokesman says
- Obama says HealthCare.gov problems are "going to get fixed"
- Secretary Sebelius expected to testify at a congressional hearing next week

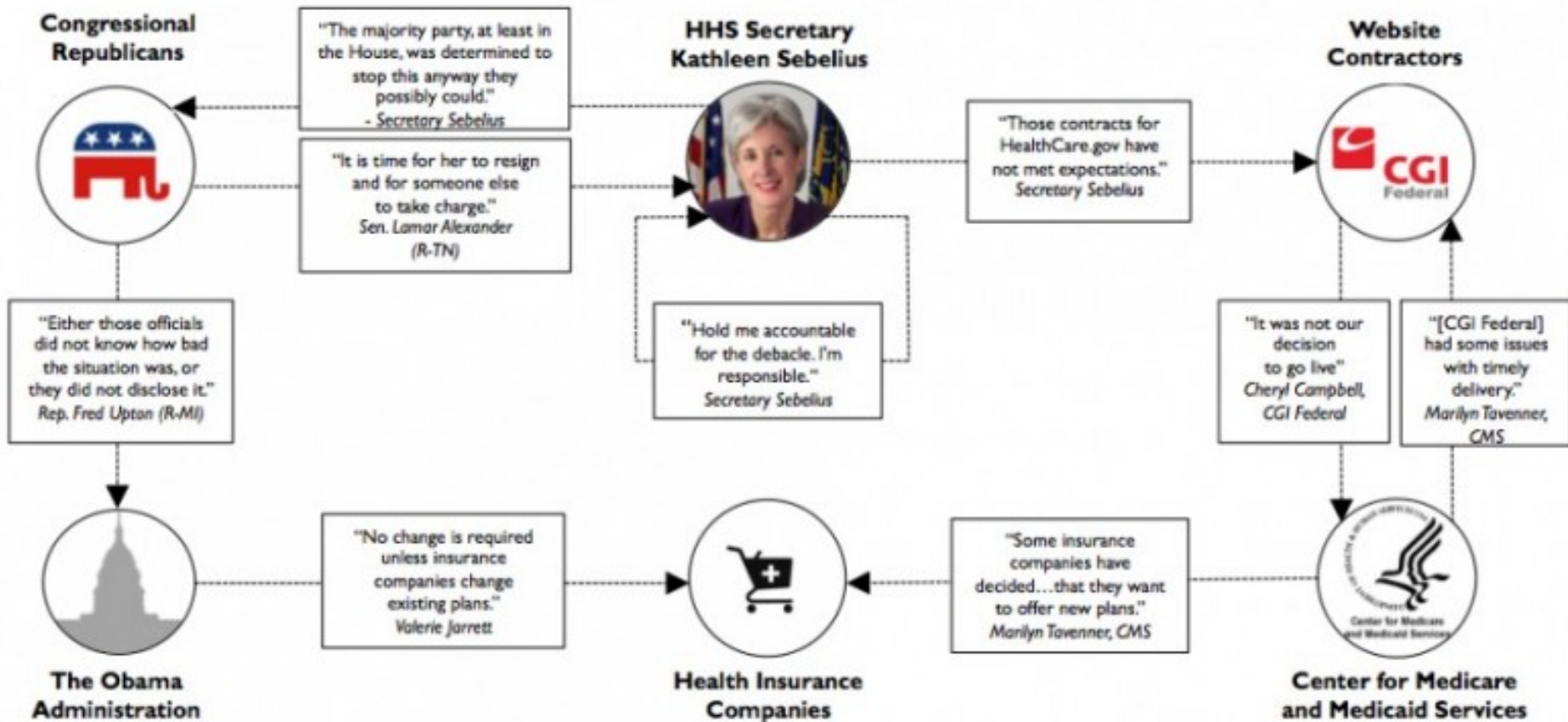
Washington (CNN) -- The President's healthcare sign-up web page was supposed to handle tens of thousands of people at once. But in a trial run days before its launch, just a few hundred users flatlined the site.

Despite the problems, federal health officials pushed aside the crash cart and rolled out [HealthCare.gov](#) on October 1 as planned, [The Washington Post](#) reported.

The result? The website crashed shortly after midnight as a couple thousand people tried to start the process, two people familiar with the project told the Post.

Requirements engineering/ Management problem

ACA Finger-Pointing Flowchart



<http://www.washingtonpost.com/blogs/wonkblog/wp/2013/11/01/thirty-one-things-we-learned-in-healthcare-govs-first-31-days/>

Thirty-one things we learned in HealthCare.gov's first 31 days

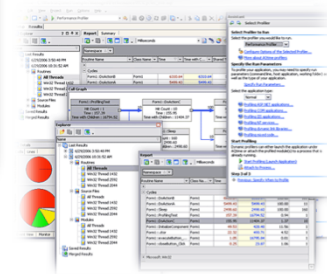
Scalability problem

Technical problems (e.g., inaccurate data, cancellation failures)

Testing issues

<http://www.washingtonpost.com/blogs/wonkblog/wp/2013/11/01/thirty-one-things-we-learned-in-healthcare-govs-first-31-days/>

Software Engineering



An overview

- Documentation, Coding Conventions
- Compile Chain and Organizations
- Debugging / Logging
- Testing
- Refactoring
- Versioning

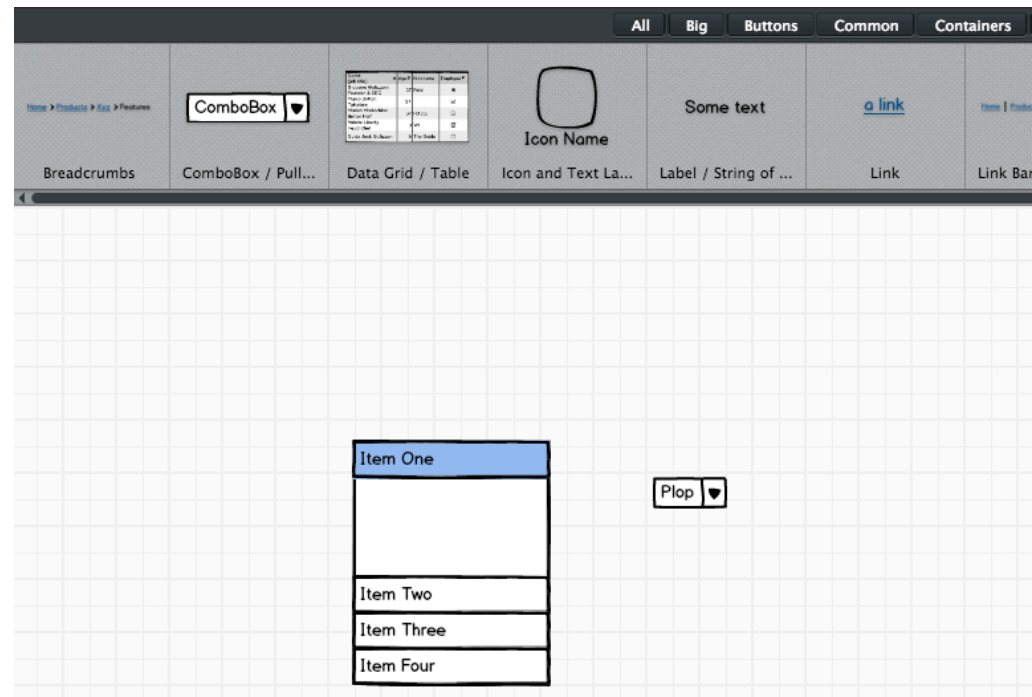
Contract

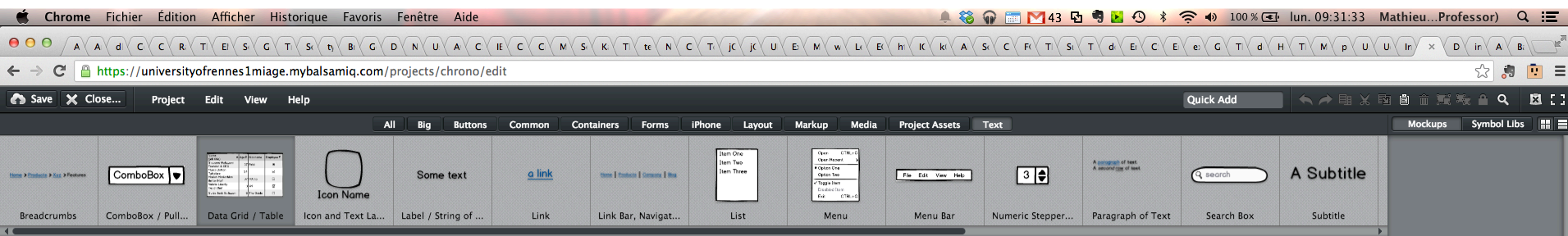
- Tools for
 - UI prototyping
 - Java Development
 - Collaborative, distributive software development
- Documenting, refactoring, testing
 - Why it matters
 - How tools can speed up the process and assist you

Requirements Engineering (A1)

Prototyping UI

- Intuitive, not cheap
- Iterative
- Exploration
- Focus on purpose
- Modeling
 - Forget technology a few seconds





Rapid prototypes of screens

Web

Time (job title)	Age	Nickname	Employee
Giacomo Guilizzoni Founder & CEO	37	Peldi	<input type="radio"/>
Marco Bolton Tuttofare	34		<input checked="" type="checkbox"/>
Mariah MacLachlan Better Half	37	Patata	<input type="checkbox"/>
Valerie Liberty Head Chef)	Val	<input checked="" type="checkbox"/>
Guido Jack Guilizzoni	6	The Guide	<input type="checkbox"/>

Drag and Drop

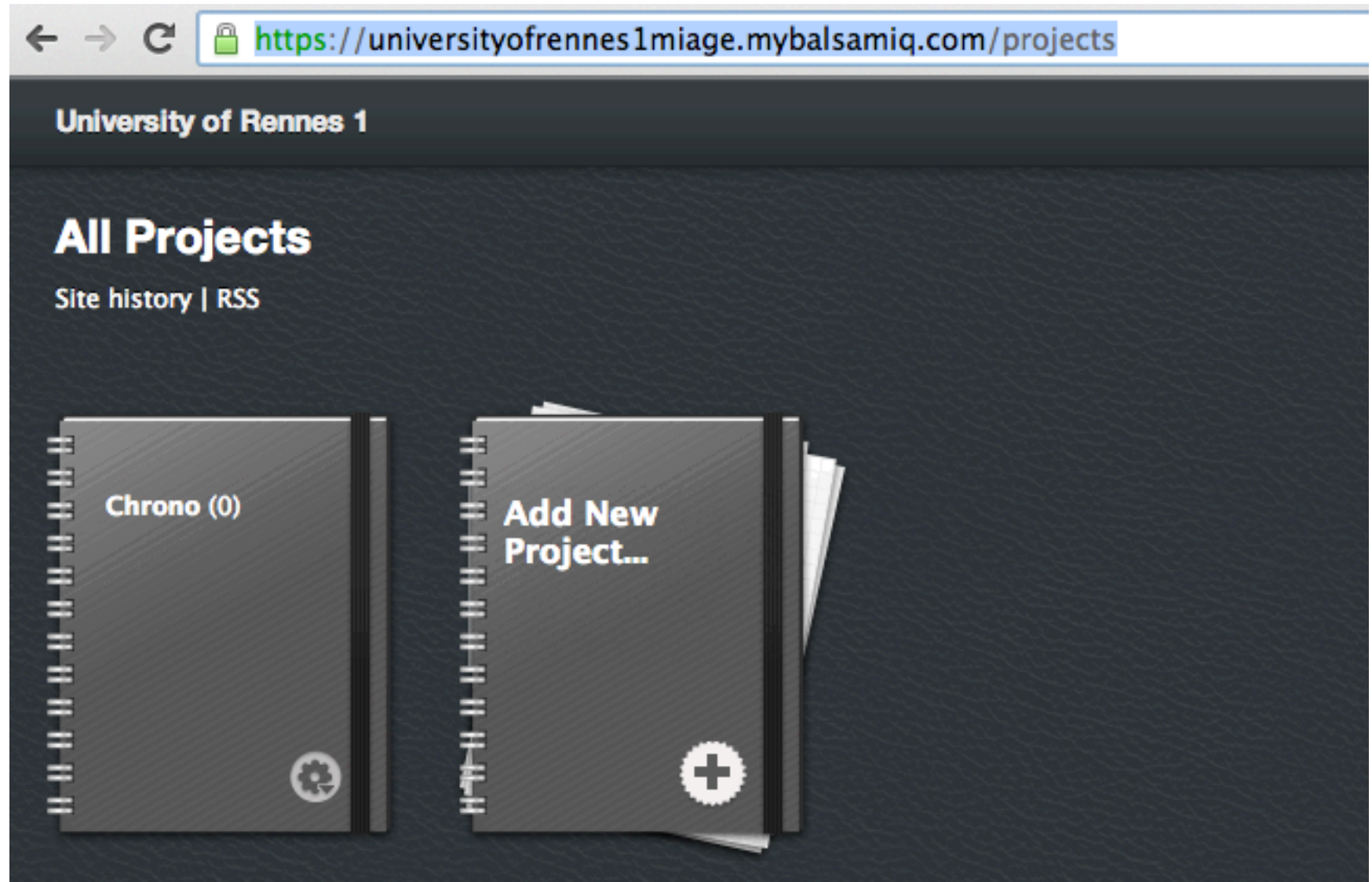
Easy

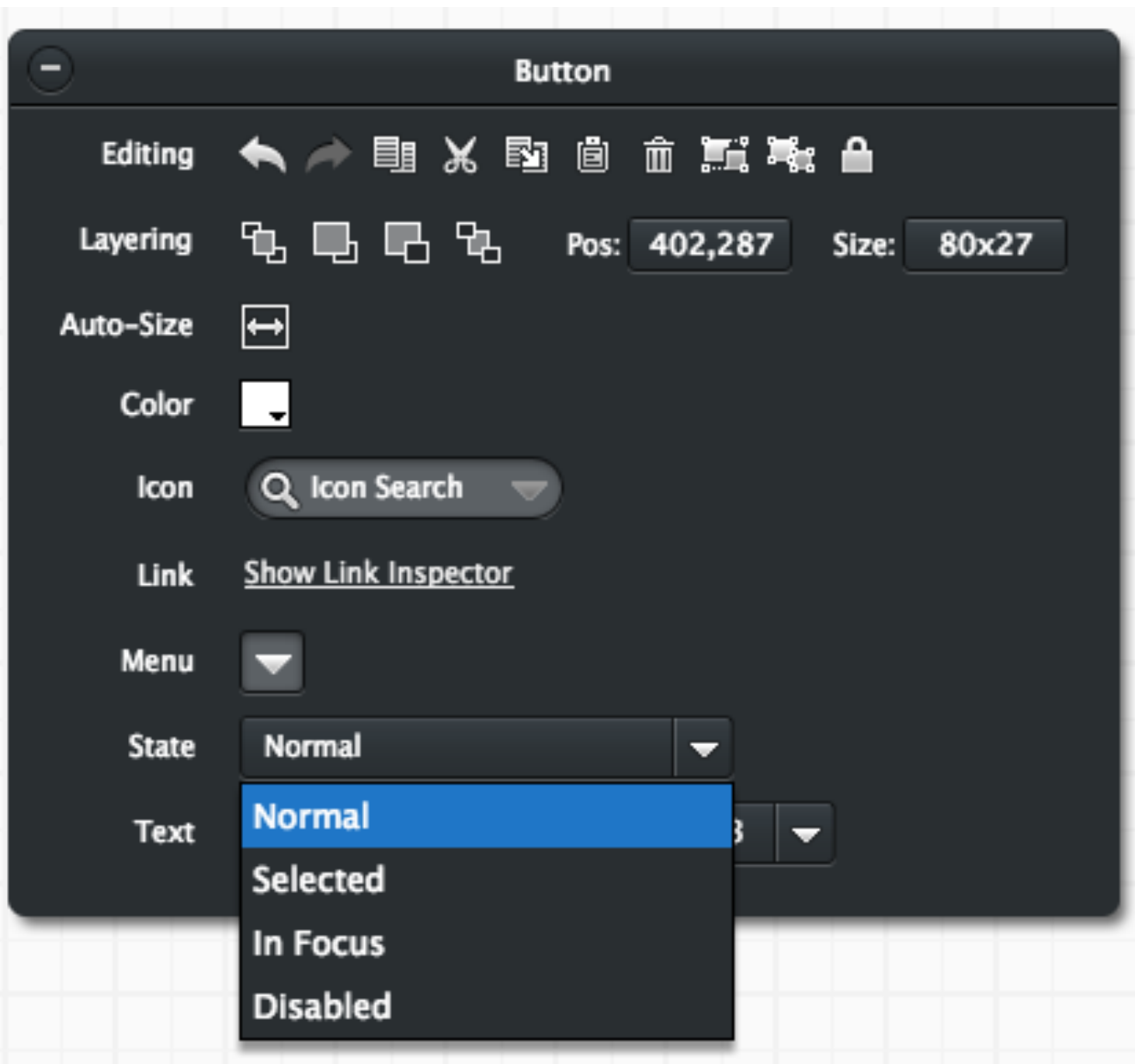
Recovered 2 mockups from auto-save.

Enter mockup notes here. You can hide this panel from the View menu.



<https://universityofrennes1miage.mybalsamiq.com/>





All

Big

Buttons

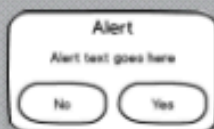
Common

Containers

Forms



Accordion



Alert Box



Arrow / Line

[Home](#) > [Products](#) > [XYZ](#) > Features

Breadcrumbs



Browser Window

Button

Button

One Two Three

Button Bar / Tab ...

iPhone

Layout

Markup

Media

Project Assets

Text



Calendar



Callout

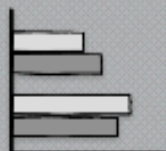


Chart: Bar Chart

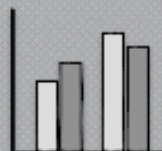


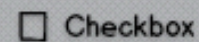
Chart: Column C...



Chart: Line Chart



Chart: Pie Chart



Checkbox

Button

Button

One Two Three

Button Bar / Tab ...

Checkbox


Checkbox

- not selected
- selected
- indeterminate
- disabled
- disabled selected
- disabled indeterminate
- A row without a checkbox

Checkbox Group



Color Picker

ComboBox 

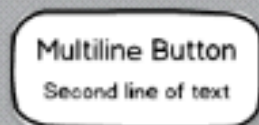
ComboBox / Pull..



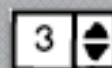
Date Chooser / ...



Help Button



Multiline Button



Numeric Stepper...



ON/OFF Switch /...



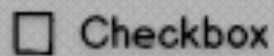
Playback Controls



Browser Window



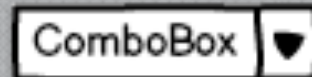
Button



Checkbox

- not selected
- selected
- indeterminate
- disabled
- disabled selected
- disabled indeterminate
- A row without a checkbox

Checkbox Group



ComboBox / Pull.



Geometric Shape



Icon



Icon Name
Icon and Text La...



Image

Some text

Label / String of ...

Home > Products > XYZ > Features

ComboBox / Pull...

Code	App #	Screen	Display #
248-100	1	100	1
248-101	1	101	1
248-102	1	102	1
248-103	1	103	1
248-104	1	104	1
248-105	1	105	1
248-106	1	106	1
248-107	1	107	1
248-108	1	108	1
248-109	1	109	1
248-110	1	110	1

Icon Name

Some text

[a link](#)

Home | Products | Contacts | More

Breadcrumbs ComboBox / Pull... Data Grid / Table Icon and Text La... Label / String of ... Link Link Bar, Navigat...

Button

Button Bar / Tab ...

Checkbox

- not selected
- selected
- indeterminate
- disabled
- disabled selected
- disabled indeterminate

A row without a checkbox

Color Picker

ComboBox / Pull...

Date Chooser / ...

Field Set / G

Button Button Bar / Tab ... Checkbox Checkbox Group Color Picker ComboBox / Pull... Date Chooser / ... Field Set / G

Browser Window

Field Set / Group...

Geometric Shape

Rectangle / Canv...

Tabs Bar / Ribbon

Vertical Tabs

Window / Dialog

Browser Window Field Set / Group... Geometric Shape Rectangle / Canv... Tabs Bar / Ribbon Vertical Tabs Window / Dialog

All

Big

Buttons

Common

Containers

[Home](#) > [Products](#) > [XYZ](#) > [Features](#)

ComboBox 

Item	Value	Icon	Enabled
System Windows	10/10/10		☑
Feature A - DE C	11		☑
Feature B - DE C	12		☑
Feature C - DE C	13		☑
Feature D - DE C	14		☑
Feature E - DE C	15		☑
Feature F - DE C	16		☑
Feature G - DE C	17		☑
Feature H - DE C	18		☑
Feature I - DE C	19		☑
Feature J - DE C	20		☑



Icon Name

Some text

[a link](#)

[Home](#) | [Features](#)

Breadcrumbs

ComboBox / Pull...

Data Grid / Table

Icon and Text La...

Label / String of ...

Link

Link Ba

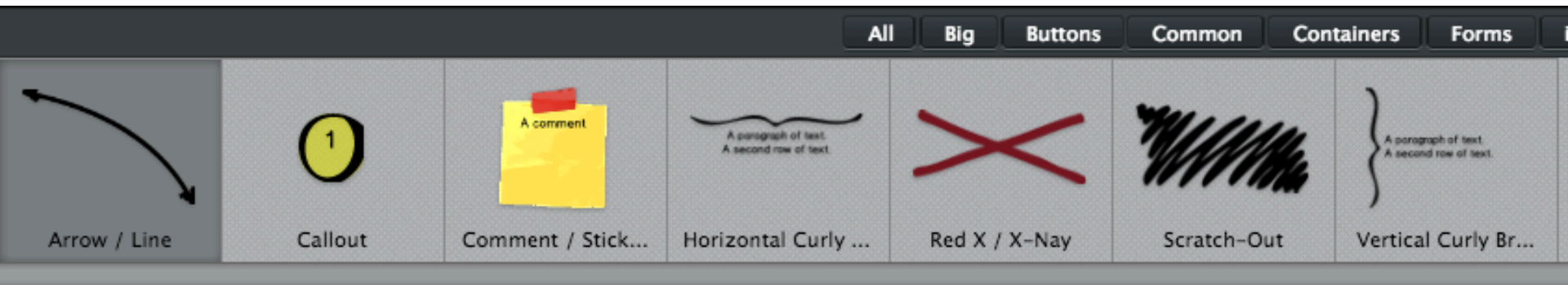
Item One

Item Two

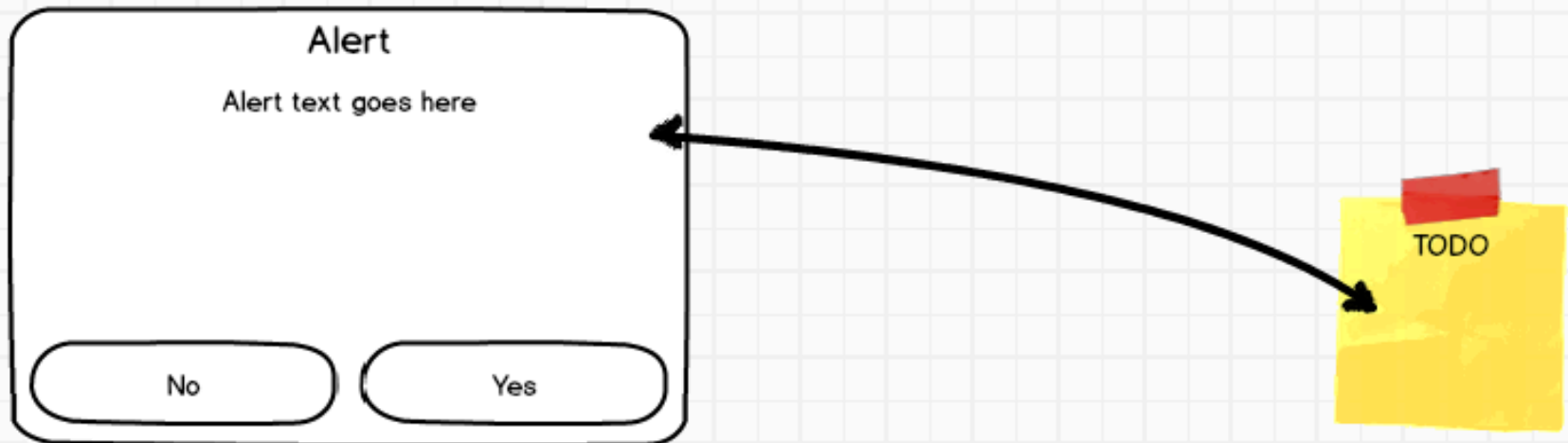
Item Three

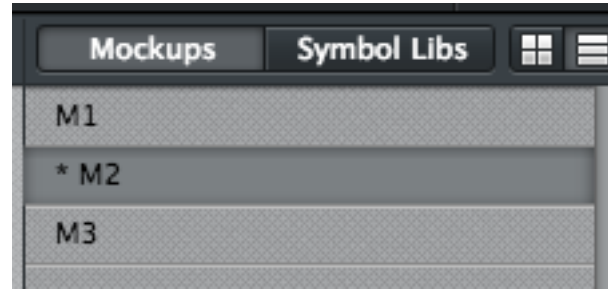
Item Four

Plop 

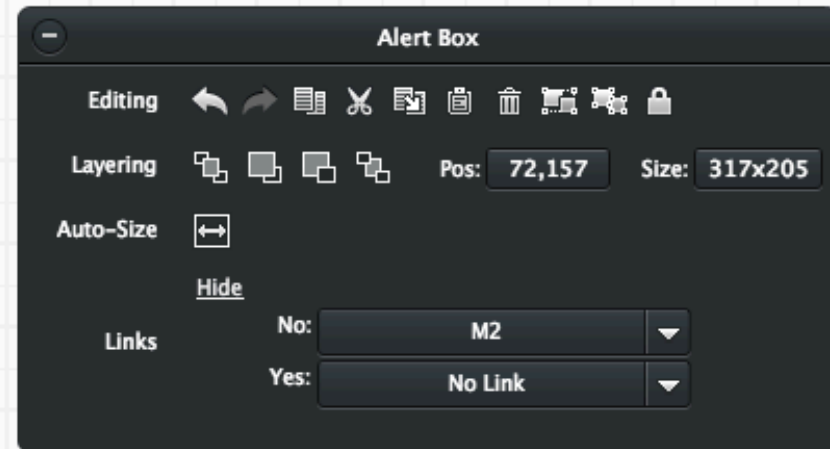
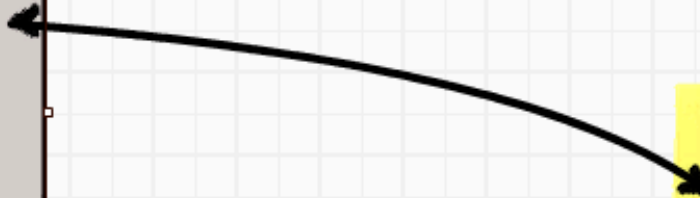


Collaborative



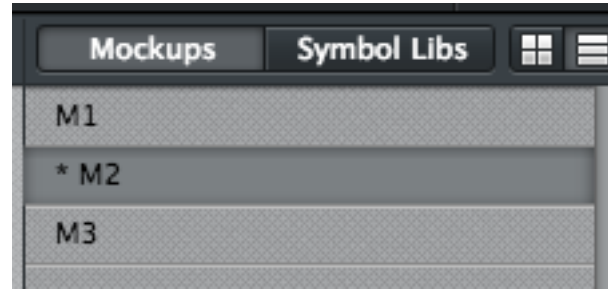


Interaction between Mockups

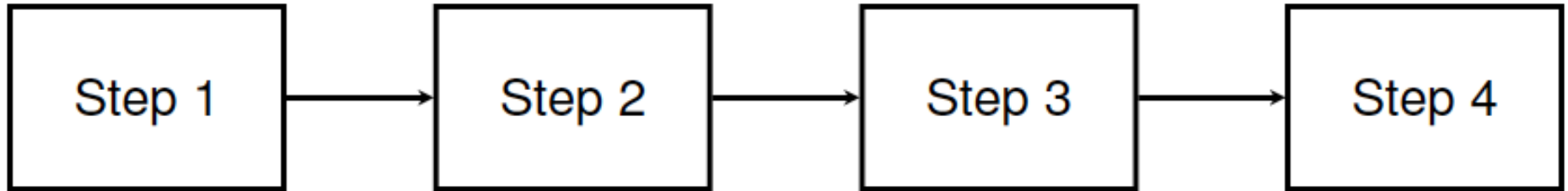


Modélisation UML

- Modélisation selon 4 points de vue principaux :
 - Aspects statiques du système (*le QUI?*)
 - » Description des objets et de leurs relations
 - Modularité, contrats, relations, généricité, héritage
 - » Structuration en paquetages
 - Vision utilisateur du système (*le QUOI?*)
 - » Cas d'utilisation
 - Aspects dynamiques du système (*le QUAND?*)
 - » Diagramme de séquences (scénarios)
 - » Diagramme de collaborations (entre objets)
 - » Diagramme d'états-transitions (Harel)
 - » Diagramme d'activités
 - Vision implantation (*le OÙ?*)
 - » Diagramme de composants et de déploiement



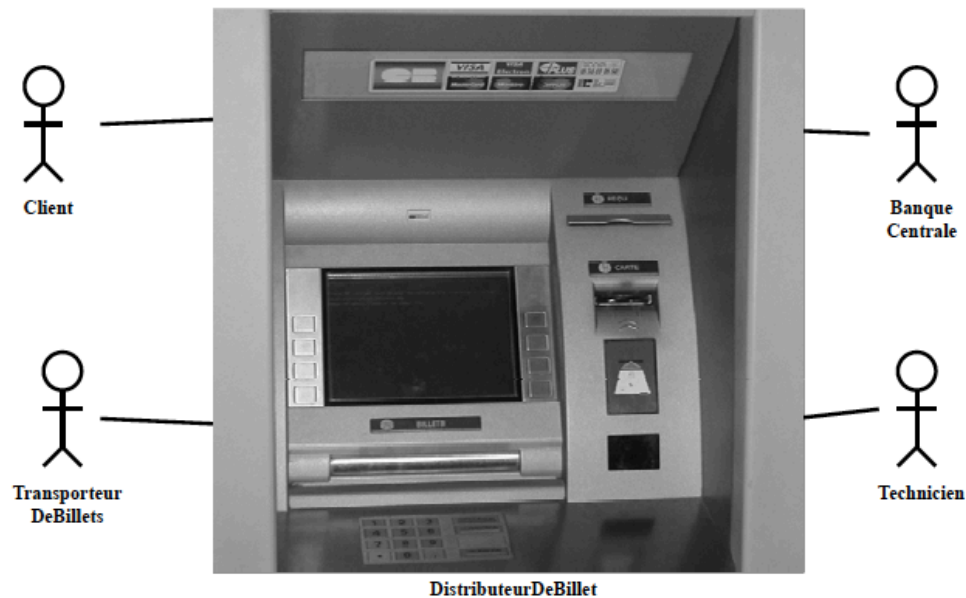
Interaction between Mockups

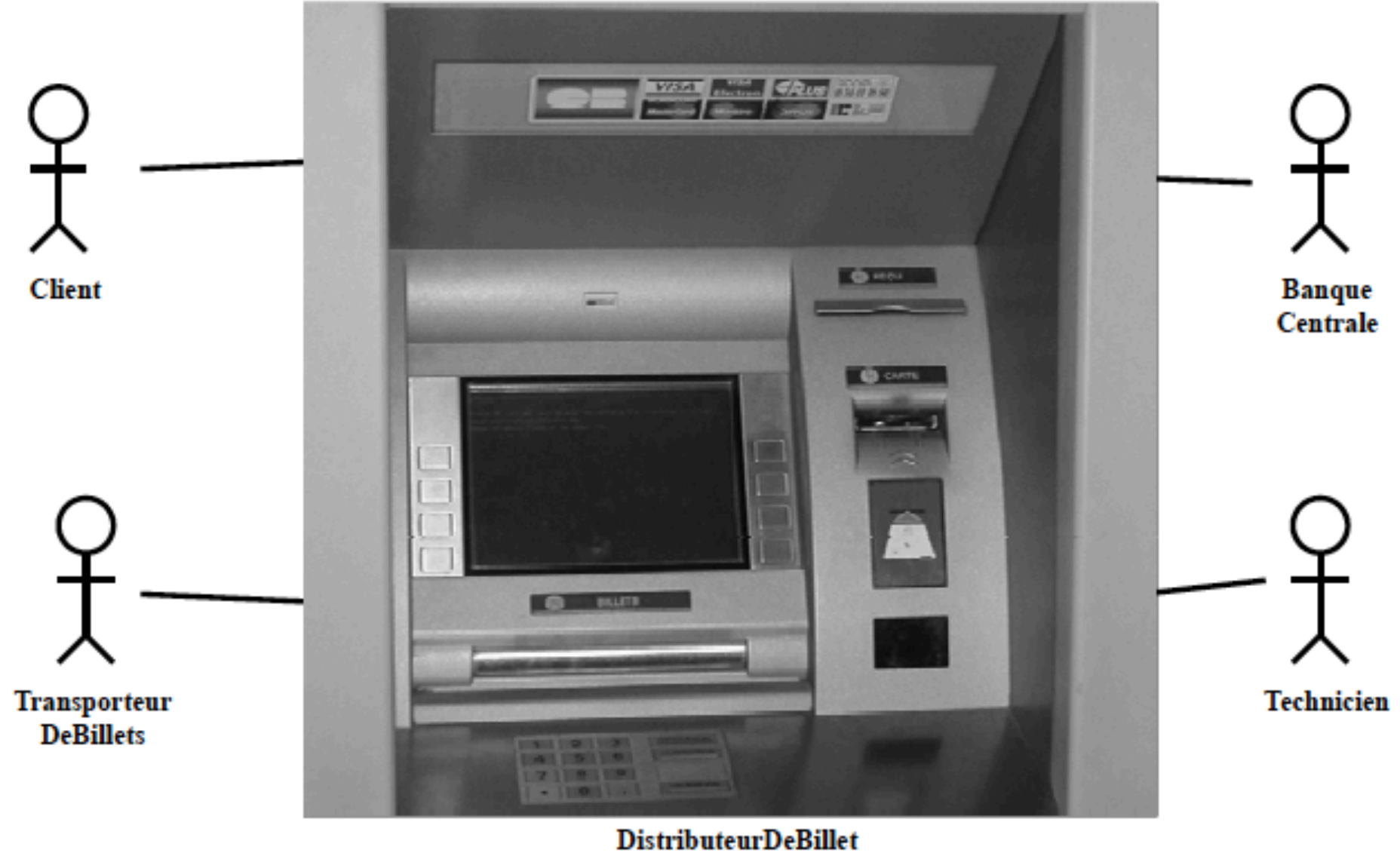


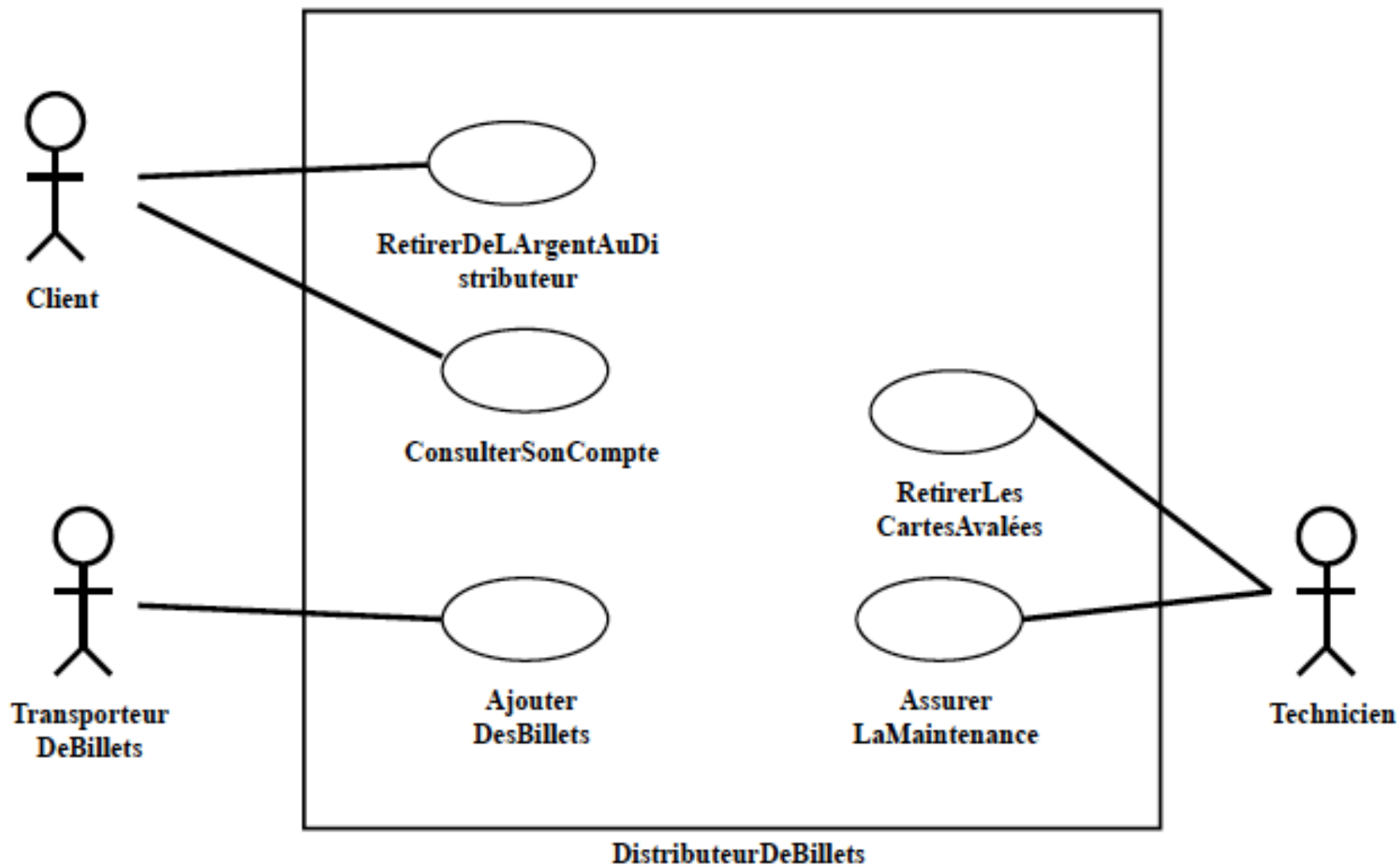
Modélisation UML

- Modélisation selon 4 points de vue principaux :
 - Aspects statiques du système (*le QUI?*)
 - » Description des objets et de leurs relations
 - Modularité, contrats, relations, généricité, héritage
 - » Structuration en paquetages
 - Vision utilisateur du système (*le QUOI?*)
 - » Cas d'utilisation
 - Aspects dynamiques du système (*le QUAND?*)
 - » Diagramme de séquences (scénarios)
 - » Diagramme de collaborations (entre objets)
 - » Diagramme d'états-transitions (Harel)
 - » Diagramme d'activités
 - Vision implantation (*le OU?*)
 - » Diagramme de composants et de déploiement

"A use case is a sequence of transactions in a system whose task is to yield a measurable value to an individual actor of the system." [– Jacobson et al., 1995]







Client

RetirerDeLArgentAuDistributeur

ConsulterSonCompte

TransporteurDeBillets

AjouterDesBillets

RetirerLesCartesAvalées

AssurerLaMaintenance

Technicien

DistributeurDeBillets

■ Pour chaque cas d'utilisation

- ◆ choisir un identificateur représentatif
- ◆ donner une **description textuelle simple**
- ◆ la fonction réalisée doit être comprise de tous
- ◆ préciser ce que fait le système, ce que fait l'acteur
- ◆ pas trop de détails, se concentrer sur le **scénario "normal"**



Retirer
DeLArgent
AuDistributeur

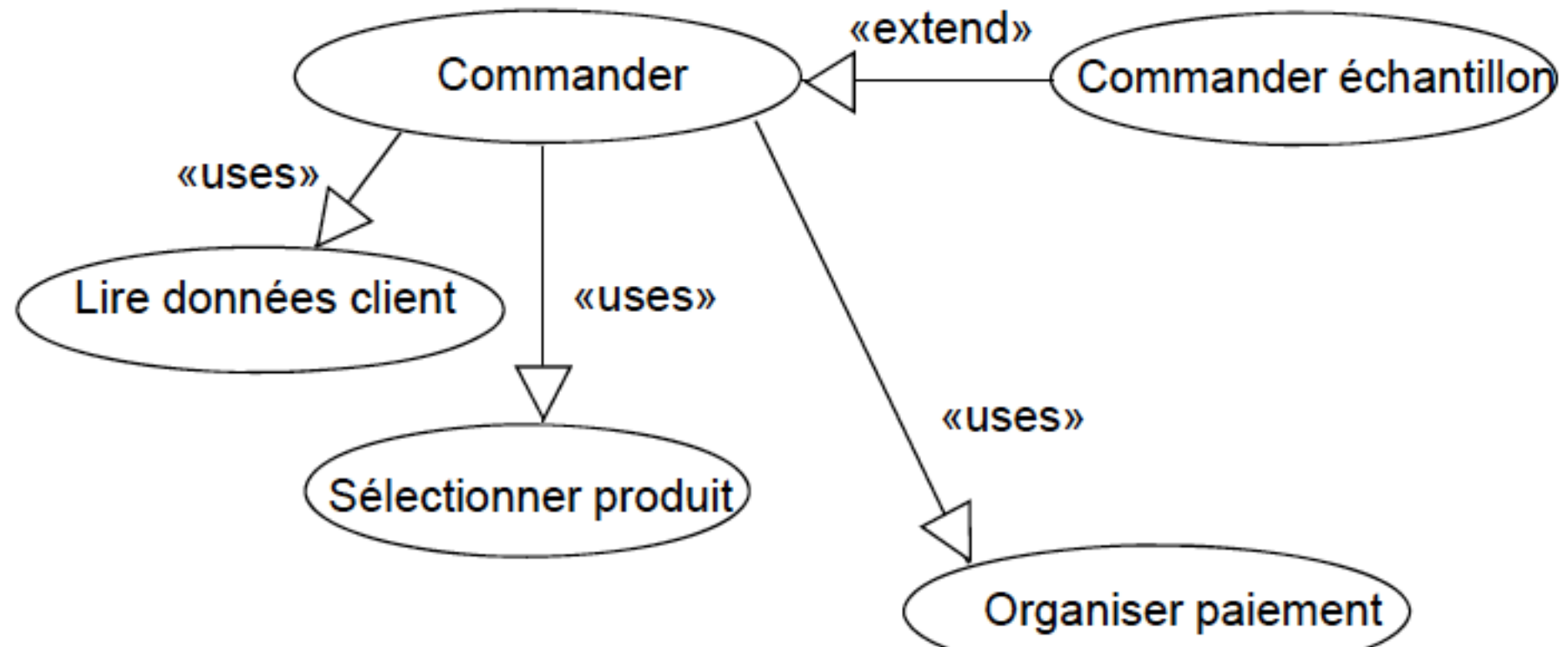
Lorsqu'un *client* a besoin de liquide il peut en utilisant un distributeur retirer de l'argent de son compte. Pour cela :

- le *client* insère sa carte bancaire dans le distributeur
- le *système* demande le code pour l'identifier
- le *client* choisit le montant du retrait
- le *système* vérifie qu'il y a suffisamment d'argent
- si c'est le cas, le *système* distribue les billets et débite le compte du client
- le *client* prend les billets et retire sa carte

Uses: refine the case by use of other cases

Extends: specialization of another use case

Relations sur les *use-cases* : notation

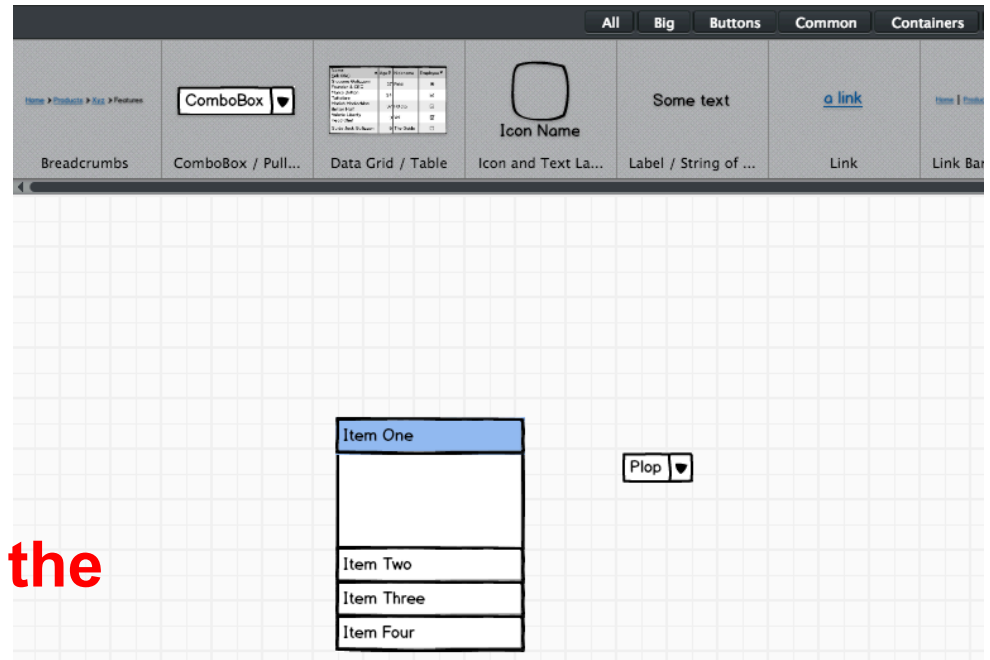


Modélisation UML

- Modélisation selon 4 points de vue principaux :
 - Aspects statiques du système (*le QUI?*)
 - » Description des objets et de leurs relations
 - Modularité, contrats, relations, généricité, héritage
 - » Structuration en paquetages
 - Vision utilisateur du système (*le QUOI?*)
 - » Cas d'utilisation
 - Aspects dynamiques du système (*le QUAND?*)
 - » Diagramme de séquences (scénarios)
 - » Diagramme de collaborations (entre objets)
 - » Diagramme d'états-transitions (Harel)
 - » Diagramme d'activités
 - Vision implantation (*le OU?*)
 - » Diagramme de composants et de déploiement

© J.-M. Jézéquel

UML diagrams



Complementary!
Focus on some aspects of the system/requirements

Software Engineering



Visual Basic



Code::Blocks Studio

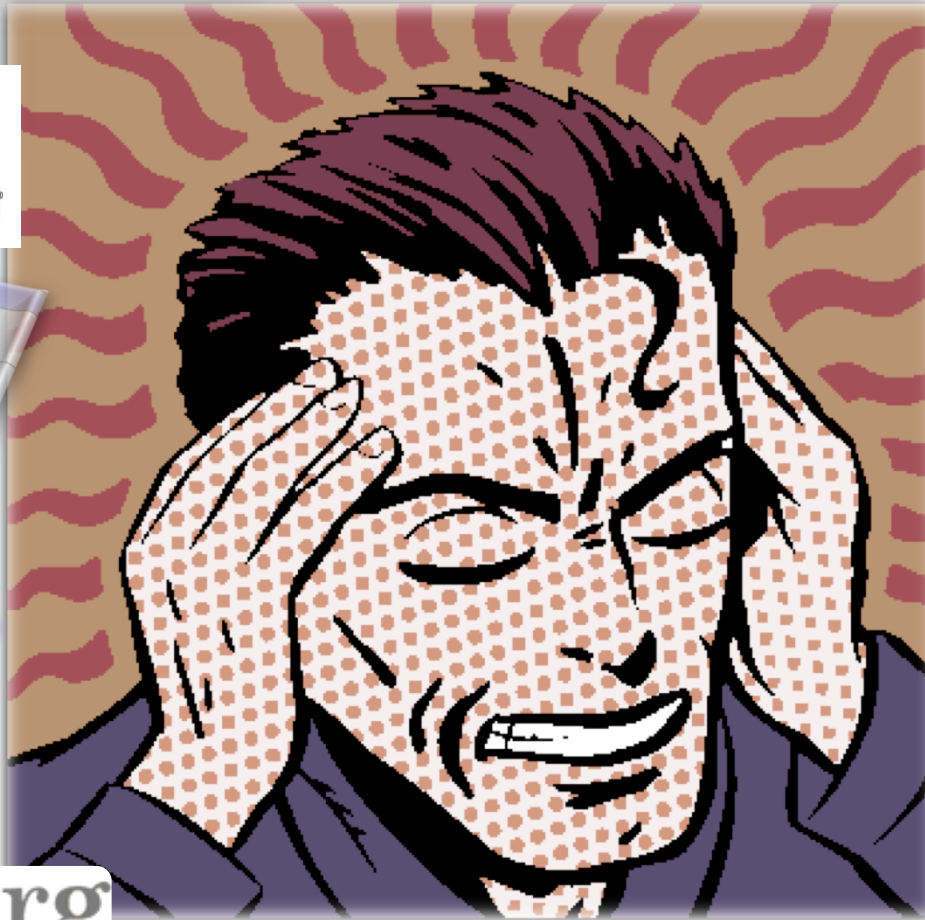
eclipse



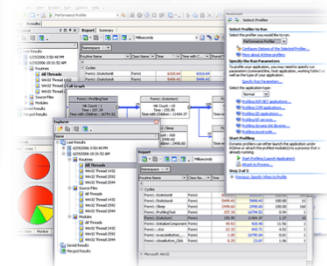
Microsoft Visual Studio



JUnit.org



git



Documentation

- Source code: one of the best artefact for documenting a project
- Javadoc (JDK)
 - Automatic **generation** of HTML documentation
 - Using comments in java files
- Syntax

```
/**  
 * This is a <b>doc</b> comment.  
 * @see java.lang.Object  
 * @todo fix {@underline this !}  
 */
```
- Includes
 - class hierarchy, interfaces, packages
 - detailed summary of class, interface, methods, attributes
- Note
 - Add doc generation to your favorite **compile chain**



Package javax.swing

Provides a set of "lightweight" (all-Java language) components that, to the maximum degree possible, work the same on all platforms.

See:

[Description](#)

Interface Summary

Action	The <code>Action</code> interface provides a useful extension to the <code>ActionListener</code> interface in cases where the same functionality may be accessed by several controls.
BoundedRangeModel	Defines the data model used by components like Sliders and ProgressBars.
ButtonModel	State Model for buttons.
CellEditor	This interface defines the methods any general editor should be able to implement.
ComboBoxEditor	The editor component used for JComboBox components.
ComboBoxModel	A data model for a combo box.
DesktopManager	DesktopManager objects are owned by a JDesktopPane object.
Icon	A small fixed size picture, typically used to decorate components.
JComboBox.KeySelectionManager	The interface that defines a <code>KeySelectionManager</code> .
ListCellRenderer	Identifies components that can be used as "rubber stamps" to paint the cells in a JList.
ListModel	This interface defines the methods components like JList use to get the value of each cell in a list and the length of the list.
ListSelectionModel	This interface represents the current state of the selection for any of the components that display a list of values with stable indices.
MenuItem	Any component that can be placed into a menu should implement this interface.
MutableComboBoxModel	A mutable version of <code>ComboBoxModel</code> .
Renderer	Defines the requirements for an object responsible for "rendering" (displaying) a value.
RootPaneContainer	This interface is implemented by components that have a single <code>JRootPane</code> child: <code>JDialog</code> , <code>JFrame</code> , <code>JWindow</code> , <code>JApplet</code> , <code>JInternalFrame</code> .
Scrollable	An interface that provides information to a scrolling container like <code>JScrollPane</code> .
ScrollPaneConstants	Constants used with the <code>JScrollPane</code> component.
SingleSelectionModel	A model that supports at most one indexed selection.
SpinnerModel	A model for a potentially unbounded sequence of object values.
SwingConstants	A collection of constants generally used for positioning and orienting components on the screen.
UIDefaults.ActiveValue	This class enables one to store an entry in the defaults table that's constructed each time it's looked up with one of the <code>getXXX(key)</code> methods.
UIDefaults.LazyValue	This class enables one to store an entry in the defaults table that isn't constructed until the first time it's looked up with one of the <code>getXXX(key)</code> methods.
WindowConstants	Constants used to control the window closing operation.

public class **JFrame**
extends [Frame](#)
implements [WindowConstants](#), [Accessible](#), [RootPaneContainer](#)

An extended version of `java.awt.Frame` that adds support for the JFC/Swing component architecture. You can find task-o

The `JFrame` class is slightly incompatible with `Frame`. Like all other JFC/Swing top-level containers, a `JFrame` contains a `JFrame` content pane, unlike the AWT `Frame` case. For example, to add a child to an AWT frame you'd write:

```
frame.add(child);
```

However using `JFrame` you need to add the child to the `JFrame`'s content pane instead:

```
frame.getContentPane().add(child);
```

The same is true for setting layout managers, removing components, listing children, and so on. All these methods should not throw an exception. The default content pane will have a `BorderLayout` manager set on it.

update

```
public void update(Graphics g)
```

Just calls `paint(g)`. This method was overridden to prevent an unnecessary call to clear the background.

Overrides:

[update](#) in class [Container](#)

Parameters:

`g` - the Graphics context in which to `paint`

See Also:

[Component.update\(Graphics\)](#)



Kornel Kisielewicz @epyoncf

12 Aug

ProTip: "//" is the speedup operator. Use // before the statement you want to speed up. Works in C++, Java and a few others!

 Retweeted by Mathieu Acher

[Collapse](#)

[← Reply](#) [↻ Retweeted](#) [★ Favorite](#) [⋮ More](#)

1,253

RETWEETS

295

FAVORITES



12:31 AM - 12 Aug 13 · [Details](#)

Coding Conventions

- Rules on the coding style :
 - Apache, Oracle and others template
 - e.g.
 - <http://www.oracle.com/technetwork/java/codeconv-138413.html>
 - <http://geosoft.no/development/javastyle.html>
- Verification tools
 - CheckStyle, PMD, JackPot, Spoon Vsuite...
 - Some integrated into IDEs

Why Coding Standards are Important?

- Lead to greater **consistency** within your code and the code of your teammates
- Easier to **understand**
- Easier to **develop**
- Easier to **maintain**
- Reduces overall cost of application

Example

8. Private class variables should have underscore suffix.

```
class Person
{
    private String name_;
    ...
}
```

Apart from its name and its type, the *scope* of a variable is its most higher significance than method variables, and should be treated w

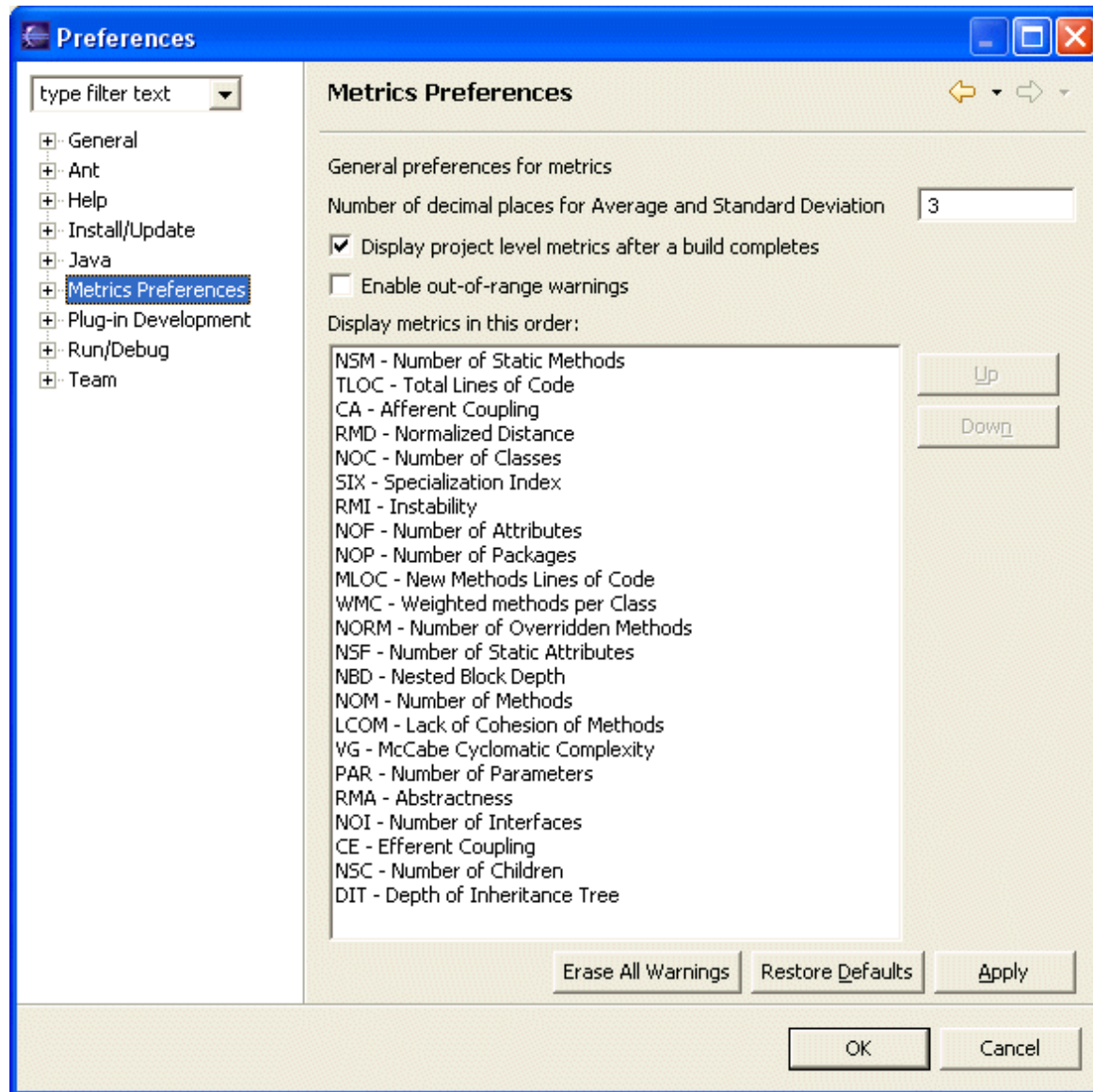
A side effect of the underscore naming convention is that it nicely r

```
void setName(String name)
{
    name_ = name;
}
```

Tools to Improve your Source code

- Formatting tools
 - Indenteurs (Jindent), beautifiers, stylers (JavaStyle), ...
- « Bug fixing » tools
 - Spoon VSuite, Findbugs (sourceforge) ...
- Quality report tools : code metrics
 - Number of Non Comment Code Source, Number of packages, Cyclomatic numbers, ...
 - JavaNCCS, Eclipse Metrics ...

Code Quality Metrics



Code Quality Metrics

- Others (standalone or as IDE plugins)
 - <http://metrics.sourceforge.net/>
 - <http://qjpro.sourceforge.net/>
 - http://www.geocities.com/sivaram_subr/index.htm
 - ...

Refactoring

What's Code Refactoring?

“A series of *small* steps, each of which changes the program's *internal structure* without changing its *external behavior*“



Martin Fowler

Example

Which code segment is easier to read?

Sample 1:

```
if (markT>=0 && markT<=25 && markL>=0 && markL<=25) {  
    float markAvg = (markT + markL) / 2;  
    System.out.println("Your mark: " + markAvg);  
}
```

Sample 2:

```
if (isValid(markT) && isValid(markL)) {  
    float markAvg = (markT + markL) / 2;  
    System.out.println("Your mark: " + mark);  
}
```


Why do we Refactor?

- Improves the design of our software
 - Design pattern!
- Minimizes technical debt
- Keep development at speed
- To make the software easier to understand
- To help find bugs
- To “Fix broken windows”

Non exhaustive (code smell)

(and not necessarily smells in all situations)

- Duplicated code
- Feature Envy
- Inappropriate Intimacy
- Comments
- Long Method
- Long Parameter List
- Switch Statements
- Improper Naming

Code Smell examples (1)

```
public void display(String[] names) {
    System.out.println("-----");
    for(int i=0; i<names.length; i++){
        System.out.println(" + " + names[i]);
    }
    System.out.println("-----");
}
```

```
public void listMember(String[] names) {
    System.out.println("List all member: ");
    System.out.println("-----");
    for(int i=0; i<names.length; i++){
        System.out.println(" + " + names[i]);
    }
    System.out.println("-----");
}
```

Duplicated code

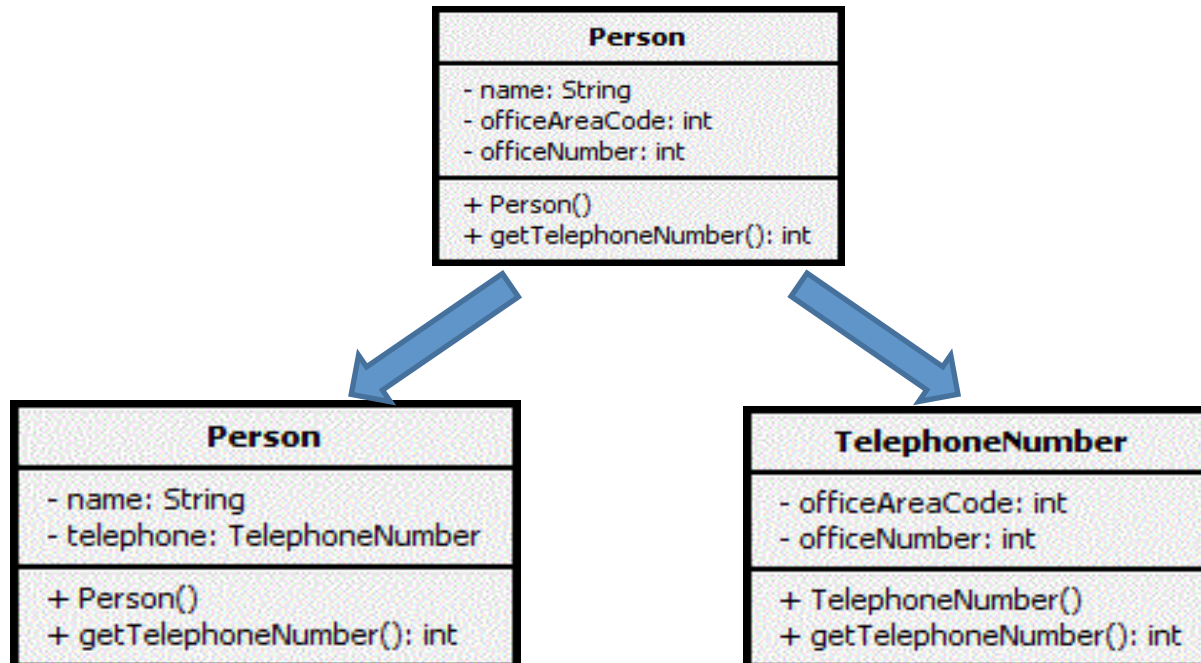
Code Smell examples (2)

```
public String formatStudent( int id,  
                             String name,  
                             Date dob,  
                             String province,  
                             String address,  
                             String phone ){  
  
    //TODO:  
    return null;  
}
```

Long list of parameters

Improving design

- Move Method or Move Field – move to a more appropriate Class or source file
- Rename Method or Rename Field – changing the name into a new one that better reveals its purpose
 - Pull Up – in OOP, move to a superclass
 - Push Down – in OOP, move to a subclass



How do we Refactor?

- Manual Refactoring
 - Code Smells
- Automated/Assisted Refactoring
 - Refactoring by hand is time consuming and prone to error
 - Tools (IDE)
- In either case, **test your changes**

```
package de.vogella.eclipse.ide.first;

public class MyFirstClass {

    public static void main(String[] args) {
        System.out.println("Hello Eclipse!");
        int sum = 0;
        for (int i = 0; i <= 100; i++) {
            sum += i;
        }
        System.out.println(sum);
    }
}
```

Problems Javadoc Declaration Console Error Log

<terminated> MyFirstClass [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe

Hello Eclipse!
5050

Extract Method

Method name:

Access modifier: public protected default private

Parameters:

Type	Name
int	sum

Declare thrown runtime exceptions
 Generate method comment
 Replace additional occurrences of statements with method

Method signature preview:
private static int calculateSum(int sum)

Preview > OK Cancel

```
package de.vogella.eclipse.ide.first;

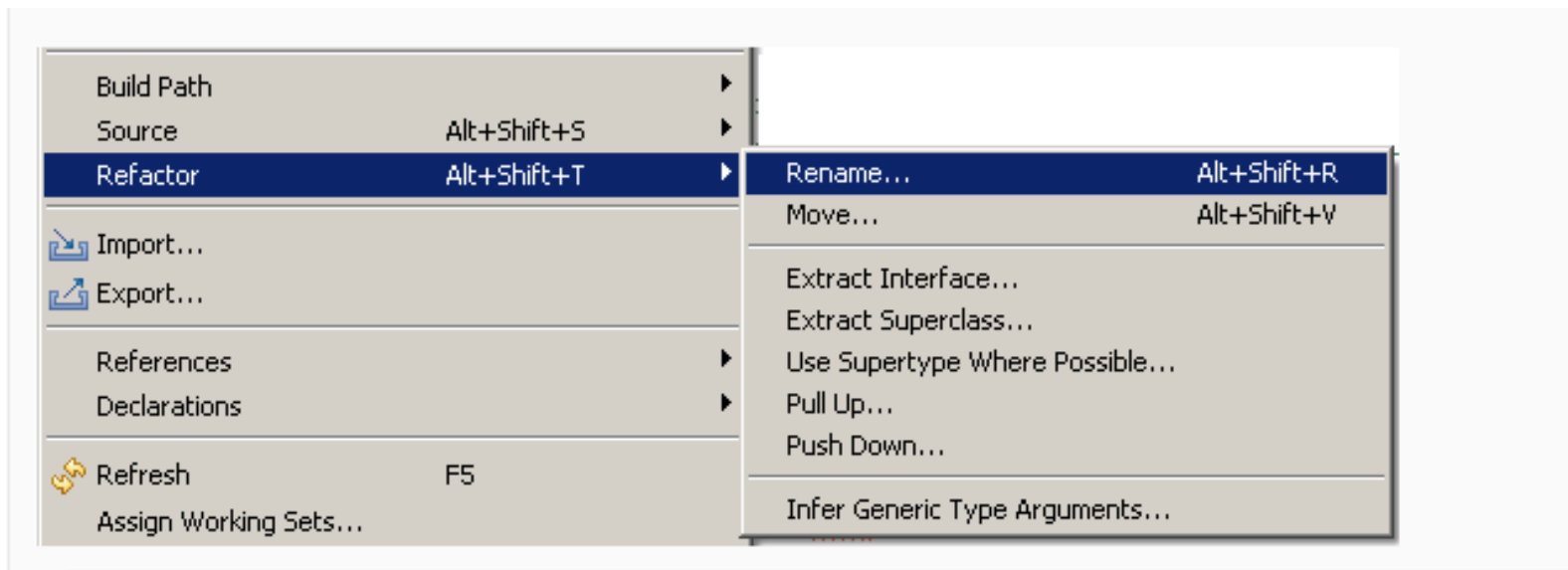
public class MyFirstClass {

    public static void main(String[] args) {
        System.out.println("Hello Eclipse!");
        int sum = 0;
        sum = calculateSum(sum);
        System.out.println(sum);
    }

    private static int calculateSum(int sum) {
        for (int i = 0; i <= 100; i++) {
            sum += i;
        }
        return sum;
    }
}
```

Typical refactoring patterns

- Rename variable / class / method / member
- Extract method
- Extract constant
- Extract interface
- Encapsulate field



You have constructors on subclasses with mostly identical bodies.

Create a superclass constructor; call this from the subclass methods.

Pull Up Constructor Body

```
class Manager extends Employee...  
    public Manager (String name, String id, int grade) {  
        _name = name;  
        _id = id;  
        _grade = grade;  
    }
```

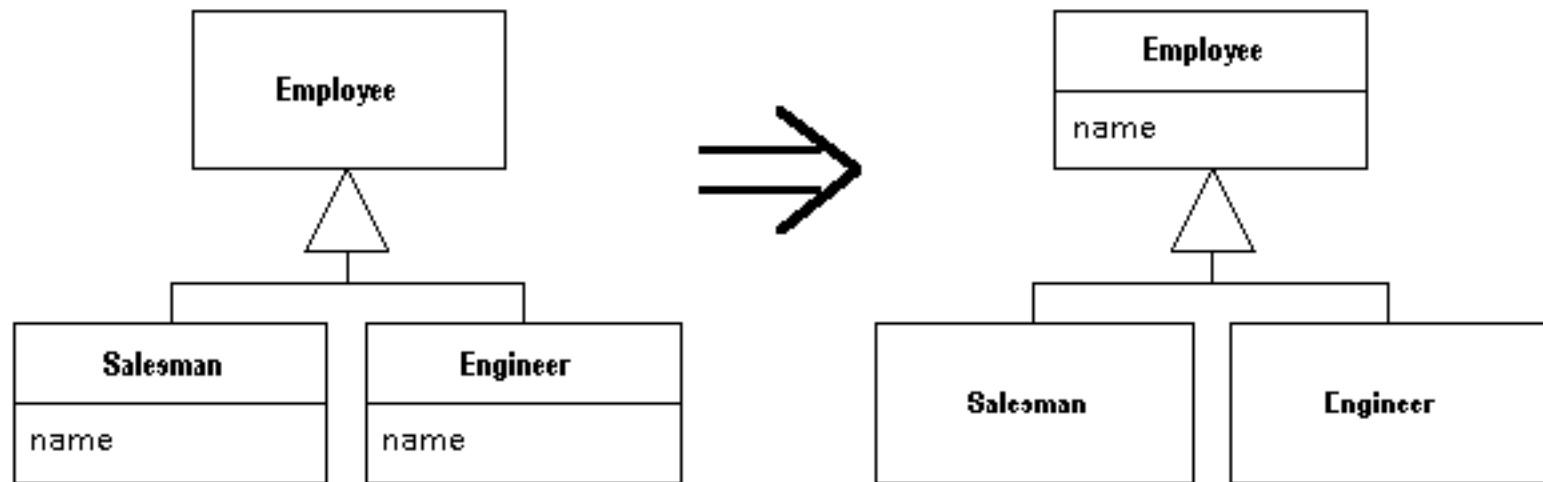
```
    public Manager (String name, String id, int grade) {  
        super (name, id);  
        _grade = grade;  
    }
```

You

Create

Two subclasses have the same field.

Move the field to the superclass.



You have a complicated expression.

Put the result of the expression, or parts of the expression, in a temporary variable with a name that explains the purpose.


```
if ( (platform.toUpperCase().indexOf("MAC") > -1) &&
      (browser.toUpperCase().indexOf("IE") > -1) &&
      wasInitialized() && resize > 0 )
{
    // do something
}

final boolean isMacOs      = platform.toUpperCase().indexOf("MAC") > -1;
final boolean isIEBrowser = browser.toUpperCase().indexOf("IE") > -1;
final boolean wasResized  = resize > 0;

if (isMacOs && isIEBrowser && wasInitialized() && wasResized)
{
    // do something
}
```

Testing

...the activity of finding out whether a piece of code (a method, class or program) produces the intended behavior

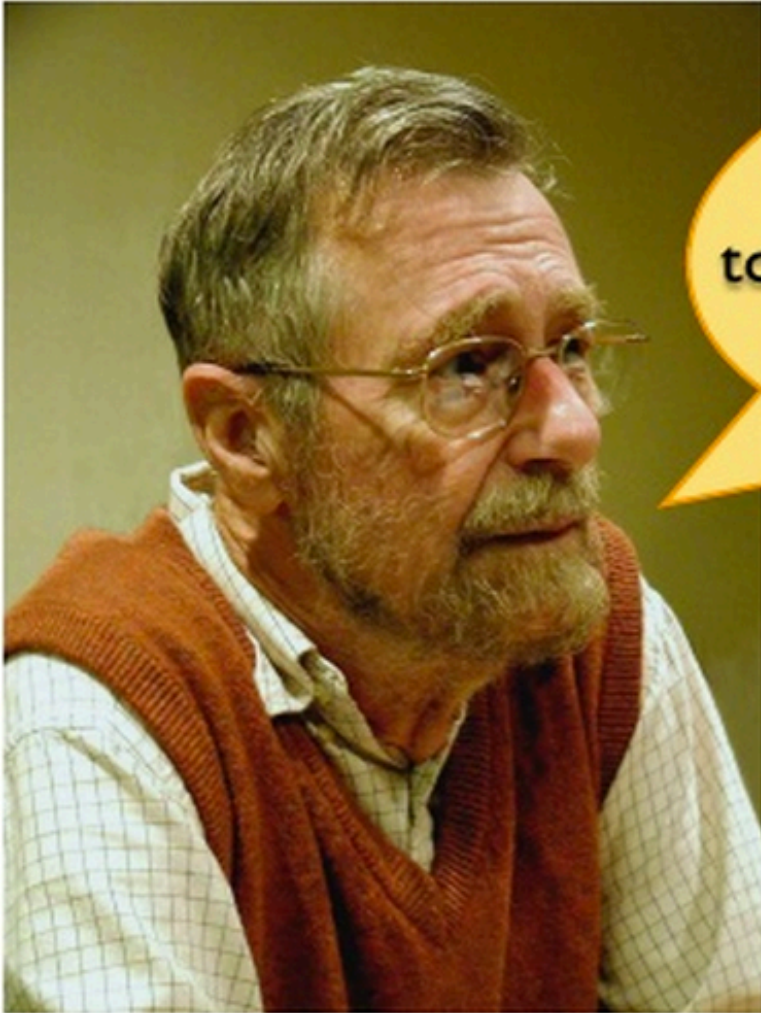
A blue starburst shape with multiple points, centered on a white background. The text is written in a black, sans-serif font within the starburst.

This part is largely
inspired by Thomas
Zimmermann slides

Your hope as a programmer

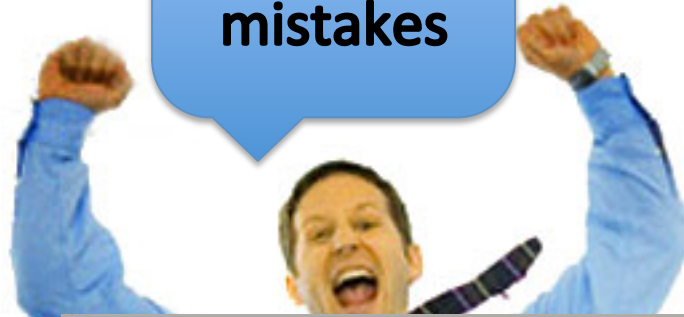
« A program does
exactly what you
expected to do »

Dijkstra



Program testing can be used to show the presence of bugs, but never to show their absence!

**I don't
make
mistakes**



Master 2 (Apprentis)

15 « jobs », 15 aim at

Testing (critical or non critical) applications

Correcting anomalies and ensuring that they won't appear in the future

Maintaining

« 1 day of producing code
= 3 days of testing code »

« 70% of a software project = maintenance »

10. HealthCare.gov didn't have enough testing before going live.

This became clear in a series of Congressional hearings, where federal contractors testified that end-to-end testing only began in the final weeks of September, right before the Oct. 1 launch. When pressed on how much time would have been ideal for testing, one contractor told lawmakers that “months would have been nice.”

<http://www.washingtonpost.com/blogs/wonkblog/wp/2013/11/01/thirty-one-things-we-learned-in-healthcare-govs-first-31-days/>

Test phases



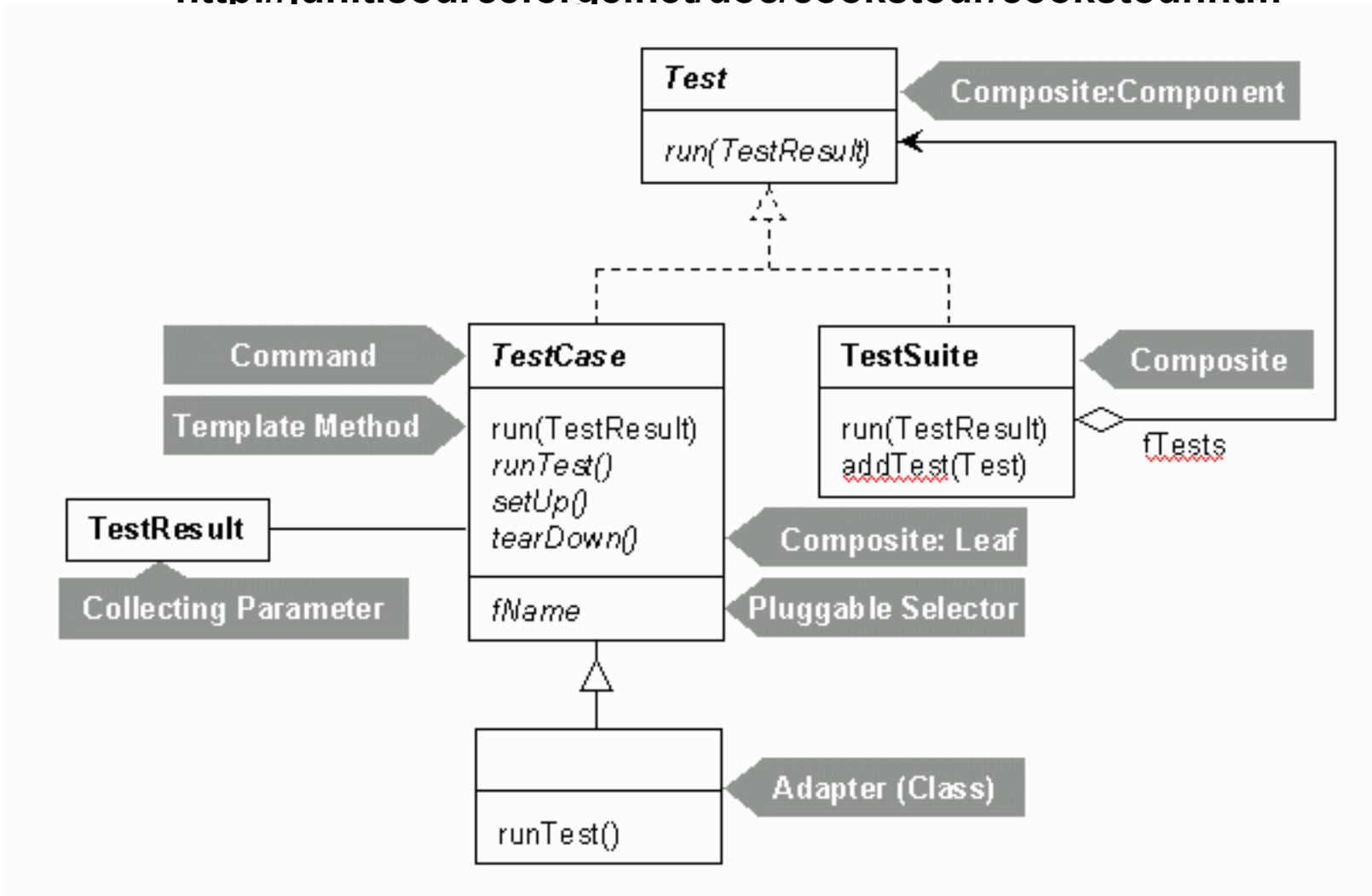
Unit testing on individual units of source code (=smallest testable part).

Integration testing on groups of individual software modules.

System testing on a complete, integrated system (evaluate compliance with requirements)

Junit and... Design Patterns

<http://junit.sourceforge.net/doc/cookstour/cookstour.htm>



Running example

- 1 Set of products
- 2 Number of products
- 3 Balance

Price: **CDN\$ 10.94**
In Stock
Ships from and sold by Amazon.ca

Quantity:

 Add to Shopping Cart

or
[Sign in](#) to turn on 1-Click ordering.

1 Add product

 **Shopping Cart** Already a customer?
[Sign in](#)

 See more items like those in your cart

Subtotal: CDN\$ 10.94

Make any changes below?

Shopping Cart Items--To Buy Now

Item added on
April 26 2007

Harry Potter and the Half-Blood Prince (Book 6) [Adult Edition] - J. K. Rowling; **Mass Market Paperback**
In Stock

Price: **CDN\$ 10.94**

Qty:
You Save:
CDN\$ 4.05
(27%)

2 Remove product

Init

Constructor + Set up and tear down of fixture.

```
import junit.framework.Test;
import junit.framework.TestCase;
import junit.framework.TestSuite;

public class ShoppingCartTest extends TestCase {

    private ShoppingCart _bookCart;

    // Creates a new test case
    public ShoppingCartTest(String name) {
        super(name);

        // Creates test environment (fixture).
        // Called before every testX() method.
        protected void setUp() {
            _bookCart = new ShoppingCart();

            Product book = new Product("Harry Potter", 23.95);
            _bookCart.addItem(book);
        }

        // Releases test environment (fixture).
        // Called after every testX() method.
        protected void tearDown() {
            _bookCart = null;
        }
    }
}
```

Assertions

`fail(msg)` – triggers a failure named *msg*

`assertTrue(msg, b)` – triggers a failure, when condition *b* is false

`assertEquals(msg, v1, v2)` – triggers a failure, when $v1 \neq v2$

`assertEquals(msg, v1, v2, ϵ)` – triggers a failure, when $|v1 - v2| > \epsilon$

`assertNotNull(msg, object)` – triggers a failure, when *object* is not *null*

`assertNotNull(msg, object)` – triggers a failure, when *object* is *null*

Example #1

```
// Tests adding a product to the cart.  
public void testProductAdd() {  
    Product book = new Product("Refactoring", 53.95);  
    _bookCart.addItem(book);  
  
    assertTrue(_bookCart.contains(book));  
  
    double expected = 23.95 + book.getPrice();  
    double current = _bookCart.getBalance();  
  
    assertEquals(expected, current, 0.0);  
  
    int expectedCount = 2;  
    int currentCount = _bookCart.getItemCount();  
  
    assertEquals(expectedCount, currentCount);  
}
```


Example #2

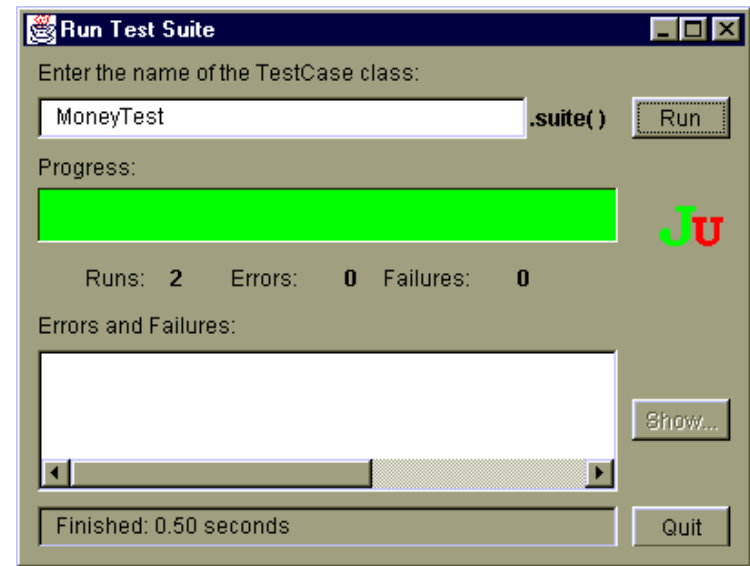
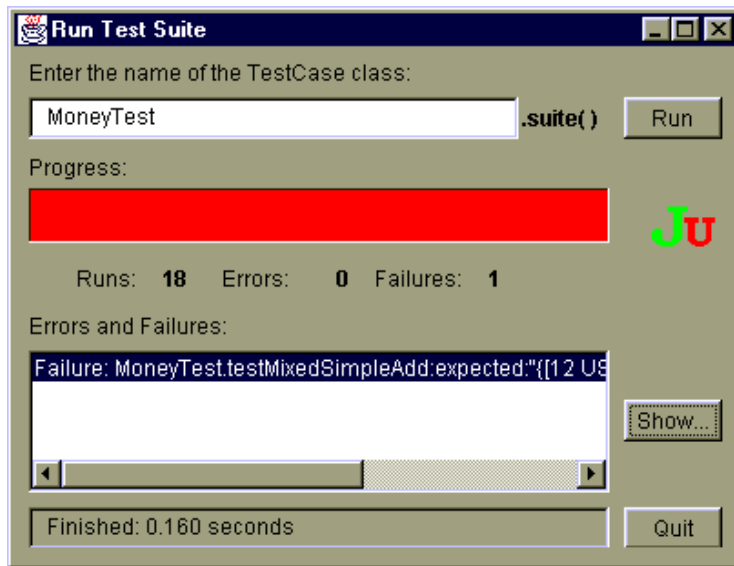
```
// Tests removing a product from the cart.  
public void testProductRemove() throws NotFoundException {  
    Product book = new Product("Harry Potter", 23.95);  
    _bookCart.removeItem(book);  
  
    assertTrue(!_bookCart.contains(book));  
  
    double expected = 23.95 - book.getPrice();  
    double current = _bookCart.getBalance();  
  
    assertEquals(expected, current, 0.0);  
  
    int expectedCount = 0;  
    int currentCount = _bookCart.getItemCount();  
  
    assertEquals(expectedCount, currentCount);  
}
```

```
public static Test suite() {  
    // Here: add all testX() methods to the suite (reflection).  
    TestSuite suite = new TestSuite(ShoppingCartTest.class);  
  
    // Alternative: add methods manually (prone to error)  
    // TestSuite suite = new TestSuite();  
    // suite.addTest(new ShoppingCartTest("testEmpty"));  
    // suite.addTest(new ShoppingCartTest("testProductAdd"));  
    // suite.addTest(...);  
  
    return suite;  
}
```

Unit Test

JUnit 3 and 4 <http://www.junit.org>

- Test pattern
 - Test, TestSuite, TestCase
 - Assertions (assertXX) that must be verified
- TestRunner
 - Chain tests and output a report.



- See JUnit course:
 - <http://membres-liglab.imag.fr/donsez/cours/junit.pdf>

You can't test everything (so one advice by Martin Fowler)

Whenever you are tempted to type something into a print statement or a debugger expression, **write it as a test instead.**



Documenting,
Testing,
Design Patterns,
Refactoring,
Debugging

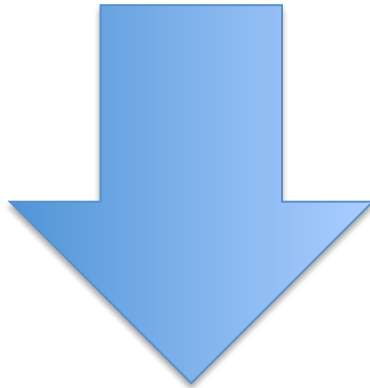
#1 What is the link?

- Documenting
 - Understanding (readability, maintainability)
- Refactoring
 - Improving the design (readability, maintainability, extensibility)
- The activity of documenting can somehow be replaced by the activity of refactoring
 - if the code and architecture is comprehensible by itself

refactoring.com

Documentation and Refactoring

```
if ( (platform.toUpperCase().indexOf("MAC") > -1) && // platform is MacOS
      (browser.toUpperCase().indexOf("IE") > -1) && // browser is IE
      wasInitialized() && resize > 0 )
{
    // do something
}
```



```
final boolean isMacOs      = platform.toUpperCase().indexOf("MAC") > -1;
final boolean isIEBrowser = browser.toUpperCase().indexOf("IE") > -1;
final boolean wasResized  = resize > 0;

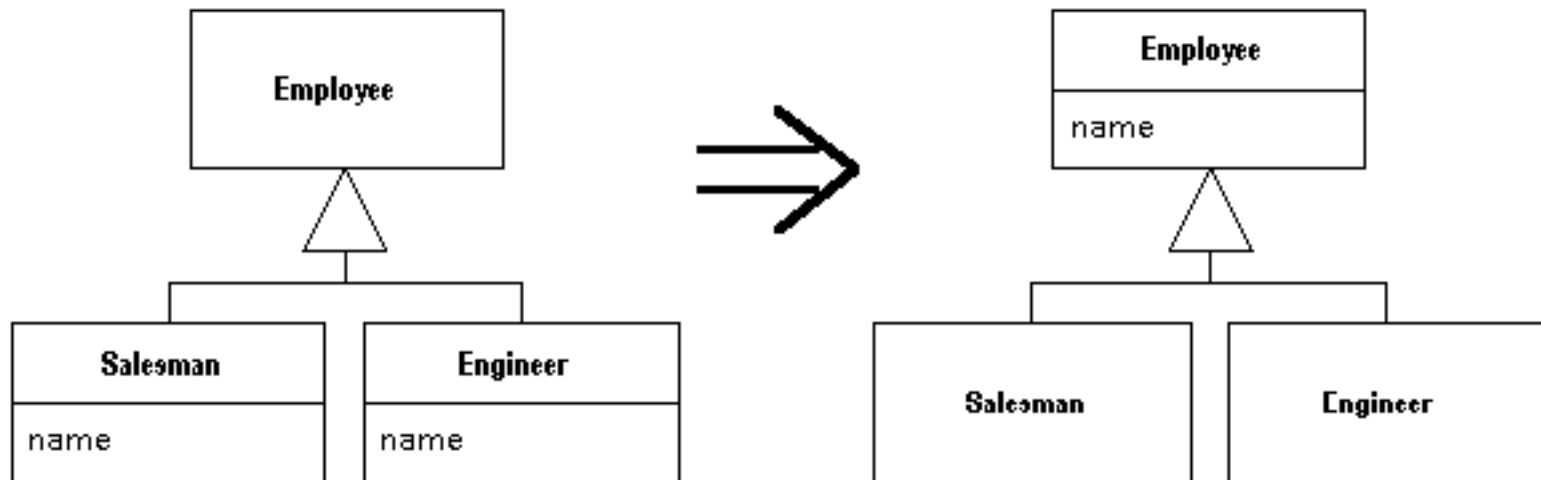
if (isMacOs && isIEBrowser && wasInitialized() && wasResized)
{
    // do something
}
```

#2 What is the link?

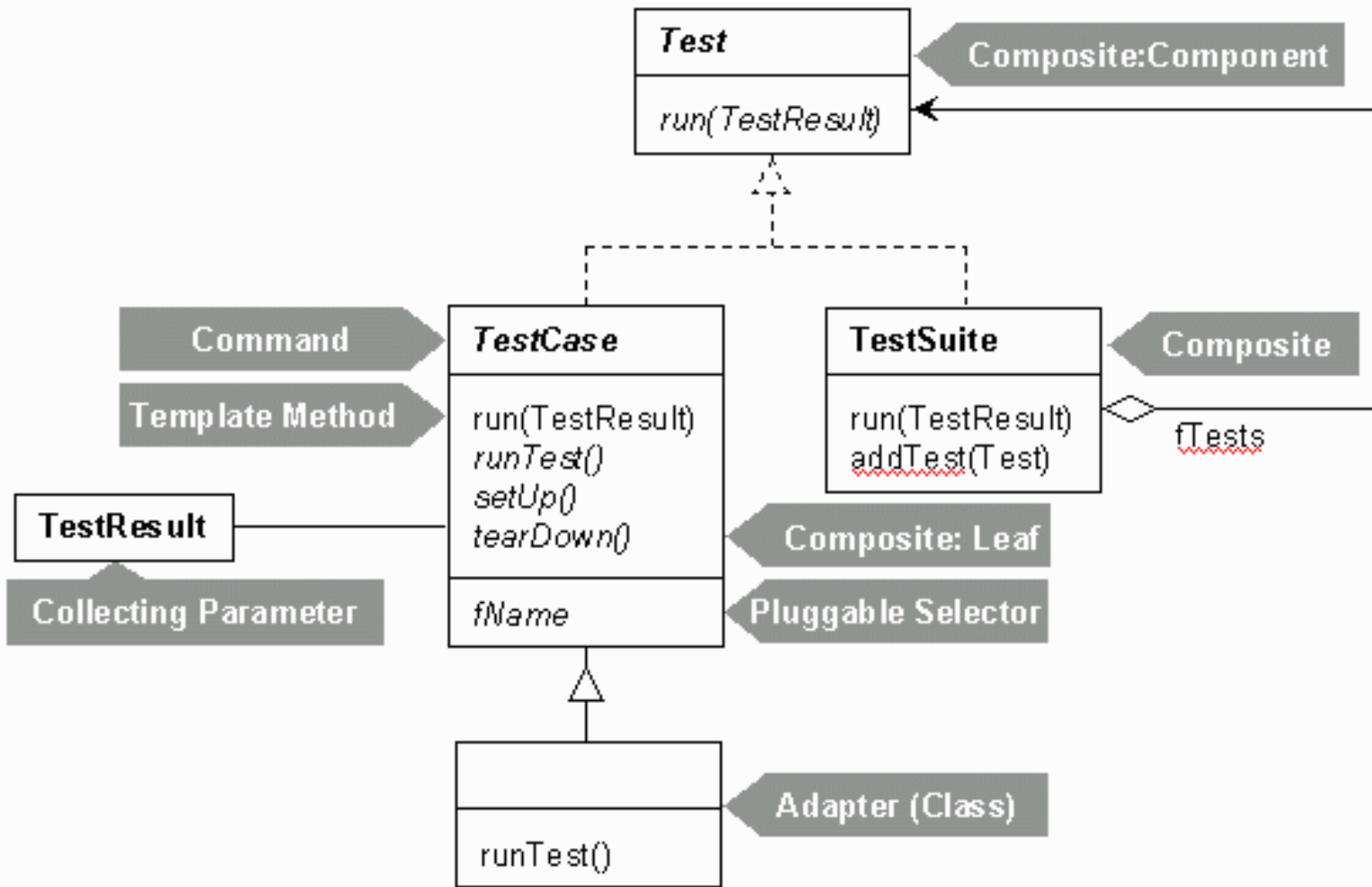
Design patterns: there are refactorings

Two subclasses have the same field.

Move the field to the superclass.



JUnit and... Design patterns



Worth reading!

<http://junit.sourceforge.net/doc/cookstour/cookstour.htm>

#3 What is the link?

- Testing: “the activity of finding out whether a piece of code produces the intended behavior”
 - Debugging can help
 - Testing is better than debugging

Whenever you are tempted to type something into a print statement or a debugger expression, **write it as a test instead.**



What is the link?

- Testability
 - degree to which a system or component **facilitates the establishment** of test criteria and the performance of tests to determine whether those criteria have been met.”
 - Controllability + Observability
- **Controllability** ability to manipulate the software’s input as well as to place this software into a particular state
- **Observability** deals with the possibility to observe the outputs and state changes that
- How to improve Testability?
 - Refactoring, Design patterns

What is the link?

Testing/Refactoring/Design Patterns

- How to improve testability?
- Test-driven Development
 - Write tests first ~ Test-driven design

Let say your first piece of code is... a test

```
// Tests removing a product from the cart.
public void testProductRemove() throws NotFoundException {
    Product book = new Product("Harry Potter", 23.95);
    _bookCart.removeItem(book);

    assertTrue(!_bookCart.contains(book));

    double expected = 23.95 - book.getPrice();
    double current = _bookCart.getBalance();

    assertEquals(expected, current, 0.0);

    int expectedCount = 0;
    int currentCount = _bookCart.getItemCount();

    assertEquals(expectedCount, currentCount);
}
```

What is the link?

- Testing
- Documenting
- Unit tests are one of the best source of documentation
 - One of the entry point to understand a framework
 - It documents the properties of methods, how objects collaborate, etc.

What is the link?

Documenting

Refactoring

Debugging

Testing

Readability
Understandability
Maintainability

Design

Document, refactor... Execute your tests... Debug.. Write test..

And so on!

Documenting

Refactoring

Debugging

Testing

With modern IDE and tools!

```
package de.vogella.eclipse.ide.first;

public class MyFirstClass {

    public static void main(String[] args) {
        System.out.println("Hello Eclipse!");
        int sum = 0;
        for (int i = 0; i <= 100; i++) {
            sum += i;
        }
        System.out.println(sum);
    }
}
```

Extract Method dialog box:

Method name:

Access modifier: public protected default private

Parameters:

Type	Name
int	sum

Declare throw name exceptions
 Generate method comment
 Replace additional occurrences of statements with method

Method signature preview:
`private static int calculateSum(int sum)`

Buttons: Preview >, OK, Cancel

Run Test Suite dialog box:

Enter the name of the Test Case class:
 .suite()

Progress:

Runs: 2 Errors: 0 Failures: 0

Errors and Failures:

Finished: 0.50 seconds