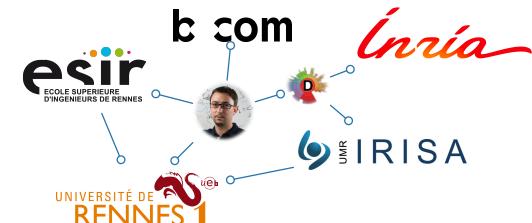


TOOLS FOR (JAVA) DEVELOPMENT INDUSTRIALIZATION

UNIV. RENNES 1, ESIR, 2020-2021

BENOIT COMBEMALE
PROFESSOR, UNIV. RENNES 1 & INRIA, FRANCE

[HTTP://COMBEMALE.FR](http://COMBEMALE.FR)
BENOIT.COMBEMALE@IRISA.FR
[@BCOMBEMALE](https://www.twitter.com/BCOMBEMALE)



WHO WE ARE?

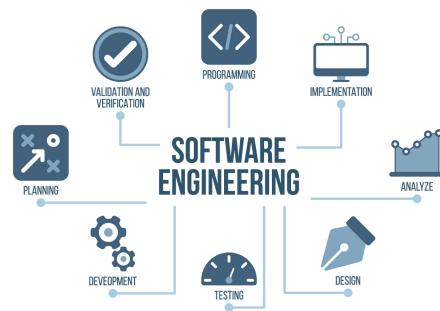
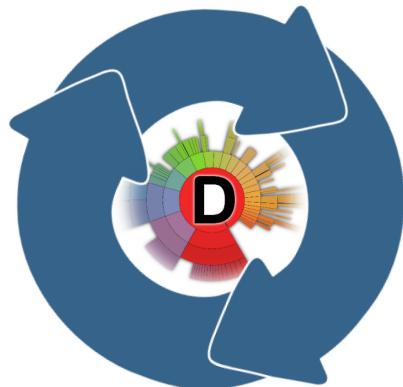


The DiverSE team

- Inria/IRISA project-team in **Software Engineering**
- Strong background in Model-Driven software/systems Eng.
- Software languages, architecture, simulation, variability, testing, resilience eng.
- Applied to smart, heterogeneous, and distributed CPS (e.g., IoT, Industry 4.0)
- 9 Prof. and Inria/CNRS researchers, 1 Inria RSE, ~20 PhD, 3 Post-doc, 3 SE
- Deductive and empirical scientific approaches
- Open source software development
- Strong contractual activity (esp. EU and industry projects)

The DiverSE team

A Software Engineering Group

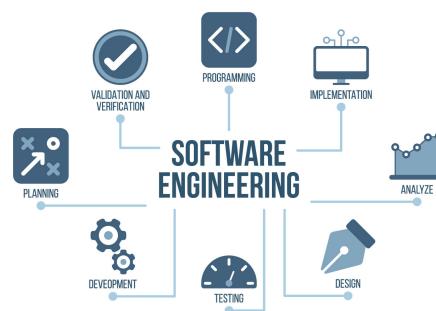


The DiverSE team

A Software Engineering Group



- Diversity of...
- stakeholders
- concerns
- configurations
- platforms
- environments
- requirements...



- Multi-engineering approach
- Domain-specific modeling
- High variability and customization
- Platform heterogeneity
- Openness and dynamicity

The DiverSE team

Software...

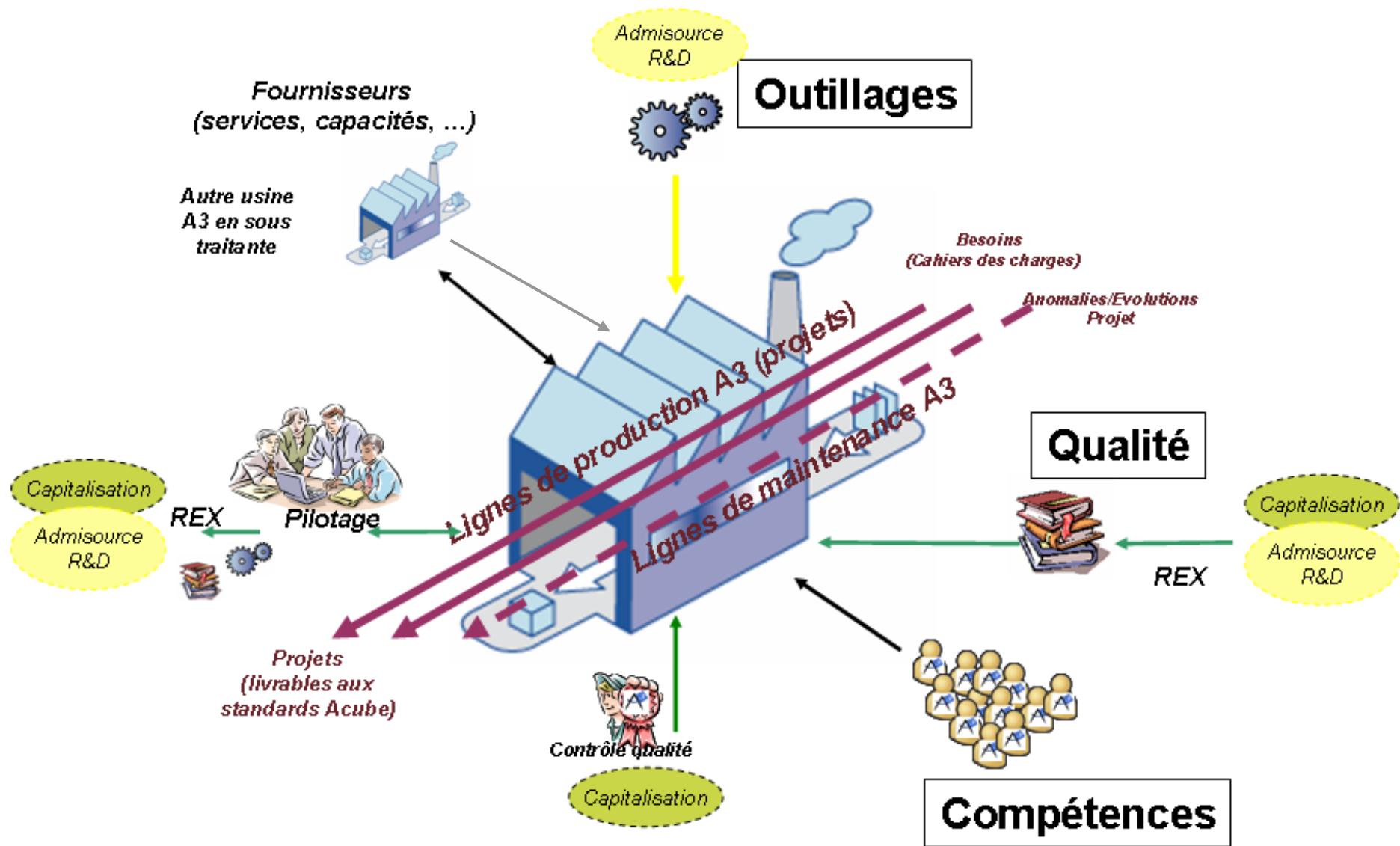
*modeling, architecture, testing, variability, reuse,
continuous deployment, adaptation and languages*



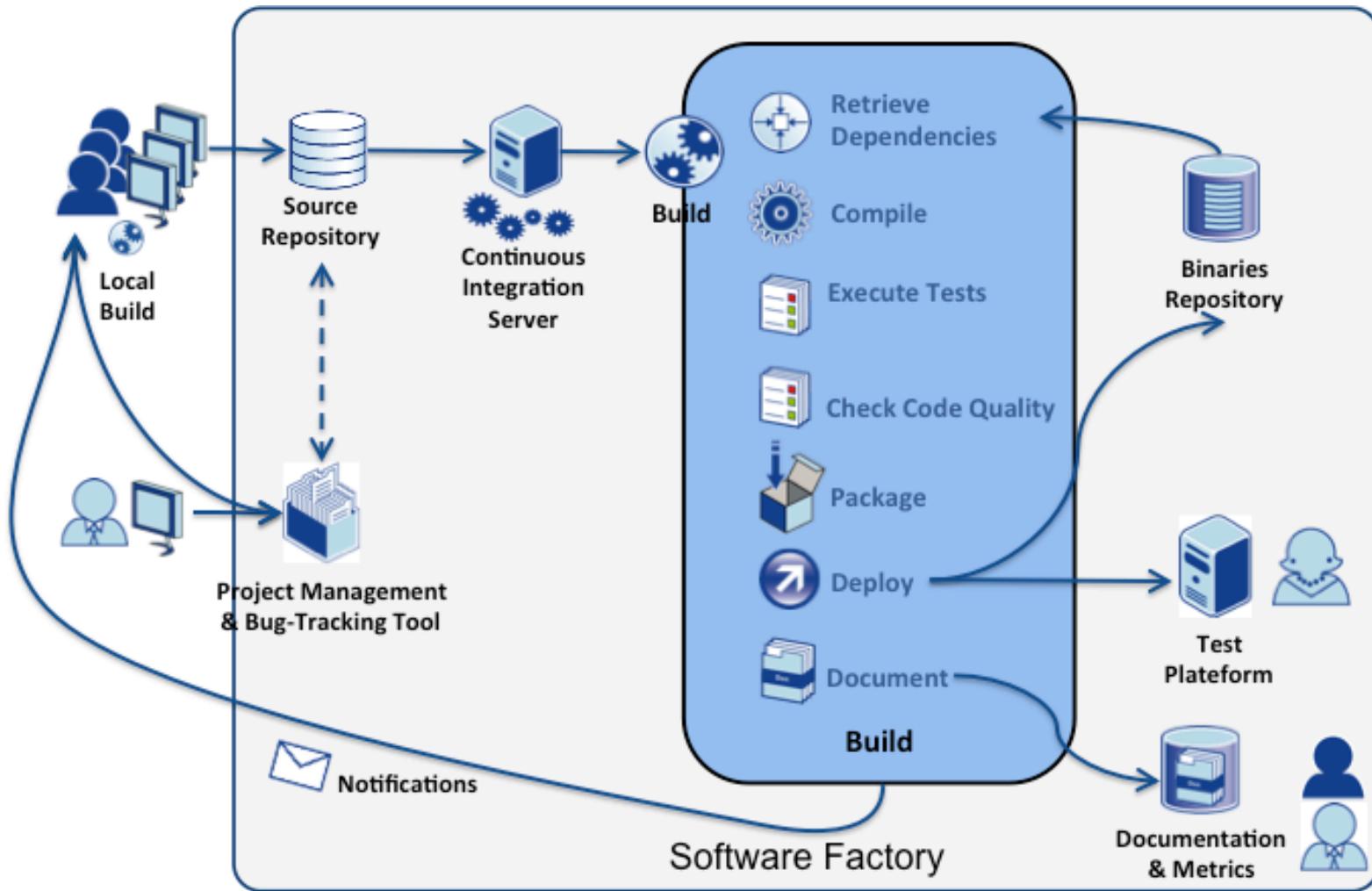
<https://www.diverse-team.fr>

Exemple d'organisation de
projet dans la pratique...

Usine Logicielle



Usine Logicielle



Approche industrielle – un cas classique

Plate-forme collaborative / forge

Applications Internet / Intranet / Extranet

Framework A³

Client Riche : FRED-A³, yahoo-ui, GWT AJAX LISE (Java / J2EE)

Socle technique

Linux / Apache / Tomcat / JBoss / Base de données (MySQL, Oracle, SQL Server, ...)

Plate-forme d'industrialisation

Eclipse
Plug-in

Accessibilité
Firefox
Plug-in

IDE & Design

Junit

Cobertura

JMeter

Selenium

Tests

Continuum

Maven

*Intégration
continue
automate*

Alfresco
Liferay
CAS

Mantis

XRadar

Checkstyle QAlab

CPD FindBugs

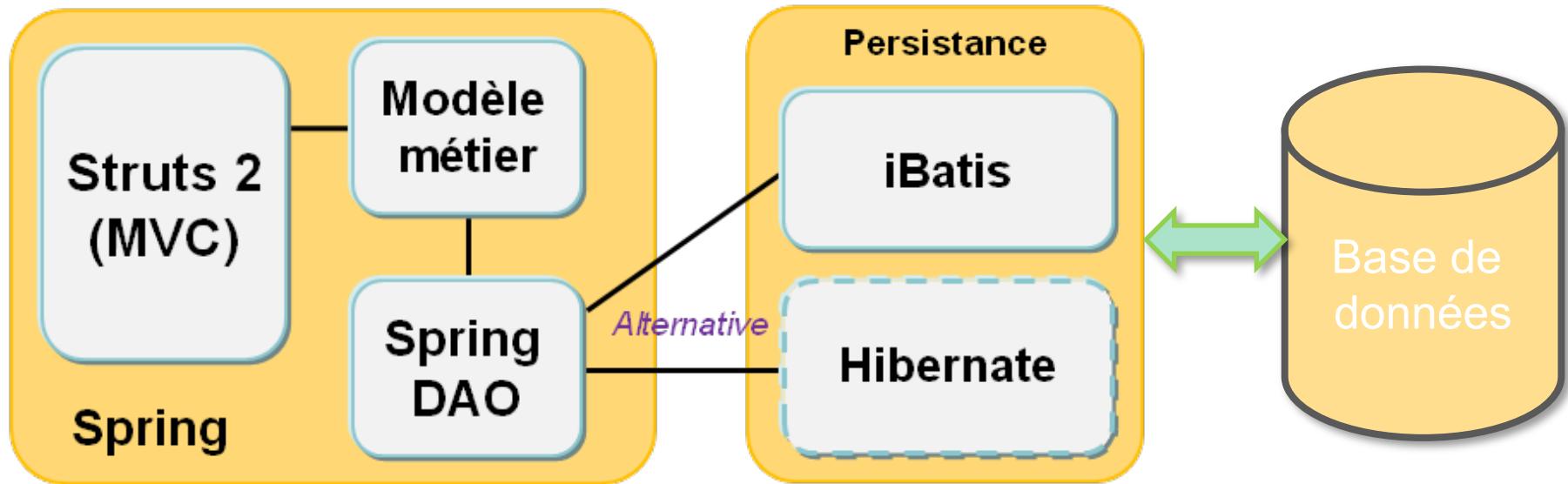
*« Application Mining »
Qualité*

Eclipse, Plug-in, Struts, Log 4J, Hibernate, ...

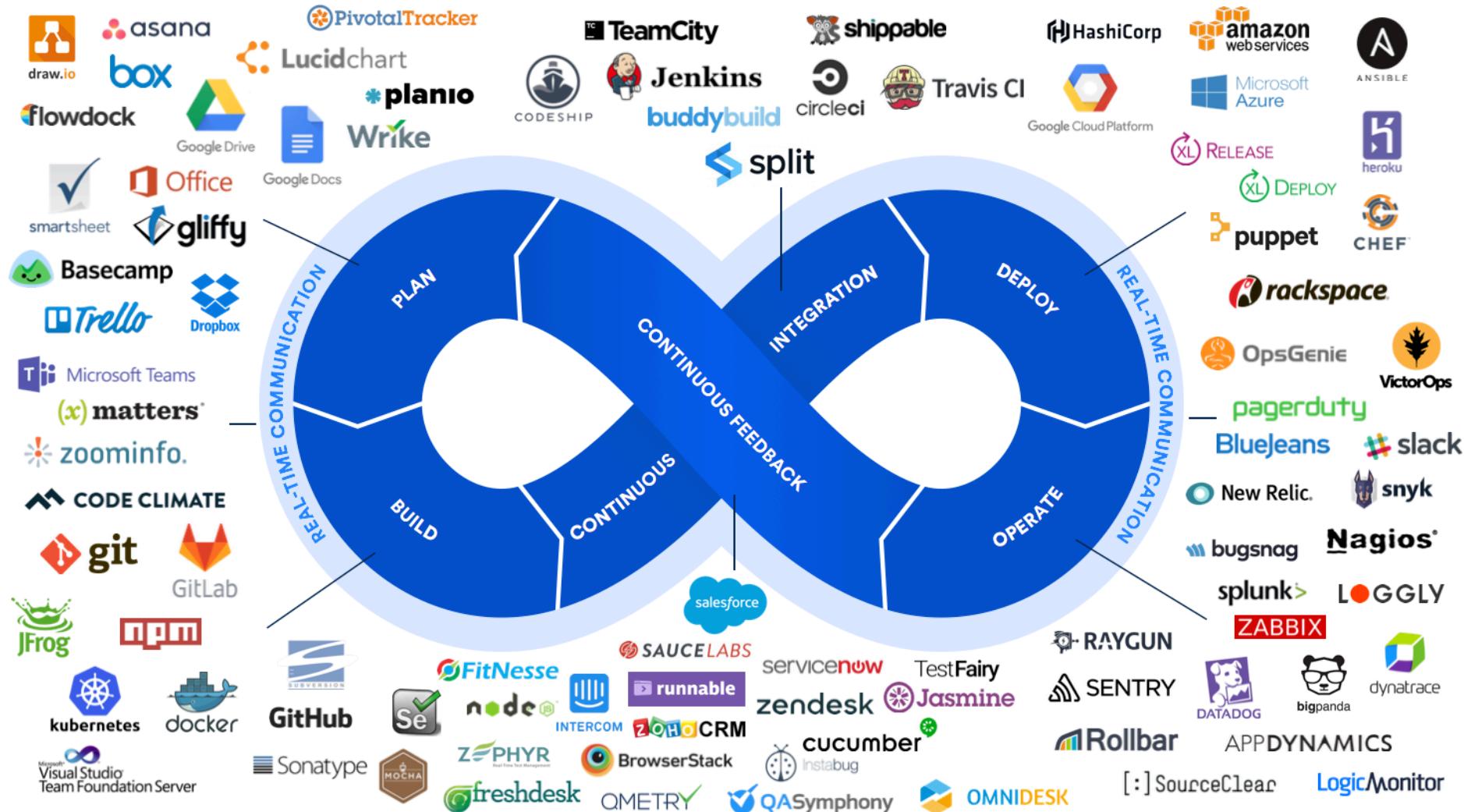
Subversion

Référentiel de sources et de composants

Serveur JEE (exemple)



DevOps (exemple)



Approche méthodologique 1/2

- ▶ Intérêts pour l'entreprise
 - ▶ Standardisation pour une meilleure maintenabilité
 - ▶ Cohérence entre les projets
 - ▶ Capitalisation

Approche méthodologique 2/2

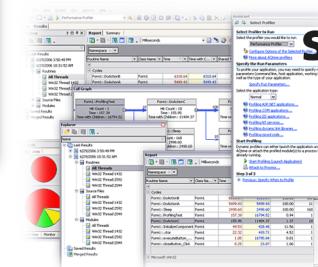
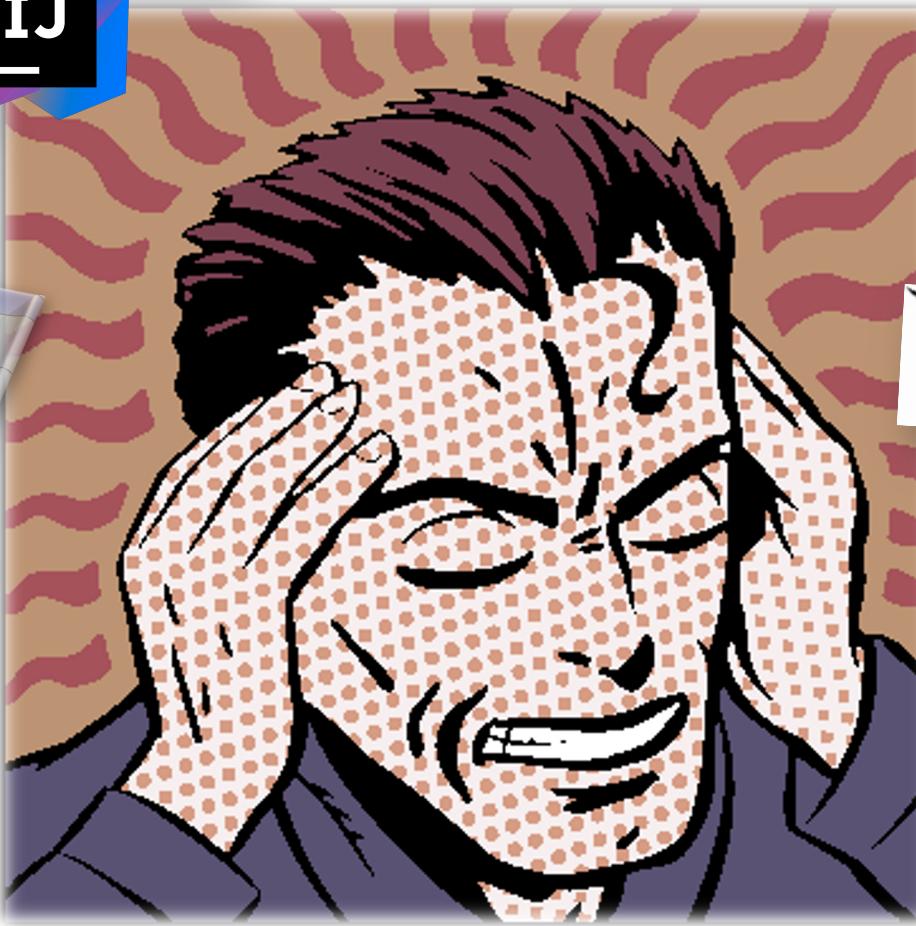
- ▶ Intérêt pour le client
 - ▶ Voir l'application se construire
 - ▶ Faire ses remarques au fur et à mesure
 - ▶ Prioriser les exigences

Software Engineering



Maven™

JUnit



Highlights

- Documentation
- Logging
- Testing
- Static analysis
- Refactoring
- Build automation
- Versioning
- Continuous integration
- *And towards DevOps and modern architectures*

Documentation and Source Code

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 controllers.
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.
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 JRootPane child: JDialog, JFrame, JWindow, JApplet, JInternalFrame.
Scollable	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 <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-oriented documentation about using `JFrame` in *The Java Tutorial*, in the section [How to Make Frames](#).

The `JFrame` class is slightly incompatible with `Frame`. Like all other JFC/Swing top-level containers, a `JFrame` contains a `JRootPane` as its only child. The `content pane` provided by the root pane should, as a rule, contain all the components in 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 normally be sent to the content pane instead of the `JFrame` itself. The content pane will always be non-null. A `NullPointerException` 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 @eypyoncf

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
 - <https://www.oracle.com/java/technologies/javase/codeconventions-contents.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
 - Indentateurs (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 ...

Logging



Logging

- Logging is chronological and systematic record of data processing events in a program
 - e.g. the Windows Event Log
- Logs can be saved to a persistent medium to be studied at a later time
- Use logging in the development phase:
 - Logging can help you **debug** the code
- Use logging in the production environment:
 - Helps you **troubleshoot problems**

Logging, why? (claims)

- Logging is easier than debugging
- Logging is faster than debugging
- Logging can work in environments where debugging is not supported
- Can work in production environments
- Logs can be referenced anytime in future as the data is stored

Logging Methods, How?

- The evil `System.out.println()`
- Custom Solution to Log to various datastores,
eg text files, db, etc...
- Use Standard APIs
 - Don't reinvent the wheel

Apache



- Popular logging frameworks for Java
- Designed to be reliable, fast and extensible
- Simple to understand and to use API
- Allows the developer to control which log statements are output with arbitrary granularity
- Fully configurable at runtime using external configuration files

Log4J Architecture

- Log4J has three main components: loggers, appenders and layouts
 - **Loggers**
 - Channels for printing logging information
 - **Appenders**
 - Output destinations (console, File, Database, Email/SMS Notifications, Log to a socket, and many others...)
 - **Layouts**
 - Formats that appenders use to write their output
- Priorities

Logger

- Responsible for Logging
- Accessed through java code
- Configured Externally
- Every Logger has a name
- Prioritize messages based on level
 - TRACE < DEBUG < INFO < WARN < ERROR < FATAL
- Usually named following dot convention like java classes do.
 - Eg com.foo.bar.ClassName
- Follows inheritance based on name

Logger API

- Factory methods to get Logger

- `Logger.getLogger(Class c)`
 - `Logger.getLogger(String s)`

- Method used to log message

- `trace()`, `debug()`, `info()`, `warn()`, `error()`, `fatal()`
 - Details
 - `void debug(java.lang.Object message)`
 - `void debug(java.lang.Object message, java.lang.Throwable t)`
 - Generic Log method
 - `void log(Priority priority, java.lang.Object message)`
 - `void log(Priority priority,
 java.lang.Object message, java.lang.Throwable t)`

Root Logger

- The root logger resides at the top of the logger hierarchy. It is exceptional in two ways:
 1. it always exists,
 2. it cannot be retrieved by name.
- `Logger.getLogger()`

Appender

- Appenders put the log messages to their actual destinations.
- No programmatic change is required to configure appenders
- Can add multiple appenders to a Logger.
- Each appender has its Layout.
- ConsoleAppender, DailyRollingFileAppender, FileAppender, JDBCAppender, JMSAppender, NTEventLogAppender, RollingFileAppender, SMTPAppender, SocketAppender, SyslogAppender, TelnetAppender

Layout

- Used to customize the format of log output.
- Eg. HTMLLayout, PatternLayout, SimpleLayout, XMLLayout
- Most commonly used is PatternLayout
 - Uses C-like syntax to format.
 - Eg. "%-5p [%t]: %m%n
 - DEBUG [main]: Message 1 WARN [main]: Message 2

Log4j Basics

- Who will log the messages?
 - The *Loggers*
- What decides the priority of a message?
 - *Level*
- Where will it be logged?
 - Decided by *Appender*
- In what format will it be logged?
 - Decided by *Layout*

Log4j in Action

```
// get a logger instance named "com.foo"
Logger logger = Logger.getLogger("com.foo");

// Now set its level. Normally you do not need to set the
// level of a logger programmatically. This is usually done
// in configuration files.
logger.setLevel(Level.INFO);

Logger barlogger = Logger.getLogger("com.foo.Bar");

// This request is enabled, because WARN >= INFO.
logger.warn("Low fuel level.");

// This request is disabled, because DEBUG < INFO.
logger.debug("Starting search for nearest gas station.");

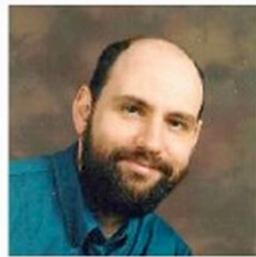
// The logger instance barlogger, named "com.foo.Bar",
// will inherit its level from the logger named
// "com.foo" Thus, the following request is enabled
// because INFO >= INFO.
barlogger.info("Located nearest gas station.");

// This request is disabled, because DEBUG < INFO.
barlogger.debug("Exiting gas station search");
```

Log4j Optimization & Best Practises

- Use logger as private static variable
- Only one instance per class
- Name logger after class name
- Don't use too many appenders
- Don't use time-consuming conversion patterns
- Use `Logger.isDebugEnabled()` if need be
- Prioritize messages with proper levels

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.**

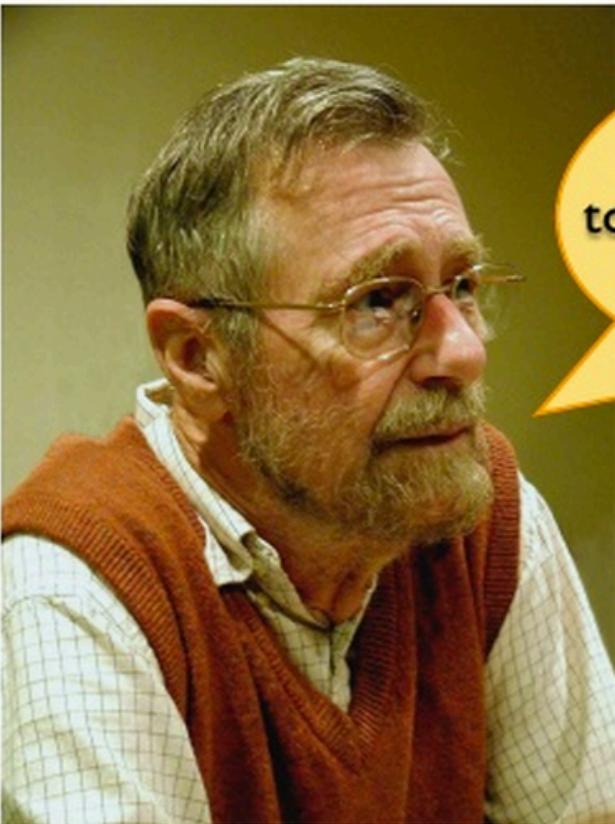
Testing and Static Analysis

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

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



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

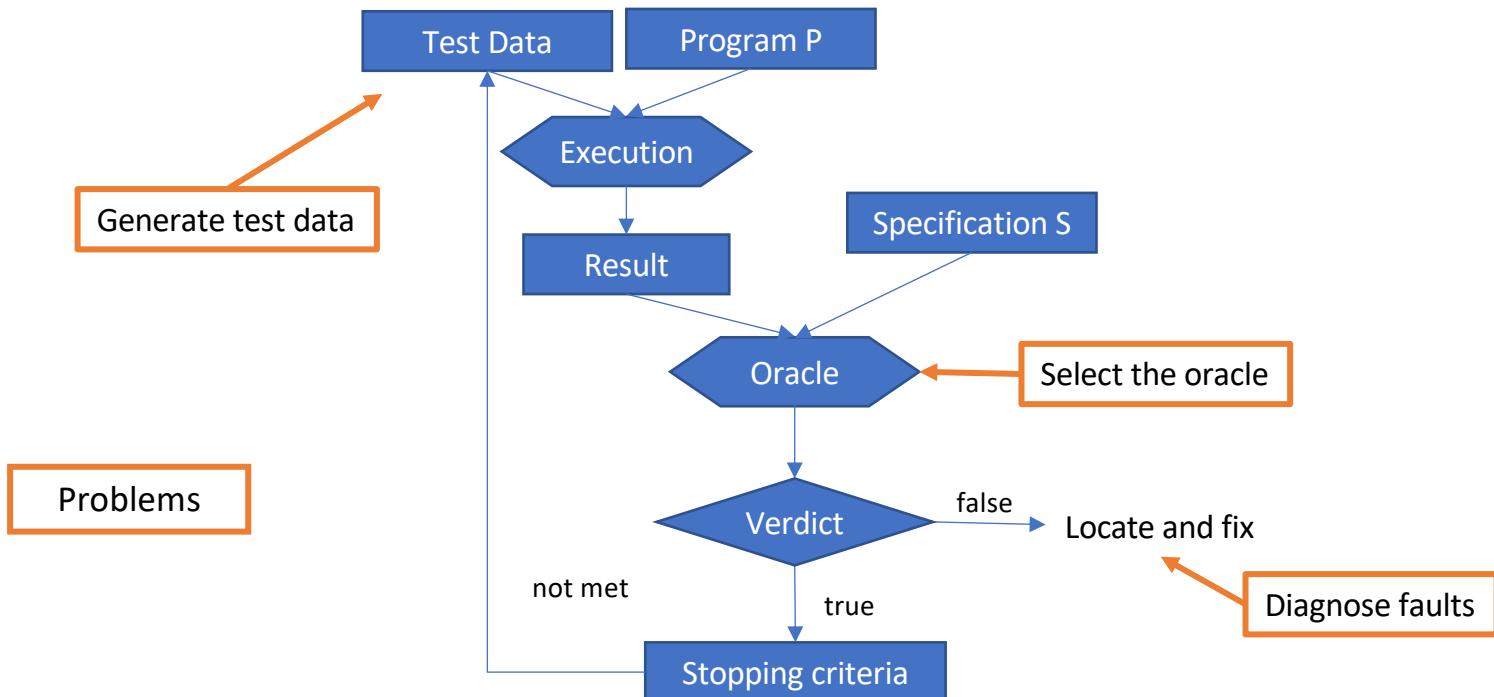


Unit testing on individual units of source code (=smallest testable part).

Integration testing on groups of individual software modules.

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

Le test dynamique : processus



Test unitaire OO

- Tester une unité isolée du reste du système
- L'unité est la classe
 - Test unitaire = test d'une classe
- Test du point de vue client
 - les cas de tests appellent les méthodes depuis l'extérieur
 - on ne peut tester que ce qui est public
 - Le test d'une classe se fait à partir d'une classe extérieure
- Au moins un cas de test par méthode publique
- Il faut choisir un ordre pour le test
 - quelles méthodes sont interdépendantes?

Test unitaire OO

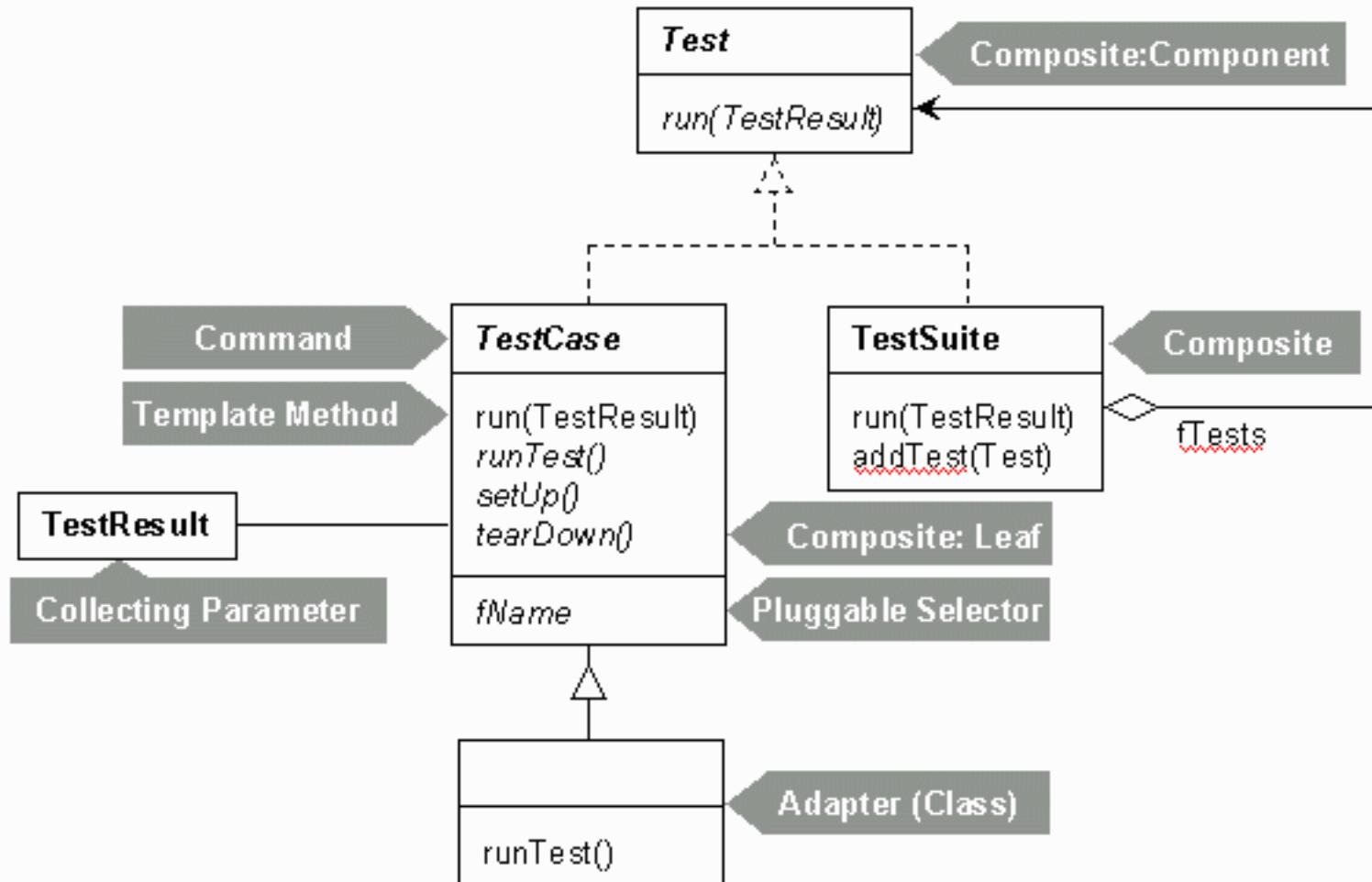
- Problème pour l'oracle :
 - Encapsulation : les attributs sont souvent privés
 - Difficile de récupérer l'état d'un objet
- Penser au test au moment du développement (« testabilité »)
 - prévoir des accesseurs en lecture sur les attributs privés
 - des méthodes pour accéder à l'état de l'objet

Cas de test unitaire

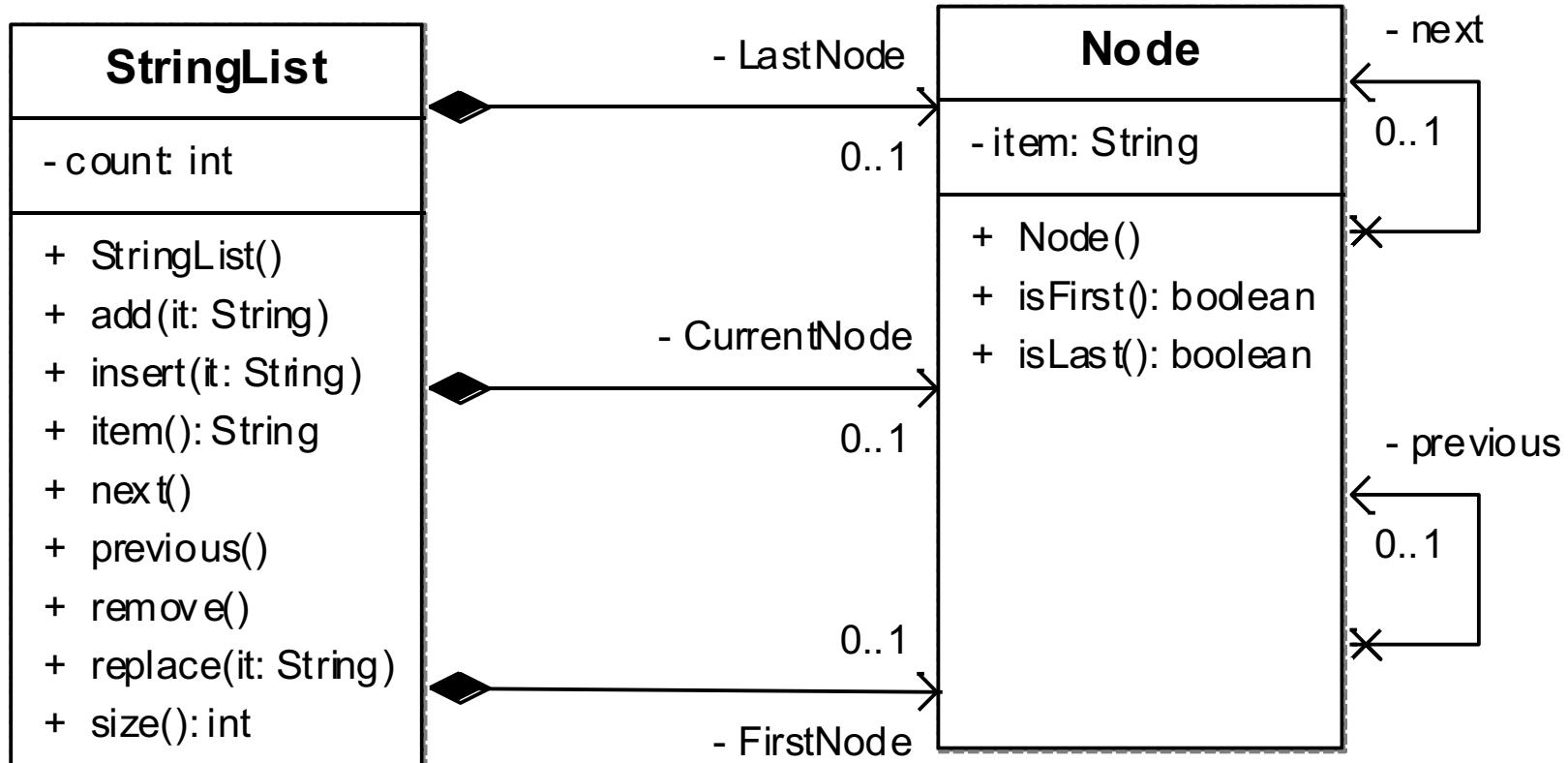
- Cas de test = une méthode
- Corps de la méthode
 - Configuration initiale
 - Une donnée de test
 - un ou plusieurs paramètres pour appeler la méthode testée
 - Un oracle
 - il faut construire le résultat attendu
 - ou vérifier des propriétés sur le résultat obtenu
- Une classe de test pour une classe testée
 - Regroupe les cas de test
 - Il peut y avoir plusieurs classes de test pour une classe testée

Junit and... Design Patterns

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

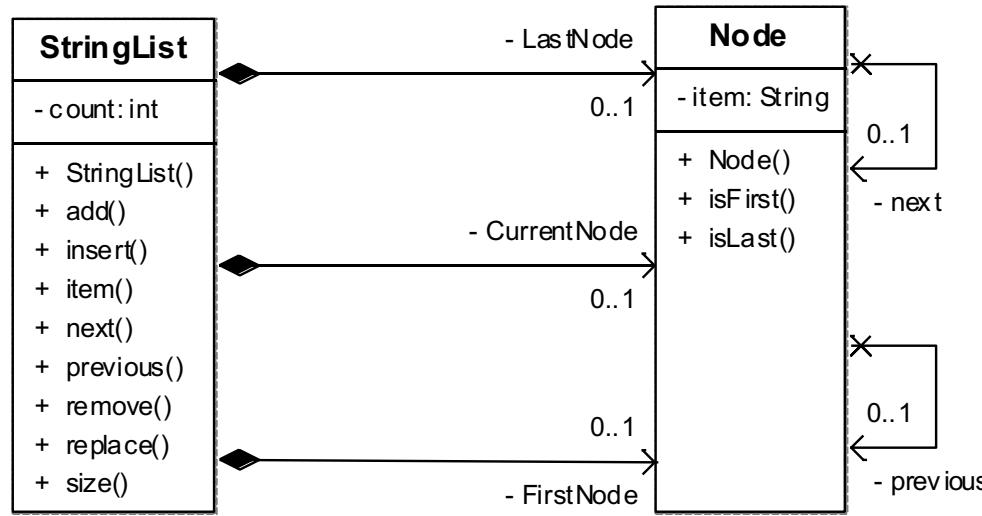


Exemple : test de StringList



- Créer une classe de test qui manipule des instances de la classe StringList
- Au moins 9 cas de test (1 par méthode publique)
- Pas accès aux attributs privés : count, LastNode, CurrentNode, FirstNode

Exemple : insertion dans une liste



spécification du cas de test {

initialisation {

appel avec donnée de test {

oracle {

```
//first test for insert: call insert
//and see if current element is the
//one that's been inserted
public void testInsert1(){
    list.add("first");
    list.add("second");
    list.insert("third");
    assertTrue(list.size()==3);
    assertTrue(list.item() == "third");
```

JUnit: codage

- Organisation du code des tests
 - cas de Test: TestCase
 - setUp() et tearDown()
 - les méthodes de test
 - suite de Test: TestSuite
 - Méthodes de test
 - Cas de test
 - Suite de Test

JUnit

- Une classe de test regroupe des cas de test pour une classe

```
import org.junit.Before;  
import org.junit.Test;  
  
public class TestStringList {  
    //déclaration des instances  
    private StringList list;  
  
    //setUp()  
    //tearDown()  
    //méthodes de test  
    //main()  
}
```

JUnit

- la méthode setUp:

```
//appelée avant chaque cas de test
//permet de factoriser la construction de l'état initial
@Before
protected void setUp() throws Exception {
    list = new StringList();
}
```

- la méthode tearDown:

```
//appelée après chaque cas de test
//permet de défaire « l'état du monde »
@After
protected void tearDown() throws Exception {
    super.tearDown();
}
```

JUnit

- les méthodes de test:

```
//test add two elements
@Test
public void testAdd2 () {
    list.add("first");
    list.add("second");
    assertTrue(list.size()==2);
    assertTrue(list.item()=="second");
}
```

Les assertions de JUnit

- fail() / fail(String message)
- assertTrue(...), assertFalse(...)
- assertEquals(Object, Object)
- assertSame(Object, Object)
- assertNull(...)
- assertEquals(double expected, double actual, double delta)

Voir la liste des méthodes de la classe Assert

JUnit v3 : détail d'implémentation

- Pour exécuter une suite de tests, JUnit utilise l'introspection

```
public TestSuite (final Class theClass){  
    ...  
    Method[] = theClass.getDeclaredMethods  
    ...  
}  
  
private boolean isTestMethod(Method m) {  
    String name= m.getName();  
    Class[] parameters= m.getParameterTypes();  
    Class returnType= m.getReturnType();  
    return parameters.length == 0 && name.startsWith("test") &&  
    returnType.equals(Void.TYPE);  
}
```

JUnit version 4/5

- Fonctionne avec Java 5+
- Utilisation intensive des annotations
- Tests paramétrés, timeouts, etc

JUnit v4/5 : classe et méthode de test

- Classe de test :
 - `import org.junit.Test;`
 - `import static org.junit.Assert.*;`
- Méthodes de test :
 - Nom de méthode quelconque
 - Annotation `@Test`
 - Publique, type de retour `void`
 - Pas de paramètre, peut lever une exception
 - Annotation `@Test(expected = Class)` pour indiquer l'exception attendue
 - Annotation `@Ignore` pour ignorer un test

JUnit

- Permet de structurer les cas de test
 - cas de test / suite de test
- Permet de sauvegarder les cas de test
 - important pour la non régression
 - quand une classe évolue on ré-exécute les cas de test

Debugging

- Symbolic debugging
 - javac options: -g, -g:source,vars,lines
 - command-line debugger : jdb (JDK)
 - commands look like those of dbx
 - graphical « front-ends » for jdb (IDE or external)
 - Misc
 - Multi-threads, Cross-Debugging (-Xdebug) on remote VM , ...

Monitoring

- Tracer
 - TRACE options of the program
 - can slow-down .class with TRACE/←TRACE tests
 - solution : use a pre-compiler (excluding trace calls)
 - Kernel tools, like OpenSolaris DTrace (coupled with the JVM)
- Logger
 - Record events on a registry, to be used at execution time or later on (via some event handlers)
 - Tools
 - Apache Log4J, ObjectWeb MonoLog
 - Package `java.util.logging` since J2SE1.4
 - `Logger`, `LogRecord`, `Handler`

Validation

- Assertion
 - Pre-Condition, Post-Condition, Invariant
 - EIFFEL, CLU ... built-in
 - Java since SE 1.4
- Other tools
 - AssertMate (Reliable Software Technologies)
 - <http://www.ddj.com/articles/1998/9801d/9801d.htm#rel>
 - JML (Java Modeling Language)
 - <http://www.eecs.ucf.edu/~leavens/JML/>
 - tool support with <https://www.openjml.org/>
 - iContract (Reliable Systems)
 - ...

Performances

- Measure/Analyze
 - Benchmark
 - `java.awt.Robot` (to build clients for testing)
 - Accounting : <http://abone.unige.ch/jraf/index.htm>
 - JProfiler, Optimiszelt ...
 - JMH
- Optimization
- See books:
 - Steve Wilson, Jeff Kesselman, « Java Platform Performance: Strategies and Tactics (The Java Series) », 1 edition (May 25, 2000), Addison-Wesley Pub Co; ISBN: 0-201-70969-4
 - Jack Shirazi, “Java Performance Tuning”, Ed Oreilly, 2000, ISBN 0-596-00015-4

Choosing the JVM and JRE

- Some criteria
 - License, redistribution, supports, performances, contraintes (embedded, servers, real-time, ...), runtimes, ...
- Examples (cf. https://en.wikipedia.org/wiki/List_of_Java_virtual_machines)
 - Azul Zulu
 - Bck2Brwsr
 - CACAO
 - Codename One
 - DoppioJVM
 - Eclipse OpenJ9
 - GraalVM
 - HaikuVM
 - HotSpot
 - Jamiga
 - JamVM
 - Jikes RVM (Jikes Research Virtual Machine)
 - JVM.go
 - leJOS
 - Maxine
 - Multi-OS Engine
 - RopeVM
 - ...

Measurements and Analysis of Performances

- Java Profiler
 - Use to profile an application
 - CPU usage, Memory, Network, time spent, and Garbage collection
 - In Total or by threads, or called methods

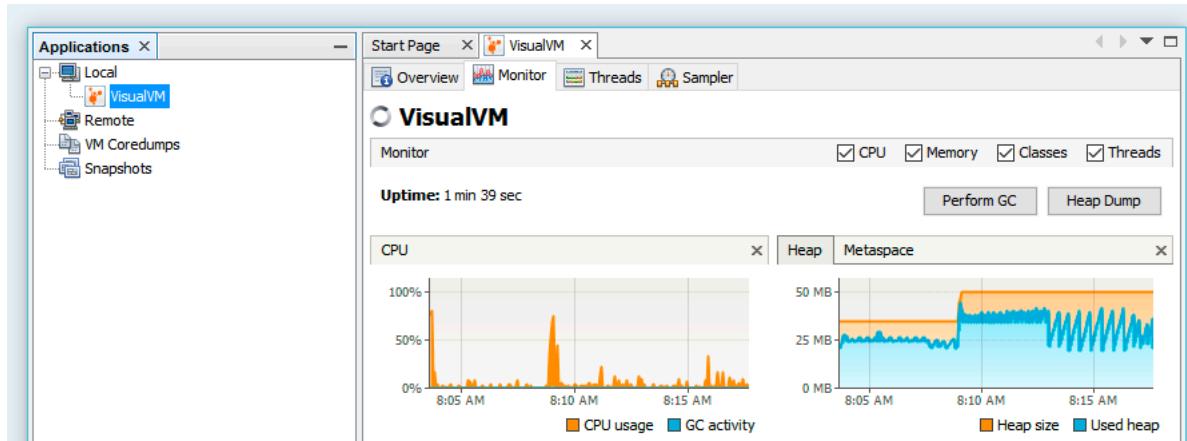


JProfiler

YourKit

VisualVM
<https://visualvm.github.io/>

- « VisualVM is a visual tool that integrates several existing JDK software tools and lightweight memory and CPU profiling capabilities. This tool is designed for both production and development time use and further enhances the capability of monitoring and performance analysis for the Java SE platform.»
- **VisualVM includes the JConsole.**



Performance Optimizations

- Java's Script Engine
 - Jython (jython.org), ...
- Bytecode interpreter, JIT and compiler
 - GraalVM, OpenJ9, Azul...
- Native compiler (static)
 - .class to .c to .s to .exe
- On-the-fly compiler (dynamic)
 - Compilation JIT (Just-In-Time) de Symantec
- HotSpot™ Optimizer
 - garbage collector
 - « method inlining »
 - with load-time verification (dynamic) of class bytecode
- Benchmark de JVM (e.g., JMH, Krun)

Code Quality Metrics

- Metrics on project source code to evaluate its quality (maintenance, reverse-engineering, evolution ...)
 - and how good the development team is :-)
- Metrics
 - LOC, LOCC, McCabe Cyclomatic Complexity, ...

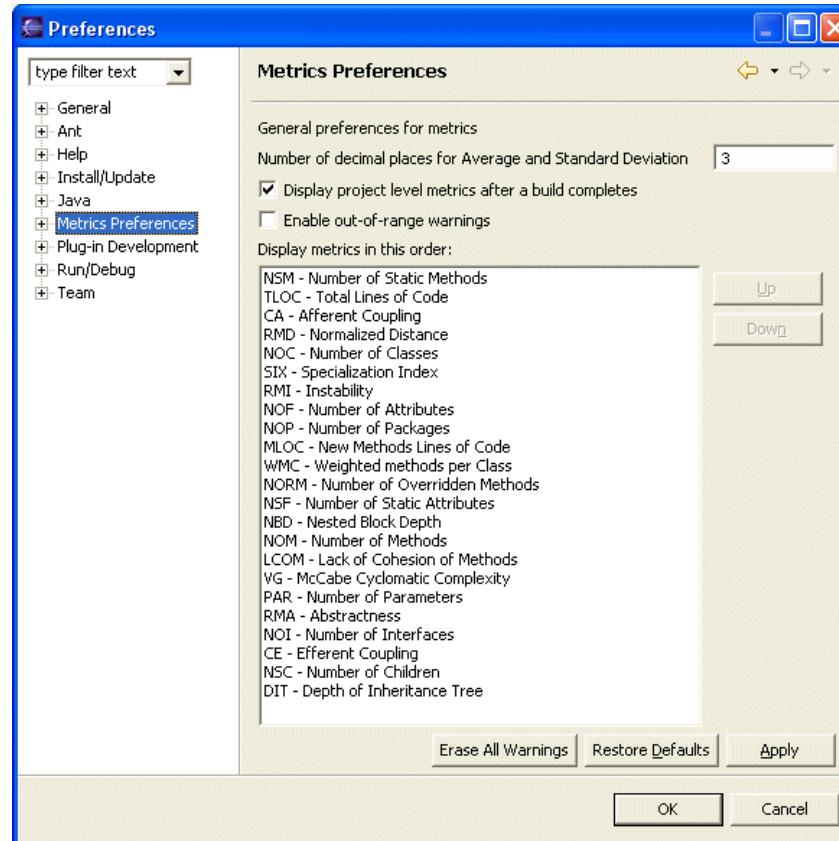


Lectures

- Brian Henderson-Sellers , "Object-Oriented Metrics, measures of Complexity", Ed Prentice Hall, 1996
- Robert Martin, "OO Design Quality Metrics, An Analysis of Dependencies", 1994, <http://www.objectmentor.com/resources/articles/oodmetrc.pdf>
- Chidamber and Kemerer, A Metrics Suite for Object Oriented Design, http://www.pitt.edu/~ckemerer/CK%20research%20papers/MetricForOOD_ChidamberKem erer94.pdf
- Mariano Ceccato and Paolo Tonella, Measuring the Effects of Software Aspectization, <http://homepages.cwi.nl/~tourwe/ware/ceccato.pdf>
- Robert Martin, "Agile Software Development, Principles, Patterns and Practices", Prentice Hall, 1st edition, 2002, ISBN: 978-0135974445

Code Quality Metrics

- Example: Metrics (Ant/Maven/Sonar + Eclipse plugin)



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
 - Apply design pattern / remove anti pattern
- Minimizes technical debt
- Keep development at speed
- To make the software easier to understand
- To help find bugs
- To “Fix broken windows”

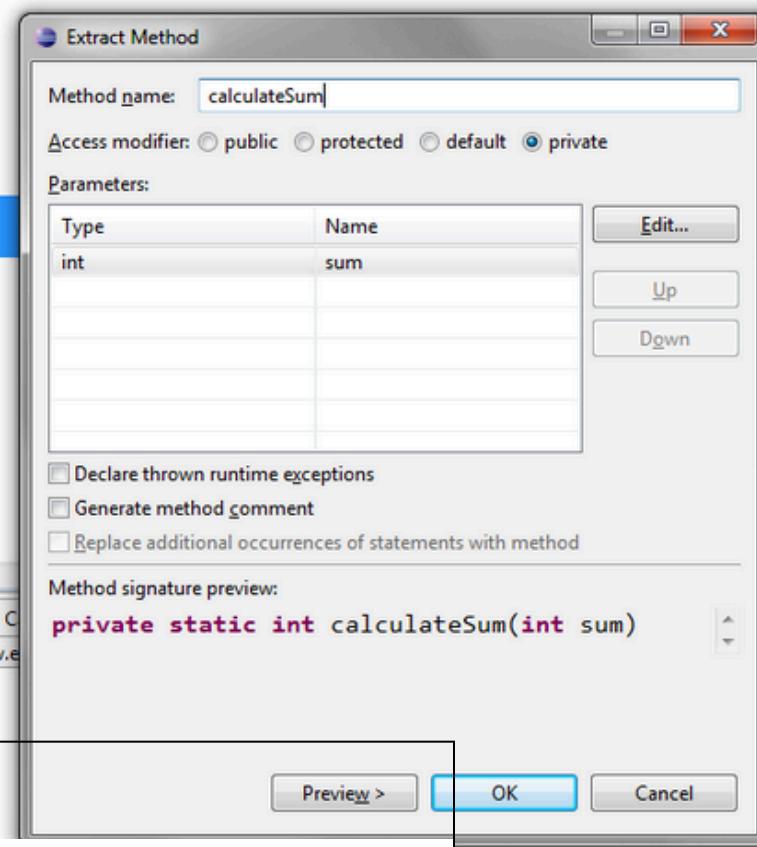
How do we Refactor?

- 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);
    }
}
```



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

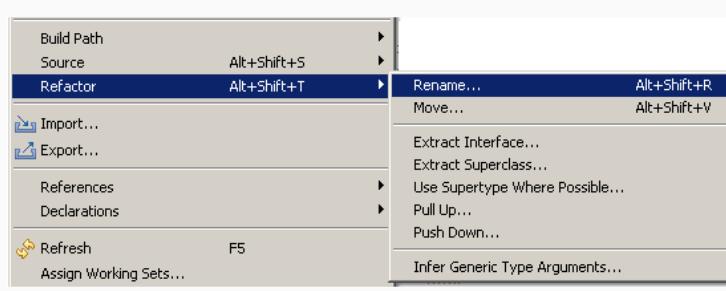
public class MyFirstClass {

    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

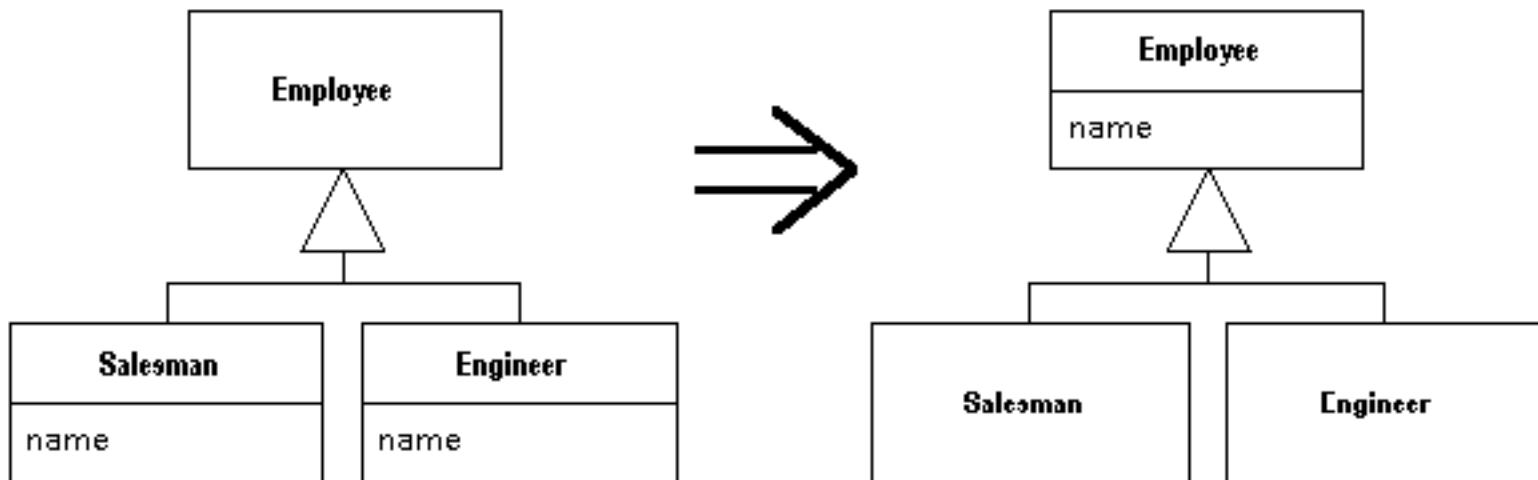


The screenshot shows the Eclipse IDE's Help menu open, displaying the 'Refactor Actions' documentation page. The page lists various refactoring actions with their descriptions, keyboard shortcuts, and availability.

Name	Description
Rename	Renames the selected element and if enabled corrects all references to the elements (also in other files). Available: Methods, method parameters, fields, local variables, types, type parameters, enum constants, compilation units, packages, source folders, projects and on a text selection resolving to one of these element types Shortcut: Alt + Shift + R Options: Renaming a type does allow to rename similarly named variables and methods. Enable 'Update similarly named variables and methods' in the Rename Type dialog. Select 'Configure...' to configure the strategy for matching type names Renaming a package does allow to rename its subpackages. Enable 'Rename subpackage' in the Rename Package dialog Edit original method as delegate to changed method to keep the original method. Optionally you can deprecate the old method.
Move	Moves the selected element and if enabled corrects all references to the elements (also in other files). Available: Instance method from the Refactor Actions menu to the Refactor Actions menu Shortcut: Alt + Shift + V Options: You can use Drag & Drop in the Package Explorer to start this refactoring.
Change Method Signature	Changes parameter names, parameter types, parameter order and updates all references to the corresponding method. Additionally, parameters and thrown exceptions can be removed or added and method return type and method visibility can be changed. Available: Methods, method parameters, fields, local variables, types, type parameters, enum constants, compilation units, packages, source folders, projects and on a text selection resolving to one of these element types Shortcut: Alt + Shift + C Options: Enable 'Keep original method as delegate to changed method' in the Change Method Signature dialog to keep the original method.
Extract Method	Creates a new method containing the statements or expressions currently selected and replaces the selection with a reference to the new method. This feature is useful for cleaning up lengthy, cluttered, or overly-complicated methods. Available: Methods, static final fields and text selections available at the Refactor Actions menu to the Refactor Actions menu Shortcut: Alt + Shift + M
Extract Local Variable	Creates a new variable assigned to the expression currently selected and replaces the selection with a reference to the new variable. Available: Text selections that resolve to local variables. You can use Expand Selection to from the Refactor Actions menu to get a valid selection range. Shortcut: Alt + Shift + L
Extract Constant	Creates a static final field from the selected expression and substitutes a field reference, and optimally replaces other places where the same expression occurs. Available: Constant expressions or text selections available at the Refactor Actions menu to the Refactor Actions menu Shortcut: Alt + Shift + C
Inline	Inline local variables, methods or text constants. Available: Methods, static final fields and text selections that resolve to methods, static final fields or local variables Shortcut: Alt + Shift + I
Convert Anonymous Class to Nested	Converts an anonymous inner class to a member class. Available: Anonymous inner classes Shortcut: Alt + Shift + I
Move Type to New File	Creates a new Java compilation unit for the selected member type or the selected secondary type, updating all references as needed. For non-static member types, a field is added to allow access to the former enclosing instance, if necessary. Available: Member types Shortcut: Alt + Shift + F
Convert Local Variable to Field	Creates a local variable from a field, which becomes a member of the class, resulting to a member type or a secondary type. Available: Text selections that resolve to local variables Shortcut: Alt + Shift + F
Extract Superclass	Extracts a new superclass from a set of base types. Available: Types Options: Enable 'Use the extracted class where possible' to use the newly created class wherever possible. See Use SuperType Where Possible.
Extract Interface	Creates a new interface with a set of methods and makes the selected class implement the interface. Available: Types Options: Enable 'Use the extracted interface type where possible' to use the newly created interface wherever possible. See Use SuperType Where Possible.
Use Supertype Where Possible	Resolves occurrences of a type or its supertypes after identifying at places where this replacement is possible. Available: Types
Push Down	Moves a set of methods and fields from a class to the subclasses. Available: One or more methods and fields declared in the same type or on a text selection made a field or method
Pull Up	Moves a field or method to a superclasses of its declaring class or on the case of methods) declares the method as abstract in the superclass. Available: Methods, static final fields and text selections available at the Refactor Actions menu to the Refactor Actions menu Shortcut: Alt + Shift + U
Extract Class	Replaces a set of fields with new container object references to the fields are updated to access the new container object. Available: A field or a type containing fields Options: Enable 'Create Getter and Setter' to add accessors methods to the new type
Introduce Parameter Object	Creates a set of parameters with a new class, and updates all callers of the method to pass an instance of the new class as the value to the introduce parameter. Available: Methods on text selection resolving to a method Options: Enable 'Redefine all method invocations' to replace all calls to the original method by calls to the introduction method
Introduce Indirection	Creates a static induction method delegating to the selected method. Available: Methods or on text selection resolving to a method Options: Enable 'Redefine all method invocations' to replace all calls to the original method by calls to the induction method
Introduce Factory	Creates a new factory method, which will call a selected constructor and return the created object. All references to the constructor will be replaced by calls to the new factory method. Available: Constructors on text selection resolving to a constructor
Introduce Parameter	Replaces an expression with a reference to a new method parameter, and updates all callers of the method to pass the expression as the value of that parameter. Available: Text selections that resolve to expressions
Encapsulate Field	Replaces all references to a field with getter and setter methods. Available: Field or a text selection resolving to a field
Generalize Declared Type	Allows the user to choose a supertype of the reference's current type. If the reference can be safely changed to the new type, it is. Available: Type references and declarations of fields, local variables, and parameters with reference types
Infer Generic Type Arguments	Replaces raw type occurrences of generic types by identifying all places where this replacement is possible. Available: Projects, packages, and types Options: 'Match type arguments of an instance of the receiver type'. Well-behaved classes generally respect this rule, but if you know that your code violates it, uncheck the box 'Leave unconstrained type arguments raw (rather than inferring <?>)'. If there are no constraints on the elements of e.g. ArrayList<?>, uncheck this box will cause Eclipse to still provide a wildcard parameter, replacing the reference with ArrayList<?>
Migrate JAR File	Migrates a JAR File on the build path of a project to a newer version, possibly using refactoring information stored in the new JAR file to avoid breaking changes. Available: Always
Create Script	Creates a script of the refactorings that have been applied in the workspace. Refactoring scripts can either be saved to a file or copied to the clipboard. See Apply Script.
Apply Script	Applies a refactoring script to projects in your workspace. Refactoring scripts can either be loaded from a file or from the clipboard. See Create Script.
History	Browses the workspace refactoring history and offers the option to delete refactorings from the refactoring history. Available: Always

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
}
```

Build automation

Compilation chain

- Tools (*aka.* build automation systems)
 - make, gmake, nmake (Win),
 - Apache ANT, MAVEN, Gradle, NPM, FinalBuilder, Grunt
- To automate:
 - pre-compilation, obfuscation, verification
 - generation of .class and .jar
 - normal, tracing, debug, ...
 - documentation generation
 - « stubs » generation (rmic, idl2java, javacard ...)
 - test
 - ...

Maven

- Goal
 - Separation of concerns applied to project build
 - Compilation, code generation, unit testing, documentation, ...
 - Handle project dependencies with versions (artifacts)
- Project object model (POM)
 - abstract description of the project
 - Property inheritance from POM parents
- Tools (called plugin)
 - To compile, generate documentation, automate test ...
- Note: more and more useful !

Versioning

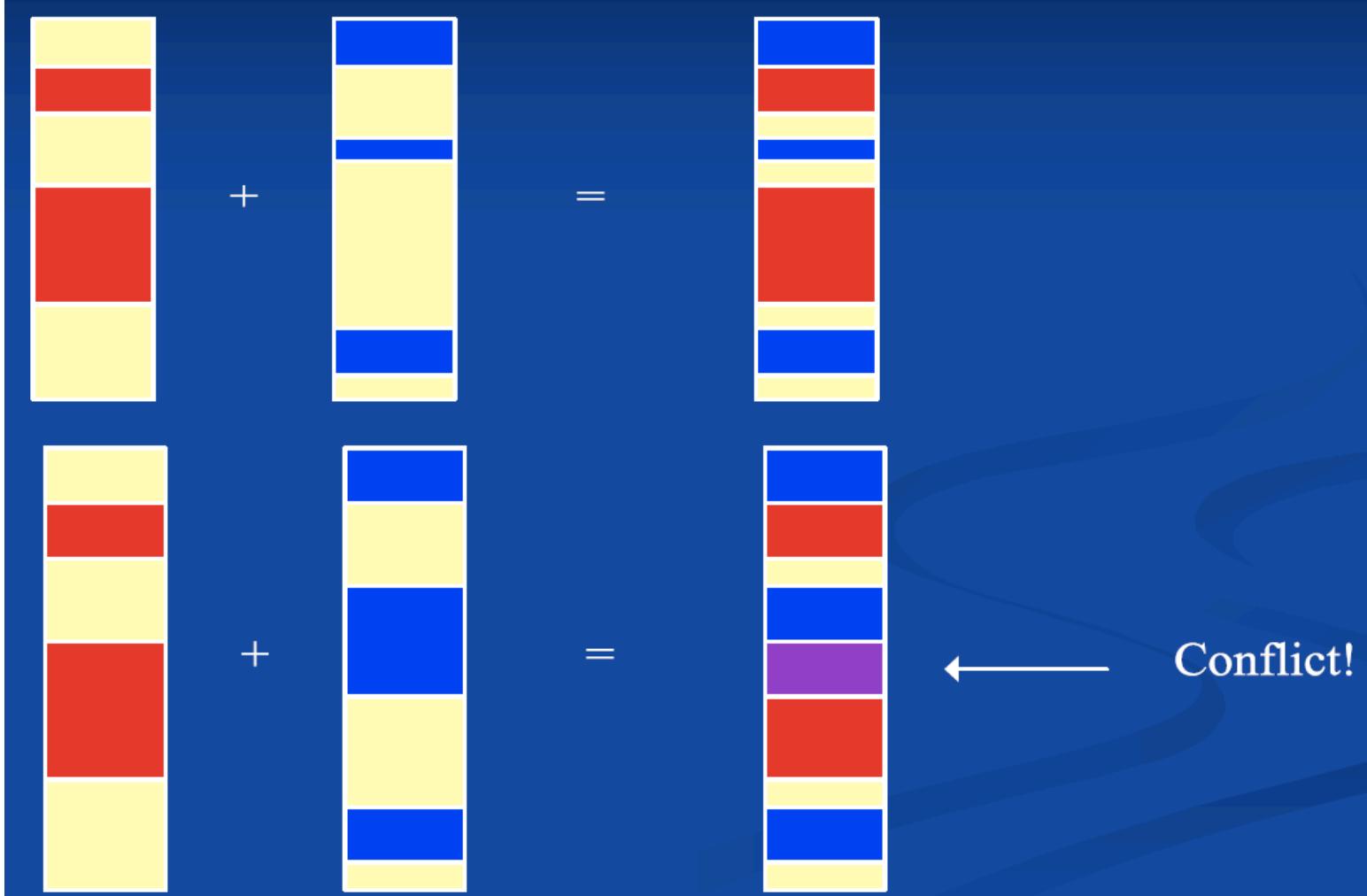
Versioning of source code

- Collaborative software engineering
- To master
 - software development by very large developer teams
 - parallel implementations (experiments, vendors)
- Goals
 - Increase productivity of developers and software robustness
 - Low-down development costs
- Manage software system configuration
 - to control software system's evolution
 - evolution tracking (time-machine)
 - issue and bug tracking

Versioning : What for?

- History of versions
 - back to an older version in case of errors
- Alternative versions (branching)
 - different design/implementations (maybe experimentals) for the same module
- Collaborative access by many developers
 - audit modification history
 - how many commits by X ?
 - when most of the commits are done?
 - ...

Concurrency management



Concurrency control

- Doing nothing!
- Lock-Modify-Unlock (Pessimistic)
 - SCCS, RCS
 - Decrease productivity
- Copy-Modify-Merge (Optimistic)
 - Conflicts resolution when concurrent modifications (which are actually rare)
 - Merge, Selection, ...
 - CVS, SVN, Git : Client level resolution
- Policy-based
 - Merging and validation process for each code contribution

Concept of Version

- Trunk
 - main development
- Branches
 - Alternatives to trunk
 - Different design/implementation (experimental), vendor-specific
- Revisions
 - Sequence of versions
- Tags
 - Symbolic references to revisions (Tiger, LongHorn, ...)
 - Represent a public release (R), a milestone (M)
- Branch merging

Tools

- Pioneers
 - SCCS, RCS, PVCS
- Current alternatives
 - CVS
 - SubVersion
 - Git
 - MS Visual SourceSafe
 - ChangeMan (Serena)
 - AllFusion Harvest (CA)
 - ClearCase (IBM Rational)
 - Perforce
 - CM Synergy (Telelogic)
 - Source Integrity (MKS)
 - PVCS (Merant)
 - TeamCode (Interwoven)
 - Surround CM (Seapine)
- Web-Oriented protocols
 - WebDAV/DeltaV

Git

- version control system
 - designed to handle very large projects with speed and efficiency
 - mainly for various open source projects, most notably the Linux kernel.
- <https://git-scm.com> (<http://git.or.cz>)

Tools to use Git

- SmartGit
- TortoiseGit
- CyberDuck
- Etc.
- GitHub => provide the whole forge with a GIT installed
 - Free for open-source project
 - Rich client: <https://desktop.github.com>
- GitLab

Centralized Version Control

- Traditional version control system
 - Server with database
 - Clients have a working version
- Examples
 - CVS
 - Subversion
 - Visual Source Safe
- Challenges
 - Multi-developer conflicts
 - Client/server communication

Distributed Version Control

- Authoritative server by convention only
- Every working checkout is a repository
- Get version control even when detached
- Backups are trivial
- Other distributed systems include
 - Mercurial
 - BitKeeper
 - Darcs
 - Bazaar

Git Advantages

- Resilience
 - No one repository has more data than any other
- Speed
 - Very fast operations compared to other VCS (I'm looking at you CVS and Subversion)
- Space
 - Compression can be done across repository not just per file
 - Minimizes local size as well as push/pull data transfers
- Simplicity
 - Object model is very simple
- Large userbase with robust tools

Git Architecture

- Index
 - Stores information about current working directory and changes made to it
- Object Database
 - Blobs (files)
 - Stored in .git/objects
 - Indexed by unique hash
 - All files are stored as blobs
 - Trees (directories)
 - Commits
 - One object for every commit
 - Contains hash of parent, name of author, time of commit, and hash of the current tree
 - Tags

Some Commands

- Getting a Repository
 - git init
 - git clone
- Get changes with
 - git fetch (fetches and merges)
 - git pull
- Commits
 - git add
 - git commit
- Propagate changes with
 - git push

Documenting,
Testing,

Design Patterns, bad smells

Refactoring,

Debugging, monitoring, logging

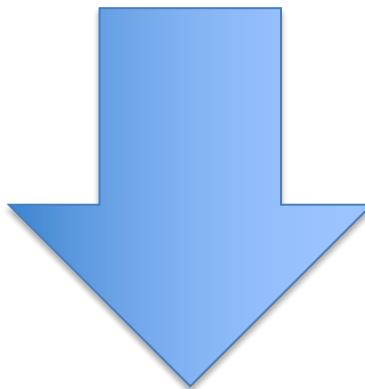
#1 What is the link?

- Documenting
 - Understanding (readability, maintainability)
- Refactoring
 - Improving the design (readability, maintainability, extensibility)
- The activity of documenting can somehow be replaced/automated 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  
}
```



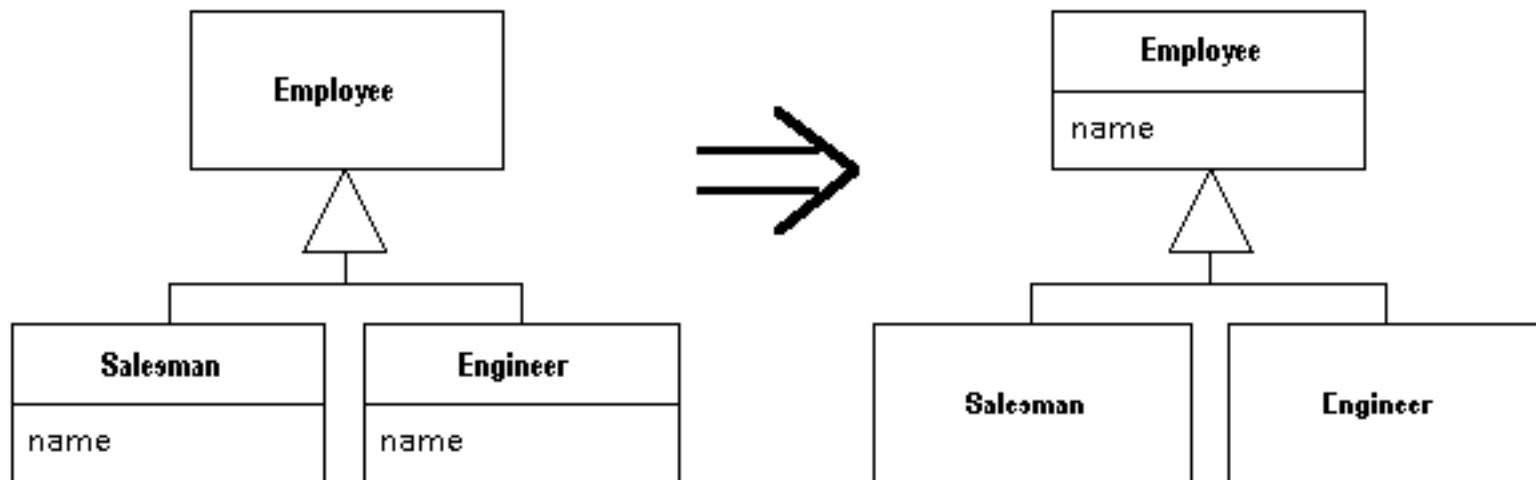
```
final boolean isMacOs      = platform.toUpperCase().indexOf("MAC") > -1;  
final boolean isIEBrowser = browser.toUpperCase().indexOf("IE") > -1;  
final boolean wasResized   = resize > 0;  
  
if (isMacOs && isIEBrowser && wasInitialized() && wasResized)  
{  
    // do something  
}
```

#2 What is the link?

Design patterns: there are refactorings

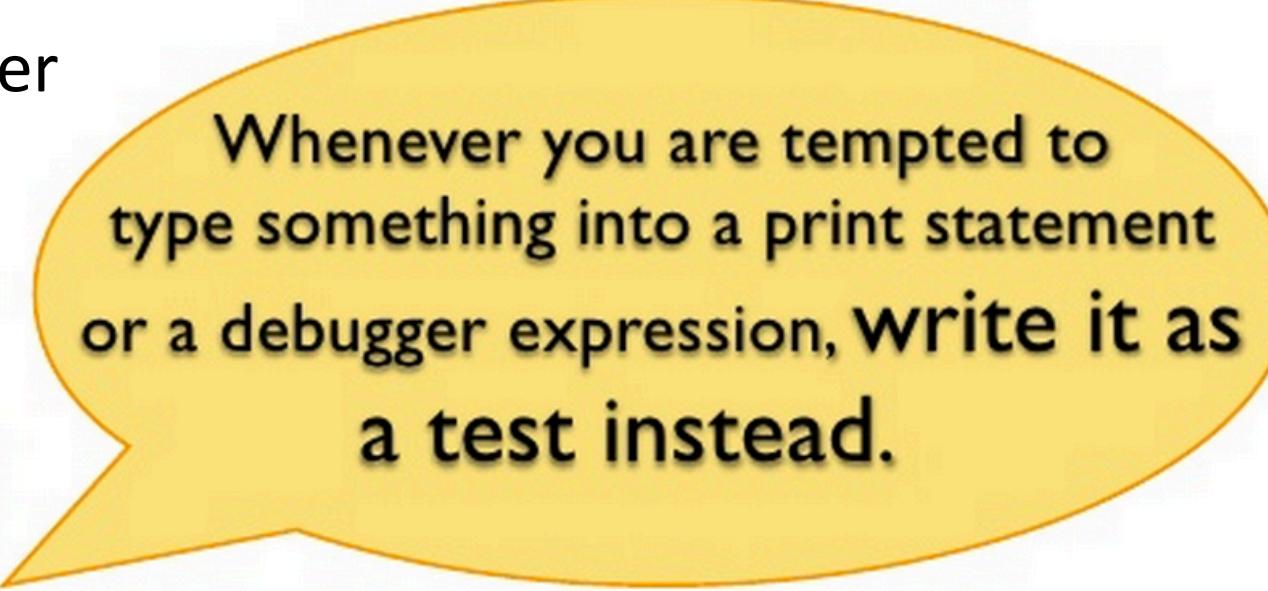
Two subclasses have the same field.

Move the field to the superclass.



#3 What is the link?

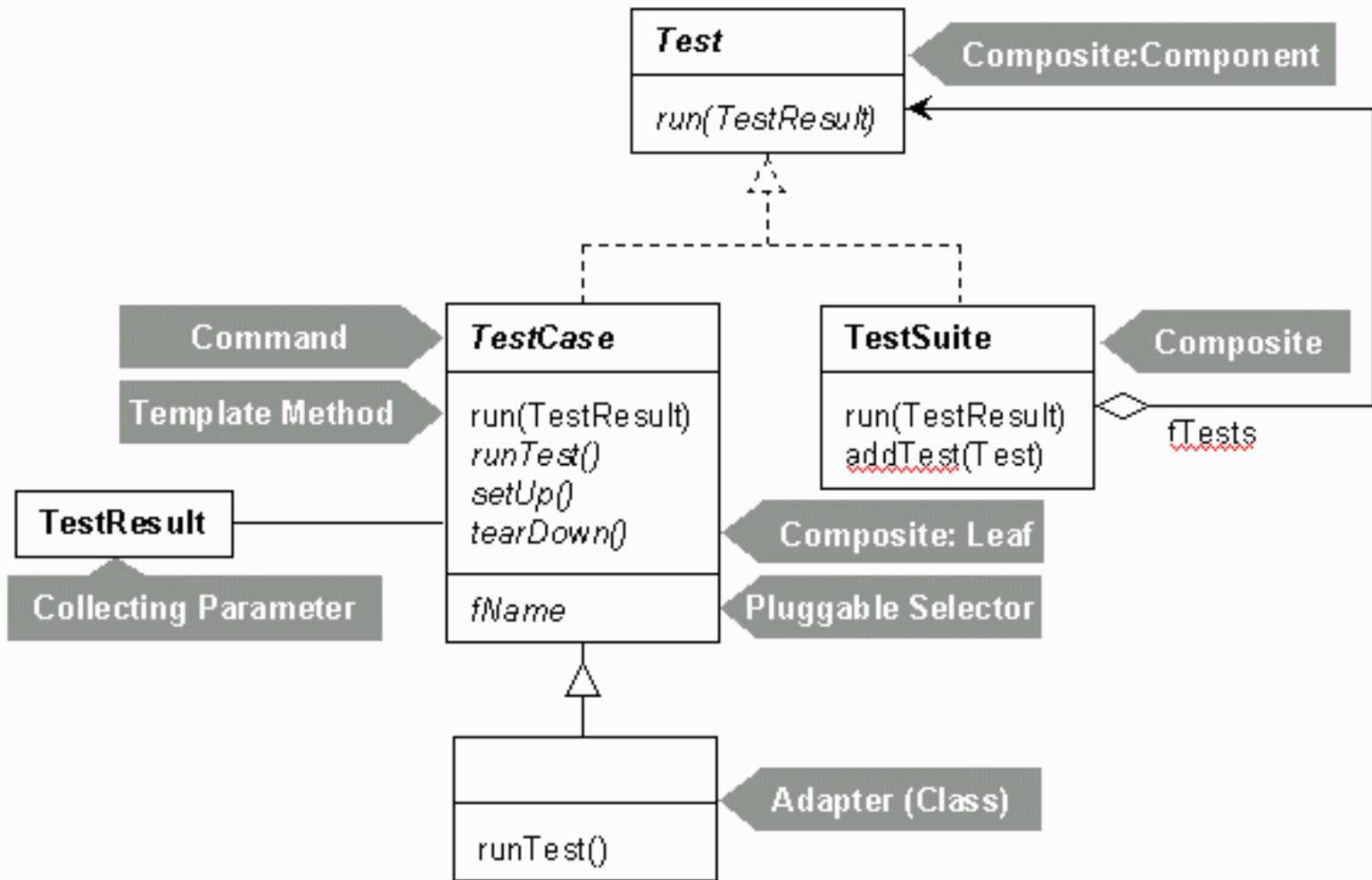
- Testing: “the activity of finding out whether a piece of code produces the intended behavior”
 - Debugging can help
 - Testing is better than debugging



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



JUnit and... Design patterns



Worth reading!

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

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
- 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!

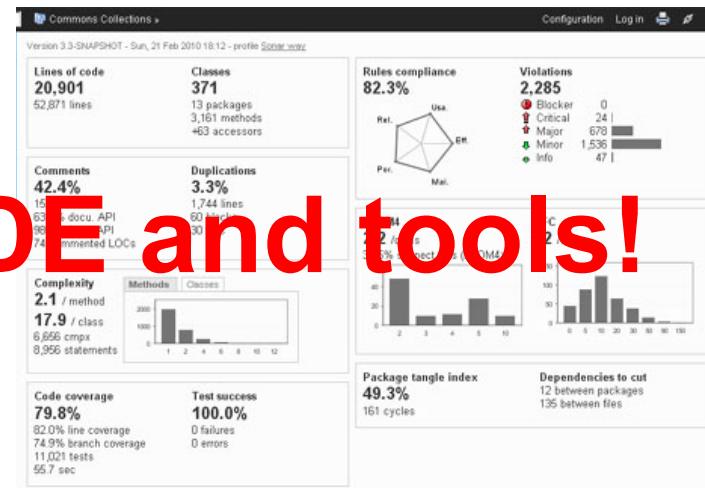
Documenting

Debugging

Refactoring

Testing

A screenshot of the Eclipse IDE interface. On the left, there is a code editor window displaying Java code. A specific line of code is highlighted with a blue selection bar. Below the code editor is a toolbar with various icons for navigating and managing projects. In the center, a modal dialog box titled "Extract Method" is open. It contains fields for "Method name" (set to "calculateSum"), "Access modifier" (set to "private"), and a "Parameters" table with one entry: "Type int" and "Name sum". There are several checkboxes at the bottom of the dialog, such as "Delete the now unused variable" and "Generate method comment", which are currently unchecked. At the bottom of the dialog are three buttons: "Preview >", "OK", and "Cancel".



With modern IDE and tools!

INTEGRATION CONTINUE



Jenkins

INTRODUCTION

- Qu'est ce que l'intégration continue ?
- Pourquoi automatiser ?
- Par où commencer ?
- Le cycle vertueux de l'intégration continue

Définitions

"L'intégration continue est un ensemble de pratiques utilisées en génie logiciel. Elles consistent à vérifier à chaque modification de code source que le résultat des modifications ne produit pas de régression de l'application en cours de développement."

Wikipedia

"Une pratique considérant différemment l'intégration, habituellement connue comme pénible et peu fréquente, pour en faire une tâche simple faisant partie intégrante de l'activité quotidienne d'un développeur."

Documentation
CruiseControl.NET

Qu'est ce que l'intégration continue ?

- Technique puissante permettant dans le cadre du développement d'un logiciel en équipes de:
 - Garder en phase les équipes de dév
 - Limiter risques de dérive
 - Limiter la complexité
- A intervalles réguliers, vous allez construire (build) et tester la dernière version de votre logiciel.
- Parrallèlement, chaque développeur teste et valide (commit) son travail en ajoutant son code dans un lieu de stockage unique.

Pourquoi automatiser ?

- Gagner du temps
 - Vous ne faites pas de tâches répétitives
- Gagner en confiance
 - Indépendant de votre efficacité du moment
 - Procédures répétables
- Diminue le besoin de documentation
 - Pour nouveaux entrants projet, utiliser scripts !
 - ... et + en analysant le script.

Par où commencer ?

- 1) Outil gestion versions code sources

- Lieu unique de partage

- Retour arrières, snapshots, branches...



- 2) Tests automatisés

- Chaque développeur



- 3) Scripts

- Coté serveur pour automatiser (Ex : crontab)

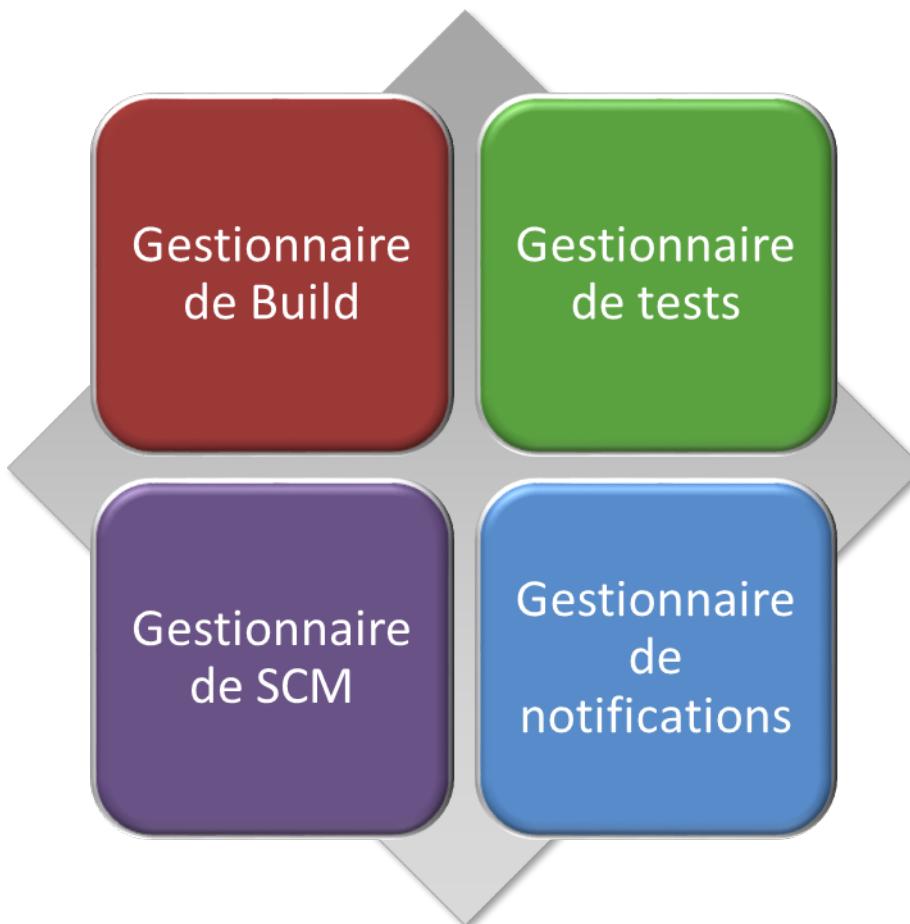


- 4) Outils de communication

- Mail, Tél, Rss...



Architecture d'un logiciel d'intégration

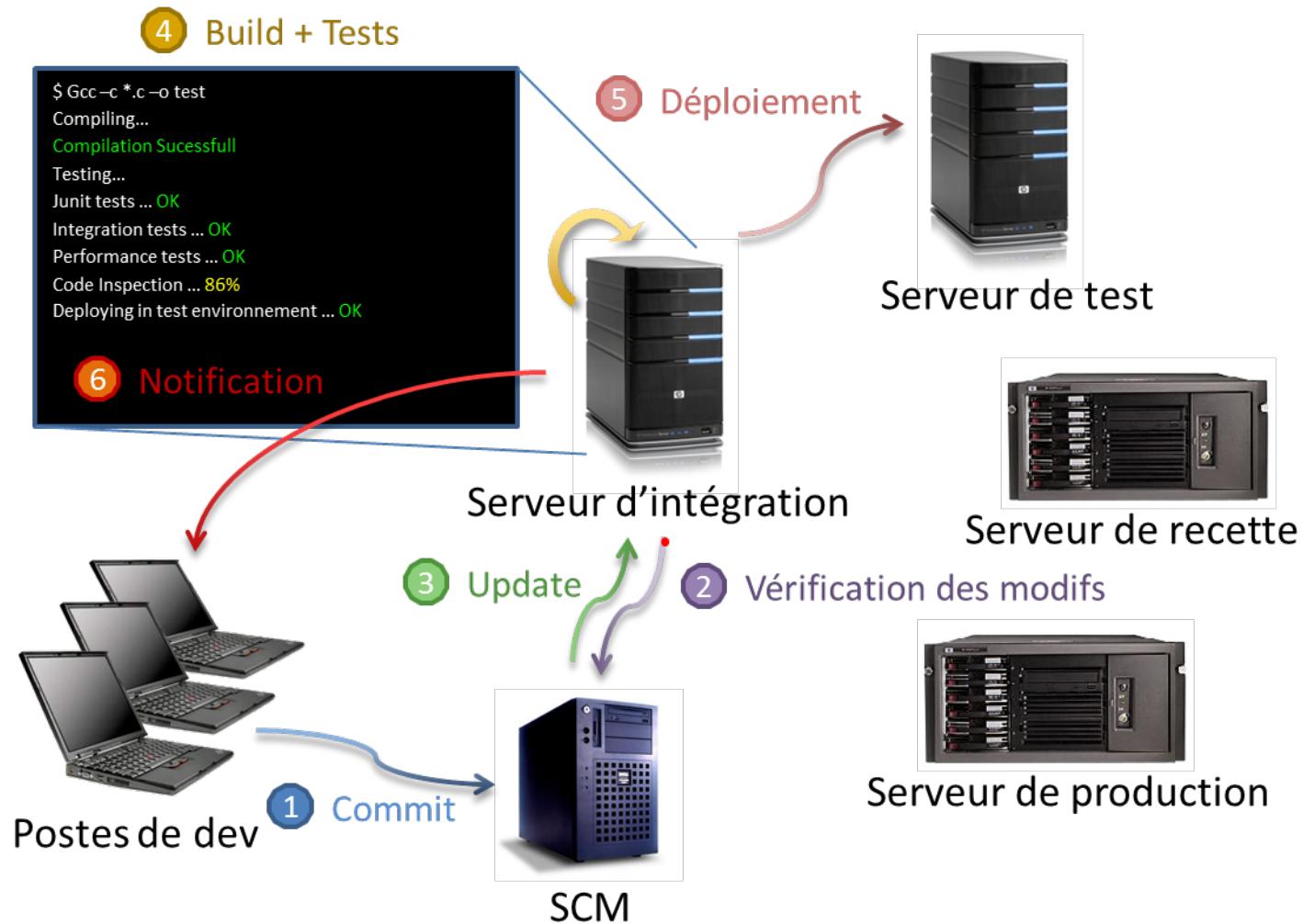


Un fonctionnement actif

- ➡ Les développeurs « commettent »
- ➡ Le serveur d'intégration surveille le serveur SCM (Cron)

Cas d'utilisation

Le développeur soumet une modification



Cas d'utilisation

Le chef de projet analyse le reporting



CoverageViewer

Search By: mediator

Branches Lines

Name	Branch Coverage	Uncovered
[Application]	40.07% (7,551/18,365)	3,814
com.noteflight.notification.controller	8.33% (83/996)	913
AnnotationMediator	0.00% (0/10)	10
BarlineDragMediator	0.00% (0/5)	5
BasicNotificationMediator	8.69% (6/69)	63
BeamMediator	23.07% (6/26)	20
DragMediator	0.00% (0/23)	23
MeasureDragMediator	0.00% (0/17)	17
MeasureEditMediator	20.68% (6/29)	23
MeasureHandleDragMediator	0.00% (0/15)	15
MeasureHandleMediator	35.29% (6/17)	11
ScoreMediator	66.66% (2/3)	1
SpriteDragMediator	0.00% (0/3)	3
StaffHandleMediator	40.00% (6/15)	9
TranspositionDragMediator	0.00% (0/12)	12
com.noteflight.notification.editor	32.26% (319/988)	669
EditorKeyMediator	41.70% (88/211)	123
NoteEditMediator	35.08% (40/114)	74

public function +19 handleViewEvents(view:ScoreO {
 _view = view as MeasureHandle;
 _view.addEventListener(MouseEvent.MOUSE_DOWN,
 _view.addEventListener(MeasureEditEvent.MEASURE_EDIT,
 if (_view.measure != +19->null)
 {
 _view.measure.addEventListener(NotationEditEvent.NOTE_EDIT,
 private function +0 handleMeasureEdit(e:MeasureEditEvent)
 {
 switch (e.kind)
 {
 case MeasureEditEvent.+0-0 ADD_AFTER:
 _view.context.document.undoHistory.
 _view.context.controller.insertBlankBreak;
 break;
 case MeasureEditEvent.+0-0 ADD_BEFORE:
 _view.context.document.undoHistory.
 _view.context.controller.insertBlankBreak;
 }
 }
 }
}

phpUnderControl

Project Metric Summary
Number of Build Attempts: 80
Number of Broken Builds: 15
Number of Successful Builds: 65

Breakdown of build types

Breakdown of build timeline

Unit coverage

Unit Tests

Test to Code ratio

Coding Violations

Hudson

JMeter Test

Project JMeter Test

Responding time

JMeter Trend

Percentage of errors

Hudson » JMeter Test

Back to Dashboard Status Changes Workspace Build Now Delete Project Configure JMeter trend

Build History (trend)

- #22 22-Apr-2009 18:25:34
- #18 13-Apr-2009 16:32:24
- #17 13-Apr-2009 16:32:03
- #16 13-Apr-2009 16:20:01
- #15 13-Apr-2009 16:18:39
- #14 13-Apr-2009 16:16:13
- #13 13-Apr-2009 16:15:11
- #12 13-Apr-2009 16:07:25
- #11 13-Apr-2009 16:06:10
- #10 13-Apr-2009 16:03:53

Workspace Last Successful Artifacts Recent Changes

PerfTest1-result.html

Responding time

JMeter Trend

Percentage of errors

Les technologies existantes

- ➡ Hudson, jenkins
- ➡ Gitlab CI
- ➡ Travis CI
- ➡ CruiseControl / CruiseControl.NET
- ➡ Apache Continuum
- ➡ QuickBuild (open-source: LuntBuild)
- ➡ Et beaucoup d'autres ...

Improving Your Productivity

- Continuous integration can help you go faster
 - Detect build breaks sooner
 - Report failing tests more clearly
 - Make progress more visible

Jenkins for Continuous Integration

- Jenkins – open source continuous integration server
- Jenkins (<http://jenkins-ci.org/>) is
 - Easy to install
 - Easy to use
 - Multi-technology
 - Multi-platform
 - Widely used
 - Extensible
 - Free

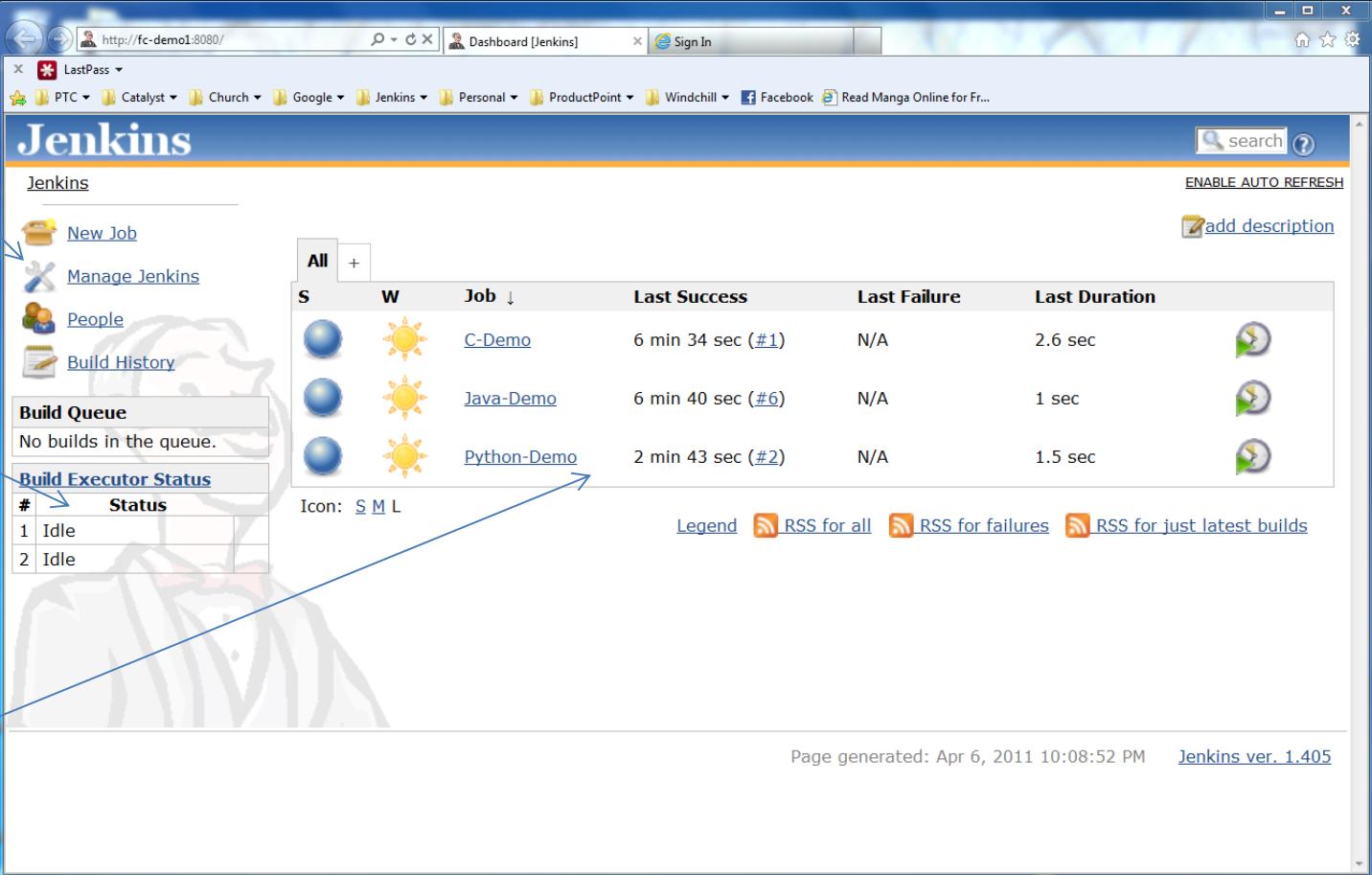


Jenkins for a Developer

- Easy to install
 - Download one file – jenkins.war
 - Run one command – java –jar jenkins.war
- Easy to use
 - Create a new job – checkout and build a small project
 - Checkin a change – watch it build
 - Create a test – watch it build and run
 - Fix a test – checkin and watch it pass
- Multi-technology
 - Build C, Java, C#, Python, Perl, SQL, etc.
 - Test with Junit, Nunit, MSTest, etc.

Jenkins User Interface

Actions



The screenshot shows the Jenkins dashboard at <http://fc-demo1:8080/>. The left sidebar contains links for New Job, Manage Jenkins, People, and Build History. The main area displays a table of build status for three jobs: C-Demo, Java-Demo, and Python-Demo. The table includes columns for Status (S), Workload (W), Job name, Last Success, Last Failure, and Last Duration. Below the table is a legend for build icons: S (blue circle), M (yellow sun), and L (green checkmark). The page footer indicates it was generated on April 6, 2011, at 10:08:52 PM, and shows Jenkins version 1.405.

#	Status	Job	Last Success	Last Failure	Last Duration
1	Idle	C-Demo	6 min 34 sec (#1)	N/A	2.6 sec
2	Idle	Java-Demo	6 min 40 sec (#6)	N/A	1 sec
		Python-Demo	2 min 43 sec (#2)	N/A	1.5 sec

Icon: S M L

Legend: RSS for all RSS for failures RSS for just latest builds

Page generated: Apr 6, 2011 10:08:52 PM Jenkins ver. 1.405

Nodes

Jobs

Jenkins Plugins - SCM

- Version Control Systems
 - Accurev
 - Bazaar
 - BitKeeper
 - ClearCase
 - Darcs
 - Dimensions
 - Git
 - Harvest
 - MKS Integrity
 - PVCS
 - StarTeam
 - Subversion
 - Team Foundation Server
 - Visual SourceSafe



Jenkins Plugins – Build & Test

- Build Tools
 - Ant
 - Maven
 - MSBuild
 - Cmake
 - Gradle
 - Grails
 - Scons
 - Groovy
- Test Frameworks
 - Junit
 - Nunit
 - MSTest
 - Selenium
 - Fitnesse

Jenkins Plugins – Analyzers

- Static Analysis
 - Checkstyle
 - CodeScanner
 - DRY
 - Crap4j
 - Findbugs
 - PMD
 - Fortify
 - Sonar
 - FXCop
- Code Coverage
 - Emma
 - Cobertura
 - Clover
 - GCC/GCOV

Jenkins Plugins – Other Tools

- Notification
 - Twitter
 - Campfire
 - Google Calendar
 - IM
 - IRC
 - Lava Lamp
 - Sounds
 - Speak
- Authorization
 - Active Directory
 - LDAP
- Virtual Machines
 - Amazon EC2
 - VMWare
 - VirtualBox
 - Xen
 - Libvirt

What about GitLab CI?



CI CD

Our journey ;)

- Documentation
- (software) Logging
- Testing and Static Analysis
- Refactoring
- Build/configuration/release automation
- Continuous integration
- *Continuous delivery*
- *Continuous deployment*
- *(runtime) monitoring*
- *Continuous improvement*