



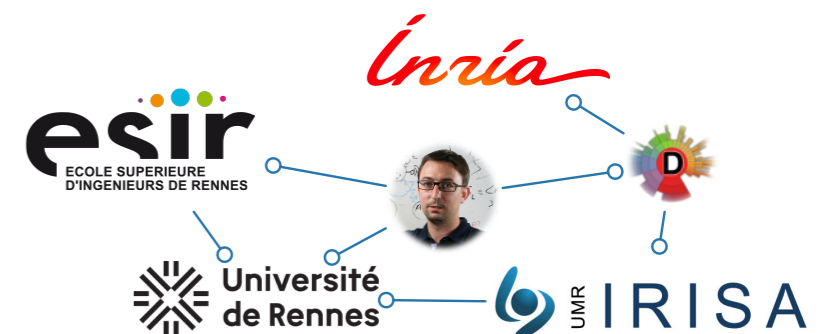
VALIDATION & VERIFICATION

INTRODUCTION TO SOFTWARE TESTING

UNIVERSITY OF RENNES, ESIR, 2023-2024

BENOIT COMBEMALE
FULL PROFESSOR, UNIVERSITY OF RENNES, FRANCE

[HTTP://COMBEMALE.FR](http://combemale.fr)
[BENOIT.COMBEMALE@IRISA.FR](mailto:benoit.combemale@irisa.fr)
[@BCOMBEMALE](https://twitter.com/bcombemale)



Tester pour prévenir...

- ... une erreur du développeur introduit ...
 - *Une erreur est une décision inappropriée ou erronée, faite par un développeur, qui conduit à l'introduction d'un défaut.*
- ... un défaut dans le système qui provoquera ...
 - *Un défaut est une imperfection dans un des aspects du système qui contribue, ou peut potentiellement contribuer, à la survenance d'une ou de plusieurs défaillances*
 - *Parfois, il faut plusieurs défauts pour causer une défaillance.*
- ... sa défaillance à l'exécution.
 - *Une défaillance est un comportement inacceptable présenté par un système.*
 - *La fréquence des défaillances reflète la fiabilité.*

Tester pour prévenir...



Le test : une première définition...

« Le test est un processus manuel ou automatique, qui vise à établir qu'un système vérifie les propriétés exigées par sa spécification, ou à détecter des différences entre les résultats engendrés par le système et ceux qui sont attendus par la spécification »

Extrait de la norme IEEE-STD729, 1983.

Le test

Essayer pour trouver des bugs.

Le test

Essayer pour voir si ça marche.

Le test

trouver des bugs

Essayer pour voir si ça marche.

- Apprendre
 - pourquoi c'est fait
 - ce que ça doit faire
 - comment c'est fait
 - comment ça marche
 - Modéliser
 - S'en faire une idée
 - Exécuter
 - Analyser
- Qu'y a-t-il à voir?
 - Que faut-il regarder?
 - Qu'est-ce qui est visible?
 - Qu'est ce qu'on cherche?
 - Comment le regarder?
- Qu'est ce qui devrait marcher?
 - Identifier une erreur
 - Diagnostiquer une erreur
 - Catégoriser ces erreurs

Qu'est-ce qu'on teste? quelles propriétés?

- Fonctionnalité
- Sécurité / intégrité
- Utilisabilité
- Cohérence
- Maintenabilité
- Efficacité
- Robustesse
- Sûreté de fonctionnement
- Etc.

Comment on teste?

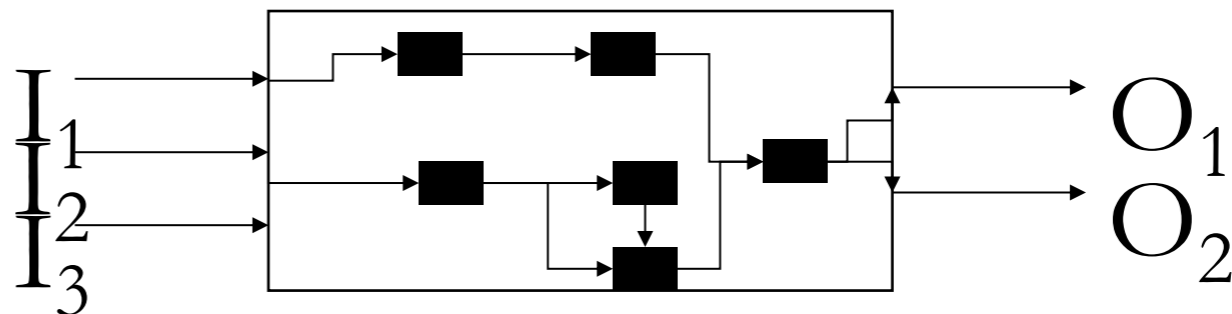
- Test statique
 - relecture / revue de code
 - analyse automatique (vérification de propriétés, règles de codage...)
- Test dynamique
 - on exécute le programme avec des valeurs en entrée et on observe le comportement

Comment on teste?

- Test fonctionnel (test boîte noire)
 - Utilise la description des fonctionnalités du programme



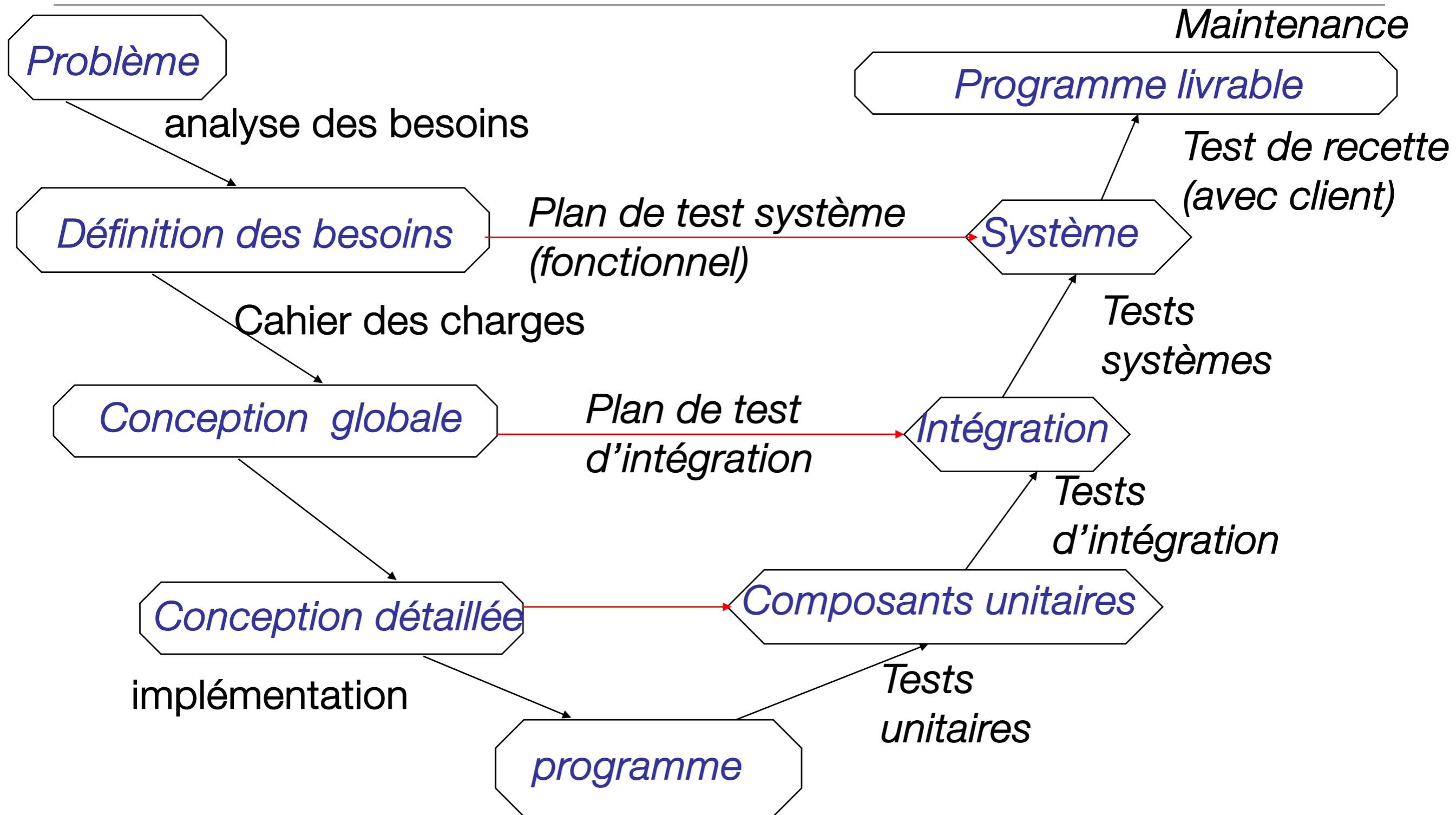
- Test structurel (test boîte blanche)
 - Utilise la structure interne du programme



Avec quoi on teste?

- Une spécification: exprime ce qu'on attend du système
 - des règles de codage
 - un cahier des charges (en langue naturelle)
 - commentaires dans le code
 - contrats sur les opérations (à la Eiffel)
 - un modèle UML
 - une spécification formelle (automate, modèle B...)

Hiérarchisation des tests



Test unitaire

- Validation d'un module indépendamment des autres
- Valider intensivement les fonctions unitaires
- Les unités sont-elles suffisamment spécifiées?
- le code est-il lisible, maintenable...?

Test unitaire

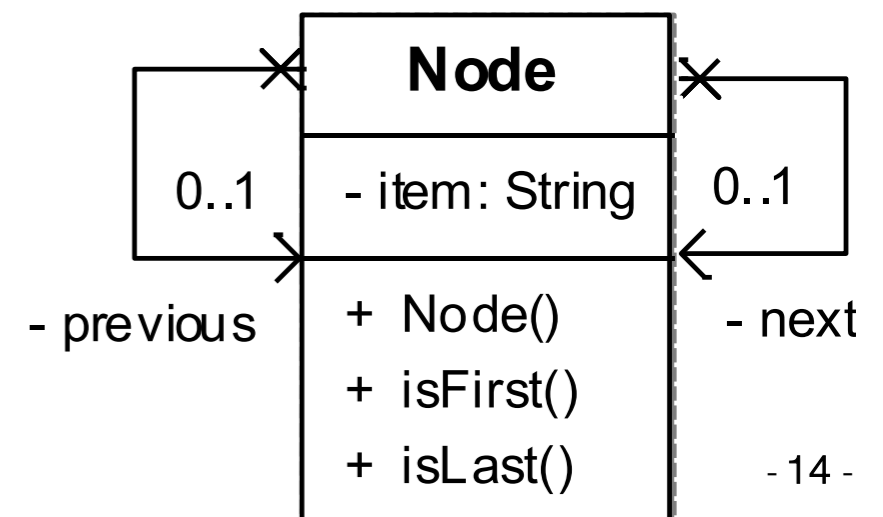
- Pour un langage procédural

- unité de test = procédure

```
void Ouvrir (char *nom, Compte *C, float S, float D )
{
    C->titulaire = AlloueEtCopieNomTitulaire(nom);
    (*C).montant = S ;
    (*C).seuil = D ;
    (*C).etat = DEJA_OUVERT ;
    (*C).histoire.nbop = 0;
    EnregistrerOperation(C);
    EcrireTexte("Ouverture du compte numero ");
    EcrireEntier(NumeroCourant+1);
    EcrireTexte(" , titulaire : \");
    EcrireTexte(C->titulaire); EcrireCar("");
    ALaLigne();
}
```

- Dans un contexte orienté objet

- unité de test = classe



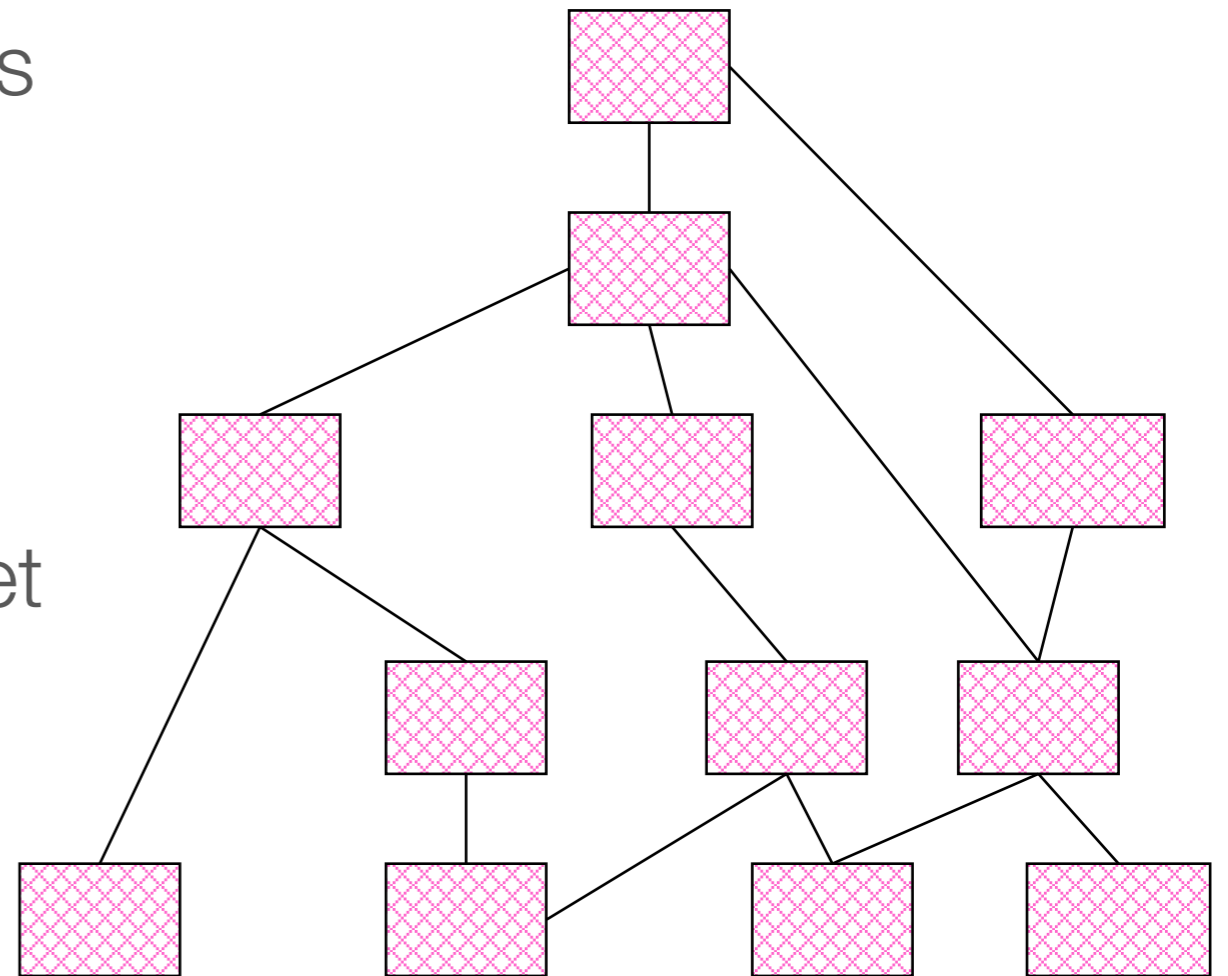
Test d'intégration

- Choisir un ordre pour intégrer et tester les différents modules du système

Test d'intégration

Cas simple: il n'y a pas de cycle dans les dépendances entre modules

Les dépendances forment un arbre et on peut intégrer simplement de bas en haut

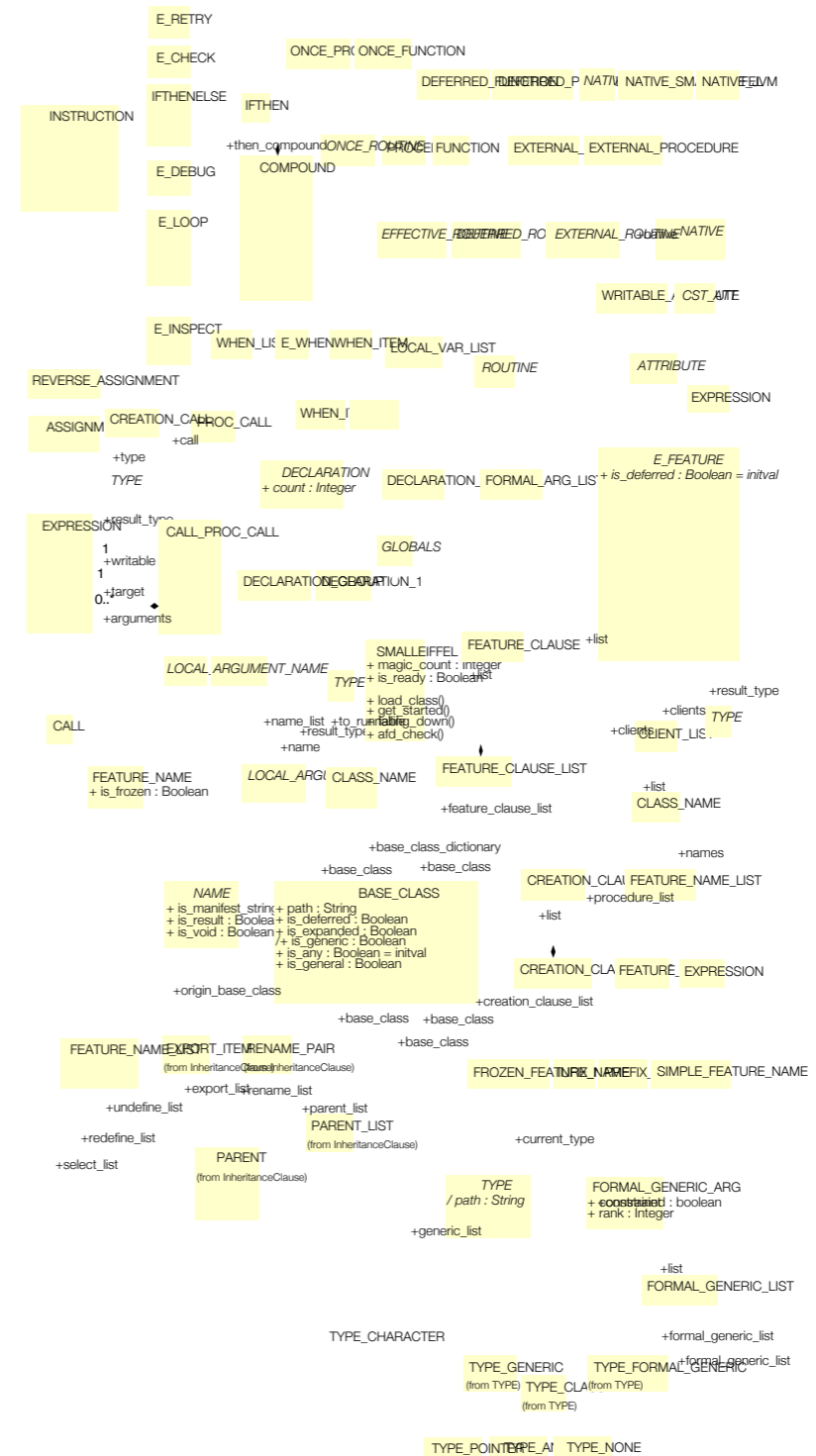


Test d'intégration

Cas plus complexe: il y a des cycles dans les dépendances entre modules

Cas très fréquent dans les systèmes à objets

Il faut des heuristiques pour trouver un ordre d'intégration



Test système

- Valider la globalité du système
 - Les fonctions offertes
 - La qualité du système
 - charge, ergonomie, sécurité, etc.
 - A partir de l'interface

Test de non-régression

- Vérifier que des modifications apportées au logiciel n'ont pas introduit de nouvelles erreurs
 - vérifier que ce qui marchait marche encore
- Dans la phase de maintenance du logiciel
 - Après refactoring, ajout/suppression de fonctionnalités
- Après la correction d'une faute

Regression:
"when you fix one bug, you
introduce several newer bugs."



Best Ways To Test System Functionality

blog.bytebytego.com

Process	Illustration	Tools
Unit Testing		 JUnit 5
Integration Testing		 cucumber
System Testing		
Load Testing		
Error Testing		
Test Automation		

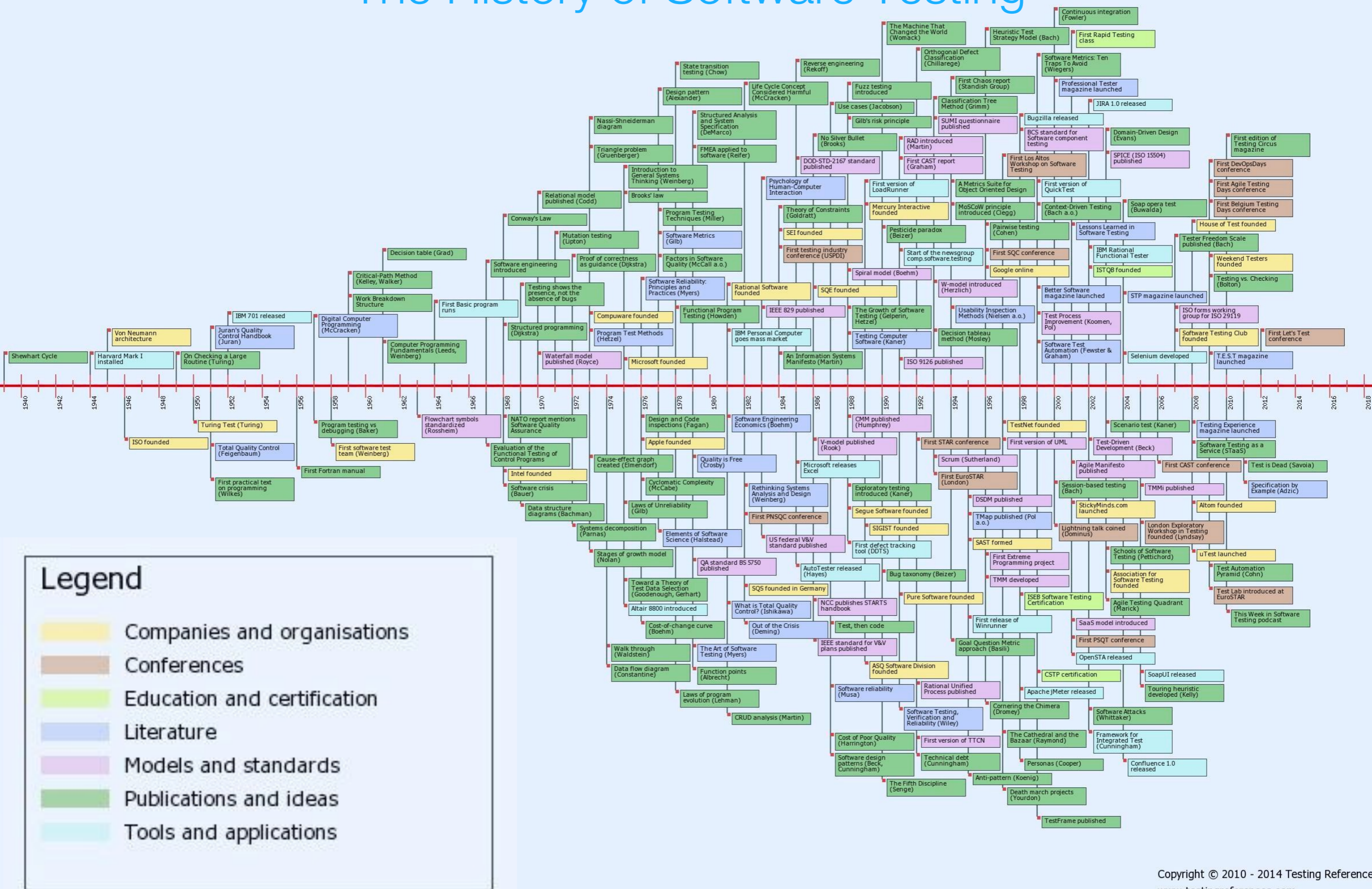
Développer du logiciel pour tester du logiciel

- Test unitaire
 - drivers (lanceur des tests), oracle (succès/échec), instrumentation (mesure couverture)
- Test d'intégration
 - idem
 - + “bouchons” de tests (stubs), pour simuler les modules non disponibles
- Test système
 - test des fonctions
 - + environnement matériel
 - + performances

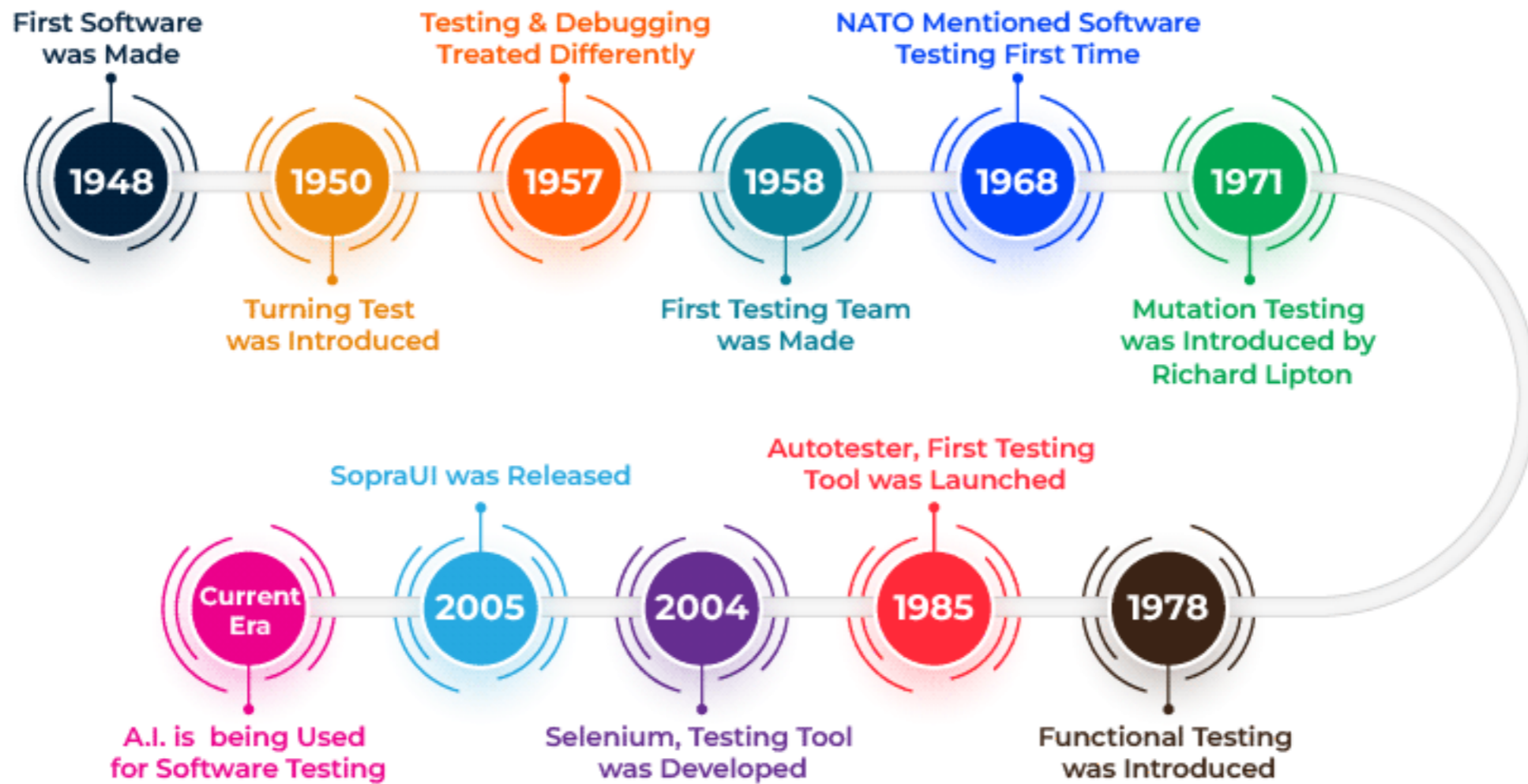
Techniques de test de logiciel

- Plusieurs techniques
 - Dynamique / statique
- Génération de test
 - Fonctionnel / structurel
- Plusieurs niveaux/étapes:
 - Unitaire, intégration, système, de non-régression

The History of Software Testing



SOFTWARE TESTING HISTORY



<https://www.geeksforgeeks.org/history-of-software-testing>

Future of Software Testing (examples)

- Testing metrics
- Test generation & amplification
- Test selection
- Process automation (DevOps)
- Immediate feedback and recommendations
- Oracle problem (e.g., metamorphic testing)
- Etc.