

# Qualité du logiciel ?

Merci à tous ceux qui ont rendu leurs cours et exposés  
disponibles sur le web & dans les livres,  
voir Biblio.

M. Blay-Fornarino  
[blay@unice.fr](mailto:blay@unice.fr),  
<http://users.polytech.unice.fr/~blay/>  
IUT Département Informatique 2<sup>e</sup> année

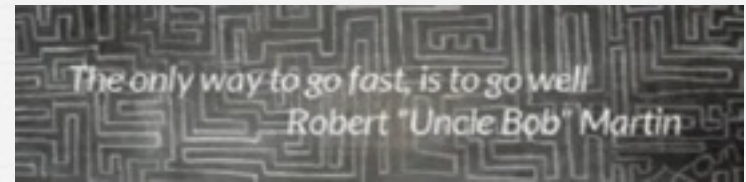
# Bibliographie

- ❖ Reflexion on Software Quality and Maintenance, Alexandre Bergel, Chili
- ❖ Cours de Production Du Logiciel, La qualité logicielle, Thierry Milan, Toulouse
- ❖ Yann-Gaël Guéhéneuc cours , Université de Montréal, <http://www-etud.iro.umontreal.ca/~ptidej/yann-gael/Work/Publications/>

# WHAT EVERY GOOD SYSTEM NEEDS



ETERNAL TYPOS



# Le logiciel ...

S'il vous fallait préparer à l'avance tous les ordres à donner à des individus totalement stupides, mais absolument obéissants, pour qu'ils réalisent une tâche complexe, vous diriez que c'est là un problème de management nouveau : les gens ne sont, en général, ni stupides, ni obéissants, ce qui aide à faire face à l'imprévu. Si, en plus, la moindre erreur provoque une catastrophe, vous diriez que la réussite d'un tel plan d'action tient du miracle. Vous venez pourtant de définir ce qu'est la fabrication d'un logiciel. Pour que la réussite ne tienne pas que du miracle, il faut donc une rigueur de fer, pas mal de culture, et de bons outils tels que ceux qui apparaissent aujourd'hui, ce qui n'est malheureusement pas encore assez connu.

## **LOGICIELS : COMMENT CHASSER LES BUGS ?**

par

**Gérard BERRY** 1996

Directeur de Centre de Mathématiques Appliquées  
de l'École des mines de Paris

membre de l'Académie des sciences française (depuis 2002),  
de l'Académie des technologies (depuis 2005),  
et de l'Academia Europaea (depuis 1998)

# Deux points de vue sur la qualité du code

Est-ce que le logiciel met en œuvre correctement les exigences?  
Est-il facile à utiliser? Plante-il?



Comment est organisé le code?  
Où est implémentée cette fonctionnalité?

```
void CruiseControl_init(i_C_CruiseControl * _C_)
{
    CruiseSpeedMgt_init(&i_C->_00_CruiseSpeedMgt);
    CruiseStateMgt_init(&i_C->_01_CB_CruiseStateMgt);
    i_C->_H_conduct_01 = true;
    ThrottleCmd_init(&i_C->_C4_ThrottleCmd);
    i_C->_H_init = true;
}

/* ===== */
/* MAIN NOCE */
/* ===== */

void CruiseControl(i_C_CruiseControl
{
    bool BrakePressed;
    bool AcceleratorPressed;
    bool SpeedOutOffLimits;
    bool LL19;
    /*code for node CruiseControl
    /* call to node not expanded DetectPedalsPressed
    i_C->_0n_DetectPedalsPressed_01_Accelerator
    i_C->_0n_DetectPedalsPressed_01_Accelerator
    DetectPedalsPressed(&i_C->_0n_DetectPedalsPressed
    BrakePressed = i_C->_0n_DetectPedalsPressed_00_01
    AcceleratorPressed =
    i_C->_0n_DetectPedalsPressed_01_AcceleratorPr
    /* call to node not expanded DetectSpeedLimits */
    i_C->_0n_DetectSpeedLimits_10_speed1 = i_C->_10
    DetectSpeedLimits(&i_C->_0n_DetectSpeedLimits);
    SpeedOutOffLimits = i_C->_0n_DetectSpeedLimits_00
    /* call to node not expanded CruiseStateMgt */
    i_C->_10_P1_CruiseStateMgt_10_01_01_01 = BrakePr
```



External (user's):  
*what?*



Internal  
(programmer's): *how?*



# User's frustration



- "after clicking all options and navigating all menus I still cannot find out how to achieve <this>"
- "why do I get this error message? I did nothing wrong – stupid software!"

# Programmer's frustration

```
void CruiseControl_init(_C_CruiseControl * _C_)
{
    CruiseSpeedMgt_init(&_C->_C0_CruiseSpeedMgt);
    CruiseStateMgt_init(&_C->_C0_CruiseStateMgt);
    !_C->_C0_contact_00 = true;
    ThrottleCmd_init(&_C->_C0_ThrottleCmd);
    !_C->_C0_init = true;
}

/* ===== */
/* HAIN NODE */
/* ===== */

void CruiseControl(_C_CruiseControl * _C_)
{
    bool BrakePressed;
    bool AcceleratorPressed;
    bool SpeedOutOffLimits;
    bool _LBP;
    /*code for node CruiseControl */
    /* call to node not expanded DetectPedalsPressed */
    !_C->_C0_DetectPedalsPressed_10_Brake = !_C->_C0_1;
    !_C->_C0_DetectPedalsPressed_11_Accelerator = !_C->_C0_1;
    DetectPedalsPressed(&_C->_C0_DetectPedalsPressed);
    BrakePressed = !_C->_C0_DetectPedalsPressed_00_Br;
    AcceleratorPressed =
        (!_C->_C0_DetectPedalsPressed_01_AcceleratorPr);
    /* call to node not expanded DetectSpeedLimits */
    !_C->_C0_DetectSpeedLimits_10_Speed = !_C->_C0_10;
    DetectSpeedLimits(&_C->_C0_DetectSpeedLimits);
    SpeedOutOffLimits = !_C->_C0_DetectSpeedLimits_00;
    /* call to node not expanded CruiseStateMgt */
    !_C->_C0_CruiseStateMgt_10_BrakePressed = BrakePr
```



- "I have to add this function, but it's impossible to find a single place where the addition should go in the intricate code structure"
- "The program produces the wrong output, but after several hours browsing the code I cannot identify which statements are guilty"

# Programmer's frustration



- Pour un développeur, la qualité d'une application réside dans la capacité de son code à être lu, compris et repris par d'autres développeurs. Un code de mauvaise qualité sera plus difficile à maintenir et source d'anomalies.
- L'incompréhension est, en effet, une cause d'erreurs non négligeable mais peut aussi devenir une surcharge importante pour le projet.



# Qualité ?

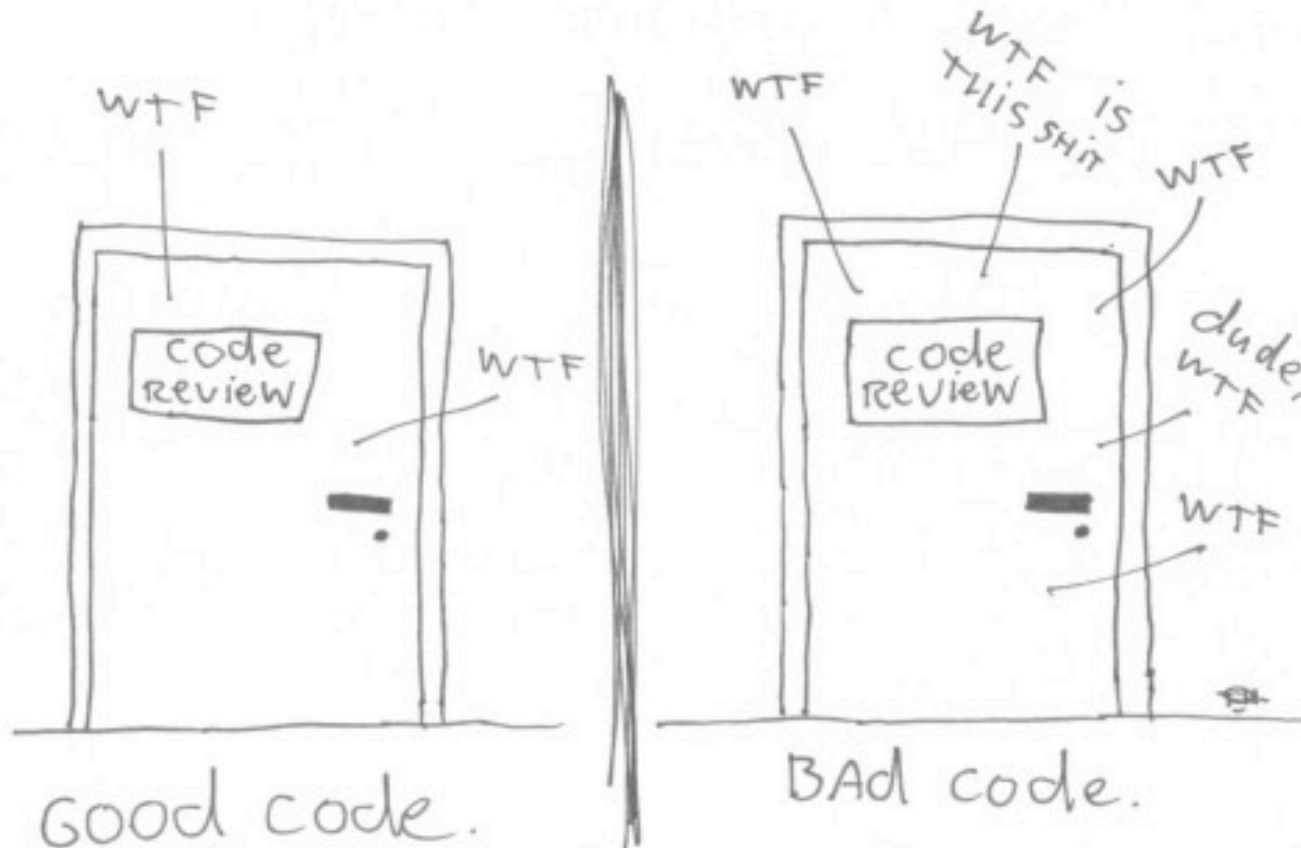
- Exemple
- On compare deux logiciels
  - Le premier a encore 10 erreurs résiduelles
  - Le second n'a que trois erreurs résiduelles
- Est-ce que le second est de meilleure qualité que le premier?

- La qualité pour n'importe quel processus est difficile à définir
- Il faut identifier
  - Des facteurs
  - Des critères
  - Des métriques

# Evaluation de la Qualité ?

# Evaluation

The ONLY VALID MEASUREMENT  
OF Code QUALITY: WTFs/minute



(c) 2008 Focus Shift/OSNews/Thom Holwerda - <http://www.osnews.com/comics>



# Intuitivement, la qualité ...

à partir du cours d'Alexandre Bergel



# QUALITÉ

```
/* Hello World program */  
  
#include<stdio.h>  
  
int main()  
{  
    printf("Hello World");  
  
}
```

**no return  
statement  
in the  
function**

```
/* Hello World program */  
  
#include<stdio.h>  
  
→ int main()  
  {  
    printf("Hello World");  
  
  }
```

**no return  
statement  
in the  
function**

```
/* Hello World program */  
  
#include<stdio.h>  
  
→ int main()  
  {  
    printf("Hello World");  
  }
```

9 lines of code  
7 seconds

In Swing:

JComponent.java

```
protected String getBorderTitle(Border b) {  
    String s;  
    if (b instanceof TitledBorder) {  
        return ((TitledBorder) b).getTitle();  
    } else if (b instanceof CompoundBorder) {  
        s = getBorderTitle(((CompoundBorder)  
            b).getInsideBorder());  
        if (s == null) {  
            s = getBorderTitle(((CompoundBorder)  
                b).getOutsideBorder());  
        }  
    }  
    return s;  
    } else {  
        return null;  
    }  
}
```

...

In Swing:

JComponent.java

```
protected String getBorderTitle(Border b) {
    String s;
    if (b instanceof TitledBorder) {
        return ((TitledBorder) b).getTitle();
    } else if (b instanceof CompoundBorder) {
        s = getBorderTitle(((CompoundBorder)
            b).getInsideBorder());
        if (s == null) {
            s = getBorderTitle(((CompoundBorder)
                b).getOutsideBorder());
        }
    }
    return s;
} else {
    return null;
}
```

...

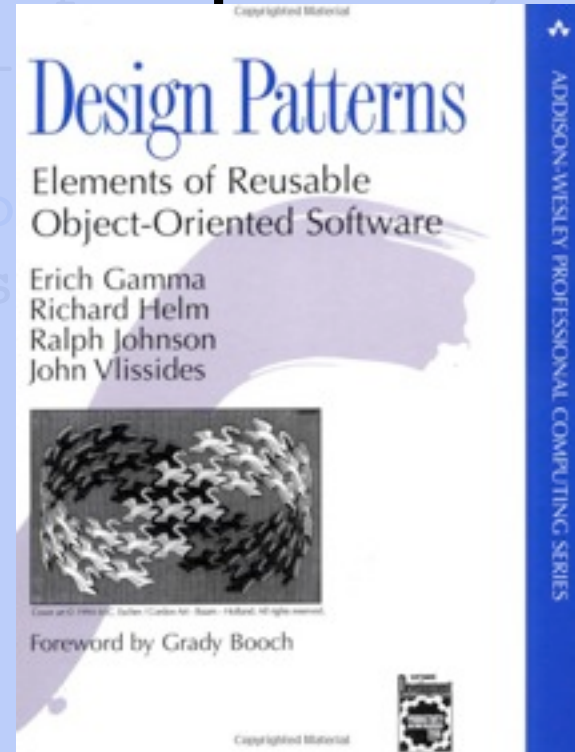


In Swing:

JComponent.java

```
protected String getBorderTitle(Border b) {  
    String s;  
    if (b instanceof TitledBorder) {  
        return ((TitledBorder) b).getTitle();  
    } else if (b instanceof ComponentBorder) {  
        s = getBorderTitle(((ComponentBorder) b).getComponent());  
    }  
    if (s == null) {  
        s = getBorderTitle(((ComponentBorder) b).getOwner());  
    }  
    return s;  
} else {  
    return null;  
}
```

## Need Double Dispatch



In Swing:

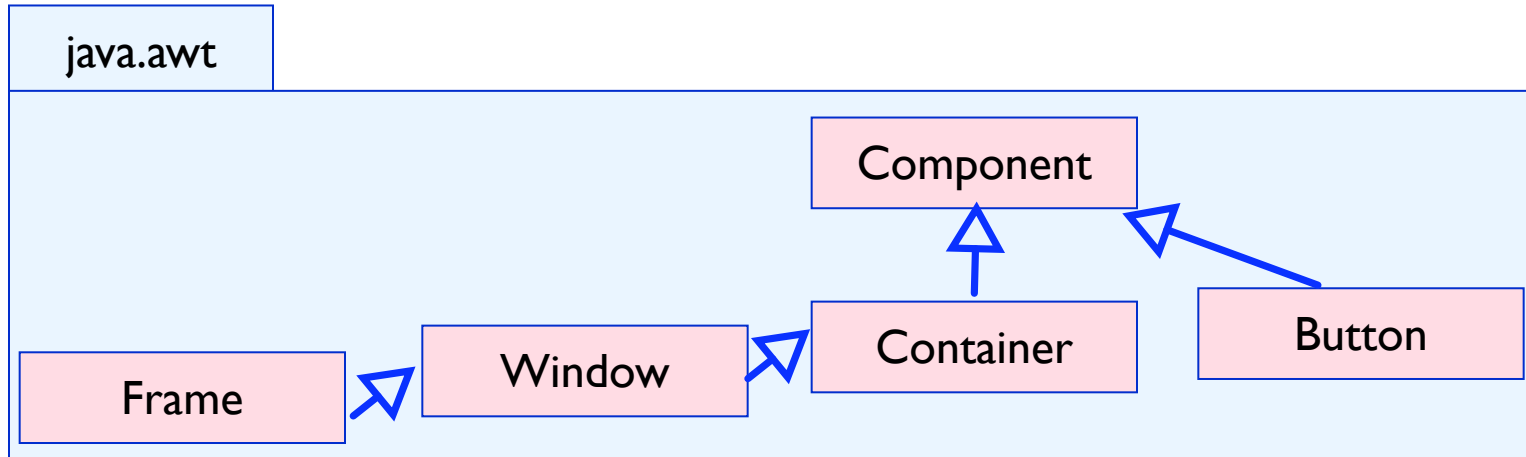
JComponent.java

```
protected String getBorderTitle(Border b) {  
    String s;  
    if (b instanceof TitledBorder) {  
        return ((TitledBorder) b).getTitle();  
    } else if (b instanceof CompoundBorder) {  
        s = getBorderTitle(((CompoundBorder)  
            b).getInsideBorder());  
        if (s == null) {  
            s = getBorderTitle(((CompoundBorder)  
                b).getOutsideBorder());  
        }  
    }  
    return s;  
} else {  
    return null;  
}
```

5471 lines of code  
12 minutes

...

# Presentation of AWT

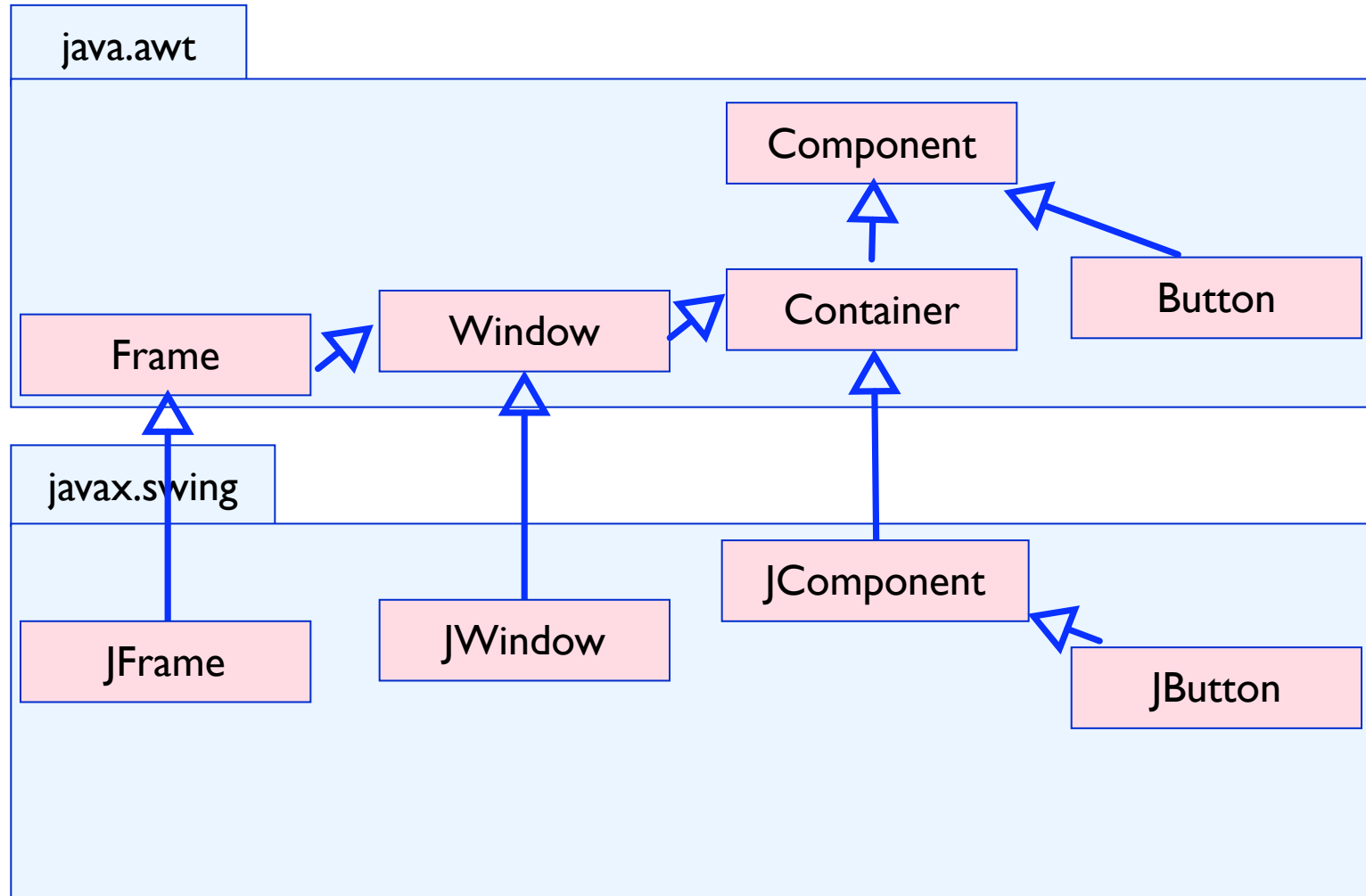


In the AWT framework:

Widgets are components (i.e., inherit from `Component`)

A frame is a window (Frame is a subclass of `Window`)

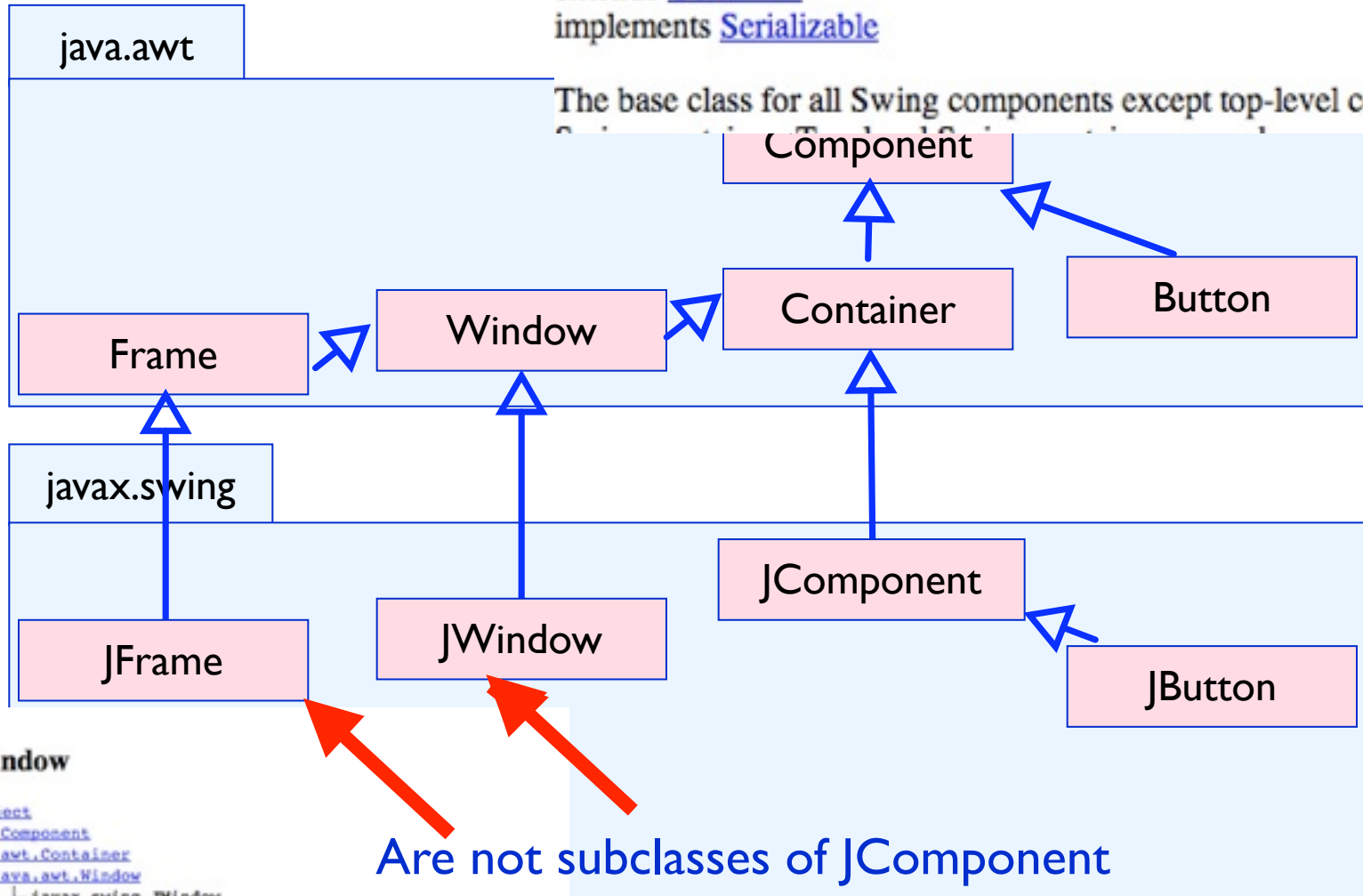
# Swing at the top of AWT



# Problem #1: Brocken Inheritance

```
public abstract class JComponent  
extends Container  
implements Serializable
```

The base class for all Swing components except top-level containers.

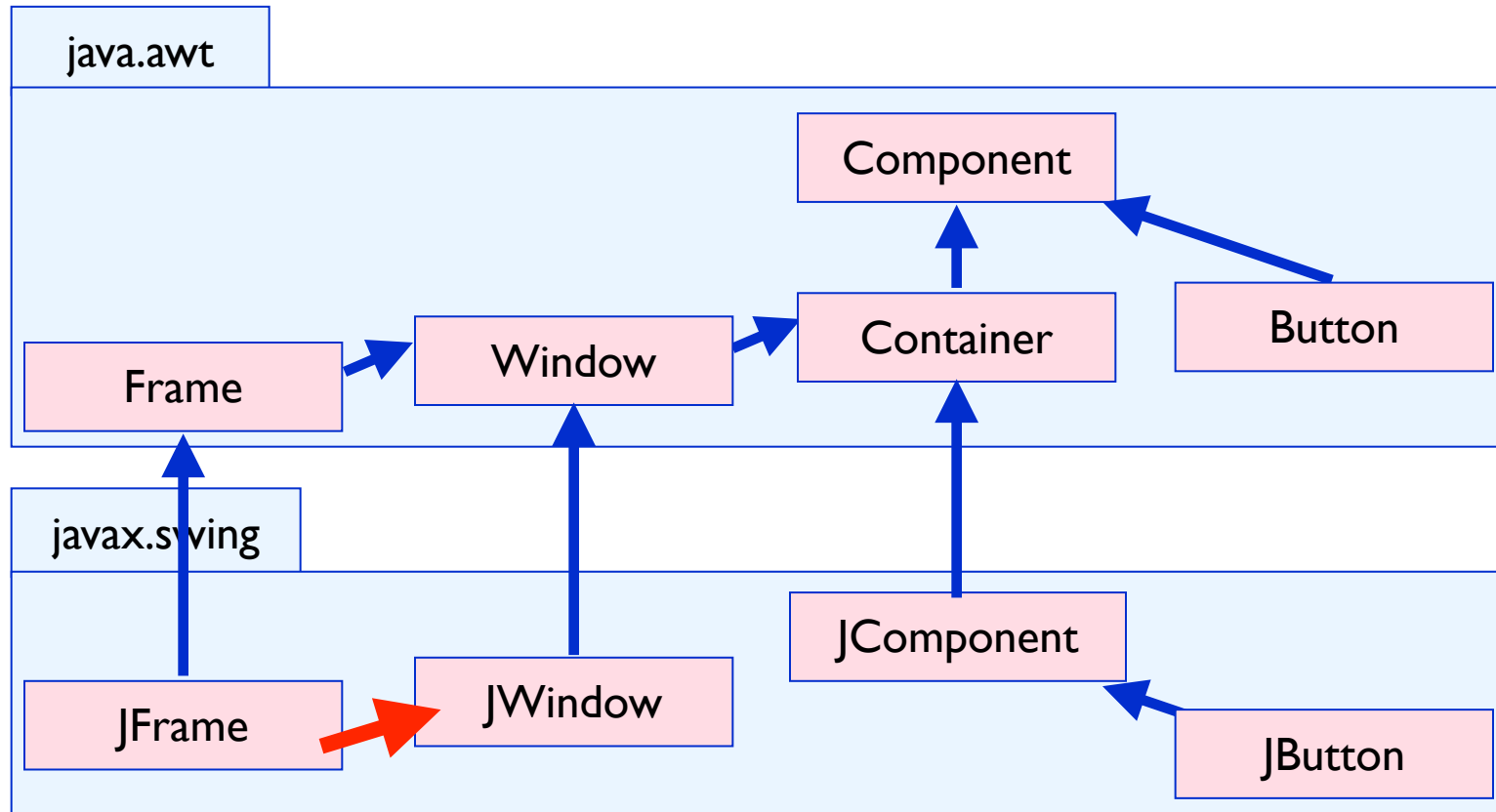


```
javax.swing  
Class JWindow  
java.lang.Object  
├─ java.awt.Component  
├─ java.awt.Container  
├─ java.awt.Window  
└─ javax.swing.JWindow
```

All Implemented Interfaces:  
Accessible, ImageObserver, MenuContainer, RootPaneContainer, Serializable

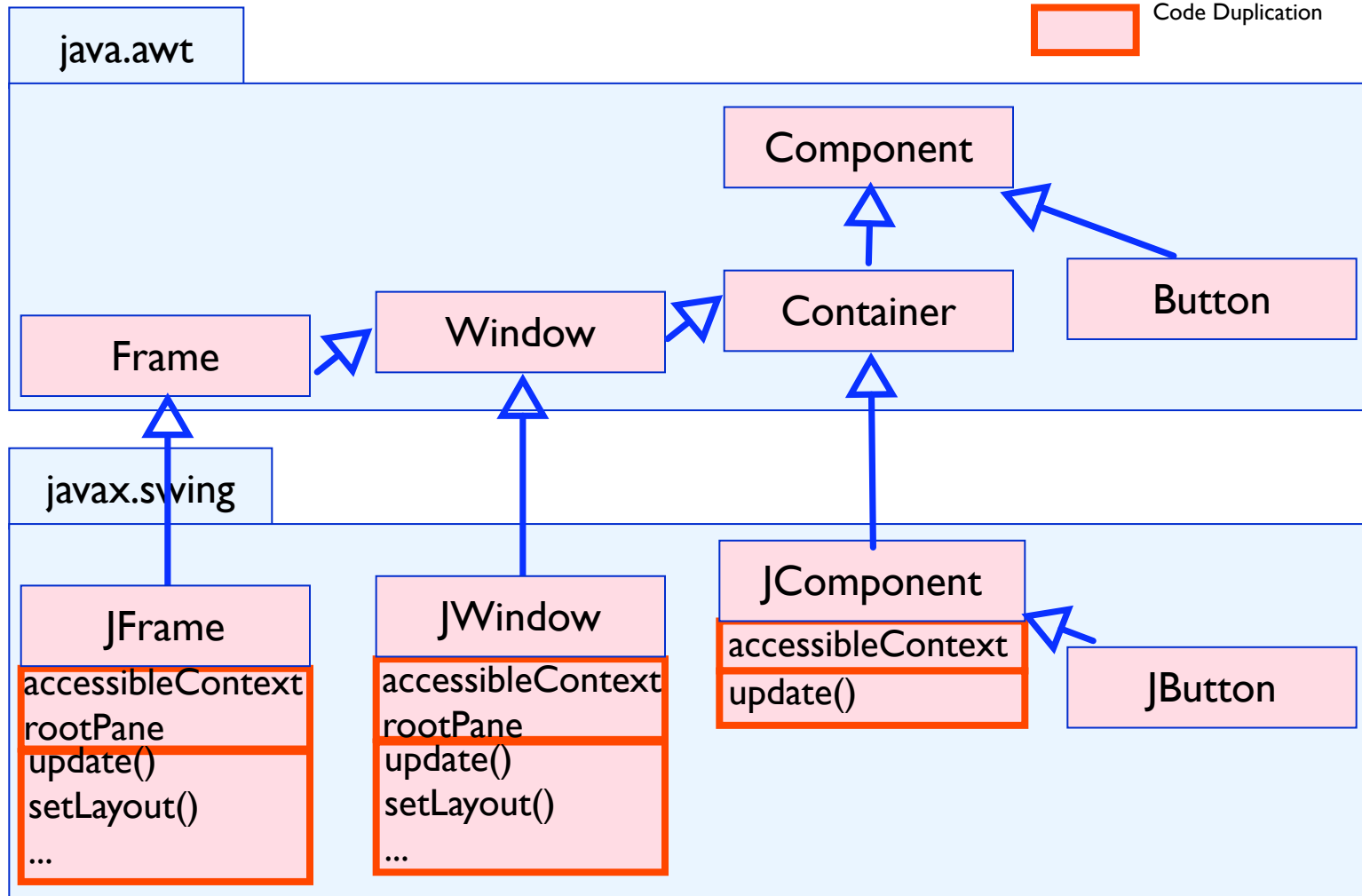


# Problem #1: Brocken Inheritance



Missing inheritance link between JFrame and JWindow

# Problem #2: Code Duplication



# Problem #3: Explicit Type Checks and Casts

---

```
public class Container extends Component {  
    Component components[] = new Component [0];  
    public Component add (Component comp) {...}  
}
```

```
public class JComponent extends Container {  
    public void paintChildren (Graphics g) {  
        for (; i>=0 ; i--) {  
            Component comp = GetComponent (i);  
            isJComponent = (comp instanceof JComponent);  
            ...  
            (JComponent comp).getBounds();  
        }  
    }  
}}
```

# Supporting Unanticipated Changes

---

AWT couldn't *be enhanced* without risk of *breaking* existing code

Swing is, therefore, *built on the top* of AWT using *subclassing*

As a result, *Swing is a big mess internally!*

# Why do we care to have a messy Swing ?

---

Swing appeared in 1998, and *has not evolved since!*

*Swing is too heavy* to be ported to PDA, cellphones, ...

SWT\* is *becoming* a *new standard*.

*Either a system evolves, or it is dead.* [Lehmans74]

\***SWT** is an [open source](#) widget toolkit for Java designed to provide efficient, portable access to the user-interface facilities of the operating systems on which it is implemented.



J2EE  
3,430,663 lines of code  
? hours



Linux 2.6.28  
7,106,111 lines of code  
? hours



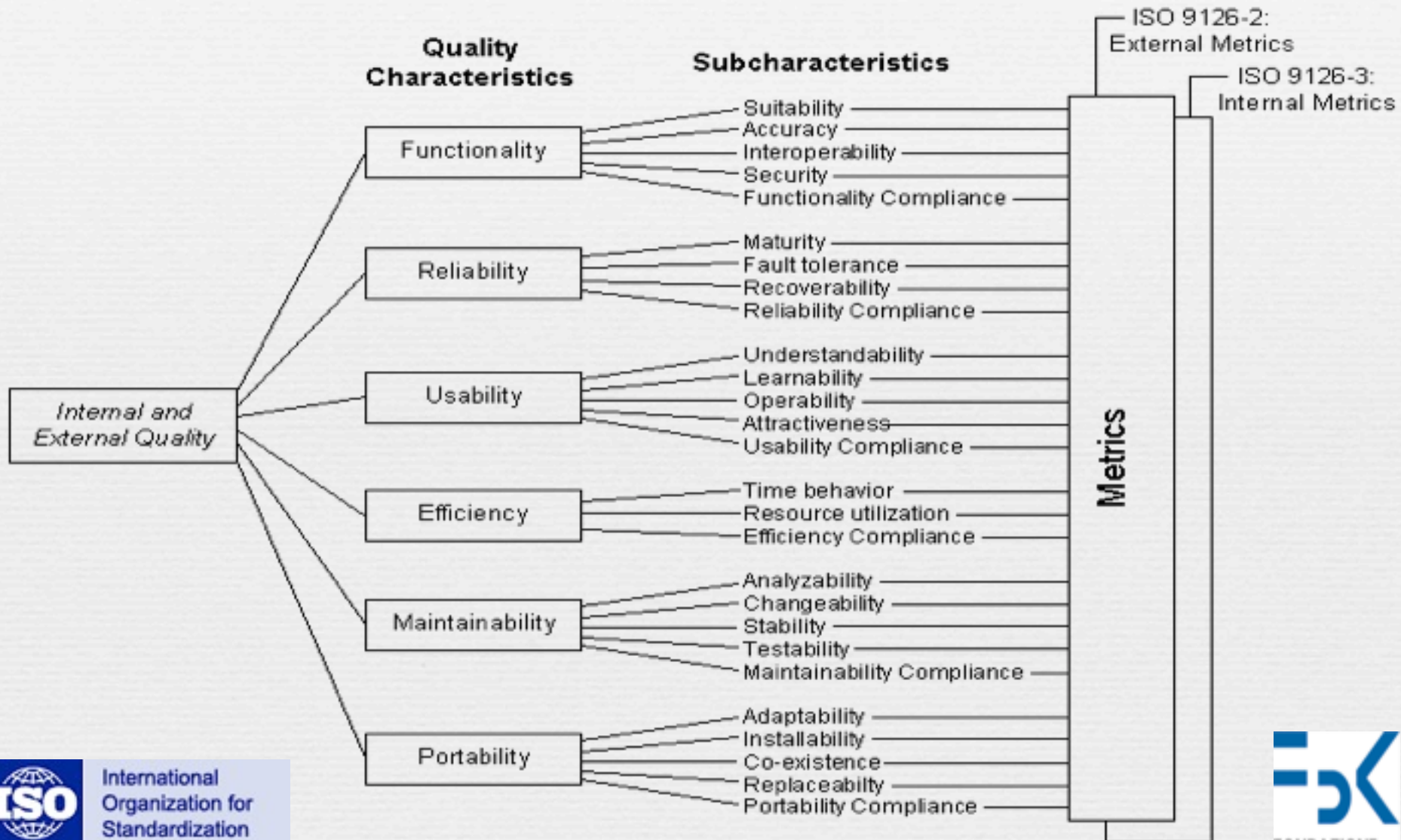
# Qu'est-ce que la qualité ?

*appréciation globale d'un logiciel,  
basée sur de nombreux indicateurs*

- ☛ ISO/CEI 9126 : **vocabulaire** visant à classifier l'ensemble des attributs d'un logiciel relevant de la qualité

ISO/IEC 9126-1,<sup>[1]</sup> classifies [software quality](#) in a structured set of characteristics and sub-characteristics.

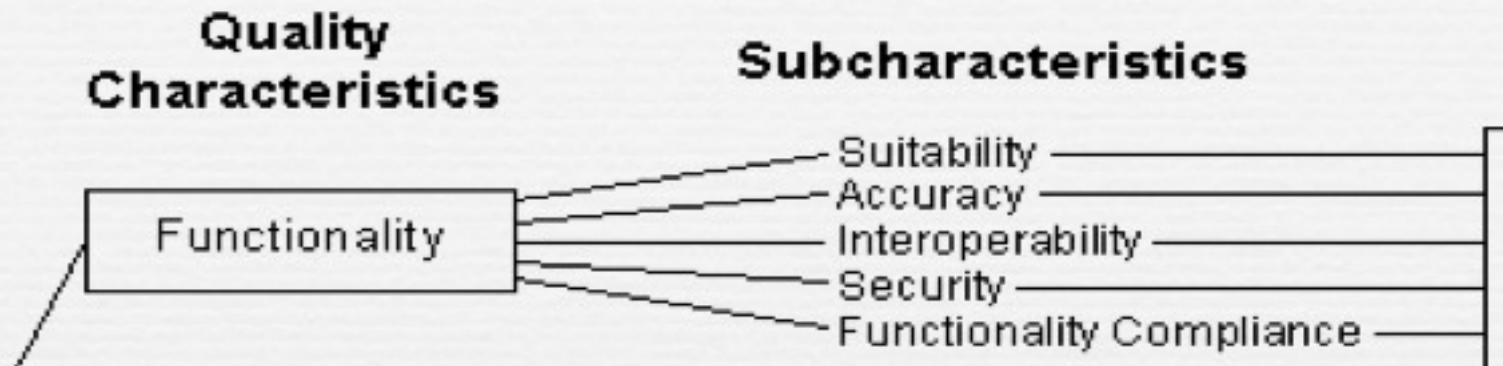
# Quality models (ISO 9126)



# ISO/CEI 9126

→ Capacité fonctionnelle (*functionality*)

✓ Répond-il aux besoins de ses utilisateurs ?



# Capacité fonctionnelle

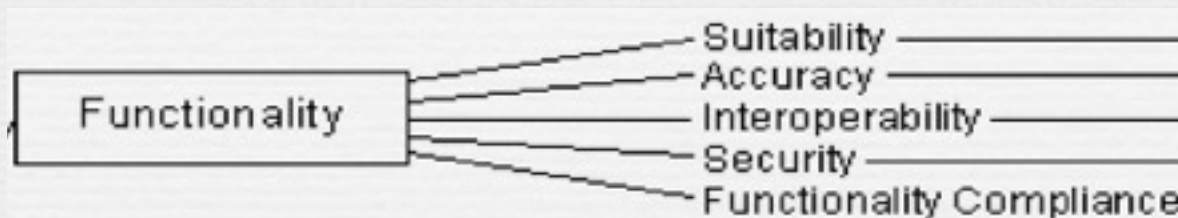


## ☞ Définition

– Ensemble d'attributs portant sur l'existence de fonctions et leurs propriétés; les fonctions sont celles qui satisfont aux besoins exprimés ou implicites

## ➔ Sous-caractéristiques

- **Aptitude** : présence et adéquation d'une série de fonctions pour les tâches données
- **Exactitude** : résultats ou effets justes ou convenus
- **Interopérabilité** : interactions avec d'autres systèmes
- **Sécurité** : accès non autorisé (accidentel ou délibéré) aux programmes et données



# ISO/CEI 9126

## → Capacité fonctionnelle (*functionality*)

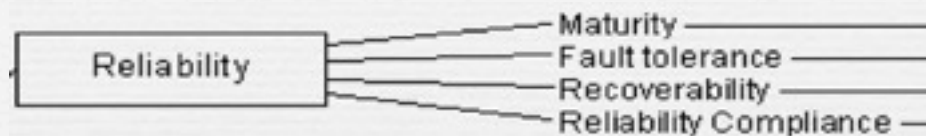
✓ Répond-il aux besoins de ses utilisateurs ?

## → Fiabilité (*reliability*)

✓ Est-il en mesure d'assurer un niveau de qualité de service suffisant pour satisfaire les besoins opérationnels de ses utilisateurs ?



# Fiabilité



## ☞ Définition

– Ensemble d'attributs portant sur l'aptitude du logiciel à maintenir son niveau de service dans des conditions précises et pendant une période déterminée

## ➡ Sous-caractéristiques

- **Maturité** : fréquence des défaillances dues à des défauts
- **Tolérance aux fautes** : aptitude à maintenir un niveau de service donné en cas de défaut ou d'attaque
- **Possibilité de récupération** : capacité à rétablir son niveau de service et de restaurer les données directement affectées en cas de défaillance ; temps et effort nécessaire pour le faire

# ISO/CEI 9126

## → Capacité fonctionnelle (*functionality*)

✓ Répond-il aux besoins de ses utilisateurs ?

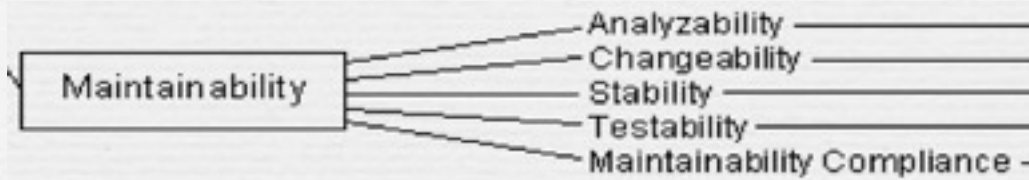
## → Fiabilité (*reliability*)

✓ Est-il en mesure d'assurer un niveau de qualité de service suffisant pour satisfaire les besoins opérationnels de ses utilisateurs ?

## → Maintenabilité (*maintainability*)

✓ Est-il facile d'adapter le logiciel à de nouveaux besoins ou à de nouvelles contraintes ?

# Maintenabilité



## ☞ Définition

– Ensemble d'attributs portant sur l'effort nécessaire pour faire des modifications données

## ➡ Sous-caractéristiques

– **Facilité d'analyse** : effort nécessaire pour diagnostiquer les déficiences et leurs causes ou pour identifier les parties à modifier

– **Facilité de modification** : effort nécessaire pour modifier, remédier aux défauts ou adapter à l'environnement

– **Stabilité** : risque des effets inattendus des modifications

– **Facilité de test** : effort pour valider le logiciel modifié

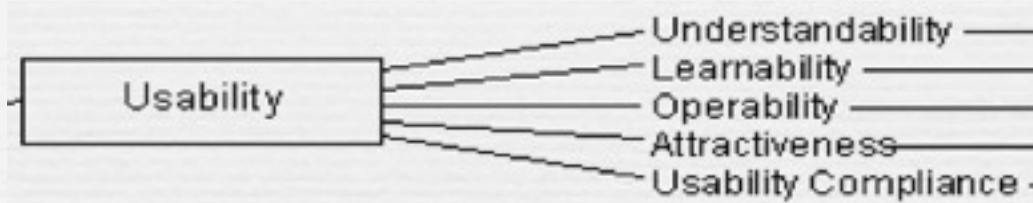


# ISO/CEI 9126

→ Facilité d'utilisation (*usability*)

✓ Peut-il être utilisé à moindre effort ?

# Facilité d'utilisation



## ☛ Définition

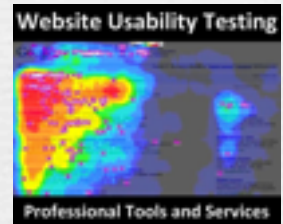
– Ensemble d'attributs portant sur l'effort nécessaire pour l'utilisation et l'évaluation individuelle de cette utilisation par un ensemble défini ou implicite d'utilisateurs

## ➡ Sous-caractéristiques

– **Facilité de compréhension** : effort de l'utilisateur pour comprendre la logique et la mise en œuvre

– **Facilité d'apprentissage** : effort de l'utilisateur pour apprendre son utilisation

– **Facilité d'exploitation** : effort que doit faire l'utilisateur pour exploiter et contrôler l'exploitation du logiciel



# ISO/CEI 9126

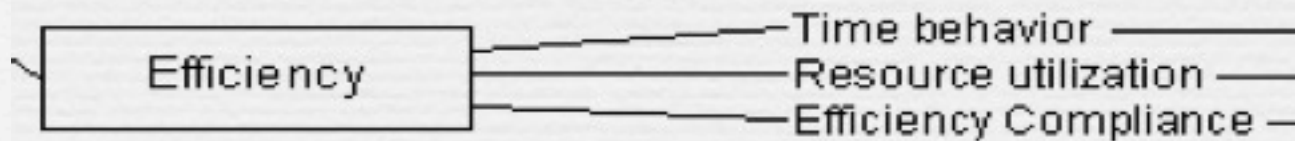
## → Facilité d'utilisation (*usability*)

✓ Peut-il être utilisé à moindre effort ?

## → Rendement / Scalabilité (*efficiency*)

✓ Les ressources matérielles nécessaires à l'exécution du logiciel sont-elles en rapport avec sa rentabilité ?

# Rendement



## ☞ Définition

– Ensemble d'attributs portant sur le rapport existant entre le niveau de service d'un logiciel et la quantité de ressources utilisées, dans des conditions déterminées

## ➡ Sous-caractéristiques

– **Temps** : temps de réponses et de traitement ; débits lors de l'exécution de sa fonction

– **Ressources** : quantité de ressources utilisées ; durée de leur utilisation par fonction

# ISO/CEI 9126

## → Facilité d'utilisation (*usability*)

✓ Peut-il être utilisé à moindre effort ?

## → Rendement / Scalabilité (*efficiency*)

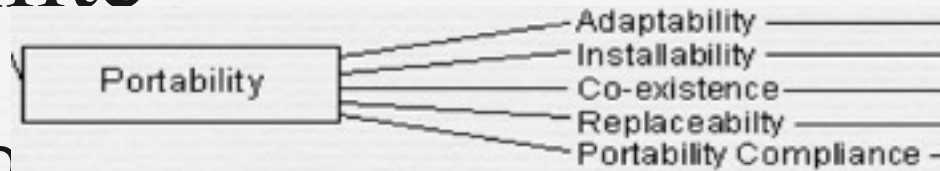
✓ Les ressources matérielles nécessaires à l'exécution du logiciel sont-elles en rapport avec sa rentabilité ?

## → Portabilité (*portability*)

✓ Peut-il être transféré facilement d'une plate-forme ou d'un environnement à un autre ?



# Portabilité



## ➤ Définition

– Ensemble d'attributs portant sur l'aptitude du logiciel à être transféré d'un environnement à un autre

## ➔ Sous-caractéristiques

- **Facilité d'adaptation** : possibilité d'adaptation à différents environnements donnés sans que l'on ait recours à d'autres actions ou moyens que ceux prévus à cet effet par le logiciel.
- **Facilité d'installation** : effort nécessaire pour installer le logiciel dans un environnement donné.
- **Conformité aux règles de portabilité** : conformité aux normes et aux conventions ayant trait à la portabilité.
- **Interchangeabilité** : possibilité et effort d'utilisation du logiciel à la place d'un autre logiciel donné dans le même environnement.

# Qualité : point de vue Utilisateur

## II Qualité du logiciel (2)



### Intégrité/sécurité

Aptitude du logiciel à se protéger contre des accès non autorisés

### Correction/validité

Aptitude à répondre aux besoins et à remplir les fonctions définies dans le cahier des charges

### Convivialité

#### Facilité

- d'apprentissage et d'utilisation,
- de préparation des données,
- de correction des erreurs d'utilisation,
- d'interprétation des retours

### Robustesse/fiabilité

Aptitude à fonctionner dans des conditions non prévues dans le cahier des charges, ou anormales

### Efficacité

Utilisation optimale des ressources matérielles (processeur, mémoire, réseau, ...)

### Extensibilité

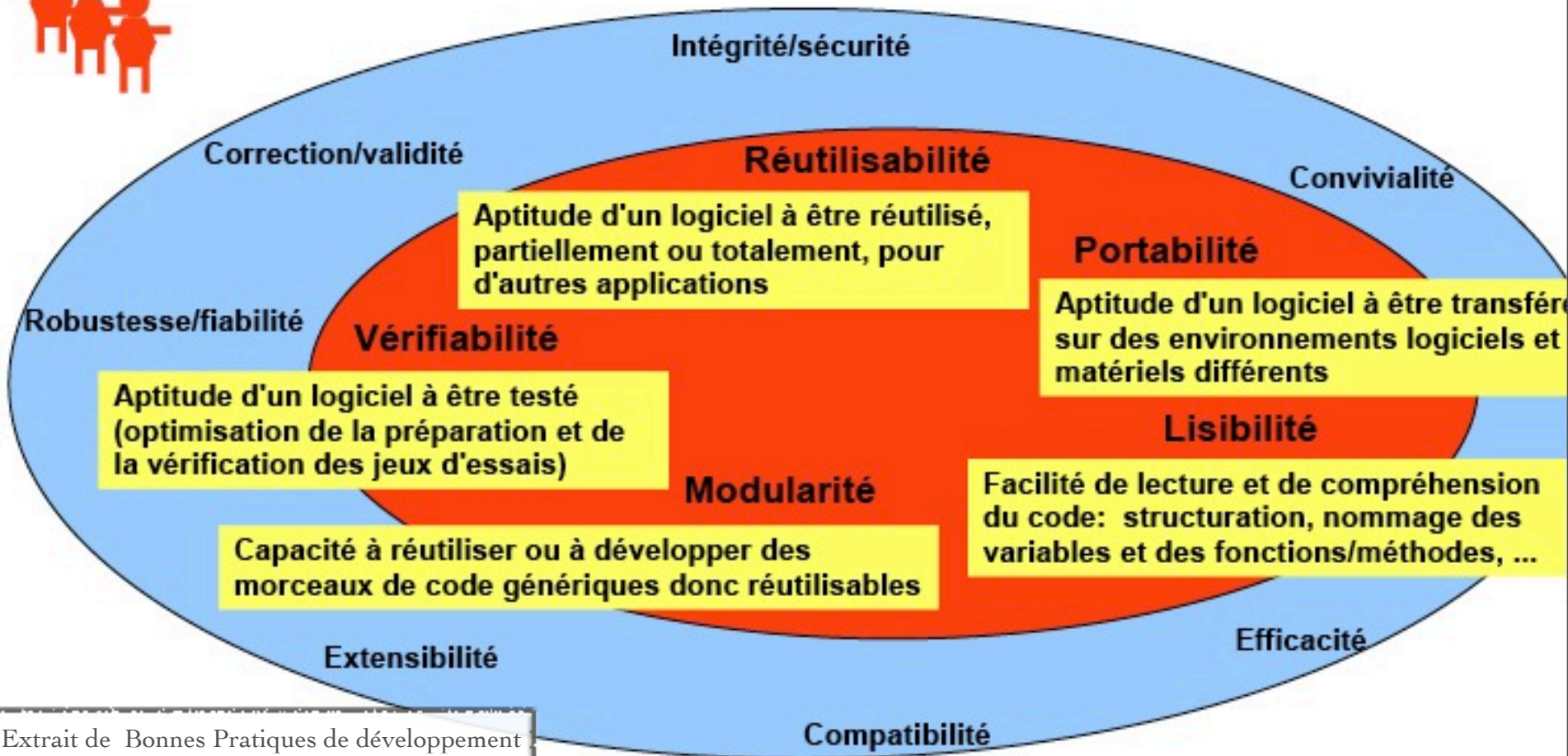
Facilité d'ajout de nouvelles fonctionnalités

### Compatibilité

Facilité de combinaison de ce logiciel avec d'autres

# Qualité : point de vue Concepteur

## II Qualité du logiciel (3)



Extrait de Bonnes Pratiques de développement  
ENVOL\_2010: 27 Septembre  
Véronique Baudin



# Comment assurer la qualité ?

## → Les tests automatisés

→ *Capacité fonctionnelle*  $\leftrightarrow$   
*Rendement*

→ Tests fonctionnels

→ Tests de charge

→ *Fiabilité*

→ Tests unitaires

→ Tests d'intégration

→ *Maintenabilité*

→ Tests de non régression

## → Les métriques, les standards de codage, ...

→ *Fiabilité*

→ *Maintenabilité*

## → & Le refactoring

→ *Maintenabilité*

## → La méthodologie

## → Normes

→ **CMMi** : Modèle de référence, ensemble structuré de bonnes pratiques, destiné à appréhender, évaluer et améliorer les activités des entreprises d'ingénierie

→ **ISO/CEI 9126**

# Mesurer la qualité logicielle

*"You can't control what you can't measure." (Tom DeMarco)*

# Qualité interne du code

- **Metriques:** couplage, cohésion ...
- **Code smells** et patterns
- **Conventions de codage:** règles de codage, de style, etc.

# Mesurer la qualité : Métriques

Approche quantitative de  
la mesure de la qualité

<http://www-igm.univ-mlv.fr/~dr/XPOSE2008/Mesure%20de%20la%20qualite%20du%20code%20source%20-%20Algorithmes%20et%20outils/files/mesure-qualite-code.pdf>

# Métrique Logicielle

- ➔ **Métrique** : mesure d'une propriété d'un logiciel (par exemple le nombre de lignes de codes)
- ➔ Approche quantitative : extraire une mesure de la qualité d'un logiciel à partir de l'analyse statistique du code source.
  - Avantage : simplicité de mise en oeuvre.
  - Principal problème : il faut trouver des **indicateurs significatifs** et les algorithmes correspondants.



# Exemples de métriques

- Nombre de Lignes de codes
  - Nombre de méthodes par classe
  - Couplage afférent/efférent
  - Niveau d'abstraction
  - Instabilité
  - ....
- Pas de métrique “absolue” : la pertinence de chaque métrique dépend du projet et surtout de l'interprétation qui en est faite.

# Tests statiques

(plugin Metrics : <http://metrics.sourceforge.net>)

- Number of Static Methods
- Total Lines of Code
- Afferent Coupling
- Normalized Distance
- Number of Classes
- Specialization Index
- Instability
- Number of Attributes
- Number of Packages
- New Methods Lines of Code
- Weighted methods per Class
- Number of Overridden Methods
- Number of Static Attributes
- Nested Block Depth
- Number of Methods
- Lack of Cohesion of Methods
- McCabe Cyclomatic Complexity
- Number of Parameters
- Abstractness
- Number of Interfaces
- Efferent Coupling
- Number of Children
- Depth of Inheritance Tree

| Metric   | Total        | Mean         | Std. Dev. | Maximum   | Resource causing Maximum                              |
|--|--------------|--------------|-----------|-----------|---|
| ▶ Number of Static Methods (avg/max per type)      | 3            | 0,188        | 0,527     | 2         | /org.eclipse.emf.examples.library/src/org             |
| ▶ Total Lines of Code                              | 3128         |              |           |           |   |
| ▶ Afferent Coupling (avg/max per packageFragm)     |              | 12           | 14,855    | 33        | /org.eclipse.emf.examples.library/src/org             |
| ▶ Normalized Distance (avg/max per packageFra)     |              | 0,259        | 0,073     | 0,333     | /org.eclipse.emf.examples.library/src/org             |
| ▶ Number of Classes (avg/max per packageFrag)      | 16           | 5,333        | 6,182     | 14        | /org.eclipse.emf.examples.library/src/org             |
| ▶ Specialization Index (avg/max per type)          |              | 0,453        | 0,339     | 1,4       | /org.eclipse.emf.examples.library/src/org             |
| ▶ Instability (avg/max per packageFragment)        |              | 0,627        | 0,22      | 0,875     | /org.eclipse.emf.examples.library/src/org             |
| ▶ Number of Attributes (avg/max per type)          | 44           | 2,75         | 3,961     | 17        | /org.eclipse.emf.examples.library/src/org             |
| ▶ Number of Packages                               | 3            |              |           |           |   |
| ▶ Method Lines of Code (avg/max per method)        | 1500         | 5,792        | 10,767    | 127       | /org.eclipse.emf.examples.library/src/org             |
| ▶ Weighted methods per Class (avg/max per typ)     | 583          | 36,438       | 22,671    | 96        | /org.eclipse.emf.examples.library/src/org             |
| ▶ Number of Overridden Methods (avg/max per        | 17           | 1,062        | 0,658     | 3         | /org.eclipse.emf.examples.library/src/org             |
| ▶ Number of Static Attributes (avg/max per type)   | 21           | 1,312        | 1,31      | 4         | /org.eclipse.emf.examples.library/src/org             |
| ▶ Nested Block Depth (avg/max per method)          |              | 1,139        | 0,469     | 4         | /org.eclipse.emf.examples.library/src/org             |
| ▶ Number of Methods (avg/max per type)             | 256          | 16           | 10,137    | 51        | /org.eclipse.emf.examples.library/src/org             |
| ▶ Lack of Cohesion of Methods (avg/max per typ)    |              | 0,301        | 0,327     | 0,924     | /org.eclipse.emf.examples.library/src/org             |
| ▶ <b>McCabe Cyclomatic Complexity (avg/max per</b> | <b>2,251</b> | <b>3,717</b> | <b>54</b> | <b>54</b> | <b>/org.eclipse.emf.examples.library/src/org</b>      |
| ▶ Number of Parameters (avg/max per method)        |              | 0,676        | 0,948     | 3         | /org.eclipse.emf.examples.library/src/org             |
| ▶ Abstractness (avg/max per packageFragment)       |              | 0,41         | 0,395     | 0,944     | /org.eclipse.emf.examples.library/src/org             |
| ▶ Number of Interfaces (avg/max per packageFra)    | 16           | 5,333        | 7,542     | 16        | /org.eclipse.emf.examples.library/src/org             |
| ▶ Efferent Coupling (avg/max per packageFragm)     |              | 11           | 6,481     | 17        | /org.eclipse.emf.examples.library/src/org             |
| ▶ Number of Children (avg/max per type)            | 10           | 0,625        | 0,992     | 3         | /org.eclipse.emf.examples.library/src/org             |
| ▶ Depth of Inheritance Tree (avg/max per type)     |              | 5,062        | 1,784     | 8         | /org.eclipse.emf.examples.library/src/org/eclipse/emt |



# Tests stati

(plugin Metrics : <http://metr>)

| Metric   | Total | Mean   | Std. Dev. | Maximum | Resource c  |
|--|-------|--------|-----------|---------|-------------|
| ▶ Number of Static Methods (avg/max per type)    | 3     | 0,188  | 0,527     | 2       | /org.eclips |
| ▶ Total Lines of Code                            | 3128  |        |           |         |             |
| ▶ Afferent Coupling (avg/max per packageFragm)   |       | 12     | 14,855    | 33      | /org.eclips |
| ▶ Normalized Distance (avg/max per packageFra)   |       | 0,259  | 0,073     | 0,333   | /org.eclips |
| ▶ Number of Classes (avg/max per packageFrag)    | 16    | 5,333  | 6,182     | 14      | /org.eclips |
| ▶ Specialization Index (avg/max per type)        |       | 0,453  | 0,339     | 1,4     | /org.eclips |
| ▶ Instability (avg/max per packageFragment)      |       | 0,627  | 0,22      | 0,875   | /org.eclips |
| ▶ Number of Attributes (avg/max per type)        | 44    | 2,75   | 3,961     | 17      | /org.eclips |
| ▶ Number of Packages                             | 3     |        |           |         |             |
| ▶ Method Lines of Code (avg/max per method)      | 1500  | 5,792  | 10,767    | 127     | /org.eclips |
| ▶ Weighted methods per Class (avg/max per typ)   | 583   | 36,438 | 22,671    | 96      | /org.eclips |
| ▶ Number of Overridden Methods (avg/max per      | 17    | 1,062  | 0,658     | 3       | /org.eclips |
| ▶ Number of Static Attributes (avg/max per type) | 21    | 1,312  | 1,31      | 4       | /org.eclips |
| ▶ Nested Block Depth (avg/max per method)        |       | 1,139  | 0,469     | 4       | /org.eclips |
| ▶ Number of Methods (avg/max per type)           | 256   | 16     | 10,137    | 51      | /org.eclips |

|                              |                  |
|------------------------------|------------------|
| Number of Static Methods     | type             |
| Total Lines of Code          | compilationUnit  |
| Afferent Coupling            | packageFragment  |
| Normalized Distance          | packageFragment  |
| Number of Classes            | compilationUnit  |
| Specialization Index         | type             |
| Instability                  | packageFragment  |
| Number of Attributes         | type             |
| Number of Packages           | packageFragmentR |
| New Methods Lines of Code    | method           |
| Weighted methods per Class   | type             |
| Number of Overridden Methods | type             |
| Number of Static Attributes  | type             |
| Nested Block Depth           | method           |
| Number of Methods            | type             |
| Lack of Cohesion of Methods  | type             |
| McCabe Cyclomatic Complexity | method           |
| Number of Parameters         | method           |
| Abstractness                 | packageFragment  |
| Number of Interfaces         | compilationUnit  |

**Number of Classes** : Total number of classes in the selected scope

**Number of Children** : Total number of direct subclasses of a class. ...

**Number of Interfaces** : Total number of interfaces in the selected scope

**Depth of Inheritance Tree (DIT)** : Distance from class Object in the inheritance hierarchy.

**Number of Overridden Methods (NORM)** : Total number of methods in the selected ...

**McCabe Cyclomatic Complexity** : Counts the number of flows through a piece of code.

**Weighted Methods per Class (WMC)** : Sum of the McCabe Cyclomatic Complexity for all methods in a class



# Indice de spécialisation

→ Se calcule sur une classe entière (puis éventuellement moyenne pour le projet)

→ **Définition :**

$$\frac{NORM \times DIT}{NOM}$$

→ **Avec**

→ **NORM** : Number of Overriden Methods

→ **DIT** : Depth of Inheritance Tree (distance depuis la classe Object)

→ **NOM** : Number of Methods

# exemple de spécialisation

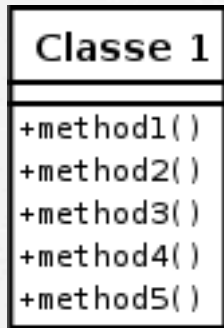
$$\frac{NORM \times DIT}{NOM}$$

```
public class Identifiable implements Comparable<Identifiable> {
    private final String ident;
    /**
     * @return le hashCode de l'identifiant de cet objet.
     */
    @Override
    public final int hashCode() {
        if (ident == null) return 0;
        else return ident.hashCode();
    }
    /**
     * @return le résultat de equals sur les identifiants.
     */
    @Override
    public final boolean equals(Object o) {
        if (!(o instanceof Identifiable)) {
            return false; }
        String oid = ((Identifiable) o).ident;
        if (ident == null)
            return oid == null;
        else
            return ident.equals(oid);
    }
    @Override
    public String toString() {
        if (ident == null)
            return "[null]";
        else
            return "[" + ident + "]";
    }
    .....
}
```

- public class Identifiable  
implements  
Comparable<Identifiable>
- 3 méthodes surchargées
  - Profondeur de l'arbre d'héritage : 1
  - 6 méthodes
  - $3/6 = 0,5$

# exemple de spécialisation

$$\frac{NORM \times DIT}{NOM}$$



→ Classe 1 :

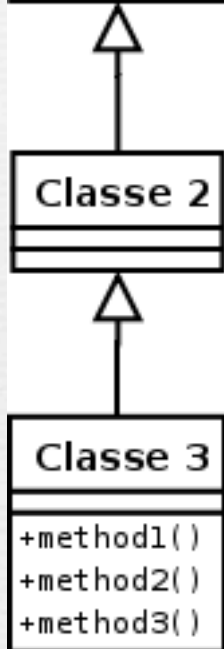
NORM : 0; DIT : 1; NOM : 5; specialisation : 0

→ Classe 2 :

NORM : 0; DIT : 2; NOM : 0; specialisation : 0

→ Classe 3 :

NORM : 3; DIT : 3; NOM : 3; specialisation : 3



# Indice de spécialisation

$$\frac{NORM \times DIT}{NOM}$$

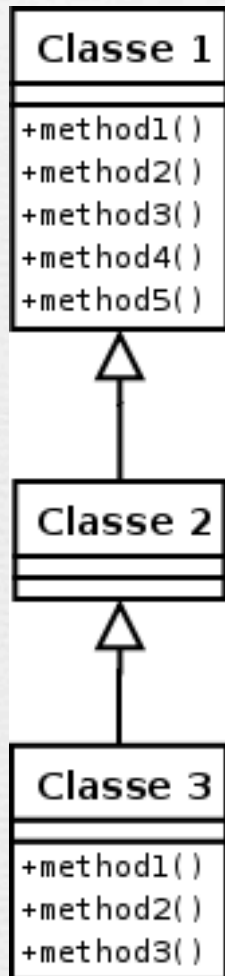
## *Augmente quand*

- Le nombre de méthodes surchargées ou la profondeur d'héritage augmente

## *Diminue quand*

- Le nombre de méthodes spécifiques à la classe augmente.
- Le nombre de méthodes redéfinies diminue.

# Indice de spécialisation



$$\frac{NORM \times DIT}{NOM}$$

Trop grand : la classe redéfinit trop de méthodes dont elle hérite : il faut penser à refactoriser en utilisant des interfaces par exemple.

- Moyenne : 0.05



# Indice d'instabilité de packages

→ Se calcule sur un paquetage ou un ensemble de paquetages.

$$\frac{C_e}{C_a + C_e}$$

• Définition :

- $C_e$  (efferent coupling) : le nombre de classes de ce paquetage qui dépendent de classes de l'extérieur. (indépendance)
- $C_a$  (afferent coupling) : le nombre de classes de l'extérieur qui dépendent de classes dans ce paquetage (responsabilité)

# Indice d'instabilité d'un package

Par exemple, le package ReseauRoutier dépend de

- $C_e = 4$  classes extérieures (Arc, Sommet, Graphe, Parcours)
- $C_a = 0$ , il n'est pas utilisé de l'extérieur.

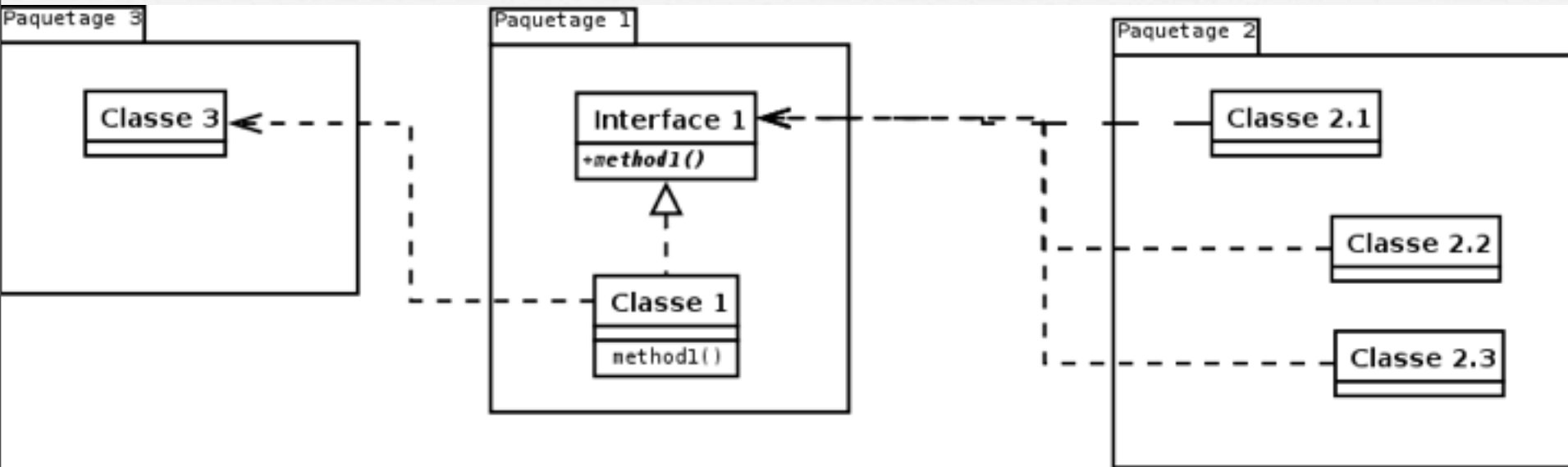
Instabilité = 1

$$\frac{C_e}{C_a + C_e}$$

GrapheX => 0;

RéseauSocial :  
1 utilisation de  
l'extérieur dans test;  
les 3 classes du  
package dépendent  
de classes extérieures:  
**Instabilité : 0,75**

# Indice d'instabilité d'un package



Package 3 :  $C_e = 0$ ;  $C_a = 1$ ; Instabilité = 0

Package 1 :  $C_e = 1$ ;  $C_a = 3$ ; Instabilité =  $1/4 = 0,25$

Package 2 :  $C_e = 3$ ;  $C_a = 0$ ; Instabilité = 1

Packages : Instabilité = 0.41

# Indice d'instabilité d'un package

## Indice d'instabilité - interprétation

- Indice compris entre 0 et 1 :
  - 0 : le paquetage est stable.
  - 1 : le paquetage n'est pas stable.
- Pour qu'un paquetage soit considéré comme stable avec cet indice, il faut qu'il y ait plus de dépendances entrantes que sortantes

$$\frac{C_e}{C_a + C_e}$$

# Niveau d'abstraction de packages

→ Se calcule sur un paquetage ou un ensemble de paquetages.

$$\frac{I}{T}$$

→ Avec

→ I : Le nombre d'interfaces et de classes abstraites.

→ T : le nombre total de types.



# Niveau d'abstraction de packages

→ Se calcule sur un paquetage ou un ensemble de paquetages.

$$\frac{I}{T}$$

- S'utilise surtout comme élément de mesure de la “normalized distance from the main sequence”

# Distance from the main sequence

➔ Se calcule sur un paquetage ou un ensemble de paquetages.

➔ Définition :

$$|Abstractness + Instability - 1|$$

➔ Avec

➔ – *Abstractness* : l'indice d'abstraction du paquetage.

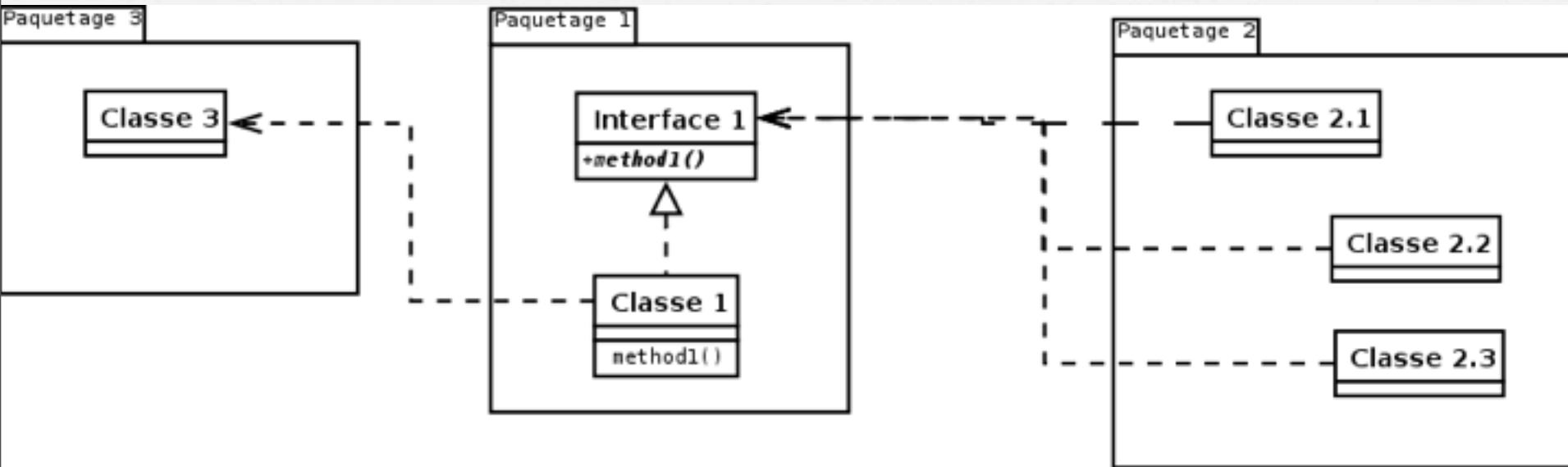
➔ – *Instability* : l'indice d'instabilité du paquetage.

# Distance from the main sequence

$$|Abstractness + Instability - 1|$$

→ Un paquetage est bien conçu si ce nombre est proche de zéro.

# Distance from the main sequence



Package 3 : asbtraction=0; instabilité=0; distance = 1

Package 1 : asbtraction=0,5; instabilité=0,25; distance = 0,25

Package 2 : asbtraction = 0; instabilité = 1; distance = 0

Packages : distance = 0,417

$$|Abstractness + Instability - 1|$$

# Assurer une bonne lisibilité du code

## → Complexité cyclomatique d'une méthode

### - Définition

▸ Nombre de chemins linéairement indépendants qu'il est possible d'emprunter dans cette méthode

☞ Nombre de points de décision de la méthode : if, case, while, ... + 1 (le chemin principal)

### - Interprétation

▸ Une méthode avec une haute complexité cyclomatique est plus difficile à comprendre et à maintenir

➡ à 30 : refactoriser la méthode

➡ à 30 : acceptable si suffisamment de tests

▸ La complexité cyclomatique est liée à la notion de « couverture de code »

☞ Nombre de tests unitaires = sa complexité cyclomatique

☞ **Un chemin = un test**

```
private ArrayList<Chemin> chemins(Sommet origine, HashMap<Sommet, ArrayList<Arc>> deJeuVu){
    ArrayList<Chemin> chemins = new ArrayList<Chemin>();
    if (deJeuVu.containsKey(origine)){
        chemins.add(new Chemin(deJeuVu.get(origine)));
        return chemins;
    }
    deJeuVu.put(origine, new ArrayList<Arc>());

    Collection<Arc<Sommet>> voisins = graphe.voisins(origine);
    HashMap<Sommet, ArrayList<Arc>> deJeuVuLocal = new HashMap<Sommet, ArrayList<Arc>>();

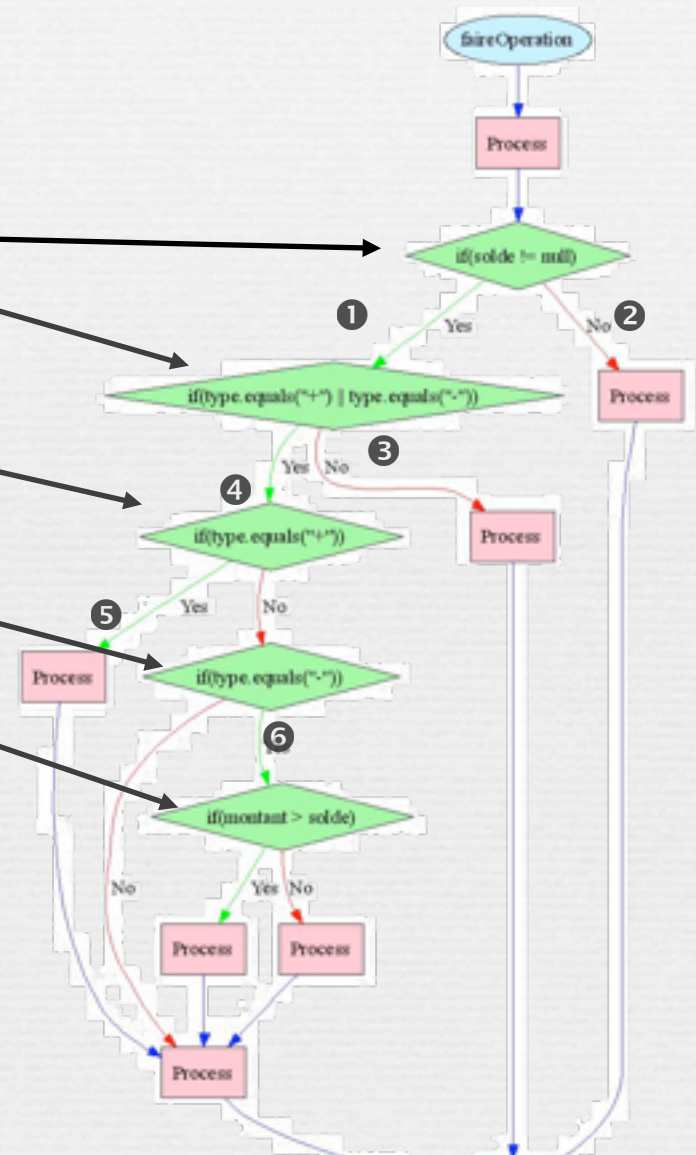
    for (Arc<Sommet> a : voisins) {
        Sommet destination = a.destination();
        deJeuVuLocal.put(destination, new ArrayList<Arc>());
    }

    if (NouvelleDestinationOuNouvelArcSansRetourCorigine(deJeuVuLocal, destination, a)) {
        deJeuVuLocal.get(origine).add(a);
    }
}
```



# Complexité cyclomatique d'une méthode

```
package banque;  
  
public class Banque {  
    private Double solde;  
  
    public void faireOperation(  
        String type, double montant) {  
        System.out.println("Début d'opération.");  
        if(solde != null) {  
            if(type.equals("+") || type.equals("-"))  
            {  
                if(type.equals("+")) {  
                    solde += montant;  
                }  
                if(type.equals("-")) {  
                    if(montant > solde) {  
                        System.err.println("!!!");  
                    }  
                    else {  
                        solde -= montant;  
                    }  
                }  
            }  
            else {  
                System.err.println("...");  
            }  
            else {  
                System.err.println(".");  
            }  
        }  
        System.out.println("Fin d'opération.");  
    }  
}
```



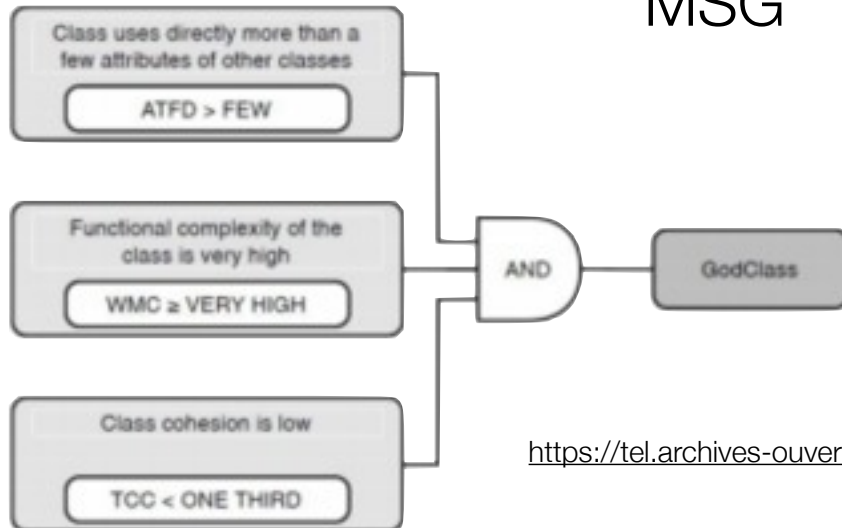
# Analyses des dépendances de packages (plugin Metrics : <http://metrics.sourceforge.net/>)



Le graphe de dépendances des packages est analysé. Les packages fortement connectés sont colorés en rouge. Les autres packages ne participant pas à des cycles sont de couleur bleue.

# Metrics compress the system into numbers

|        |       |          |
|--------|-------|----------|
| NOC    | NOM   | DUPLINES |
| NOCmts | LOC   | NAI      |
| NOPA   | TCC   | NOA      |
| WLOC   | WMC   | NI       |
| WNOC   | CYCLO | ...      |
| WOC    | ATFD  |          |
| MSG    | HNL   |          |



<https://tel.archives-ouvertes.fr/tel-00790056/document>



# Tests statiques

## Analyse de modèles avec VP

Matrix Diagram2

Sort: Nom Show rows/columns: Matches only filter row... filter column...

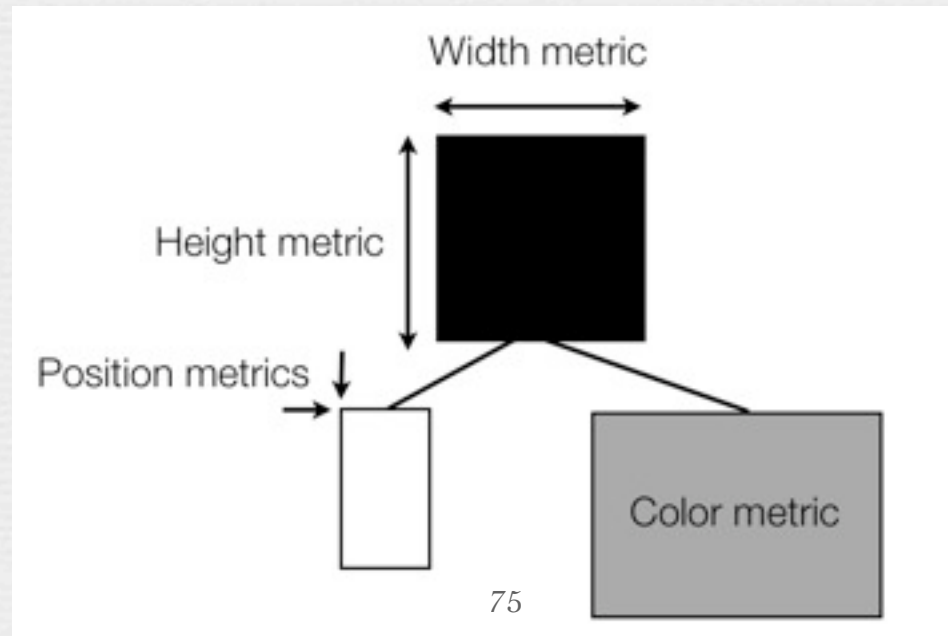
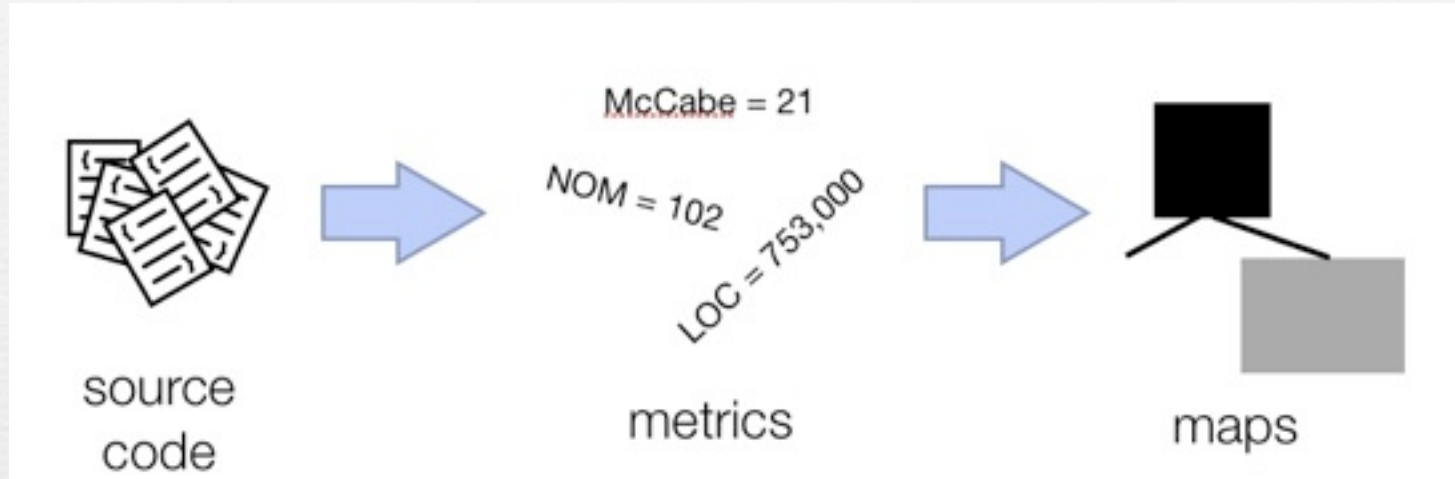
(36) Class

By: Relation  
Relation: (All)  
Include children:

| Class                    | Calendrier | Connexion | ConversanetMain | Etat | ExploreDossier | FenTc | FenTcDebrief | FenTcEcouleNouveau | FenTcEvaluation | FenTcPanelStats | FenTcRems | FenetreEcoute | FenetreFicheLancement | FenetreFicheLancement... | FenetreFicheLancement... | FenetreMatos | FenetreMysys | FenetrePermanences | FenetrePlanningComite... | FenetrePresse | FenetreRolls | FicheLancementInfos | JLabelPerso | JTablePerso | JTextAreaPerso | MenuAccuell | OuvreFichier | PanelAccueil | PanelAccueilBoutons | PanelPresseOpe | PanelPresseSaisie | PanelTA | Sql | SqlAccess | TcEcouleNouveauFoncti... | Variables |  |  |
|--------------------------|------------|-----------|-----------------|------|----------------|-------|--------------|--------------------|-----------------|-----------------|-----------|---------------|-----------------------|--------------------------|--------------------------|--------------|--------------|--------------------|--------------------------|---------------|--------------|---------------------|-------------|-------------|----------------|-------------|--------------|--------------|---------------------|----------------|-------------------|---------|-----|-----------|--------------------------|-----------|--|--|
| Calendrier               |            |           |                 |      |                |       |              |                    |                 |                 |           |               |                       |                          |                          |              |              |                    |                          |               |              |                     |             |             |                |             |              |              |                     |                |                   |         |     |           |                          |           |  |  |
| Connexion                |            |           |                 |      |                |       |              |                    |                 |                 |           |               |                       |                          |                          |              |              |                    |                          |               |              |                     |             |             |                |             |              |              |                     |                |                   |         |     |           |                          |           |  |  |
| ConversanetMain          |            |           |                 |      |                |       |              |                    |                 |                 |           |               |                       |                          |                          |              |              |                    |                          |               |              |                     |             |             |                |             |              |              |                     |                |                   |         |     |           |                          |           |  |  |
| Etat                     |            |           |                 |      |                |       |              |                    |                 |                 |           |               |                       |                          |                          |              |              |                    |                          |               |              |                     |             |             |                |             |              |              |                     |                |                   |         |     |           |                          |           |  |  |
| ExploreDossier           |            |           |                 |      |                |       |              |                    |                 |                 |           |               |                       |                          |                          |              |              |                    |                          |               |              |                     |             |             |                |             |              |              |                     |                |                   |         |     |           |                          |           |  |  |
| FenTc                    |            |           |                 |      |                |       |              |                    |                 |                 |           |               |                       |                          |                          |              |              |                    |                          |               |              |                     |             |             |                |             |              |              |                     |                |                   |         |     |           |                          |           |  |  |
| FenTcDebrief             |            |           |                 |      |                |       |              |                    |                 |                 |           |               |                       |                          |                          |              |              |                    |                          |               |              |                     |             |             |                |             |              |              |                     |                |                   |         |     |           |                          |           |  |  |
| FenTcEcouleNouveau       |            |           |                 |      |                |       |              |                    |                 |                 |           |               |                       |                          |                          |              |              |                    |                          |               |              |                     |             |             |                |             |              |              |                     |                |                   |         |     |           |                          |           |  |  |
| FenTcEvaluation          |            |           |                 |      |                |       |              |                    |                 |                 |           |               |                       |                          |                          |              |              |                    |                          |               |              |                     |             |             |                |             |              |              |                     |                |                   |         |     |           |                          |           |  |  |
| FenTcPanelStats          |            |           |                 |      |                |       |              |                    |                 |                 |           |               |                       |                          |                          |              |              |                    |                          |               |              |                     |             |             |                |             |              |              |                     |                |                   |         |     |           |                          |           |  |  |
| FenTcRems                |            |           |                 |      |                |       |              |                    |                 |                 |           |               |                       |                          |                          |              |              |                    |                          |               |              |                     |             |             |                |             |              |              |                     |                |                   |         |     |           |                          |           |  |  |
| FenetreEcoute            |            |           |                 |      |                |       |              |                    |                 |                 |           |               |                       |                          |                          |              |              |                    |                          |               |              |                     |             |             |                |             |              |              |                     |                |                   |         |     |           |                          |           |  |  |
| FenetreFicheLancement    |            |           |                 |      |                |       |              |                    |                 |                 |           |               |                       |                          |                          |              |              |                    |                          |               |              |                     |             |             |                |             |              |              |                     |                |                   |         |     |           |                          |           |  |  |
| FenetreFicheLancement... |            |           |                 |      |                |       |              |                    |                 |                 |           |               |                       |                          |                          |              |              |                    |                          |               |              |                     |             |             |                |             |              |              |                     |                |                   |         |     |           |                          |           |  |  |
| FenetreFicheLancement... |            |           |                 |      |                |       |              |                    |                 |                 |           |               |                       |                          |                          |              |              |                    |                          |               |              |                     |             |             |                |             |              |              |                     |                |                   |         |     |           |                          |           |  |  |
| FenetreMatos             |            |           |                 |      |                |       |              |                    |                 |                 |           |               |                       |                          |                          |              |              |                    |                          |               |              |                     |             |             |                |             |              |              |                     |                |                   |         |     |           |                          |           |  |  |
| FenetreMysys             |            |           |                 |      |                |       |              |                    |                 |                 |           |               |                       |                          |                          |              |              |                    |                          |               |              |                     |             |             |                |             |              |              |                     |                |                   |         |     |           |                          |           |  |  |
| FenetrePermanences       |            |           |                 |      |                |       |              |                    |                 |                 |           |               |                       |                          |                          |              |              |                    |                          |               |              |                     |             |             |                |             |              |              |                     |                |                   |         |     |           |                          |           |  |  |

# Tests statiques

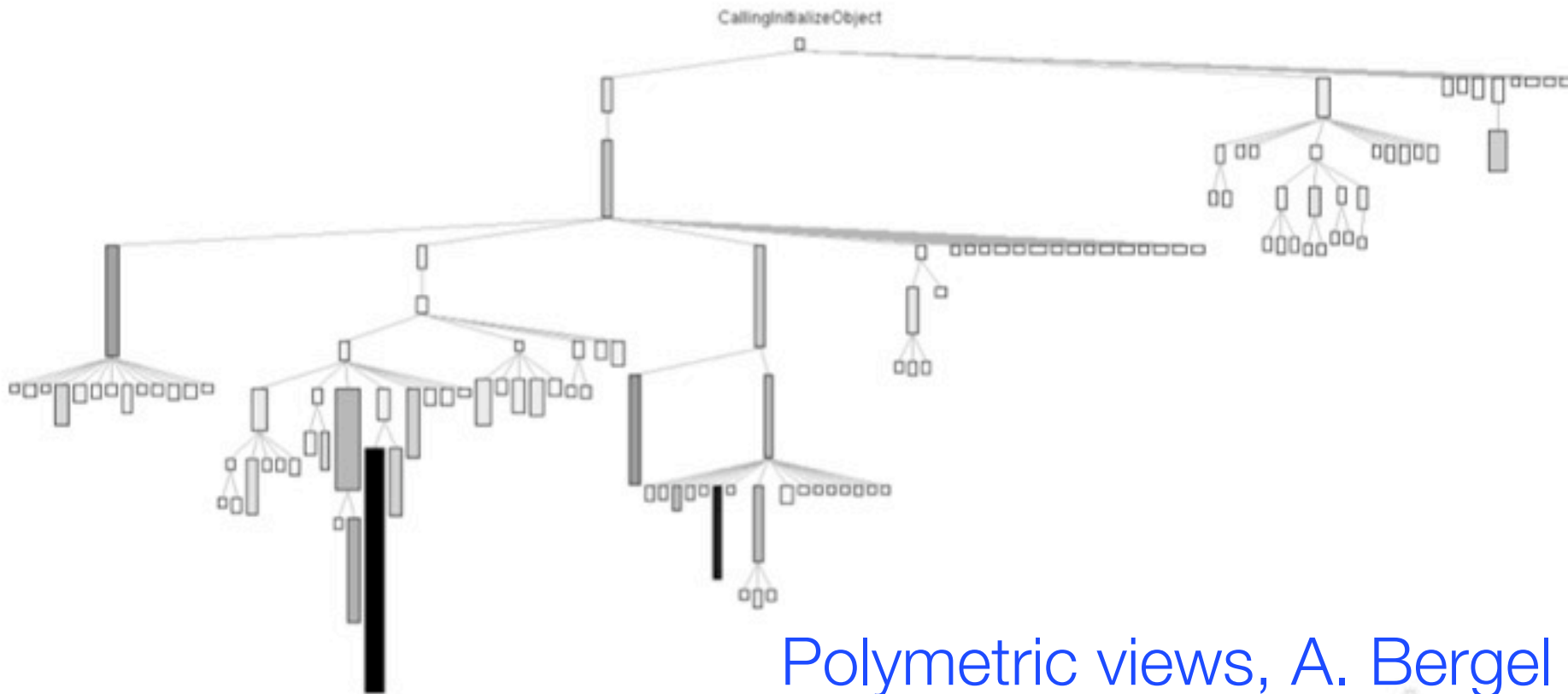
## Autres Visualisations





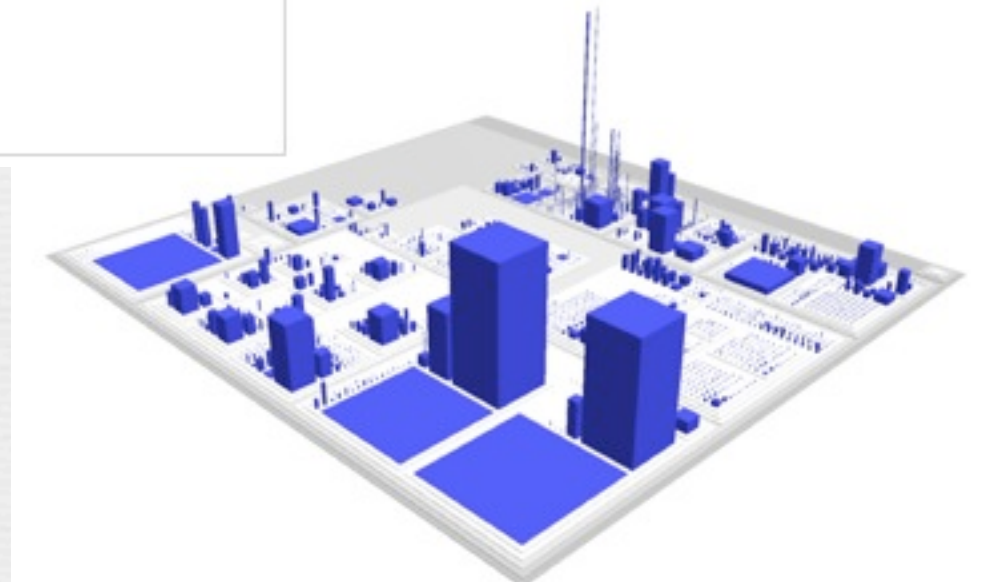
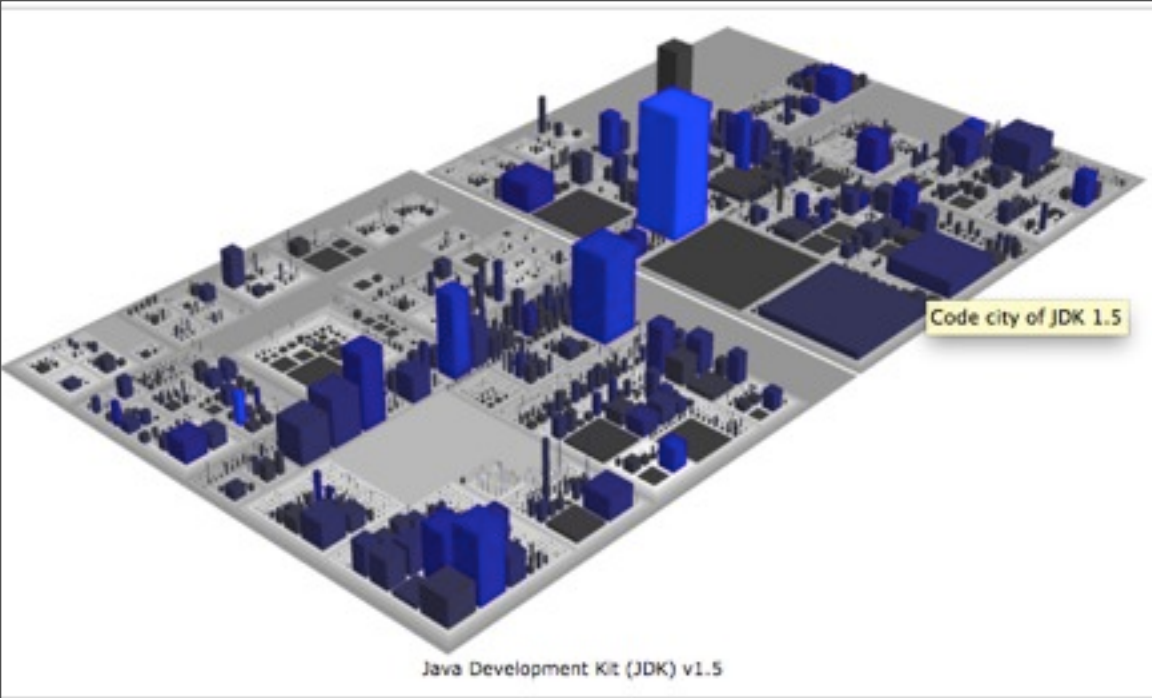
# Tests statiques

## Autres Visualisations



# Tests statiques

## Autres Visualisations



CodeCity (Wettel, Univ Lugano)

<http://www.inf.usi.ch/phd/wettel/codecity-wof.html>

# Analyse du style du code

# Assurer une bonne lisibilité du code

- ❧ S'accorder sur un ensemble de pratiques permettant d'assurer la bonne compréhension du code
- ❧ Utiliser des standards de
  - ❧ Codage : Entreprise, Projet
  - ❧ Nommage
  - ❧ Indentation

# Est-ce la classe que vous cherchez?

```
public class T01<T02,T03> extends T04<T02,T03>
    implements T05<T02,T03>, T06, T07 {

    public T03 m01(T02 x01, T03 x02) {
        if (x01 == null)
            return m02(x02);
        int x03 = m03(x01.m04());
        int x04 = m05(x03, x05.x06);
        for (T08<T02,T03> x07 = x08[x04]; x07 != null; x07 = x07.x09) {
            T09 x10;
            if (x07.x11 == x03 && ((x10 = x07.x12) == x01 || x01.m06(x10))) {
                T03 x13 = x07.x14;
                x07.x14 = x02;
                x07.m07(this);
                return x13;
            }
        }
        x15++;
        m08(x03, x01, x02, x04);
        return null;
    }
}
```



# Is this one the class you are looking for?

```
public class HashMap<K,V> extends AbstractMap<K,V>
    implements Map<K,V>, Cloneable, Serializable {

    public V put(K key, V value) {
        if (key == null)
            return putForNullKey(value);
        int hash = hash(key.hashCode());
        int i = indexOf(hash, table.length);
        for (Entry<K,V> e = table[i]; e != null; e = e.next) {
            Object k;
            if (e.hash == hash && ((k = e.key) == key || key.equals(k))) {
                V oldValue = e.value;
                e.value = value;
                e.recordAccess(this);
                return oldValue;
            }
        }
        modCount++;
        addEntry(hash, key, value, i);
        return null;
    }
}
```

# Identifiant auto-documenté

## Bons identifiants:

- fournit des indices concis sur la sémantique des données identifiées;
- évite aux programmeurs de lire le segment de code en entier;
- accélère l'acquisition de connaissances;
- facilite la compréhension du programme (recherche de mots, ...)

# Checkstyle

■ <http://checkstyle.sourceforge.net/>

- ➔ Checkstyle performs source code analysis.
  - ✓ Originally for "coding standard" (formatting)
  - ✓ Now includes design-level best practice compliances.
- ➔ Classes of checks:
  - ✓ JavaDoc, Naming Conventions, Headers, Size Violations, Imports, WhiteSpace, Modifiers, Block Checks, Coding, Class Design, Duplicate Code, Metrics, J2EE.
- ➔ Can be extended with new checks.
  - ✓ Use a configuration file to customize what checks your system should comply with.

# Java conventions

CITS1220 Software Engineering

Coding Style Rules

|                    |                      | Rules  | Example  |
|--------------------|----------------------|--|--|
| Naming Conventions | Classes              | <ul style="list-style-type: none"><li>• Use nouns</li><li>• Use whole words-avoid acronyms and abbreviations (unless the abbreviation is much more widely used than the long form, such as URL or HTML).</li><li>✓ Begin with upper case letters</li><li>• First letter of each internal word capitalized</li></ul>  | QueueBlock, ReversalADT, LinkedList, CustomerAccount, not MaintainCustomerData   |
|                    | Methods              | <ul style="list-style-type: none"><li>• Use verbs</li><li>✓ Begin with lower case letters</li><li>• First letter of each internal word capitalized</li></ul>   | examine, delete, isEmpty, toString   |
|                    | Instance Variables   | <ul style="list-style-type: none"><li>• Use nouns, plurals for collection classes</li><li>✓ Begin with lower case letters</li><li>• First letter of each internal word capitalized</li><li>• Do not start with underscore _ or dollar sign \$ characters, even though both are allowed.</li><li>• One-character variable names should be avoided except for common temporary variables.</li></ul>  | items, firstAction, list<br><br>Common temporary variables names<br><b>Integers:</b> i, j, k, m, and n<br><b>Characters:</b> c, d, and e     |
|                    | Constants            | <ul style="list-style-type: none"><li>✓ All characters in upper case</li><li>• Words are separated by underscores (" ")</li></ul>  | private static final int<br>MAX_SEQUENCE   |
|                    | Files                | <ul style="list-style-type: none"><li>✓ File names should be the same as the (principal) class stored in the file.</li></ul>   |  |
| Layout             | Classes & Interfaces | <ul style="list-style-type: none"><li>✓ No space between a method name and the parenthesis "(" starting its parameter list</li><li>✓ Open brace "{" appears at the end of the same line as the declaration statement</li><li>✓ Closing brace "}" starts a line by itself indented to match its corresponding opening statement, except when it is a null statement the "}" should appear immediately after the "{"</li><li>✓ The internal statements are indented by 2 spaces.</li></ul> | <pre>public class QueueBlock {     .     .     protected Object [] items;     .     .     public int isZero () {return 0;}     .     .</pre> |

# Java doc

## Order of Tags

Include tags in the following order:

- 1 `@author` (classes and interfaces only, required)
- 2 `@version` (classes and interfaces only, required. See [footnote 1](#))
- 3 `@param` (methods and constructors only)
- 4 `@return` (methods only)
- 5 `@exception` (`@throws` is a synonym added in Javadoc 1.2)
- 6 `@see`
- 7 `@since`
- 8 `@serial` (or `@serialField` or `@serialData`)
- 9 `@deprecated` (see [How and When To Deprecate APIs](#))

## `@version` ([reference page](#))

The Java Software convention for the argument to the `@version` tag is the SCCS string "%I%, %G%", which converts to something like "1.39, 02/28/97" (mm/dd/yy) when the file is checked out of SCCS.

```
* All coordinates which appear as arguments to the methods of this
* Graphics object are considered relative to the translation origin
* of this Graphics object prior to the invocation of the method.
* All rendering operations modify only pixels which lie within the
* area bounded by both the current clip of the graphics context
* and the extents of the Component used to create the Graphics object.
*
* @author      Sami Shaio
* @author    Arthur van Hoff
* @version    %I%, %G%
* @since     1.0
* /
```

<http://www.oracle.com/technetwork/articles/java/index-137868.html>



# Java doc

## Tag Comments

As a reminder, the fundamental use of these tags is described on the [Javadoc Reference page](#). Java Software generally uses the following additional guidelines to create comments for each tag:

### **@author** ([reference page](#))

You can provide one @author tag, multiple @author tags, or no @author tags. In these days of the community process when development of new APIs is an open, joint effort, the JSR can be consider the author for new packages at the package level. For example, the new package java.nio has "@author JSR-51 Expert Group" at the package level. Then individual programmers can be assigned to @author at the class level. As this tag can only be applied at the overview, package and class level, the tag applies only to those who make significant contributions to the design or implementation, and so would not ordinarily include technical writers.

The @author tag is not critical, because it is not included when generating the API specification, and so it is seen only by those viewing the source code. (Version history can also be used for determining contributors for internal purposes.)

If someone felt strongly they need to add @author at the member level, they could do so by running javadoc using the new 1.4 `-tag` option:

```
-tag author:a:"Author:"
```

If the author is unknown, use "unascribed" as the argument to @author. Also see [order of multiple @author tags](#).

### **@version** ([reference page](#))

The Java Software convention for the argument to the @version tag is the SCCS string "%I%, %G%", which converts to something like " 1.39, 02/28/97" (mm/dd/yy) when the file is checked out of SCCS.

<http://www.oracle.com/technetwork/articles/java/index-137868.html>



PMD

# PMD

- ➔ PMD also performs source code analysis.
  - ✓ More 'design' oriented than Checkstyle.
  - ✓ Lots of overlap.
- ➔ PMD rulesets:
  - ✓ Basic, Braces, Code Size, Clone, Controversial, Coupling, Design, Finalizers, Import, J2EE, JavaBeans, JUnit, Logging, Migrating, Naming, Optimizations, Exceptions, Strings, Security, Unused Code, JSP, JSF.

# On retient au moins

- ➔ Quelles sont les propriétés que je peux/dois vérifier lorsque je fournis un logiciel?
- ➔ Comment vérifier que mon code vérifie bien ces propriétés?
  - ✓ Ne pas se poser ces questions, revient à faire un gâteau sans se poser les questions :
    - Pour qui est ce gâteau? Pour combien de personnes? Pour quelle occasion ? Doit-il être transporté?
    - Puis/dois je y mettre du chocolat?
    - Est-il assez/trop sucré, y-en a-t-il assez pour tous, va-t-il résister au transport?