A Journey in Software Development

An overview of methods and tools

Mathieu Acher

Maître de Conférences

mathieu.acher@irisa.fr

UNIVERSITÉ DE RENNES 1

istic Informatique Électronique

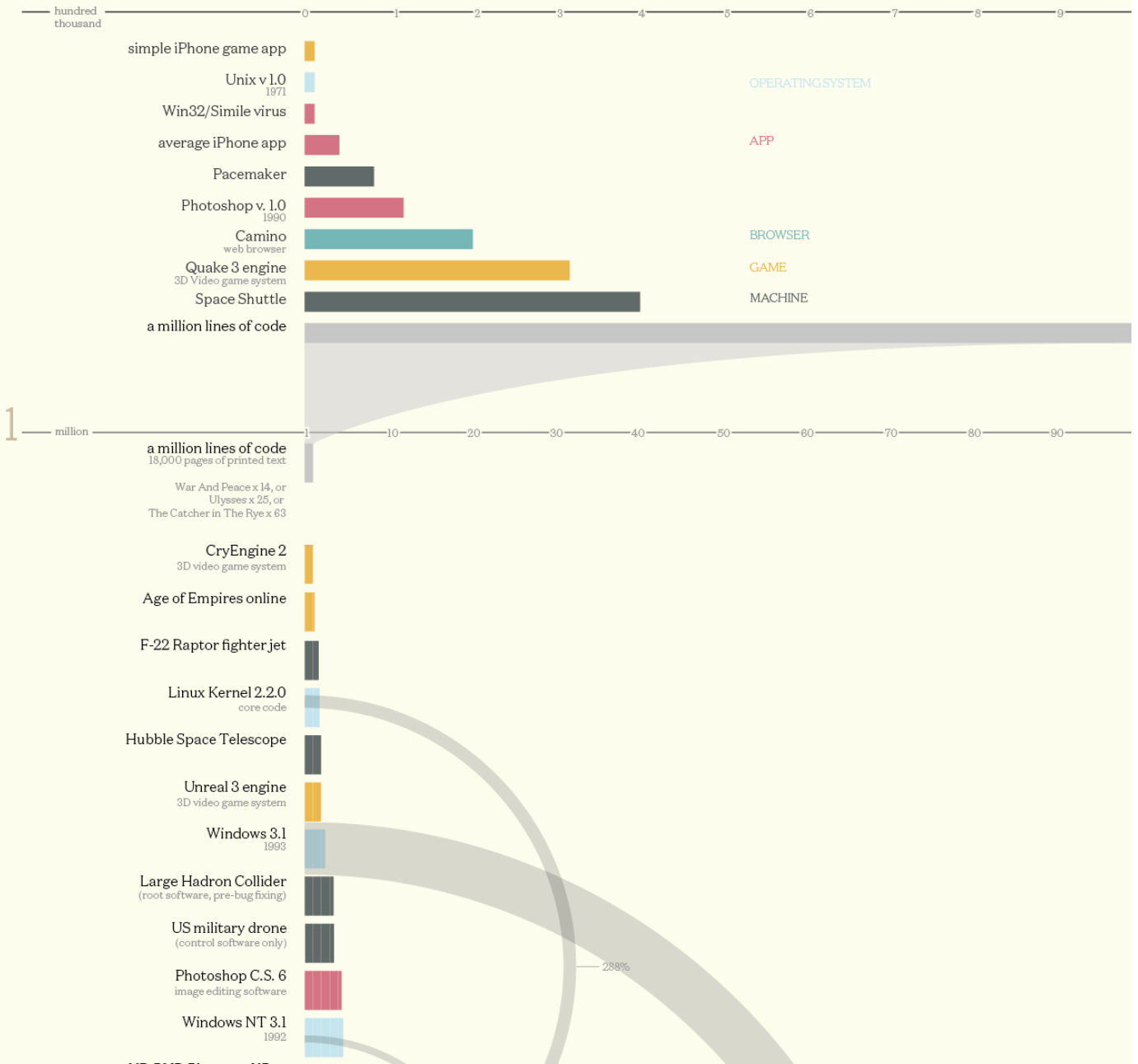
Material

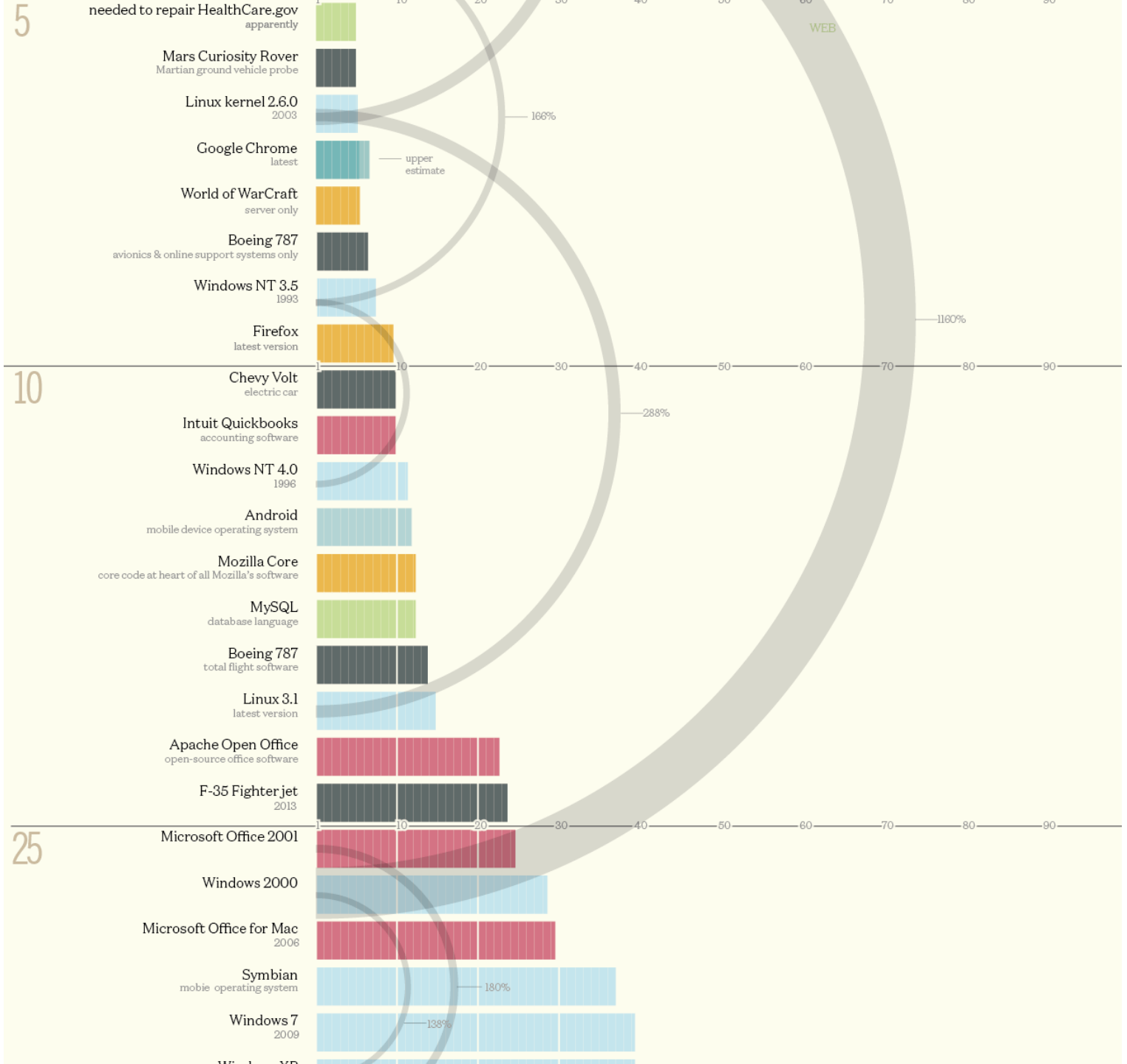
https://github.com/acherm/teaching/tree/master/PDL/

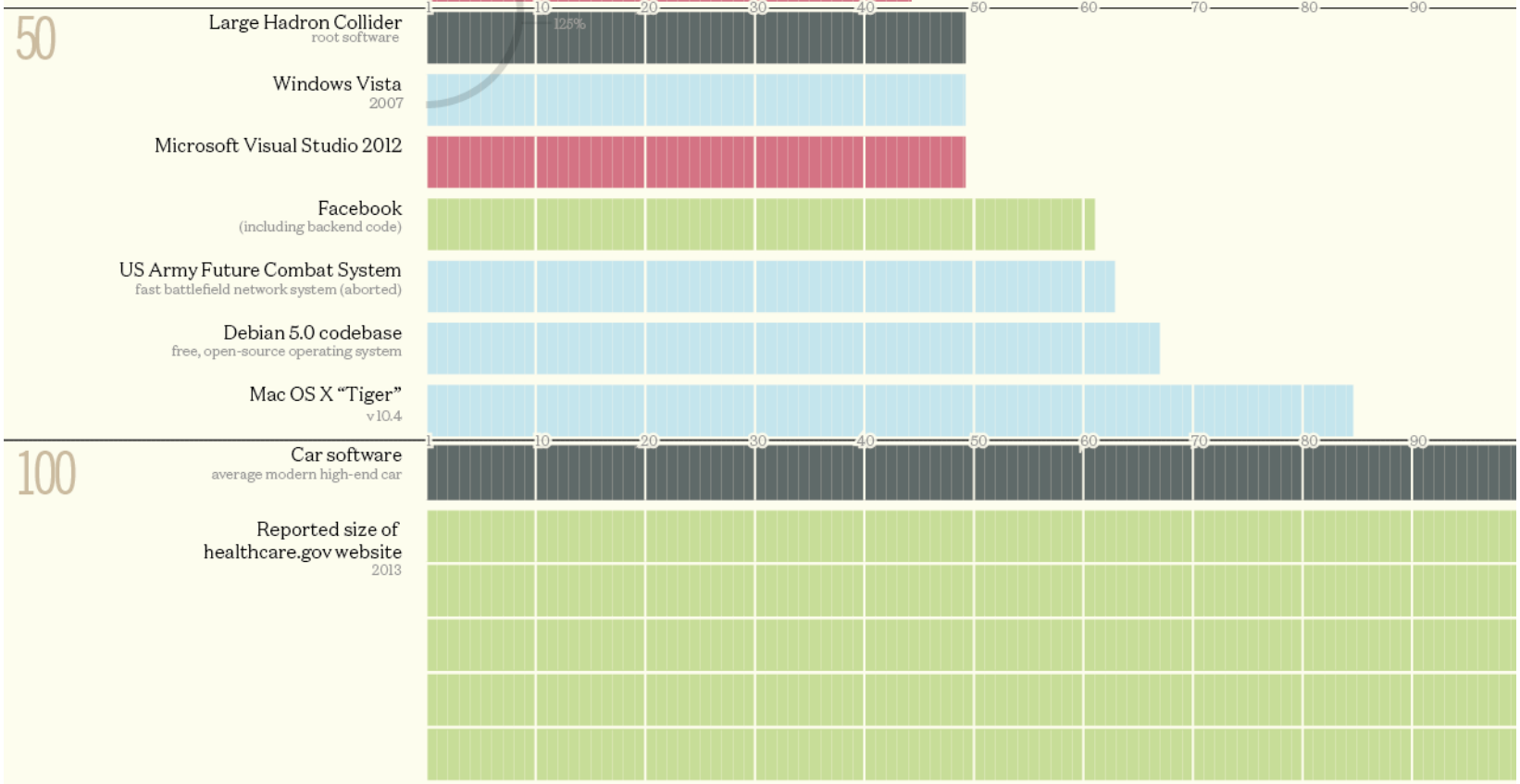
Motivation

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
upper estimate
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
2008
Symbian
mobie operating system
Windows 7
2009
1 10 20 30 40 50 60 70 80 90
WEB
166%
1160%
288%
180%
138%



concept & design: David McCandless

informationisbeautiful.net

research: Pearl Doughty-White, Miriam Quick

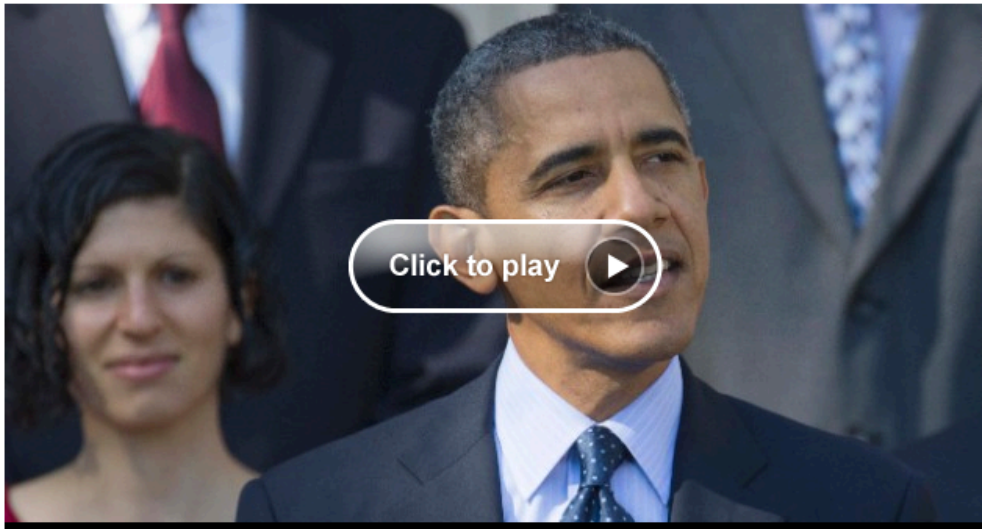
work in progress

v 0.62 // 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

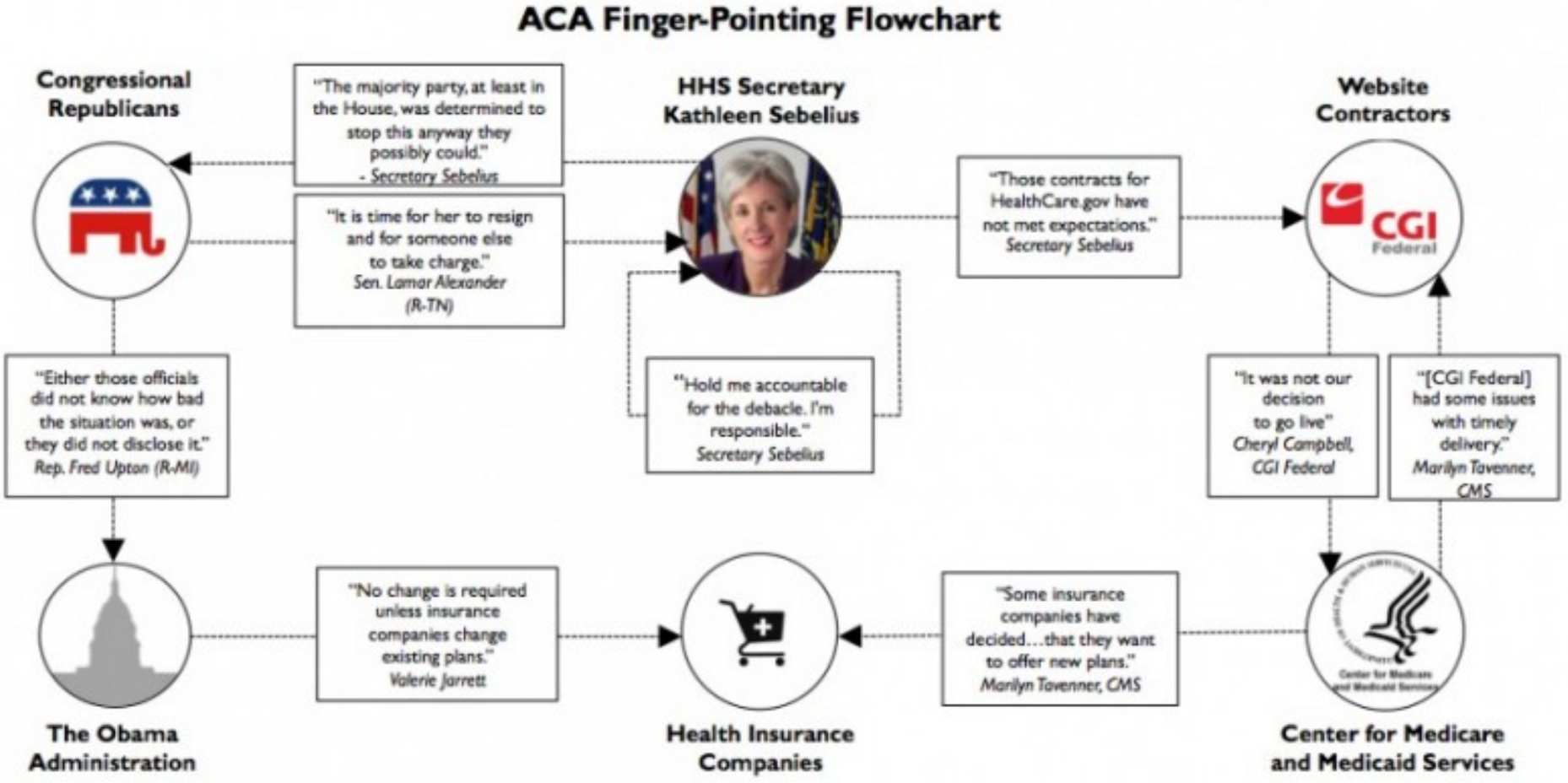
- **NEW:** Top White House official part of "tech surge" on Obamacare
- Obamacare "is not failing" despite website woes, White House spokesman says
- Obama says HealthCare.gov problems are "going to get fixed"
- Secretary Sebelius expected to testify at a congressional hearing next week

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



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

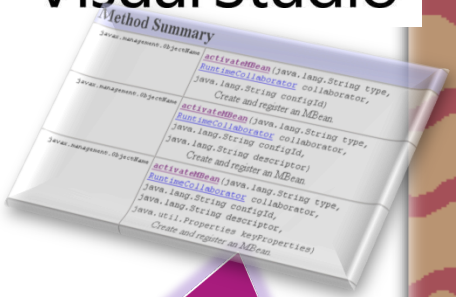
Scalability problem

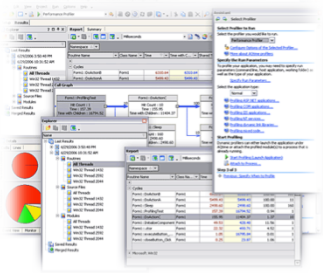
Technical problems (e.g., inaccurate data, cancellation failures)

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)

Livrable d'analyse (A1)

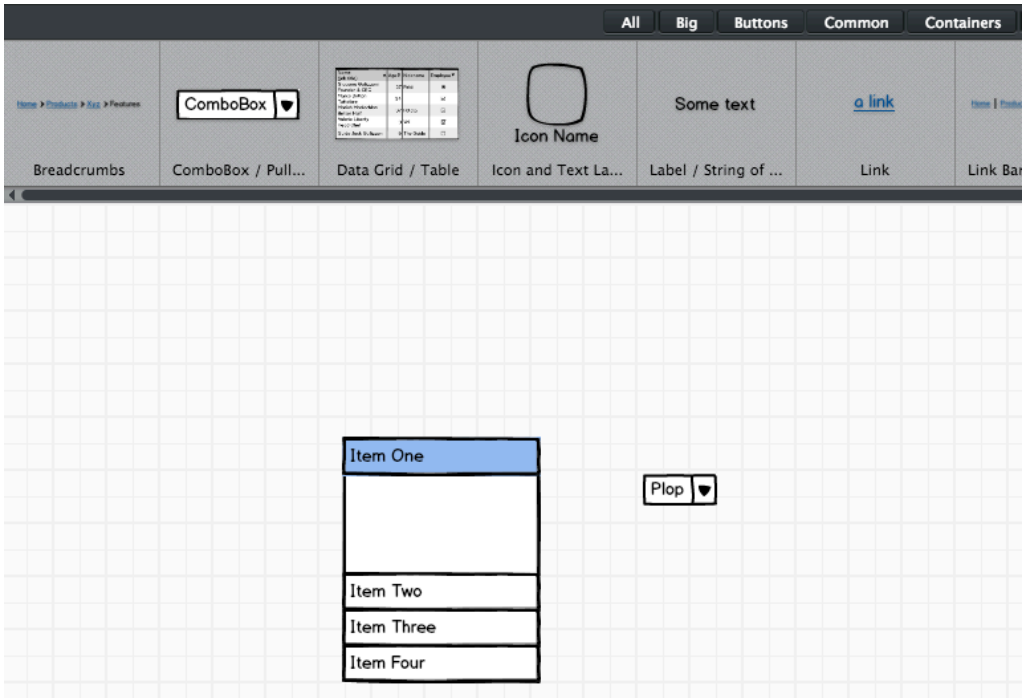
- Analyse du cahier des charges de l'application
- Qualités attendues:
  - non ambigüe
  - complet
- Contenu (typiquement) attendu:
  - **Les cas d'utilisations de l'application**
  - Pour chaque cas d'utilisation au moins un scénario nominal et un scénario exceptionnel
    - Ces scénarios doivent décrire les interactions entre les acteurs et le système (diagramme de séquences UML)
  - Un diagramme de classes d'analyse
  - **Prototype de l'interface graphique**
    - **Avec des modèles d'UI (sketchs)**
    - **Machine à états UML**

Two important tasks of A1

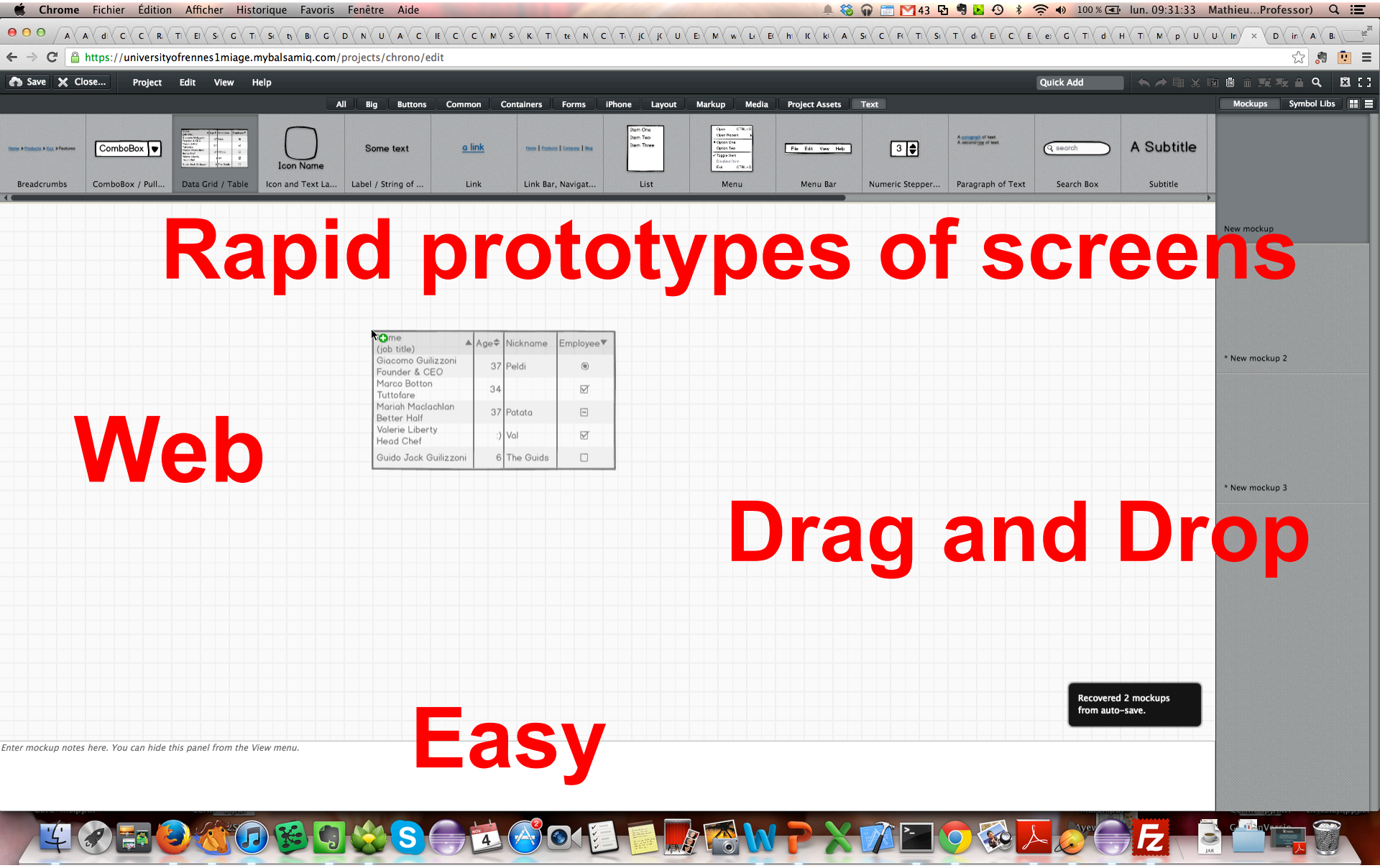
- **Modeling**
- #1 Use cases (UML)
- #2 UI prototyping (Sketching UI)
- How ?
  - Language
  - Tools
- #1 and #2 are related

Prototyping UI

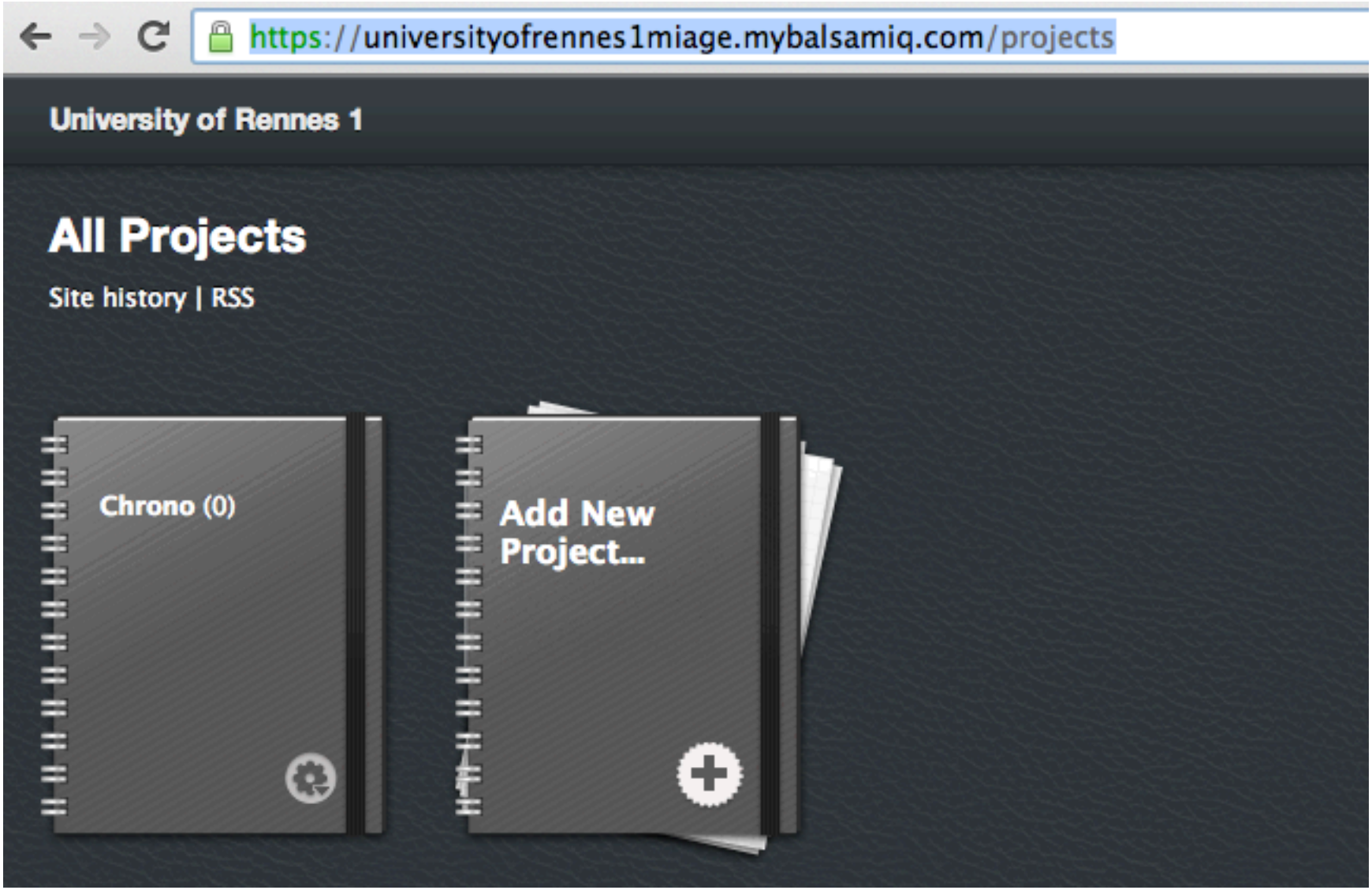
- Intuitive, not cheap
- Iterative
- Exploration
- Focus on purpose
- Modeling
  - Forget technology a few seconds

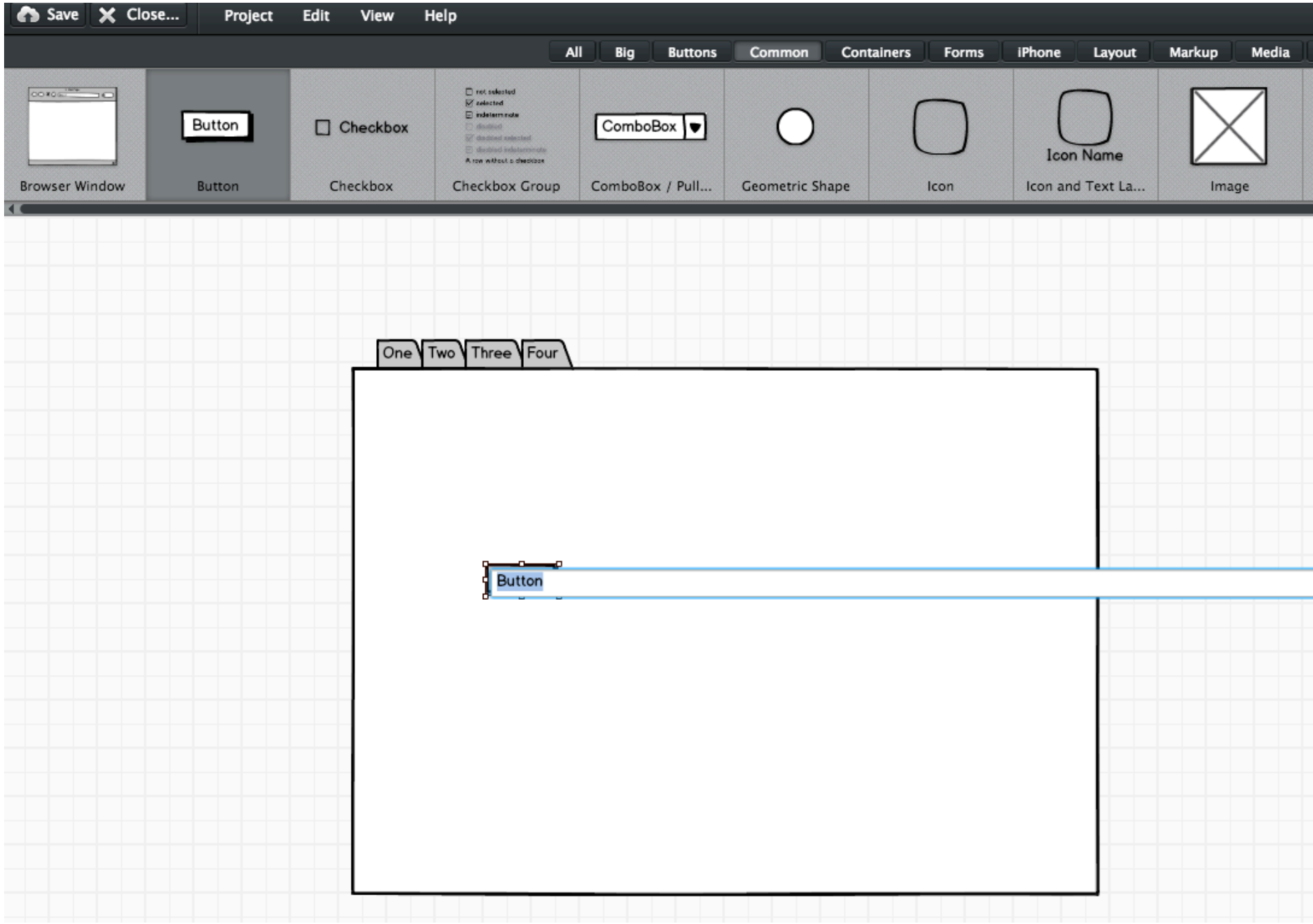


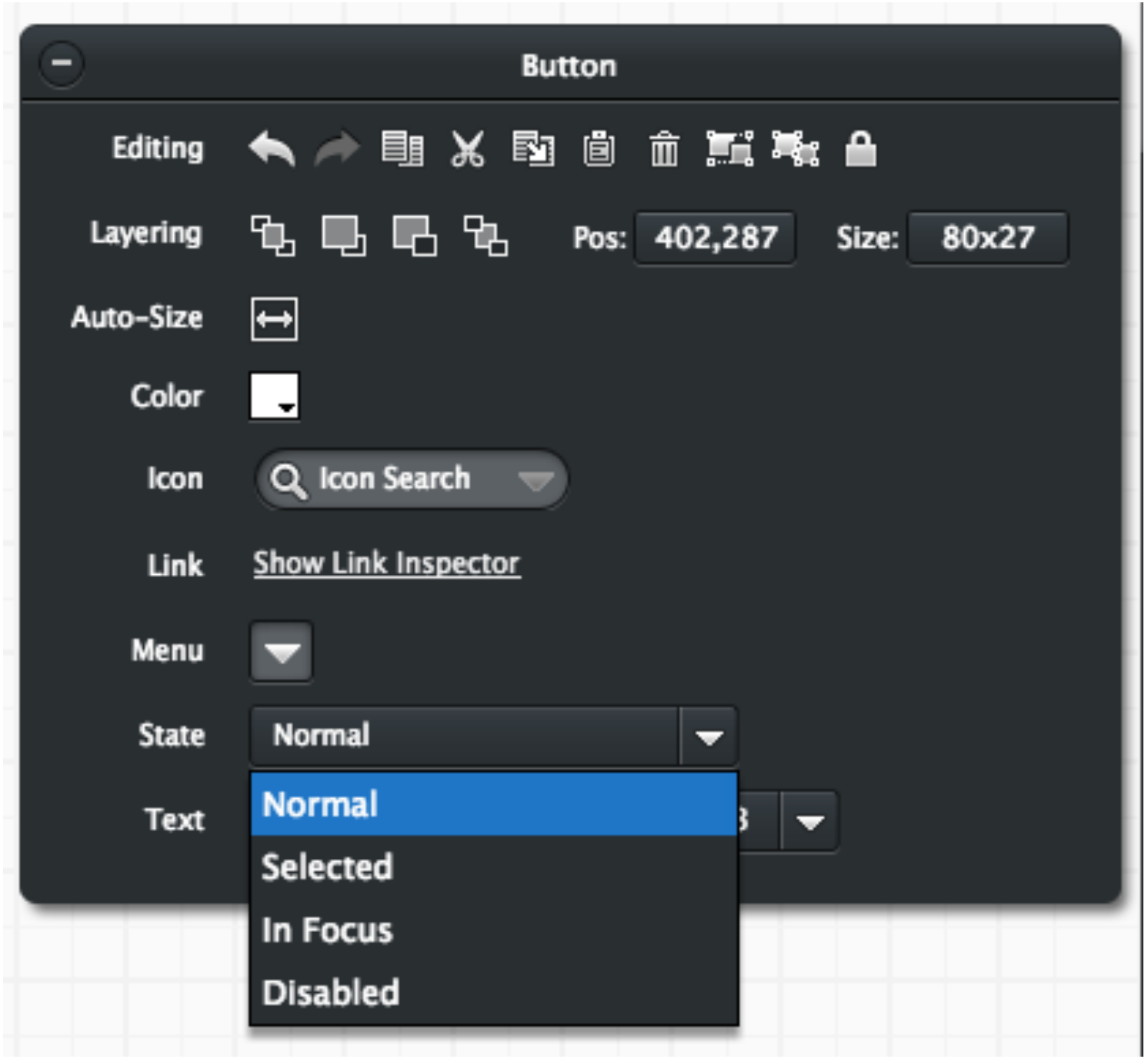
https://universityofrennes1miage.mybalsamiq.com/



https://universityofrennes1miage.mybalsamiq.com/




Save
Close...
Project
Edit
View
Help
All
Big
Buttons
Common
Containers
Forms
iPhone
Layout
Markup
Media
Browser Window
Button
Checkbox
Checkbox Group
ComboBox / Pull...
Geometric Shape
Icon
Icon Name
Icon and Text La...
Image
One
Two
Three
Four
Button

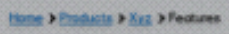

Button
Editing
Layering
Pos: 402,287
Size: 80x27
Auto-Size
Color
Icon
Icon Search
Link
Show Link Inspector
Menu
State
Normal
Text
Normal
Selected
In Focus
Disabled

All | Big | Buttons | Common | Containers | Forms | i

Accordion

Alert Box

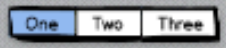

Arrow / Line



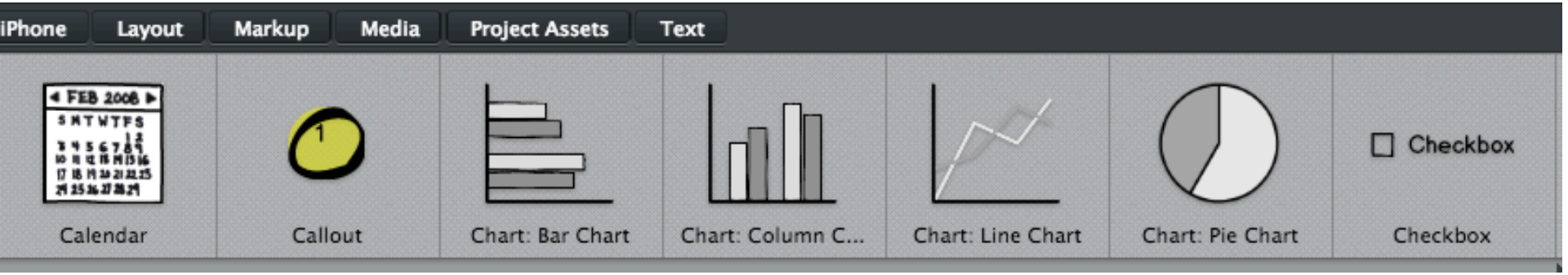
Breadcrumbs

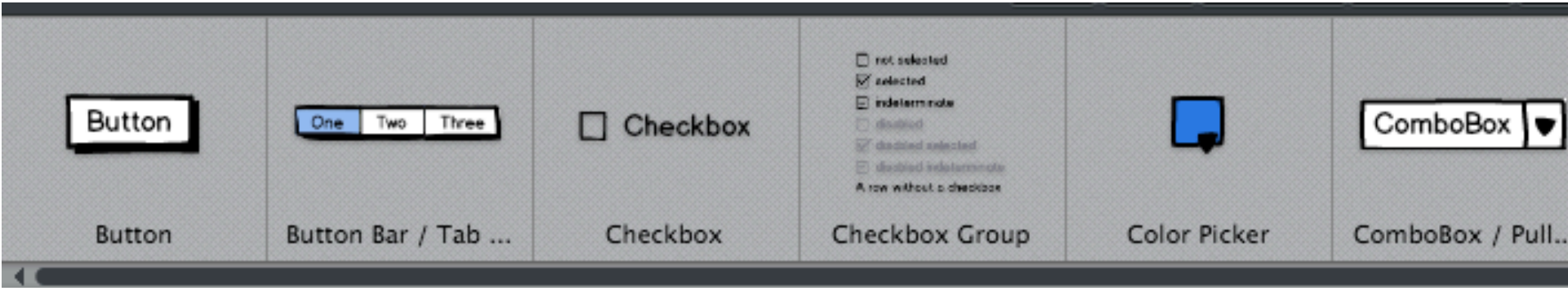

Browser Window

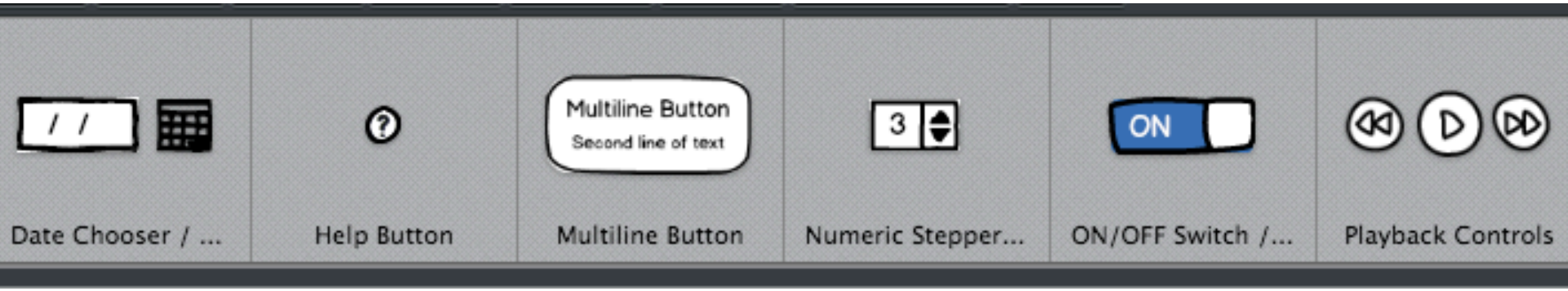
Button

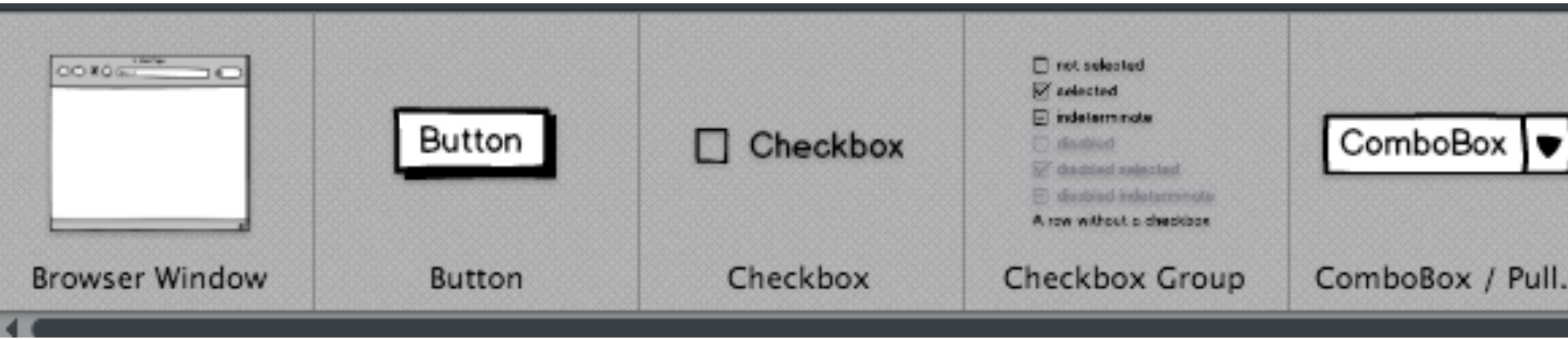


Button Bar / Tab ...


iPhone
Layout
Markup
Media
Project Assets
Text
FEB 2008
Calendar
Callout
Chart: Bar Chart
Chart: Column C...
Chart: Line Chart
Chart: Pie Chart
Checkbox
Checkbox


Button
One
Two
Three
Checkbox
not selected
selected
indeterminate
disabled
disabled selected
disabled indeterminate
A row without a checkbox
ComboBox
Button
Button Bar / Tab ...
Checkbox
Checkbox Group
Color Picker
ComboBox / Pull..

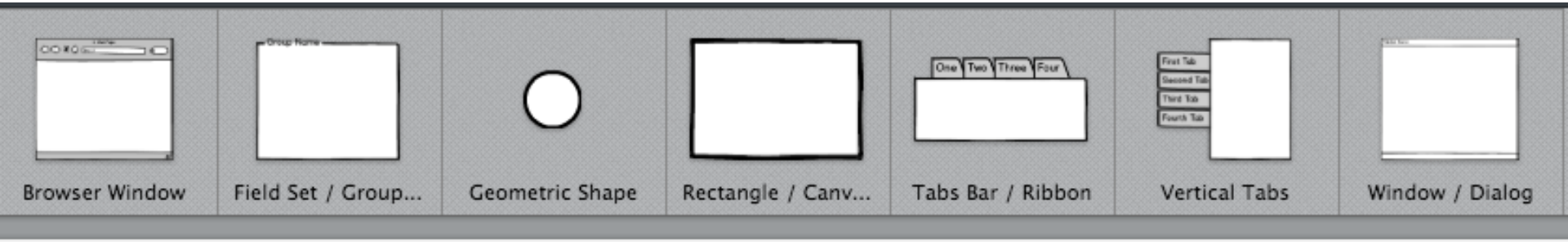

/ /
Multiline Button
Second line of text
3
ON
Date Chooser / ...
Help Button
Multiline Button
Numeric Stepper...
ON/OFF Switch /...
Playback Controls


Button
Checkbox
not selected
selected
indeterminate
disabled
disabled selected
disabled indeterminate
A row without a checkbox
ComboBox
Browser Window
Button
Checkbox
Checkbox Group
ComboBox / Pull.

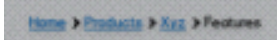

eometric Shape
Icon
Icon Name
Icon and Text La...
Image
Some text
Label / String of ...


Home ▸ Products ▸ Xyz ▸ Features
ComboBox
Icon Name
Some text
a link
Home | Products | Company | Blog
Breadcrumbs
ComboBox / Pull...
Data Grid / Table
Icon and Text La...
Label / String of ...
Link
Link Bar, Navigat...

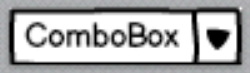

Button
One
Two
Three
Checkbox
not selected
selected
indeterminate
A row without a checkbox
ComboBox
/ /
Button
Button Bar / Tab ...
Checkbox
Checkbox Group
Color Picker
ComboBox / Pull...
Date Chooser / ...
Field Set / G


Group Name
One
Two
Three
Four
First Tab
Second Tab
Third Tab
Fourth Tab
Browser Window
Field Set / Group...
Geometric Shape
Rectangle / Canv...
Tabs Bar / Ribbon
Vertical Tabs
Window / Dialog

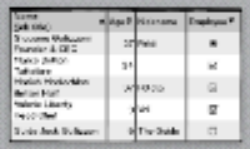
All | Big | Buttons | Common | Containers



Breadcrumbs



ComboBox / Pull...



Data Grid / Table

Icon and Text La...

Some text

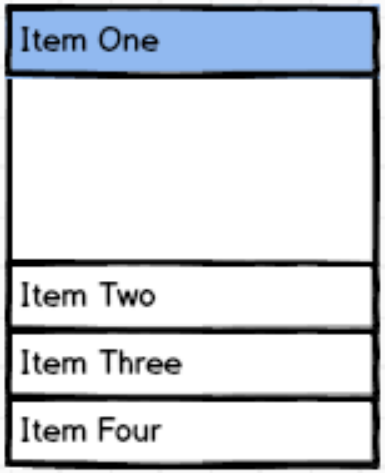
Label / String of ...

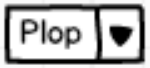
a link

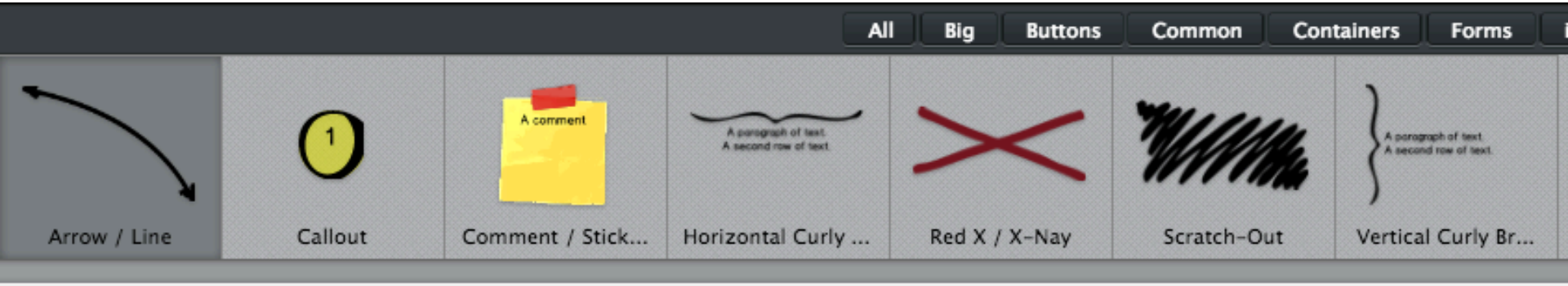
Link

Home | Produ

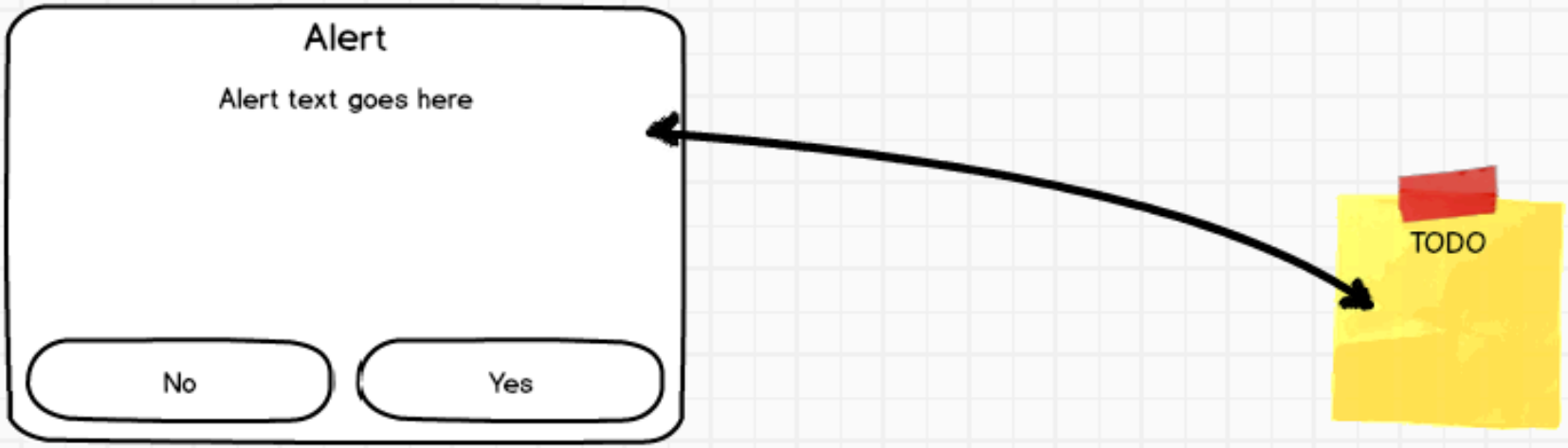
Link Ba

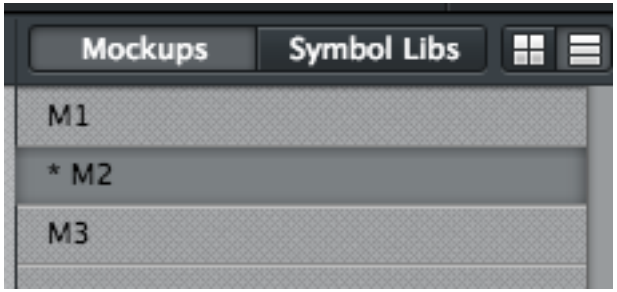




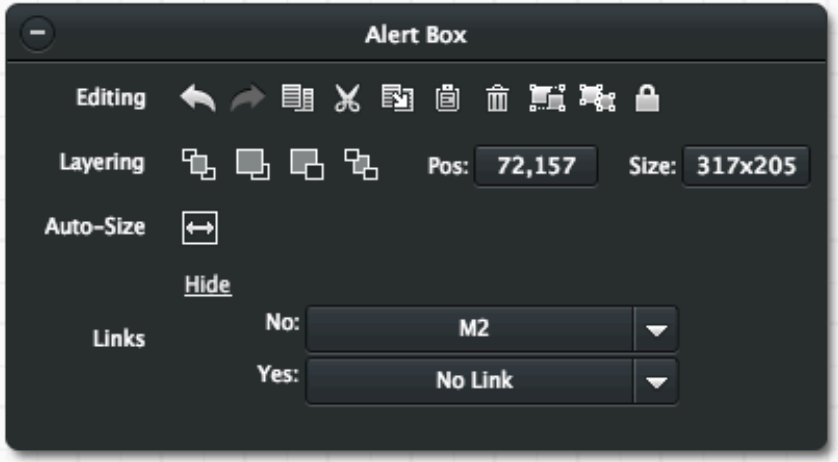


Collaborative



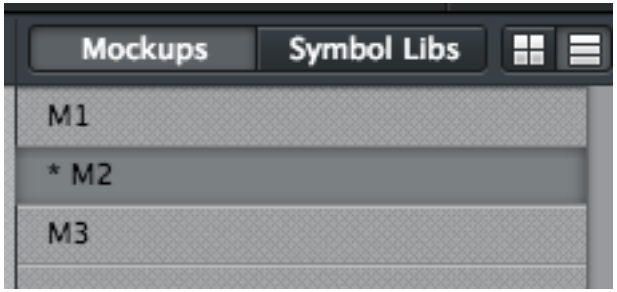


Interaction between Mockups

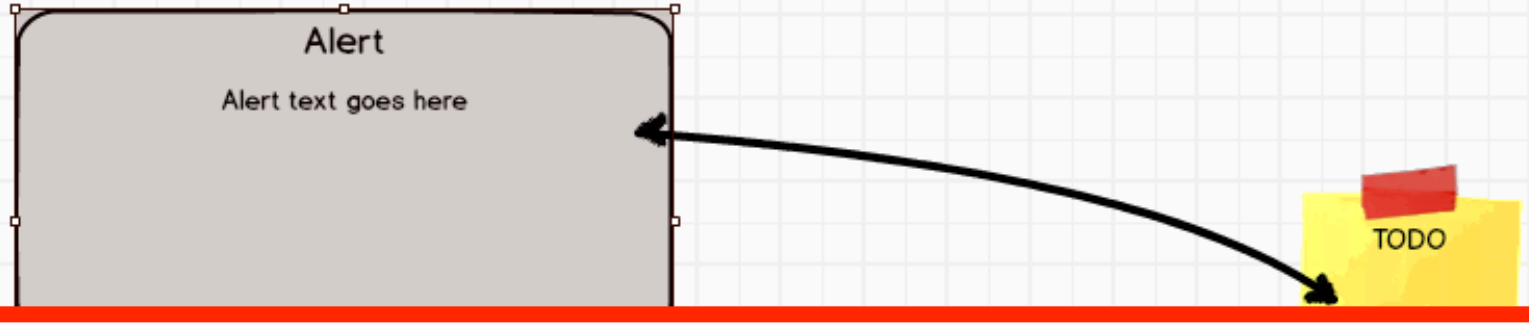


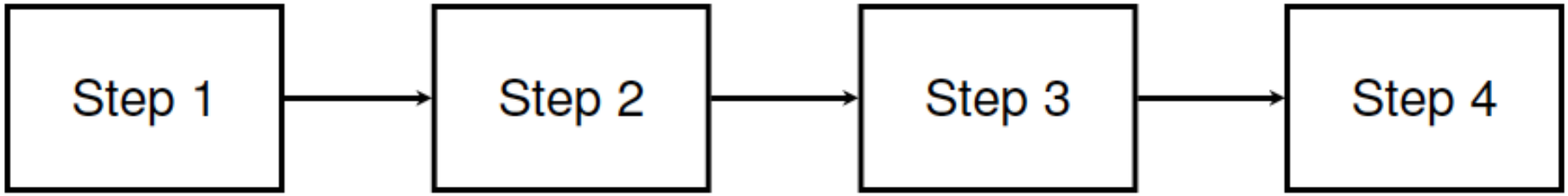
Modélisation UML

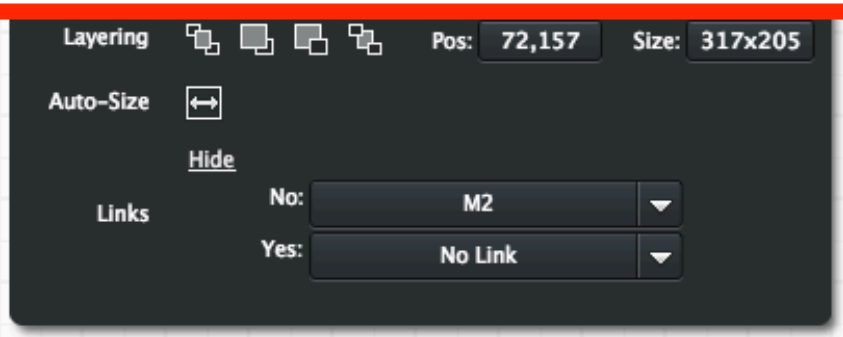
- Modélisation selon 4 points de vue principaux :
  - Aspects statiques du système *(le QUI?)*
    - » Description des objets et de leurs relations
      - Modularité, contrats, relations, généricité, héritage
    - » Structuration en paquetages
  - Vision utilisateur du système *(le QUOI?)*
    - » 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 OU?)*
    - » Diagramme de composants et de déploiement



Interaction between Mockups





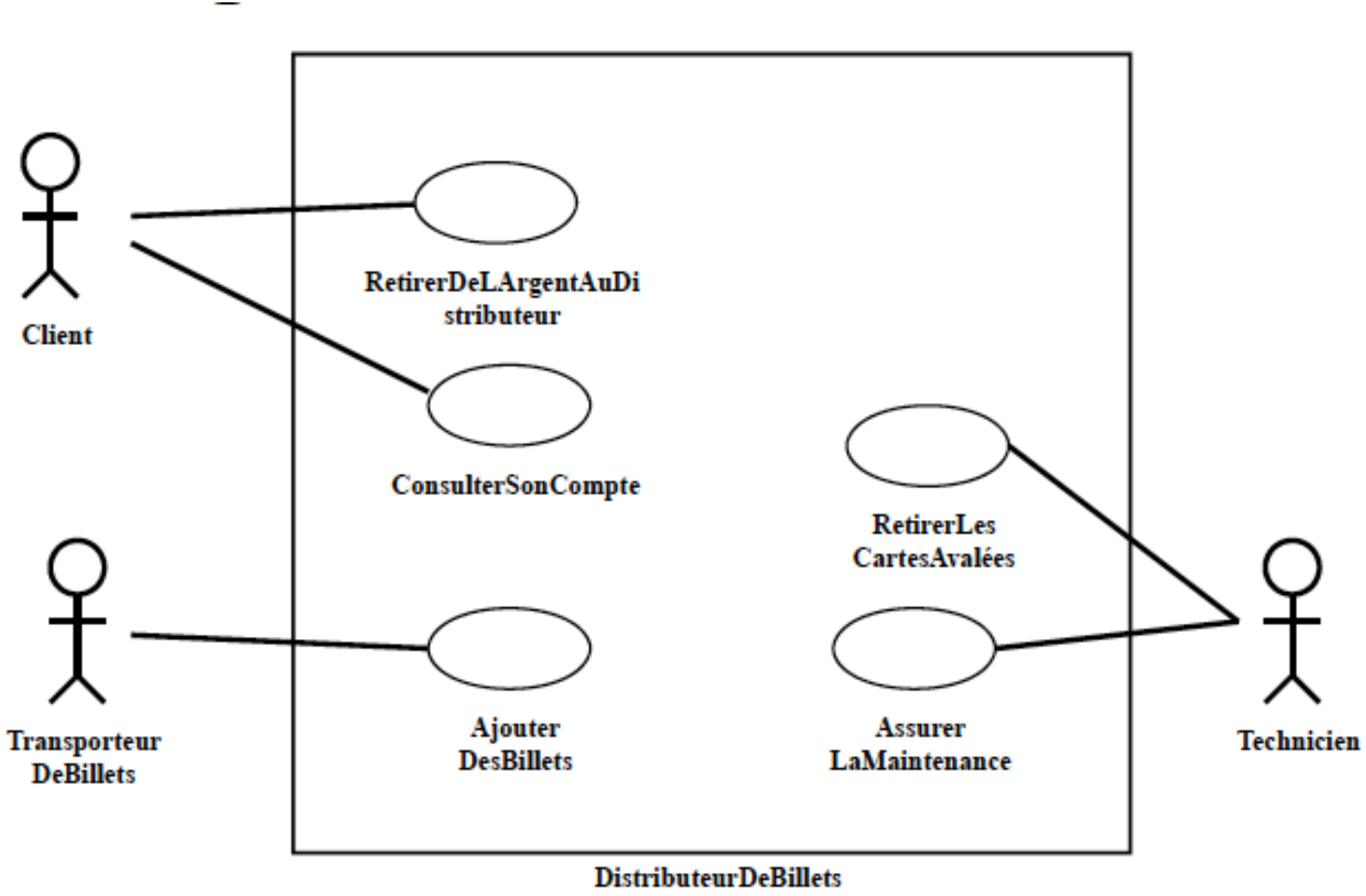


Modélisation UML

- 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
Banque Centrale
Transporteur DeBillets
Technicien
DistributeurDeBillet


Client
RetirerDeLArgentAuDistributeur
ConsulterSonCompte
Transporteur DeBillets
Ajouter DesBillets
RetirerLes CartesAvalées
Assurer LaMaintenance
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"**

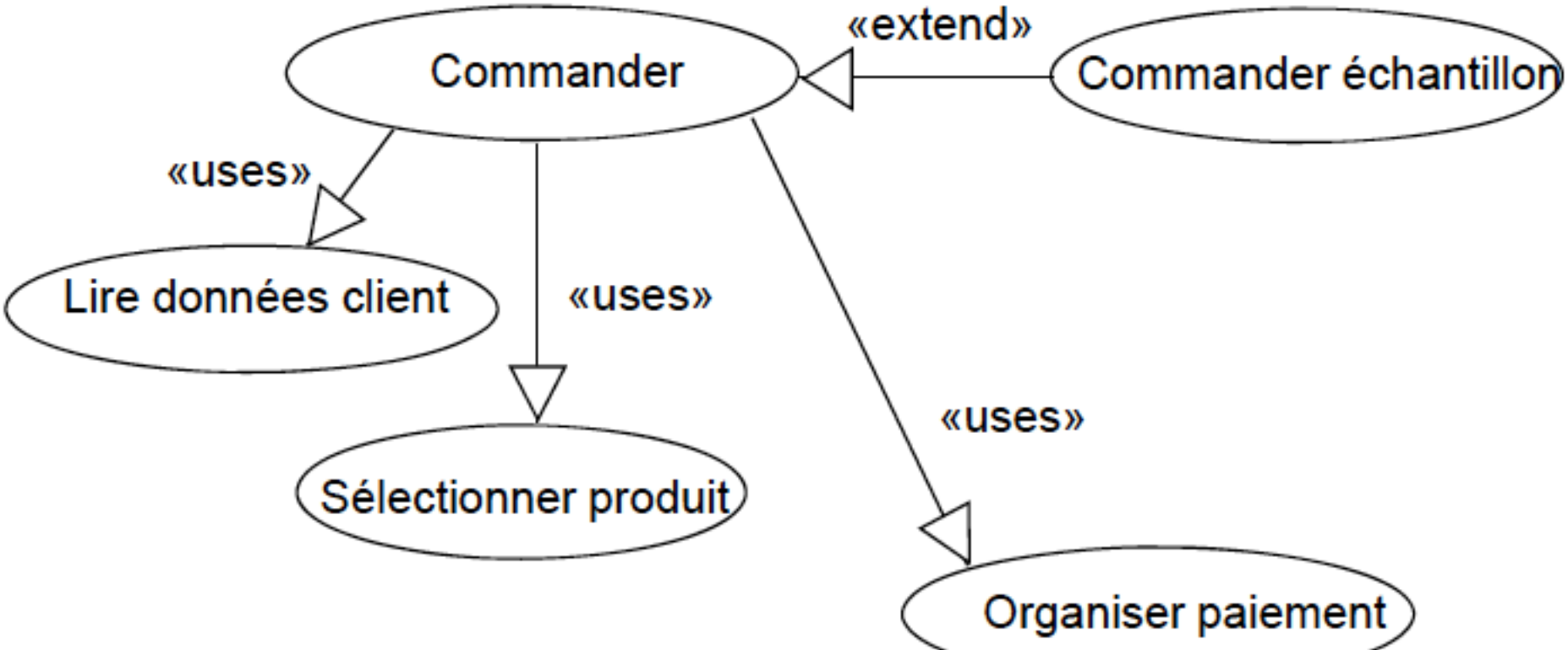
Retirer DeLArgent AuDistributeur

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

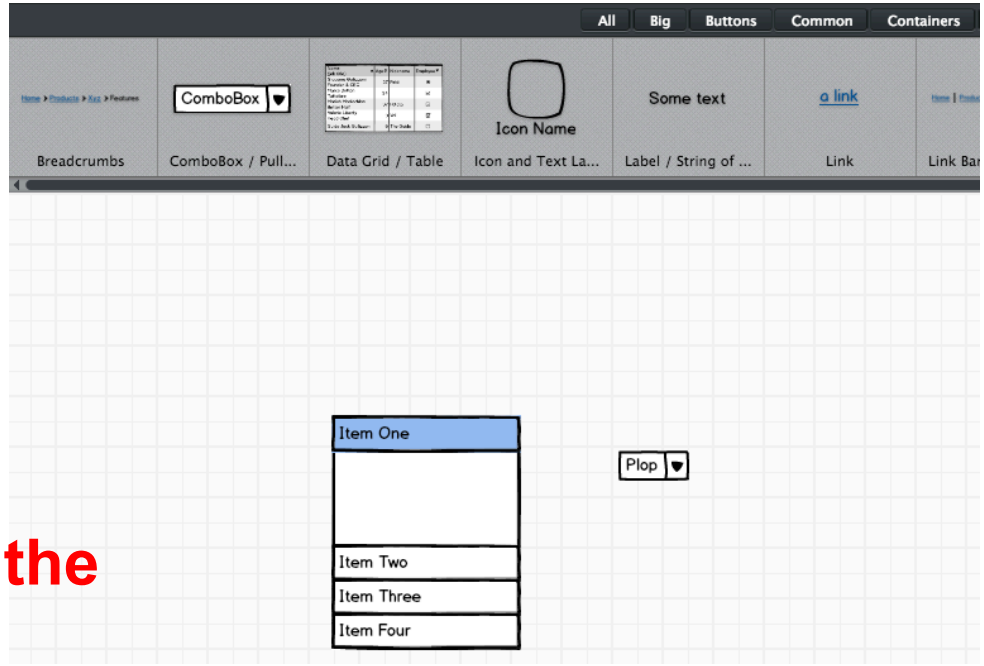
Relations sur les *use-cases* : notation



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

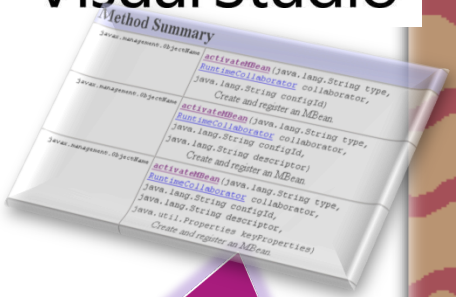
UML diagrams

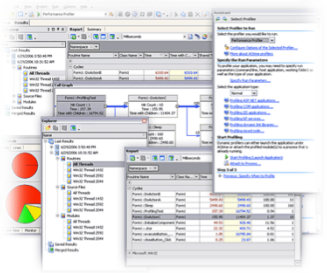


Complementary!

Focus on some aspects of the system/requirements

Software Engineering





Documentation and Source Code (for C1)

Code (C1)

- Code source de l'application
  - Cela inclue évidemment les tests
  - La documentation
  - Un packaging du tout
- Nécessairement en Java
- Liberté totale pour les "frameworks"
  - À condition qu'ils soient open source
    - À discuter avec le client

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 `Action` interface provides a useful extension to the `ActionListener` interface in cases where the same functionality may be accessed by several contro
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 `KeySelectionManager`.
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.
MenuElement	Any component that can be placed into a menu should implement this interface.
MutableComboBoxModel	A mutable version of `ComboBoxModel`.
Renderer	Defines the requirements for an object responsible for "rendering" (displaying) a value.
RootPaneContainer	This interface is implemented by components that have a single JRootPane child: JDialog, JFrame, JWindow, JApplet, JInternalFrame.
Scrollable	An interface that provides information to a scrolling container like JScrollPane.
ScrollPaneConstants	Constants used with the JScrollPane 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 `getXXX(key)` 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 `getXXX(key)` 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 `J`
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 no
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

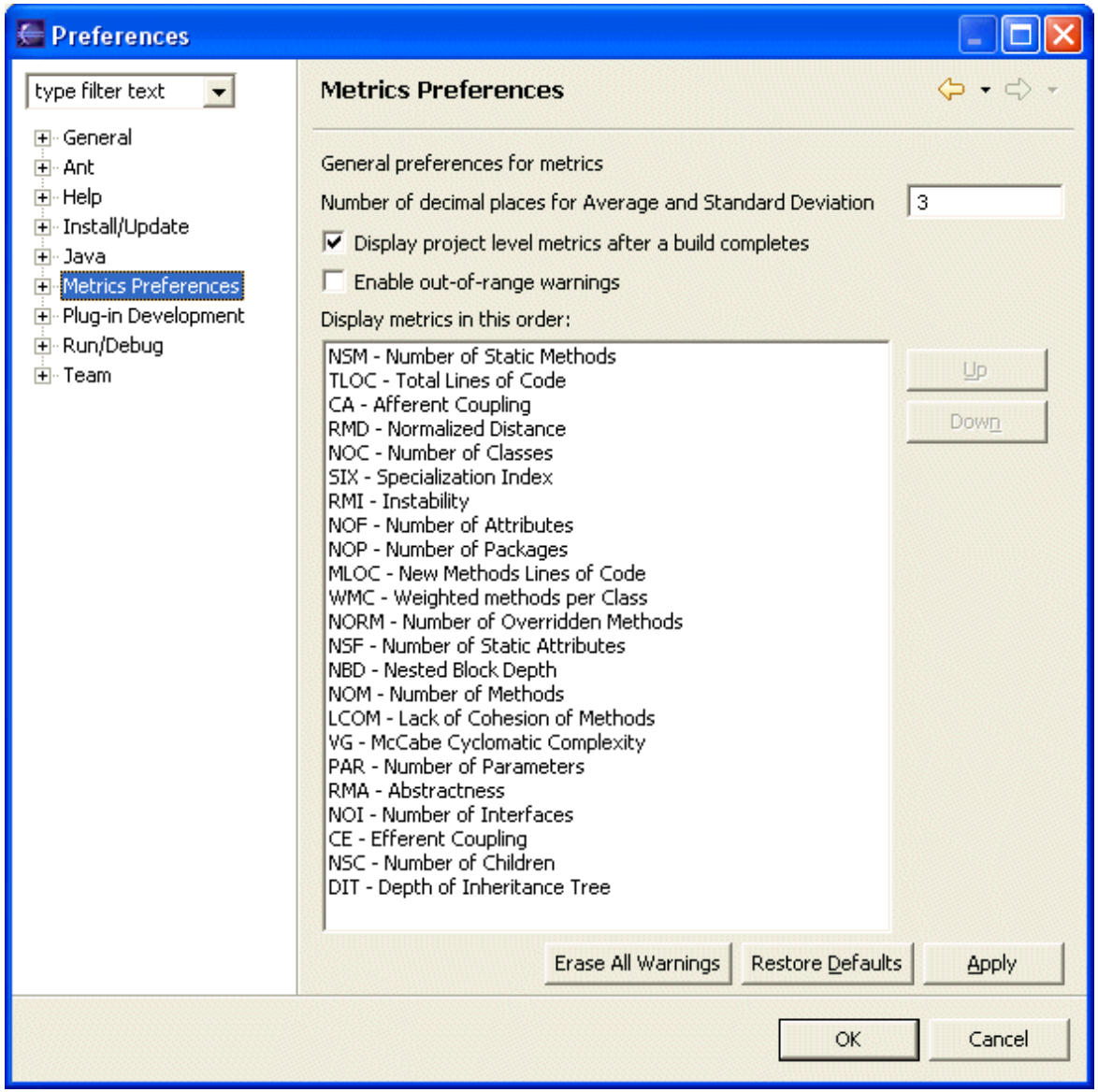
A side effect of the underscore naming convention is that it nicely r

```
  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”

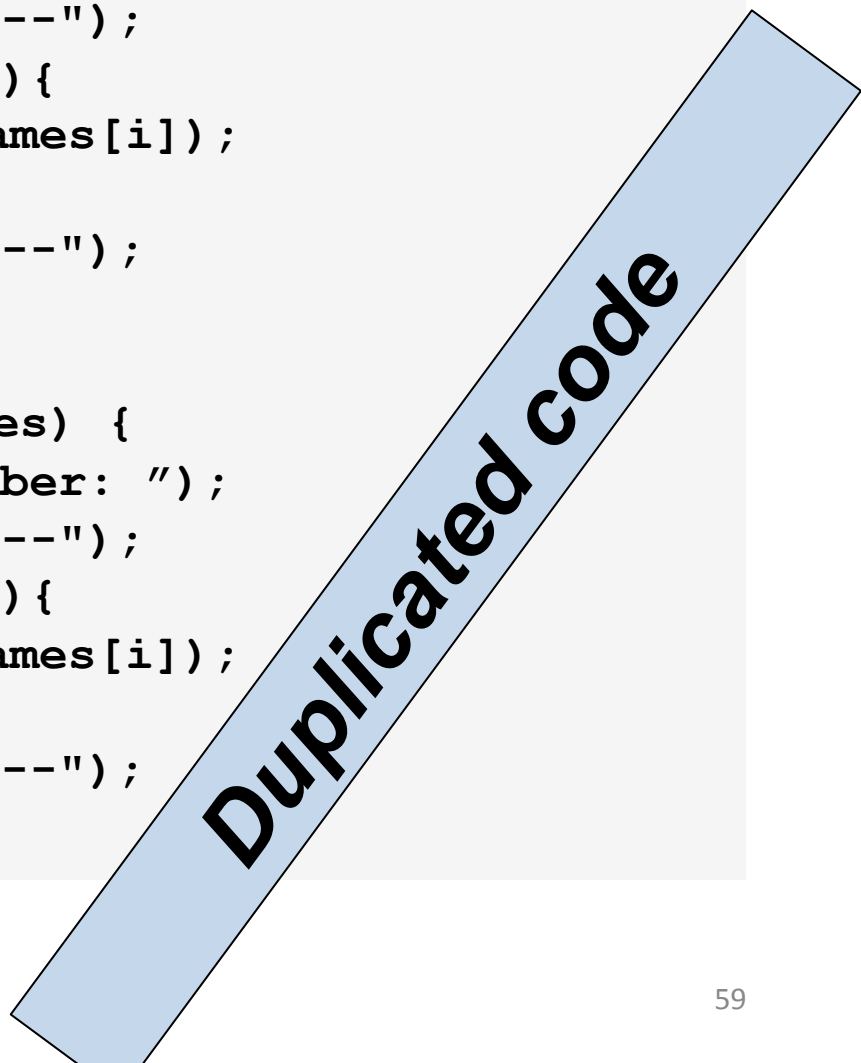
Non exhaustive (code smell)

(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("--------------");
}
```



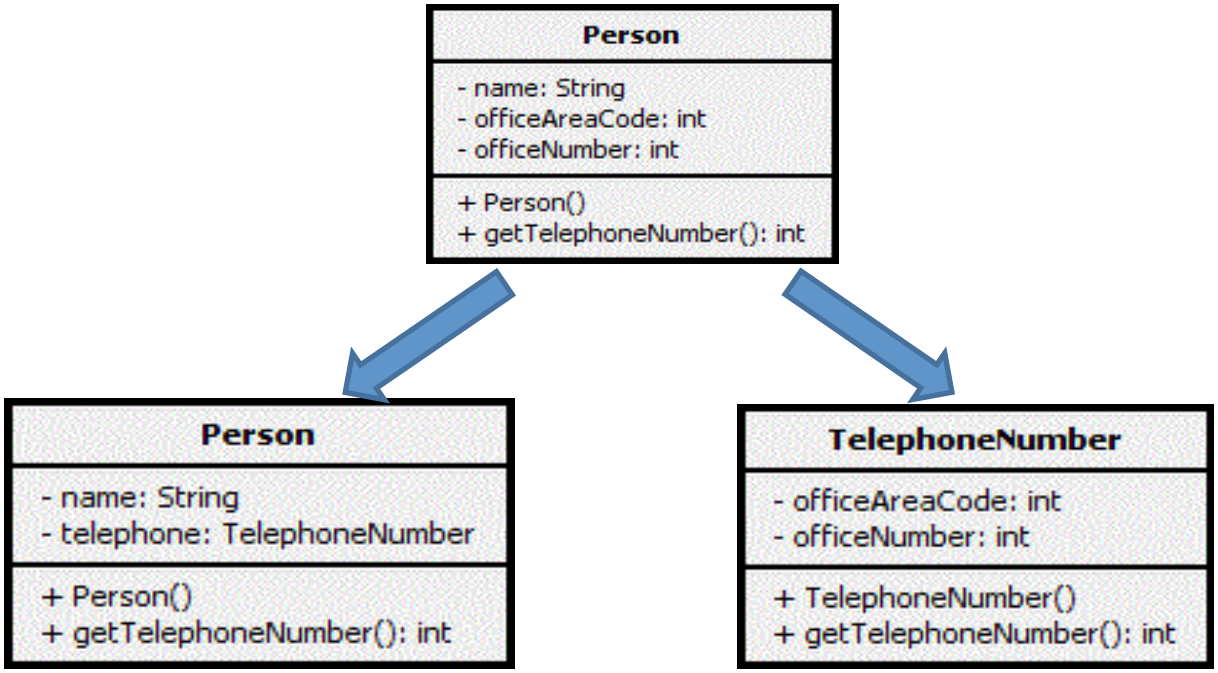
Code Smell examples (2)

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

Long list of parameters

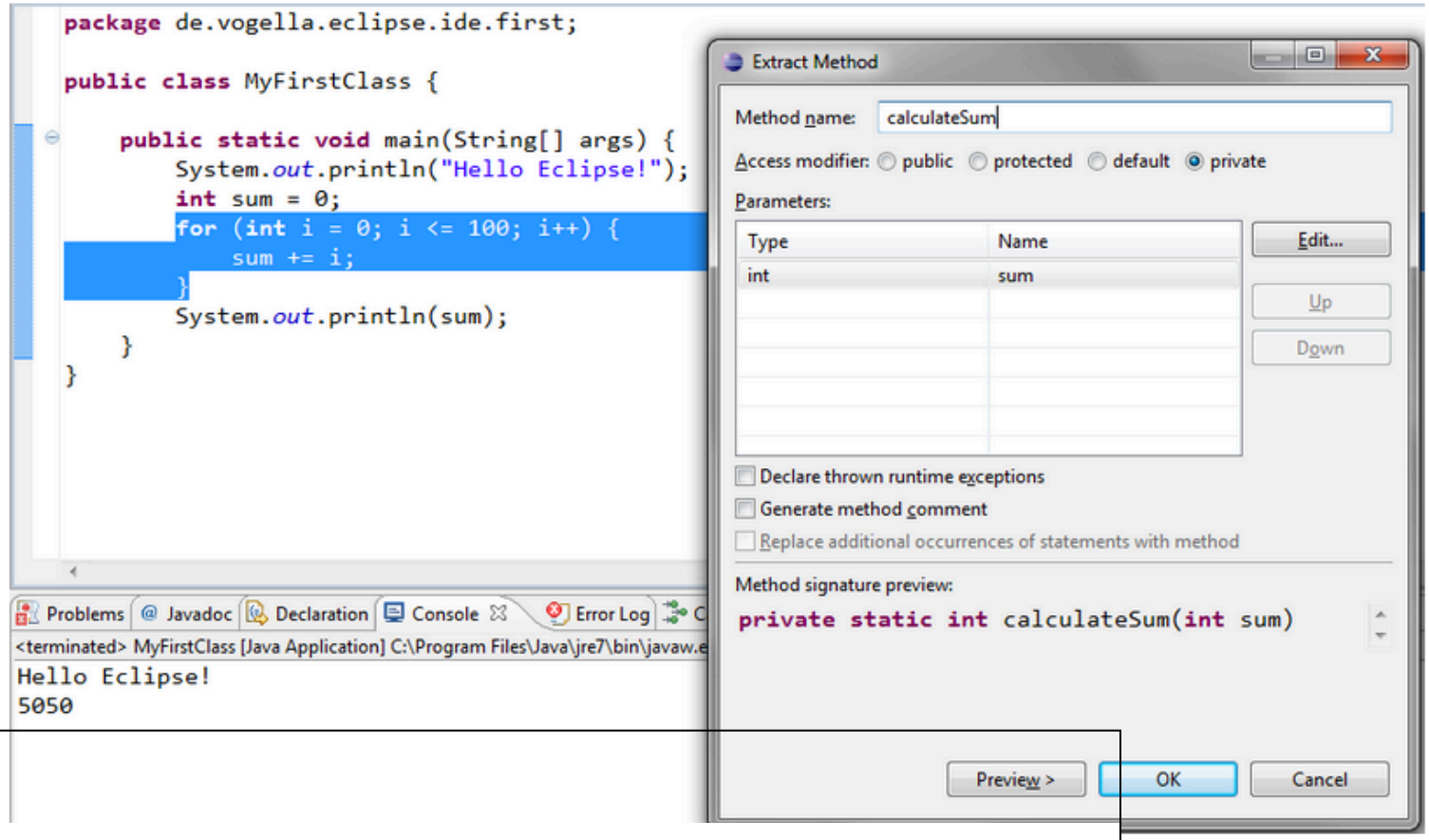
Improving design

- 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?

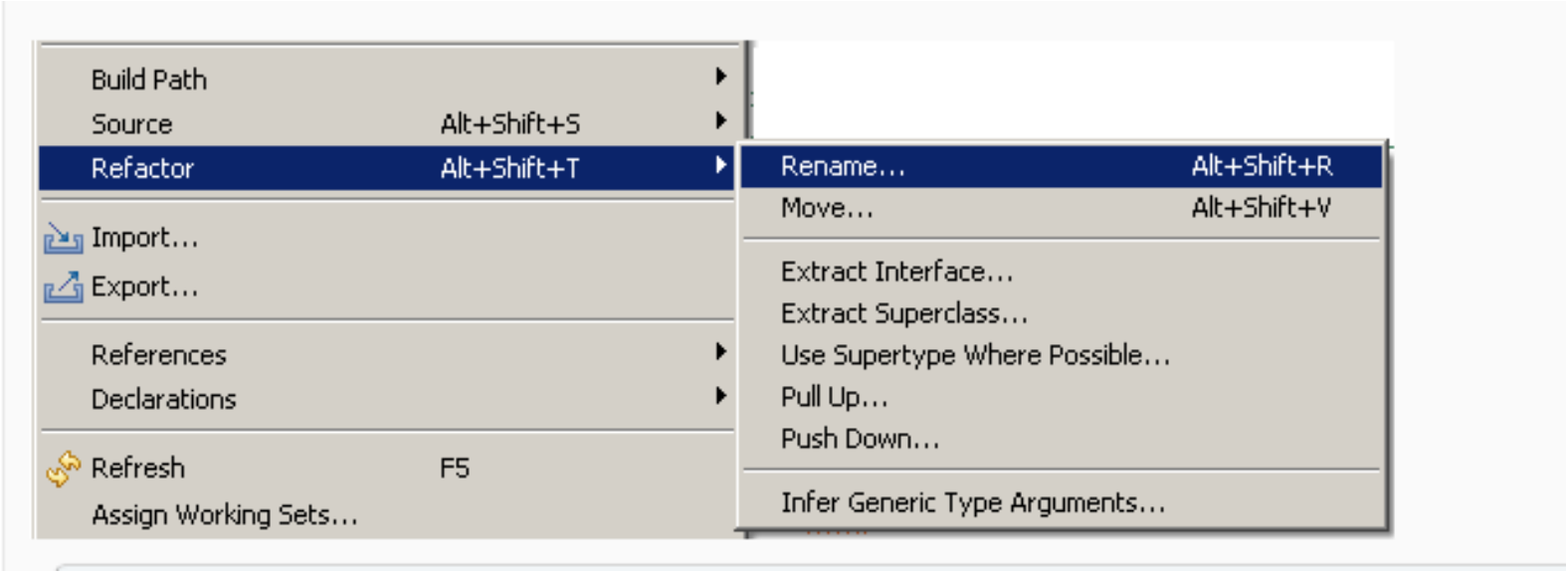
- Manual Refactoring
  - Code Smells
- Automated/Assisted Refactoring
  - Refactoring by hand is time consuming and prone to error
  - Tools (IDE)
- In either case, **test your changes**




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

public class MyFirstClass {

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

You

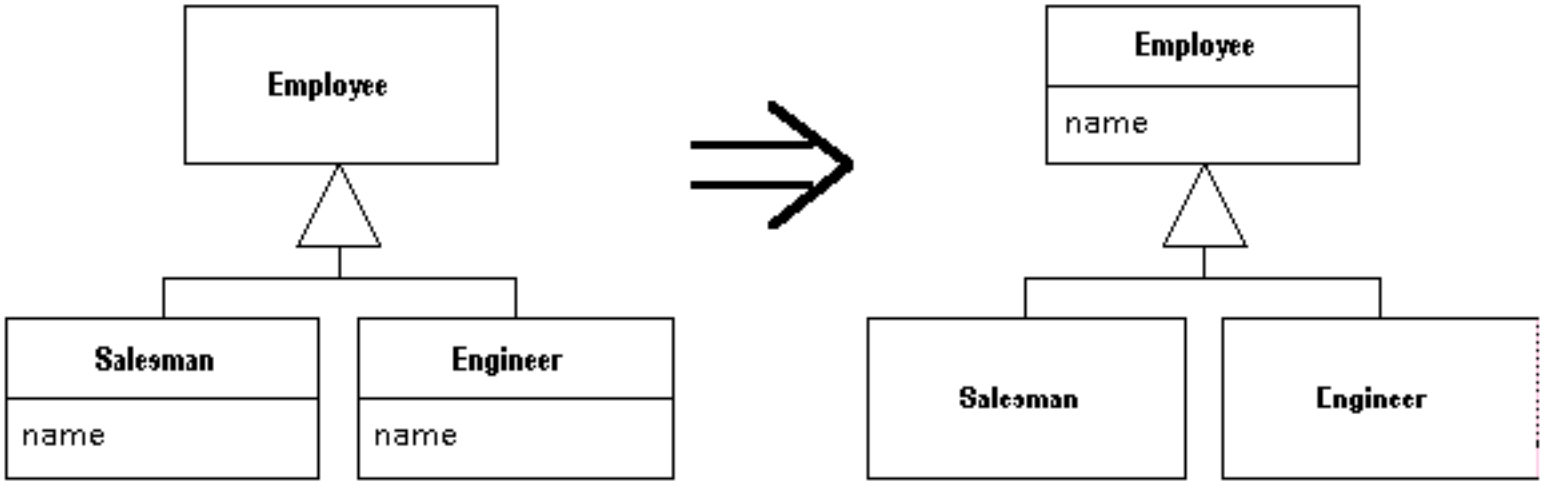
Create

```
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;
        }
```

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


This part is largely inspired by Thomas Zimmermann slides

Your **hope** as a programmer

« A program does exactly what you expected to do »

Djikstra


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


I don’t make mistakes
Microsoft Visual Studio
Package Load Failure
Package 'ArtOfTest.WebAiiVSIP.WebAiiVSIP, ArtOfTest.WebAiiVSIP, Version=1.0.0.0, Culture=neutral, PublicKeyToken=14176981d1873d86' has failed to load properly (GUID = {D5EDEC6D-8025-40CF-987C-B8EABA2C7451}). Please contact package vendor for assistance. Application restart is recommended, due to possible environment corruption. Would you like to disable loading this package in the future? You may use 'devenv /resetskippkgs' to re-enable package loading.
Yes
No

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 = maintainance »

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

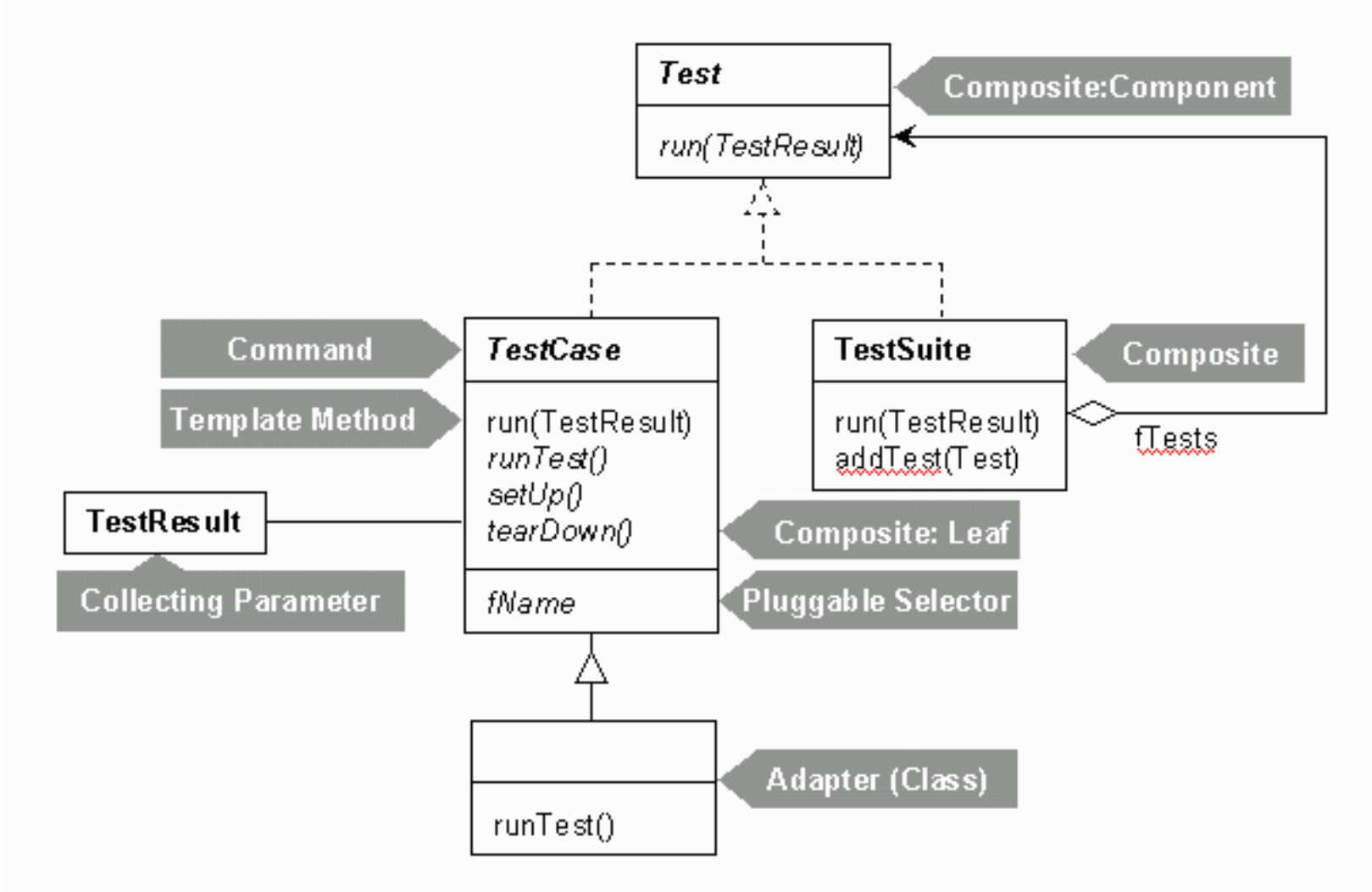
Unit testing on individual units of source code (=smallest testable part).

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

1. Add product

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(St
        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 $|v1 - v2| > \epsilon$

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

`assertNonNull(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

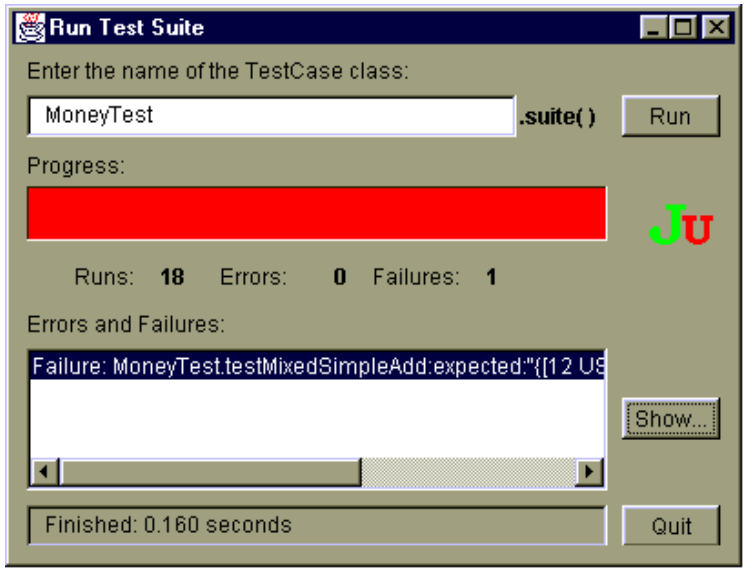
```
// Tests removing a product from the cart.
public void testProductRemove() throws NotFoundException {
    Product book = new Product("Harry Potter", 23.95);
    _bookCart.removeItem(book);

    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);
}
```

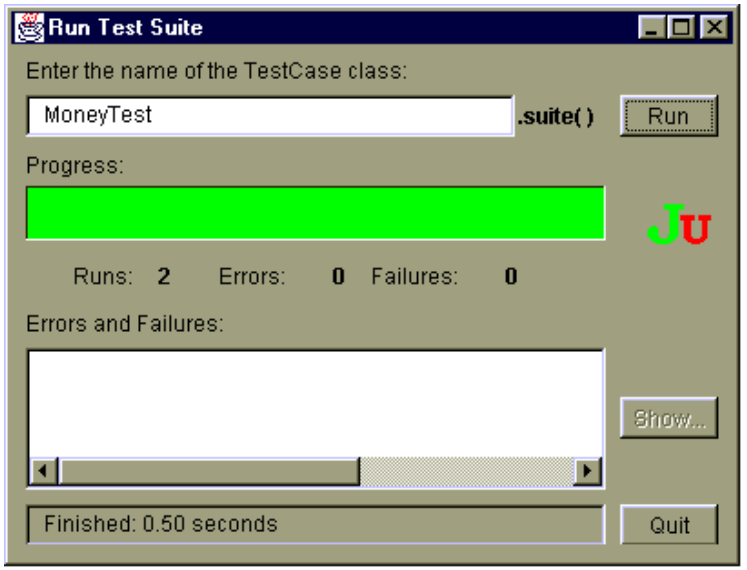
```
public static Test suite() {

    // 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.