ANTI-PATRONS ET CODE SMELLS DANS LES SYSTÈMES MULTI-LANGAGES

PALMYRA WÜRTZ

PLAN

UdM, Polytechnique Montréal puis Concordia University

Ptidej : Pattern Trace Identification, Detection, and Enhancement in Java

Internet of Things

PRÉSENTATION DU LABORATOIRE

SYSTÈMES MULTI-LANGAGES

SYSTÈMES MULTI-LANGAGES

Bonnes raisons

- Économies liées à l'utilisation de code écrit dans un autre langage
- Besoin d'inclure du code pré-existant
- Disponibilité de développeurs pour des langages spécifiques
- Langage spécifique pour besoin spécifique

Mauvaises raisons

- Résistance aux changements
- Mauvais management d'un projet (usage d'autres langages non planifié).

JAVA NATIVE INTERFACE - JAVA

```
public class HelloJNI {
    static {
        System.loadLibrary("hello");
    }
    private native void sayHello();
    public static void main(String[] args) {
        new HelloJNI().sayHello();
    }
}
```

JAVA NATIVE INTERFACE - C

```c
/* DO NOT EDIT THIS FILE - it is machine generated
*/
#include <jni.h>
/* Header for class HelloJNI */
#ifndef _Included_HelloJNI
#define _Included_HelloJNI
#ifdef __cplusplus
extern "C" {
#endif
/*
* Class: HelloJNI
* Method: sayHello
* Signature: ()V
*/
JNIEXPORT void JNICALL Java_HelloJNI_sayHello(JNIEnv
*, jobject);
#ifdef __cplusplus
}
#endif
#endif
```

```c
#include <jni.h>
#include <stdio.h>
#include "HelloJNI.h«

JNIEXPORT void JNICALL
Java_HelloJNI_sayHello(JNIEnv *env, jobject
thisObj) {
printf("Hello World!\n");
return;
}
```

DÉFAUTS DE CONCEPTION

ANTI-PATRONS ET CODE SMELLS

ANTI-PATRONS

- Erreurs de conception
- Perçus comme une solution à un problème
- Contraire des patrons de conception

- Exemple : blob

ANTI-PATRONS MULTI-LANGAGES

- Excessive inter-language communication
- Too much clustering of multi-language concerns
- Too much scattering of multi-language participants
- Unnecessary use of multi-language programming
- Language and paradigms mismatch
- Project migration language related issues

CODE SMELLS

- Mauvaises pratiques → apparition de défauts
- Mauvais choix qui complexifie le code
- *Refactoring* (réusinage)

- Exemple : code dupliqué

CODE SMELLS MULTI-LANGAGES

- Assuming safe multi-language return values
- Hard coding libraries
- Local references abuse
- Memory management mismatch
- Not caching objects' éléments
- Not handling exceptions across languages
- Not securing libraries
- Not using relative path to load the library
- Passing excessive objects
- Unused native method declaration
- Unused native method implementation
- Unnecessary parameters

IMPLÉMENTATION

UN EXEMPLE SIMPLE : TOO MUCH CLUSTERING

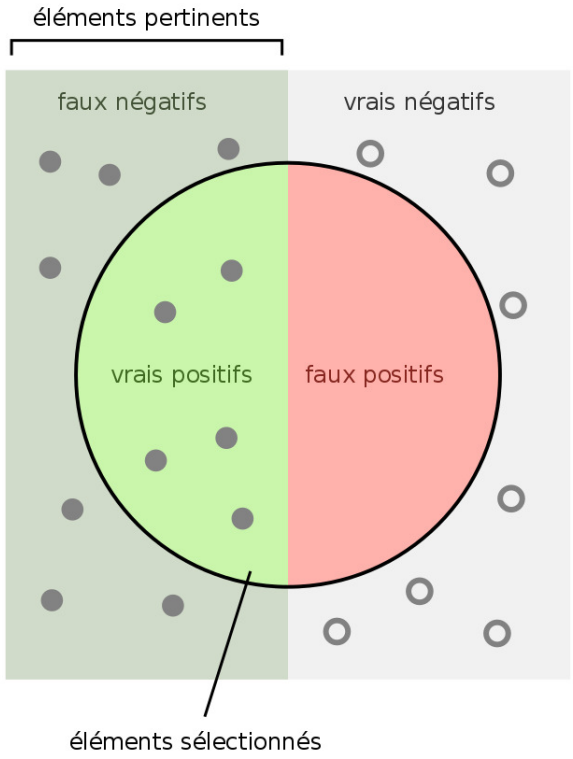
- Choix
- Complexité
- Classes `MLSAntiPattern` et `MLSCodeSmell`

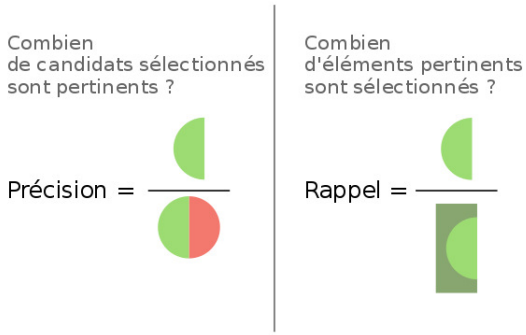
```
for (int j = 0; j < nbClasses; j++) {
      final Node thisClassNode = classList.item(j);
      // Native method declaration
      final NodeList nativeDeclList = (NodeList) NATIVE_EXP
      .evaluate(thisClassNode, XPathConstants.NODESET);
      if (nativeDeclList.getLength() > minNbOfMethodsPerClass) {
            final String thisClass = NAME_EXP.evaluate(thisClassNode);
            final String thisPackage =
            PACKAGE_EXP.evaluate(thisClassNode);
            final String thisFilePath =
            FILEPATH_EXP.evaluate(thisClassNode);
            antiPatternSet
            .add(
            new MLSAntiPattern(
                  this.getAntiPatternName(),
                  "",
                  "",
                  thisClass,
                  thisPackage,
                  thisFilePath));
      }
}
```

RÉSULTATS PRÉLIMINAIRES

Système	Précision	Rappel
Conscrypt	96,7 %	83 %
PL/Java	99,2 %	
OpenJ9	> 96 %	

$$p = \frac{VP}{VP + FP} \qquad r = \frac{VP}{VP + FN}$$





UN TRAVAIL À POURSUIVRE

PISTES D'AMÉLIORATION

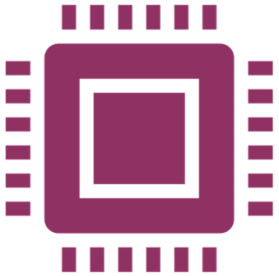
Identifier plus d'anti-patrons et de code smells

Prendre en compte plus de cas

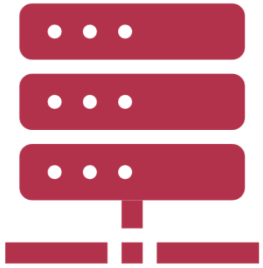
Localiser plus précisément les défauts

Accélérer l'exécution

FUTUR DE L'OUTIL



Autres combinaisons de langages de programmation



Intégration dans le logiciel Ptidej

CONCLUSION

- Contraintes matérielles, administratives, académiques
- Travail en équipe
- Un article à paraître