



Projet GL

Tests

Projet Génie Logiciel – 3 grandes dimensions

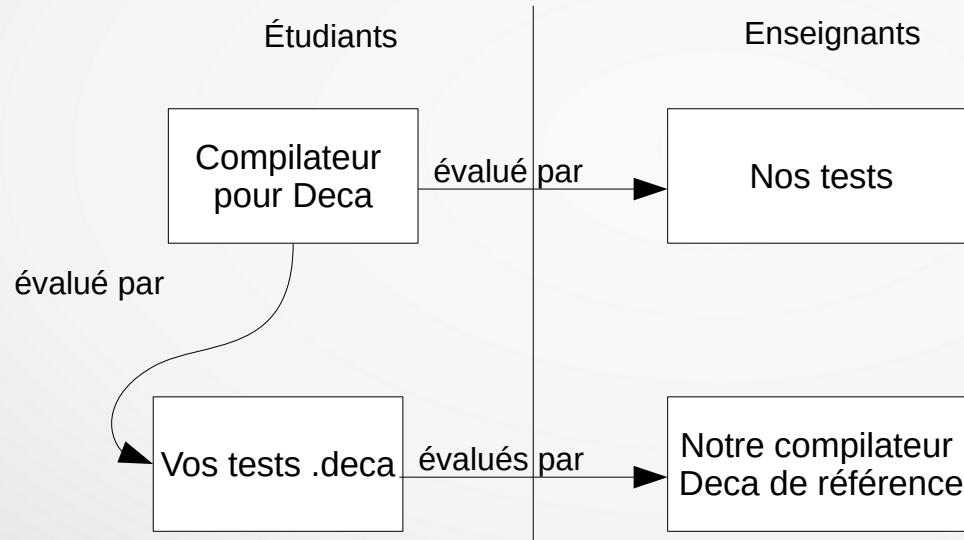
- 1) Implémenter un compilateur pour Deca.
- 2) Faire une gestion de projets pertinente.
- 3) Écrire une base de tests pertinente.

- Au delà de l'implémentation, une partie importante du génie logiciel est la **qualité** du logiciel produit.
- Une partie de l'évaluation du PGL porte sur les tests produits à fin d'assurer la qualité du compilateur implémenté.
- ch. [Tests]

- Il faut consacrer une partie importante du **temps** à la création de tests **pertinents**.
 - Pas respectif, discriminants.
- Slide #51

Base de tests

- Formée des fichiers .deca, contenant des exemples de programme deca valides et invalides.



- Ceci pour bien s'assurer de la couverture de tests.

Jacoco – couverture de tests

- On mesure la couverture de vos tests sur notre compilateur
- Score mutationnel de vos bases de tests :
 - Capacité de « tuer » des compilateur mutants
- Objectif : Couvrir toutes les lignes de code de notre compilateur (100%)
- ~500 tests

Conventions

- Conventions imposées sur le nommage et l'architecture des tests.
 - Tests pour l'étape A dans `syntax`
 - Tests pour l'étape B dans `context`
 - Tests pour l'étape C dans `codegen`
 - Slide #48
 - Section 3 du ch [Tests]
 - L'évaluation est automatique

Exemples de test pour l'étape A

(analyse lexicale et syntaxique)

- Test **valide**
 - Fichier hello.deca :

```
{println("Hello");}
```
- Test **invalide**
 - Fichier hello2.deca :

```
{println("Hello");}  
{println("world!");}
```


Exemples de test pour l'étape B (analyse contextuelle)

- Test **valide**
 - Fichier assign.deca :

```
{  
    int x;  
    x = 0 ;  
}
```
- Test **invalide**
 - Fichier assign2.deca :

```
{ x = 0 ; }
```

Exemple de test pour l'étape C (génération de code)

- Il faut déjà que le programme deca compile
- Affichage à l'écran du résultat attendu, puis vérification par script Shell.
- Test **valide**

– Fichier bool.deca :

```
{  
    boolean b1, b2 ;  
    b1 = false ;  
    b2 = b1 ;  
    if (b2) {println ("ko");}  
    else    {println ("ok");};  
}
```

Exemple de test pour l'étape C - Automation (génération de code)

- Ex : inspiré de ...script/basic-gencode.sh fourni
- Ce script lance un test deca et vérifie la sortie à l'écran. Aller plus loin : une boucle pour lancer plusieurs tests.

```
{...
  rm ./src/test/deca/codegen/valid/bool.ass ...
  decac ./src/test/deca/codegen/valid/bool.deca ...
  if [ ! -f ./src/test/deca/codegen/valid/bool.ass ]; then
    echo "Fichier cond0.ass non généré."
    exit 1 #the test failed
  fi

  resultat=$(ima ./src/test/deca/codegen/valid/bool.ass) ...
  rm ./src/test/deca/codegen/valid/bool.ass

  # On code en dur la valeur attendue.
  attendu=ok

  if [ "$resultat" = "$attendu" ]; then
    echo "Tout va bien"
  else
    echo "Résultat inattendu de ima:"
    echo "$resultat"
    exit 1
  fi
}
```

Exemple de test pour l'étape C - Automation (génération de code)

- But : pouvoir exécuter facilement tout un tas de tests facilement avec `mvn test`.
- Poly, ch[Tests], 1.6 - Automatisation des tests
- Slide #50

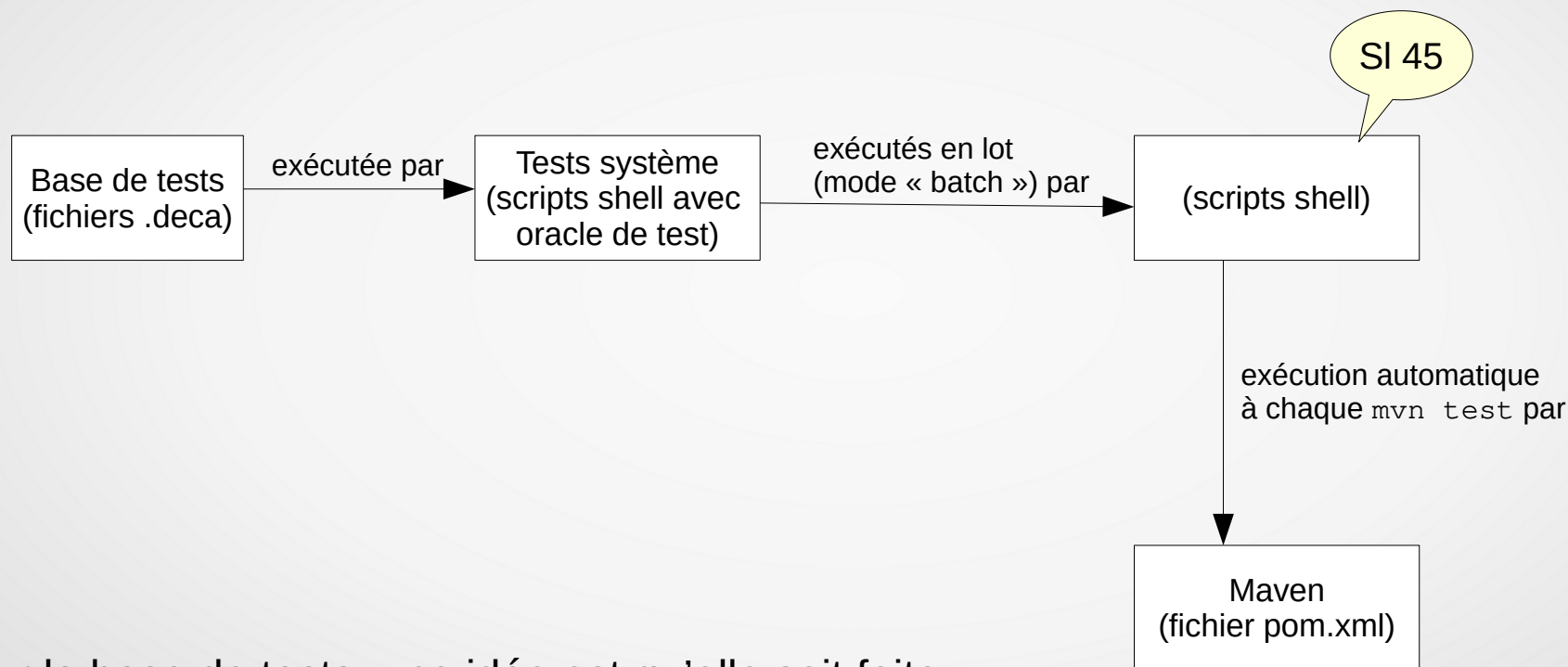
Exemple de test pour l'étape C (génération de code)

- Test **invalid**

- Fichier division.deca :

```
{  
    println("Resultat attendu : Erreur :  
           division par zero") ;  
    println(1 / 0) ;  
    println("On n'a pas le resultat attendu") ;  
}
```

Schéma globale de tests



Pour la base de tests, une idée est qu'elle soit faite par une personne <> de celle qui a codé cette partie du compilateur (meilleure confiance sur la compréhension de la spécification).

D'autres tests – en bonus

- Tests unitaires avec JUnit
 - Cas simples (profils des opérateurs binaires et unaires, relations de sous-typages, assign_compatible, ...)
 - Complicé pour l'étape C, qui requiert un AST en entrée.
 - Automatisation déjà intégrée dans Maven
- Tests d'intégration
 - Ex :
`...src/test/java/fr/ensimag/deca/syntax/ManualTestLex.java`
 - Exécuté par le script `test_lex` pour l'étape A