Test de Programmes et Inspection de Code

Pb: Écrire et maintenir du code relativement sûr, conséquent et performant.

I Analyse Dynamique

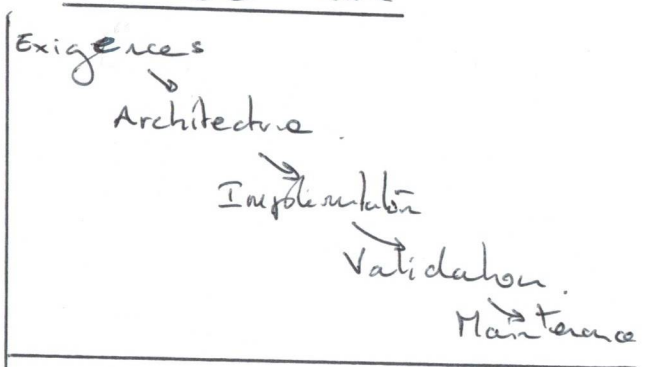
Def°: Étude par l'exécution d'une partie ou la totalité du code.

A) Processus de test

La manière dont on teste et ce qu'on teste dépend du processus de dev et de l'objectif.

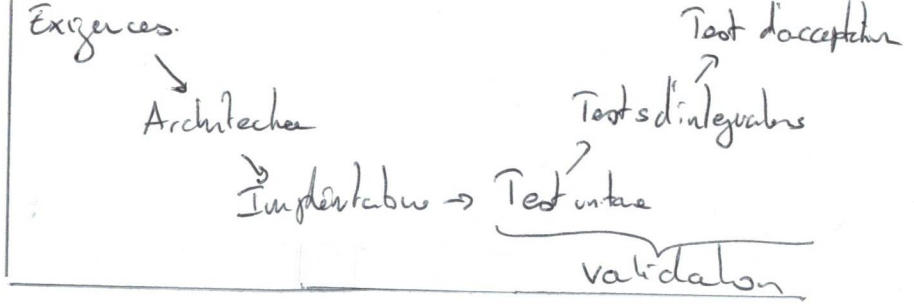
1) Modèle Linéaire

Cycle en Cascade :



Adapté de l'ingénierie classique, pour l'électronique embarquée...

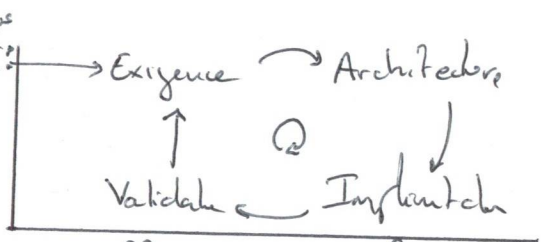
Cycle en V :



Met en évidence une hiérarchie des tests.

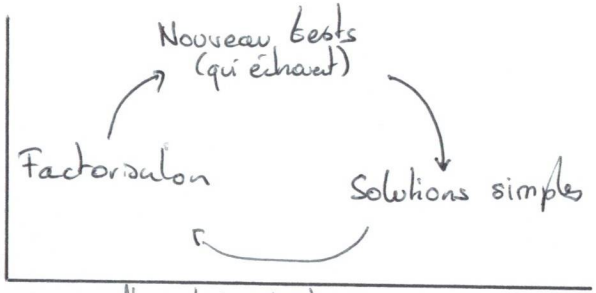
2) Processus Itératifs

Processus Itératifs :



Les raffinements successifs nous donnent plus de flexibilité mais contraignent les possibilités futures.

Dev. dirigé par les tests (TDD) :



Est lié originellement à l'extreme programming.

Intégration Continue : Automatisation dans le flot de travail pour vérifier que les mises à jour du code n'entraînent pas de conflits.

Ex: Revue de code par les pairs, tests automatiques

B) Cadre d'information

- Test en boîte blanche : on connaît parfaitement la structure
- " " " noire : aucune supposition sur la structure interne
- " " " grise : accès à la documentation ou autre information sur le code sans en disposer directement.

C) Indicateurs de performances

Def : Quantités ou qualités établies par les tests.

Ex : Taux de passer les tests, temps d'exécution, mémoire occupée.

1) Couverture

Def (Couverture de test) : Part des entrées possibles couvertes par les tests.

→ Difficile à qualifier

Def (Couverture de code) : Part du code explorée lors de l'exécution des tests.

Plusieurs unités possibles : fonction/méthodes, lignes de codes, branches, chemins ....

2) Metriques personnalisées

Peuvent demander du travail à définir correctement. Dépendent beaucoup de l'objectif du programme.

Ex : Précision/rappel en apprentissage stat., experience utilisateur, temps de réactivité ....

D) Programmation par contrat

Def : Processus de développement mettant en avant ce qui doit être vrai avant, après et au court d'un traitement.

Peut prendre la forme d'assertion dans le code au lieu des tests. Peut servir à générer des tests.

Ex : Hypothesis par python, Qcheck en Ocaml.

II Analyse statique

Def : Étude d'un programme sans recourir à son exécution directe.

A) Clarté et Maintenabilité

Un code est généralement plus lu qu'écrit.

1) Mise en page, formattage

Ex : tabulations vs espaces, largeur de code, ...
Se gère assez bien automatiquement, nécessaire pour utiliser un gestionnaire de version.

2) Conventions de langage

Conventions générales du langage, et celles spécifiques au projet

Ex : Conventions de nommage, manière d'organiser les blocs, de découper le code ...

Ce NE sont PAS les règles imposées par le langage (Ex : les majuscules en début de constructeur en Ocaml)

3) Documentation

Ex : docstring, commentaires, annotations ...

B) Optimisation

Heuristiques sur la syntaxe lors de la compilation/interpolation

Ex : Jump threading, Inlining ....

C) Typage Statique.

1) Definitions

Def (type de donnée): Attribut de données qui informe le compilateur ou interpréteur de comment les données sont utilisées. Peut être représenté comme un ensemble.

Def (typage statique): Analyse du code pour vérifier des propriétés de sûretés sur les types serait vérifié lors de l'exécution. Se distingue du typage dynamique qui se fait lors de l'execution.

Ex: Dynamique: Python ; Statique: Ocaml, C.

2) Annotation de typage en Python.

On peut annoter le code en python qu'on peut vérifier avec des outils.

Syntaxe:

```
def f(x:int) -> int:
    y:int
    y = -x
    return y.
```

Ex: MyPy, Pyright, Pytype ...

Dev 1 Exemple de programme qui vérifie certaines propriétés de typage sur un fragment de Python.

D) Preuve de programme.

Motivation: Sûreté
Pb: Indécidabilité.
Solution: Formalisme de preuve de programme:
Dev 2: Logic de Hoare

1) Vérification de modèle finis

Tests exhaustifs sur tous les états et transitions possibles.
Ex: Programme pour petit hardware,
Pb: Impensable quand le nombre d'état augmente.

2) Assistants de preuve

Def: Outil logiciel qui guide et aide l'utilisateur à prouver la correction de code.
Ex: Coq, Isabelle ...

Conclusion: Beaucoup de solutions différentes pour des problèmes différents.

Ouvertures: Approfondissement sur le typage statique,