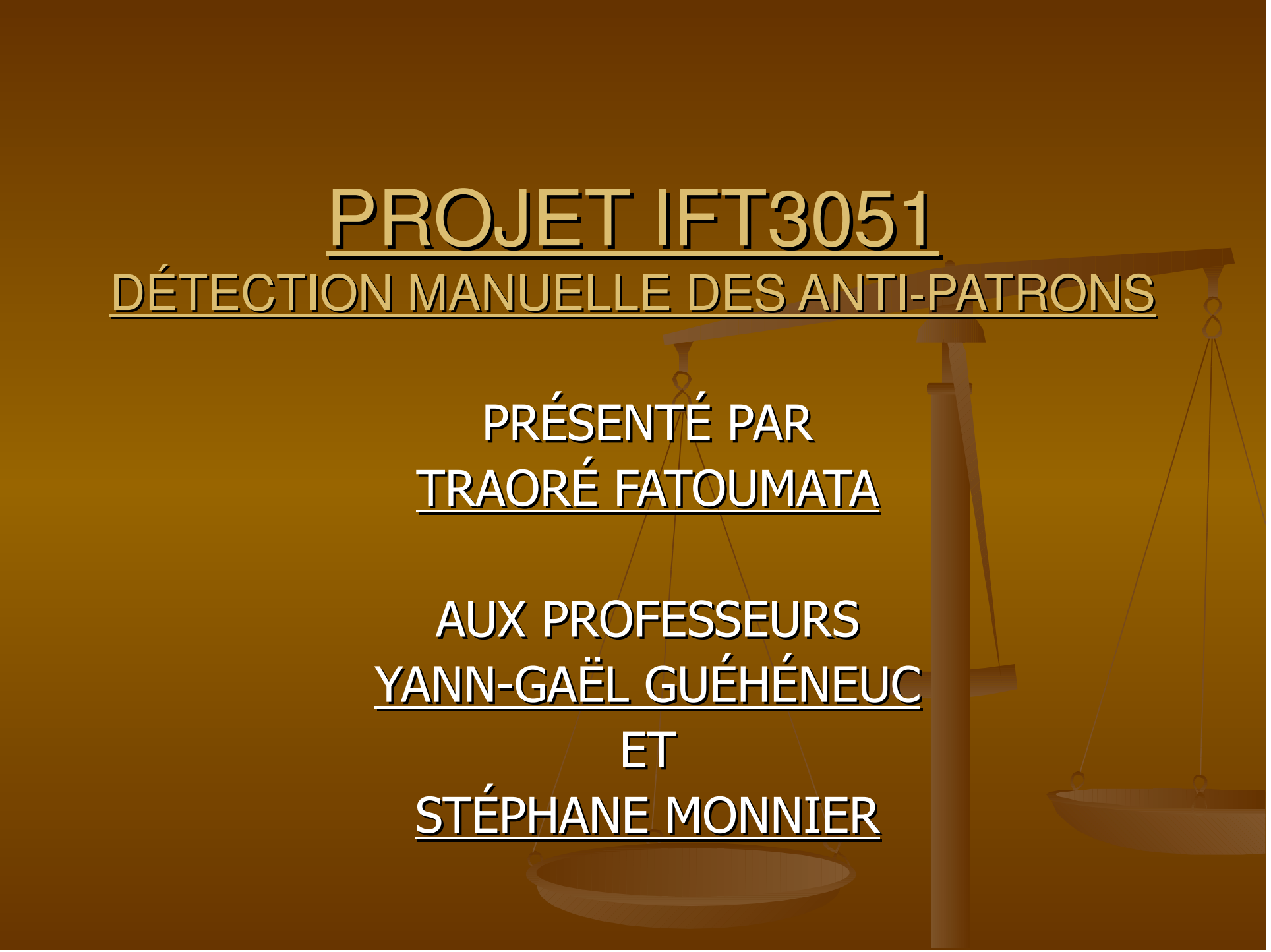
PROJET IFT3051

DÉTECTION MANUELLE DES ANTI-PATRONS

PRÉSENTÉ PAR
TRAORÉ FATOUMATA

AUX PROFESSEURS
YANN-GAËL GUÉHÉNEUC
ET
STÉPHANE MONNIER



OBJECTIFS

- Le but de ce travail est de détecter manuellement les anti-patrons de conception dans la version 0.7.1 de Nutch pour vérifier et valider les résultats obtenus par l'outil conçut par le professeur Yann-Gaël Guéhéneuc et Naouel Moha, étudiante au doctorat, et ce, par une comparaison entre les résultats obtenus par cet outil et ceux obtenus manuellement.

I-INTRODUCTION

Présentation de Nutch

- Nutch est un projet qui a pour objectif la conception d'un moteur de recherche à code source libre du genre Google mais en restant une entreprise à but non commercial pour contrecarrer les intérêts premiers des entreprises qui le sont.

- Nutch survit donc grâce à la contribution des développeurs bénévoles qui veulent bien y participer ainsi qu'à des dons de particuliers.

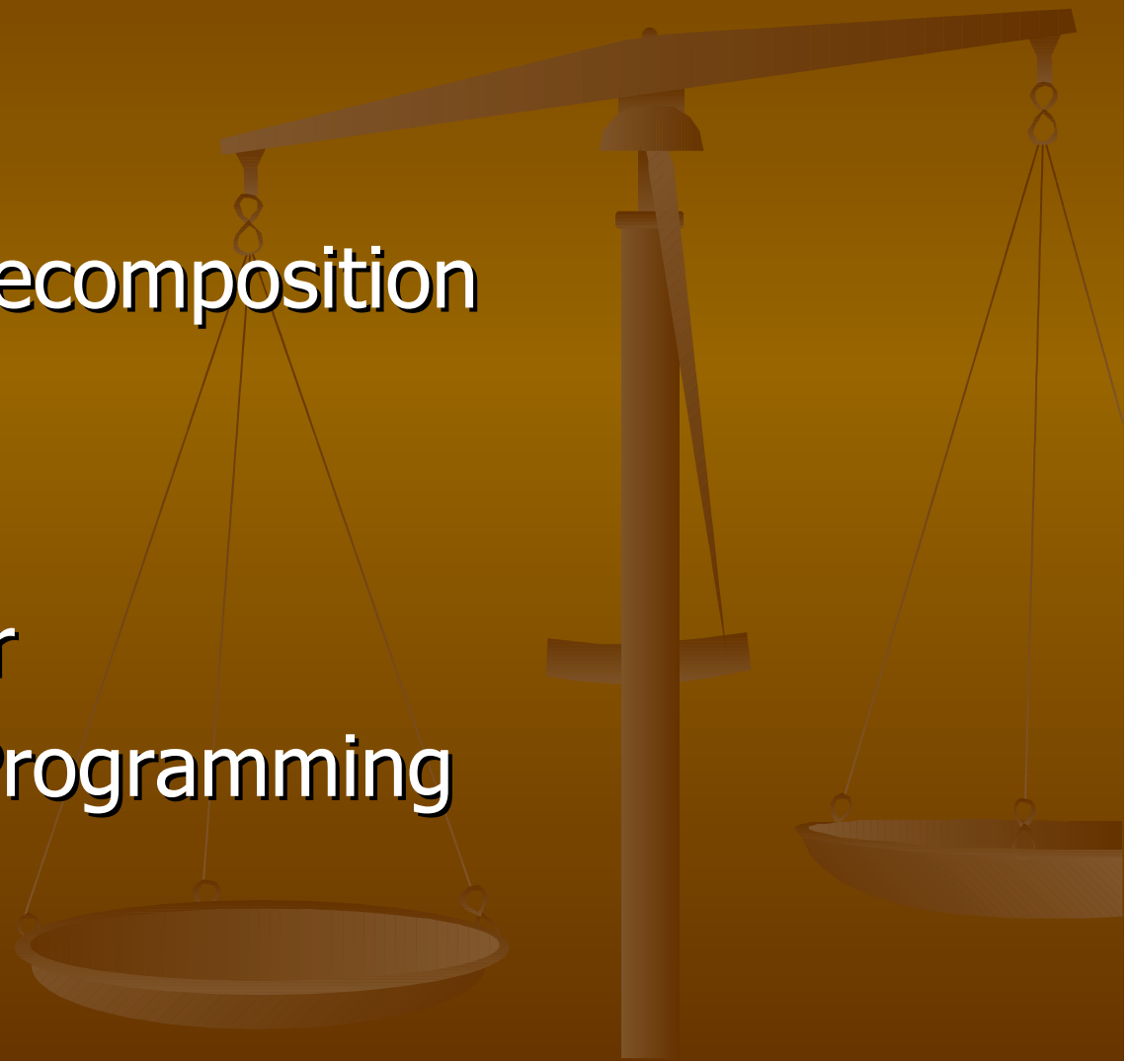
INTRODUCTION

Les anti-patrons

- Les anti-patrons sont des problèmes courants de conceptions dus soit au non-usage des patrons de conception, soit au mauvais usage de ceux-ci, soit à une mauvaise conception.
- La présence d'anti-patrons entraîne généralement dans un programme une lenteur excessive dans le logiciel, des coûts de réalisation ou de maintenance élevés, des comportements anormaux et la présence de bogues.

II- DIFFÉRENTS TYPES D'ANTI-PATRONS

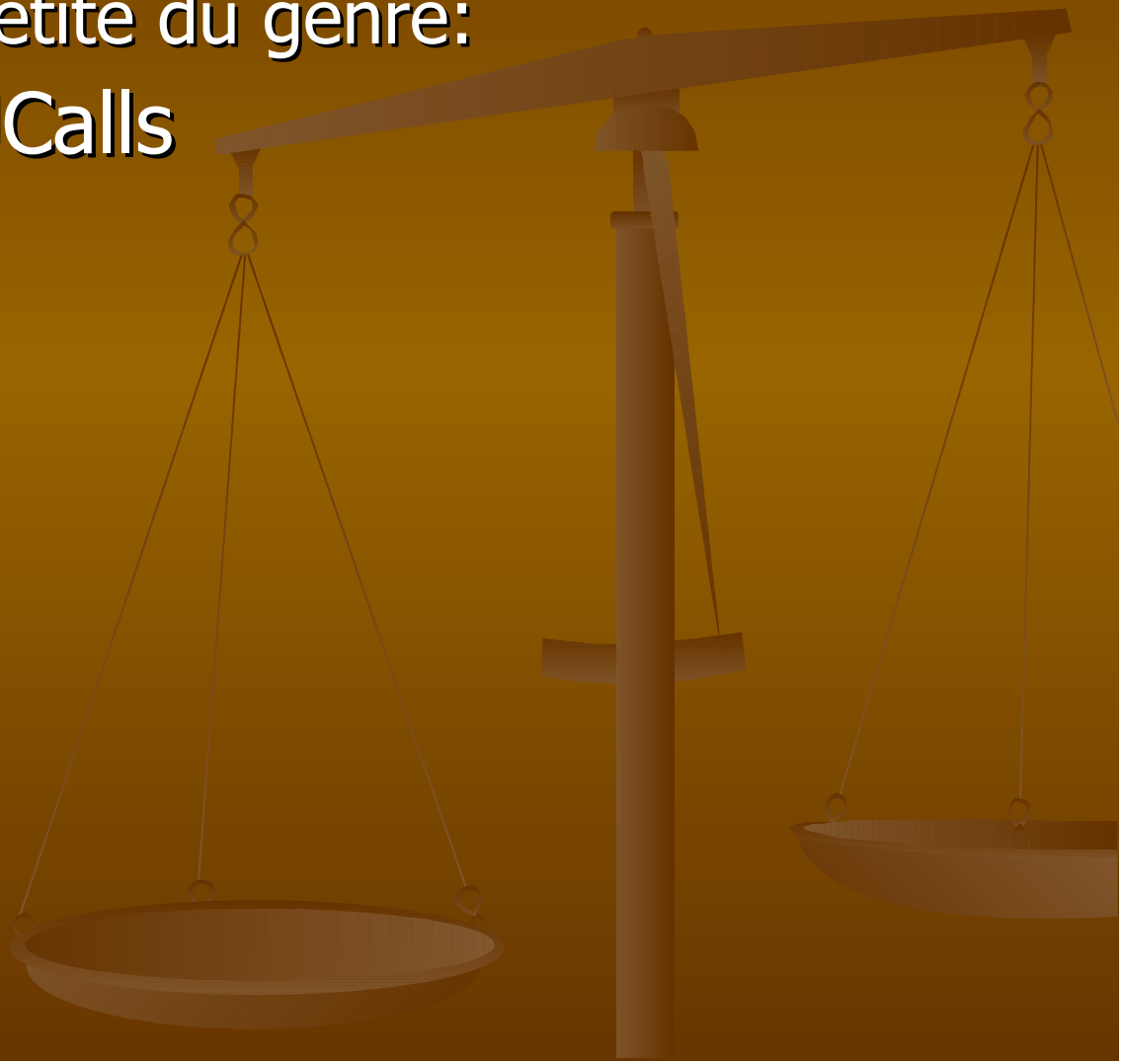
- Le Blob
- Le Lava Flow
- Le Functional Decomposition
- Le Poltergeist
- Spaghetti Code
- Golden Hammer
- Cut and Paste Programming



Exemple de Blob

Une grande classe qui fait tout avec plus de 60 attributs et une petite du genre:

```
static final class JJCalls
 {
    int gen;
    Token first;
    int arg;
    JJCalls next;
  }
```



Exemple de Functional Decomposition

Une classe qui ne fait qu'une action: avoir une fonction.

```
private static class AnchorAnalyzer extends Analyzer
 {
    public final TokenStream tokenStream(String fieldName, Reader reader)
    {
      return new AnchorFilter(CONTENT_ANALYZER.tokenStream(fieldName, reader));
    }
 }
```

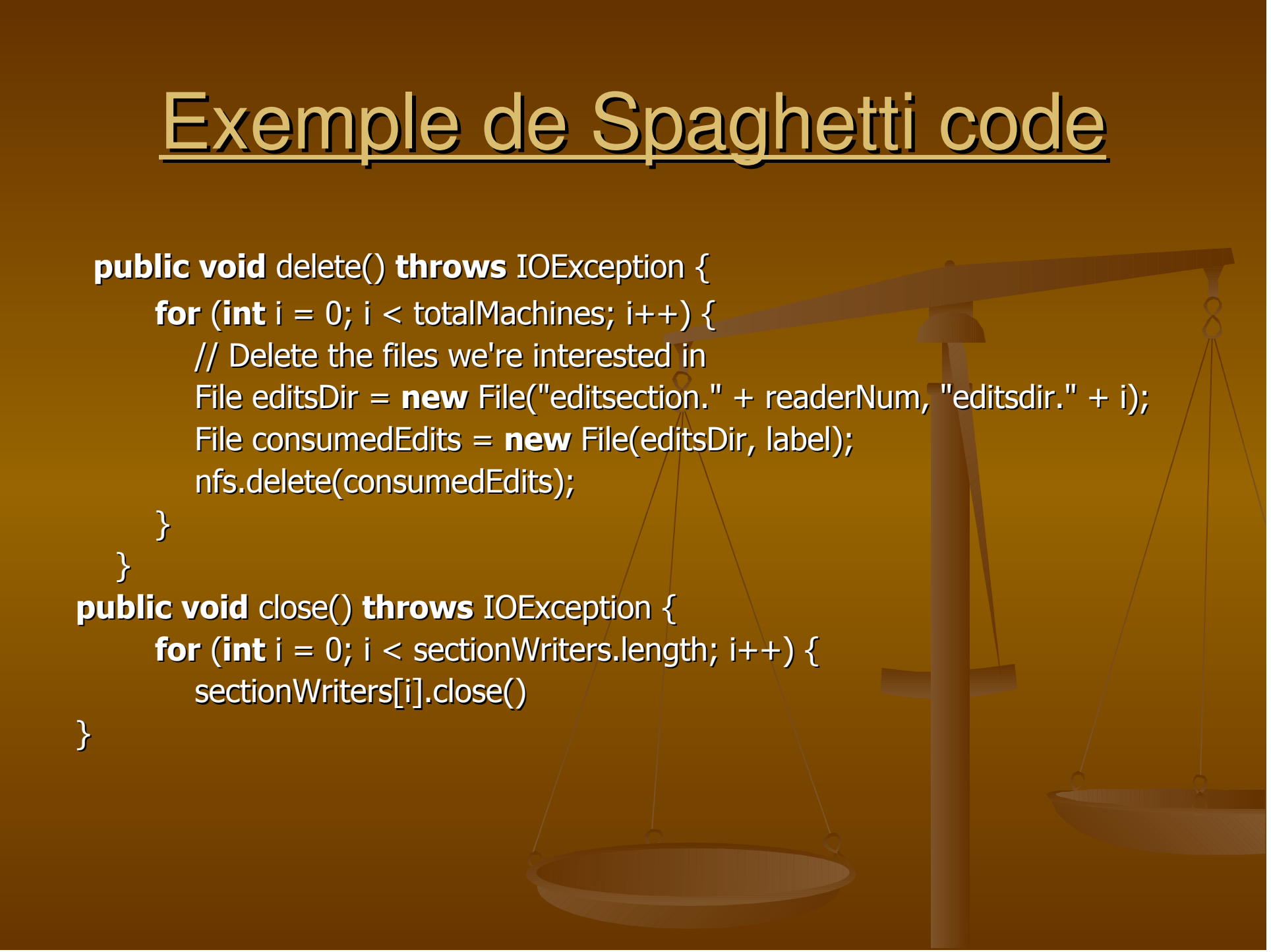
Exemple de lavaflow

Partie d'une fonction:

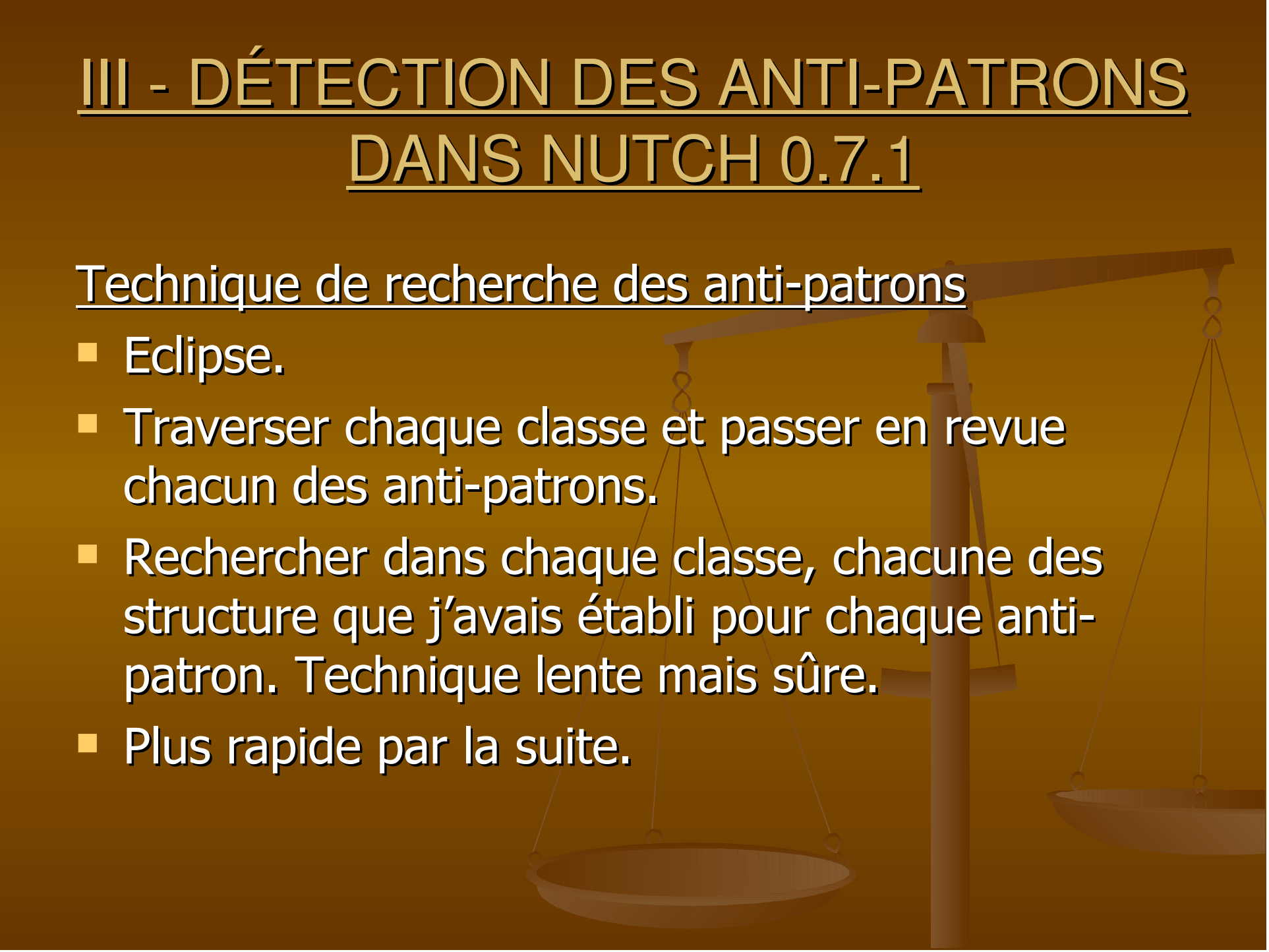
```
if (encoding != null) {                    // found an encoding header
      try {                                // try to use named encoding
        text = new String(content.getContent(), encoding);
      } catch (java.io.UnsupportedEncodingException e) {
        return new ParseStatus(e).getEmptyParse();
      }
    } else
     {
      // FIXME: implement charset detector. This code causes problem
     when
      //       character set isn't specified in HTTP header.
      text = new String(content.getContent());    // use default encoding
    }
```

Exemple de Spaghetti code

```
  public void delete() throws IOException {
        for (int i = 0; i < totalMachines; i++) {
            // Delete the files we're interested in
            File editsDir = new File("editsection." + readerNum, "editsdir." + i);
            File consumedEdits = new File(editsDir, label);
            nfs.delete(consumedEdits);
        }
    }
public void close() throws IOException {
        for (int i = 0; i < sectionWriters.length; i++) {
            sectionWriters[i].close()
}
```



III - DÉTECTION DES ANTI-PATRONS DANS NUTCH 0.7.1



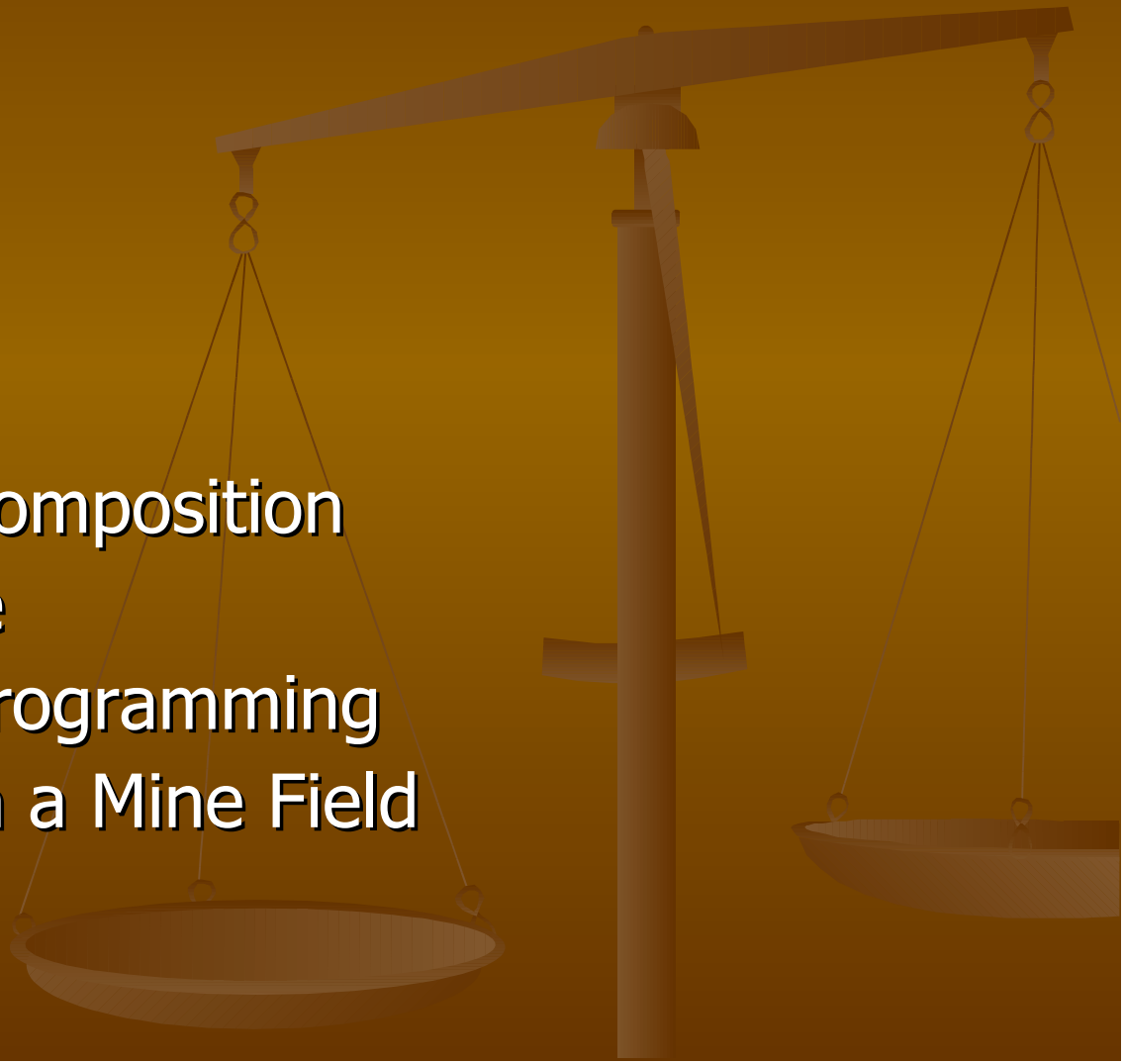
Technique de recherche des anti-patrons

- Eclipse.
- Traverser chaque classe et passer en revue chacun des anti-patrons.
- Rechercher dans chaque classe, chacune des structure que j'avais établi pour chaque anti-patron. Technique lente mais sûre.
- Plus rapide par la suite.

RÉSULTATS OBTENUS

Sur 482 classes :

- 9 Blobs
- 34 Lava Flow
- 7 Poltergeist
- 68 Functional Decomposition
- 51 Spaghetti Code
- 5 Cut and Paste Programming
- 2 Walking through a Mine Field



IV- DISCUSSION ET CONCLUSION

Après avoir travaillé 4 mois sur les anti-patrons j'ai pu faire les constats suivants:

- La recherche demande d'avoir un certain sens critique.
- La présence d'un anti-patron ou d'un autre est un sujet très discutable.
- La présence de certains types d'anti-patrons peut en entraîner d'autres.
- Certains anti-patron comme le Golden Hammer ne sont pas detectables(choix d'outil, classes qui ne servent a rien, utilisation de code plus maintenu, développeurs en contact avec les utilisateurs?).

MINI ANTI-PATRONS

- Le Continuous Obsolescence(+ieurs versions du logiciel)
- Le Boat Anchor(partie du code sans objectif)
- Le Dead End(utilisation de code qui n'est plus maintenu)
- Le Input Kludge (on laisse l'usager faire tout)
- Le Walking through a Mine Field(on pense que le logiciel marche jusqu'a ce qu'on le test sur d'autres platformes)
- Le Mushroom Management (le develloppeur n'est pas en contact avec les usagers finaux)