

## T.1

## Tester le logiciel réalisé

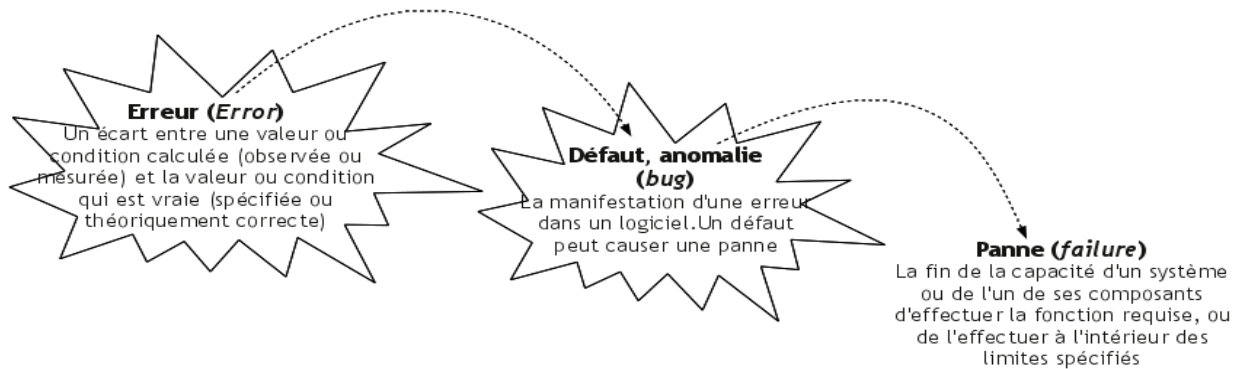
## Objectif

Rechercher une anomalie (défaut, appelé souvent bug) dans le comportement d'un logiciel.

## Préambule

## Principe d'une panne (IEEE 729)

A l'origine, il y a la Faute (mistake) : c'est la cause d'une erreur, puis :

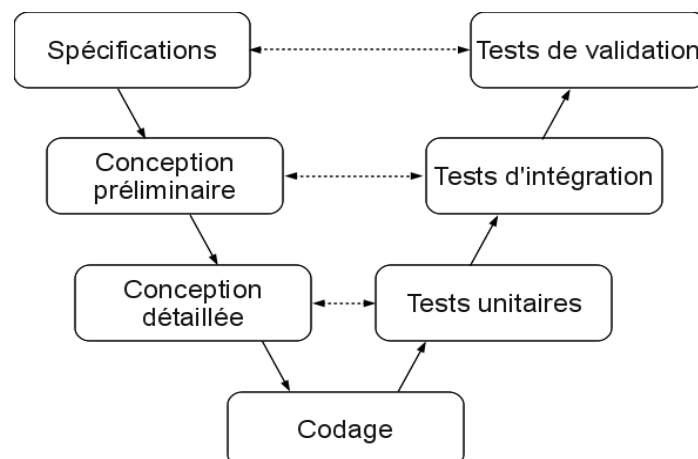


## Méthode

Si les tests se déroulent parmi les activités finales du projet, ils se préparent dès le début. Il existe différentes formes de tests :

- Tests de fonctionnements (**unitaires et intégrations**) : pour vérifier que le produit est bien fait (sans anomalies)
- Tests de **validation** : permettent de vérifier que le produit réalisé est le bon
- Tests d'architecture : pour vérifier que le produit est utilisable dans son environnement d'exploitation.

Ces tests, quels qu'ils soient, sont **planifiés à l'avance**, dans l'étape correspondante à leur niveau de détails. Cela signifie qu'il faut, dans chaque étape du processus, spécifier quels tests seront effectués, à quel moment, par qui, dans quelles conditions, avec quels moyens et surtout de quelle manière.



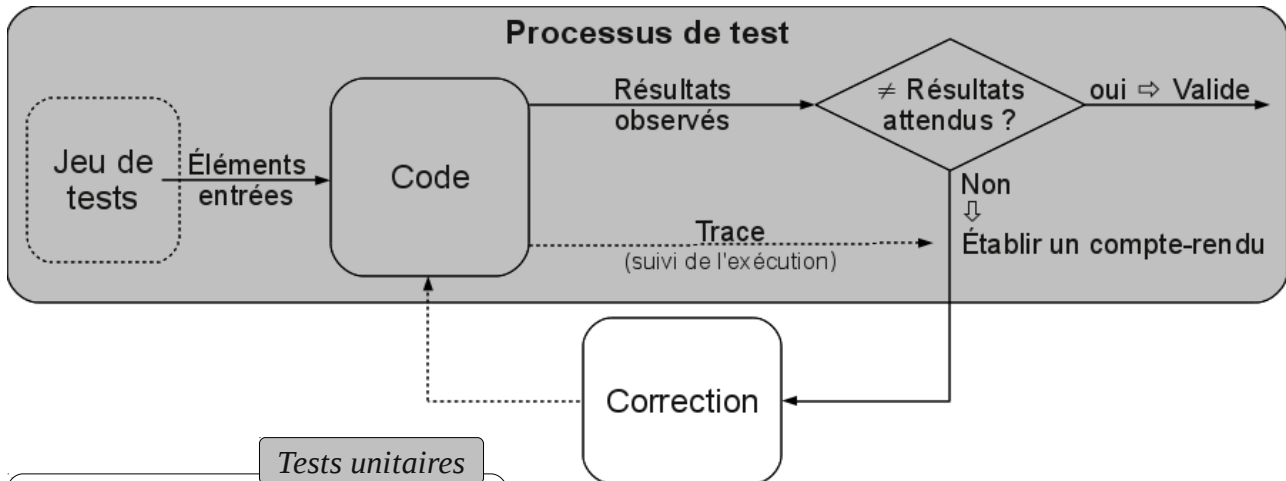
### Remarques

Si une batterie de tests ne montre pas de défaut cela n'implique pas que le logiciel est quand même exempt de défaut ...

Le test n'a pas pour objectif : de diagnostiquer la cause des erreurs, de corriger les fautes ou de prouver la correction.

On évalue à environ 40% la part des tests dans le coût d'un logiciel (et plus pour des logiciels critiques).

L'investissement dans les tests a donc un coût très important dans un projet logiciel.



### Tests unitaires

Chaque module du logiciel est testé séparément par rapport à ses spécifications. Ils sont planifiés lors de la conception détaillée. Ils permettent de tester "les plus petites unités testables" : méthodes, classes.

### Tests d'intégration

Les modules validés par les test unitaires sont rassemblés dans un composant logiciel. Le test d'intégration vérifie que l'intégration des modules n'a pas altéré leur comportement. Ils sont planifiés en conception préliminaire et ils évaluent les différentes unités intégrées (packages, groupement de classes) mais aussi les interactions matérielles-logicielles en cas de système embarqué. Ils utilisent les diagrammes d'interactions (séquence et/ou collaboration).

### Tests de validation

Conçus dès les spécifications, ils permettent de vérifier l'adéquation du produit aux exigences fonctionnelles du client. Les points de départ sont les cas d'utilisation et les scénarios. Ces tests vérifient donc que le logiciel réalisé correspond bien aux besoins exprimés par le client. La validation ou vérification d'un produit cherche donc à s'assurer qu'on a construit le bon produit.

### Tests de recette

L'application doit fonctionner dans son environnement de production, avec les autres applications présentes sur la plate-forme et avec le système d'exploitation. Les utilisateurs vérifient sur site que le système répond de manière parfaitement correcte.