

Validation et vérification de son logiciel scientifique

Introduction

Bug du vol 501 d'Ariane 5 en 1996

→ **Explosion après 36s**

A cause d'un **bug (dépassement de capacité des entiers)**



Guerre du Golfe de 1991 : un anti-missile US Patriot (dont le programme tournait depuis 100H) a raté l'interception d'un missile Irakien Scud

→ 28 morts

Problème de codage des réels
(arithmétique flottante)



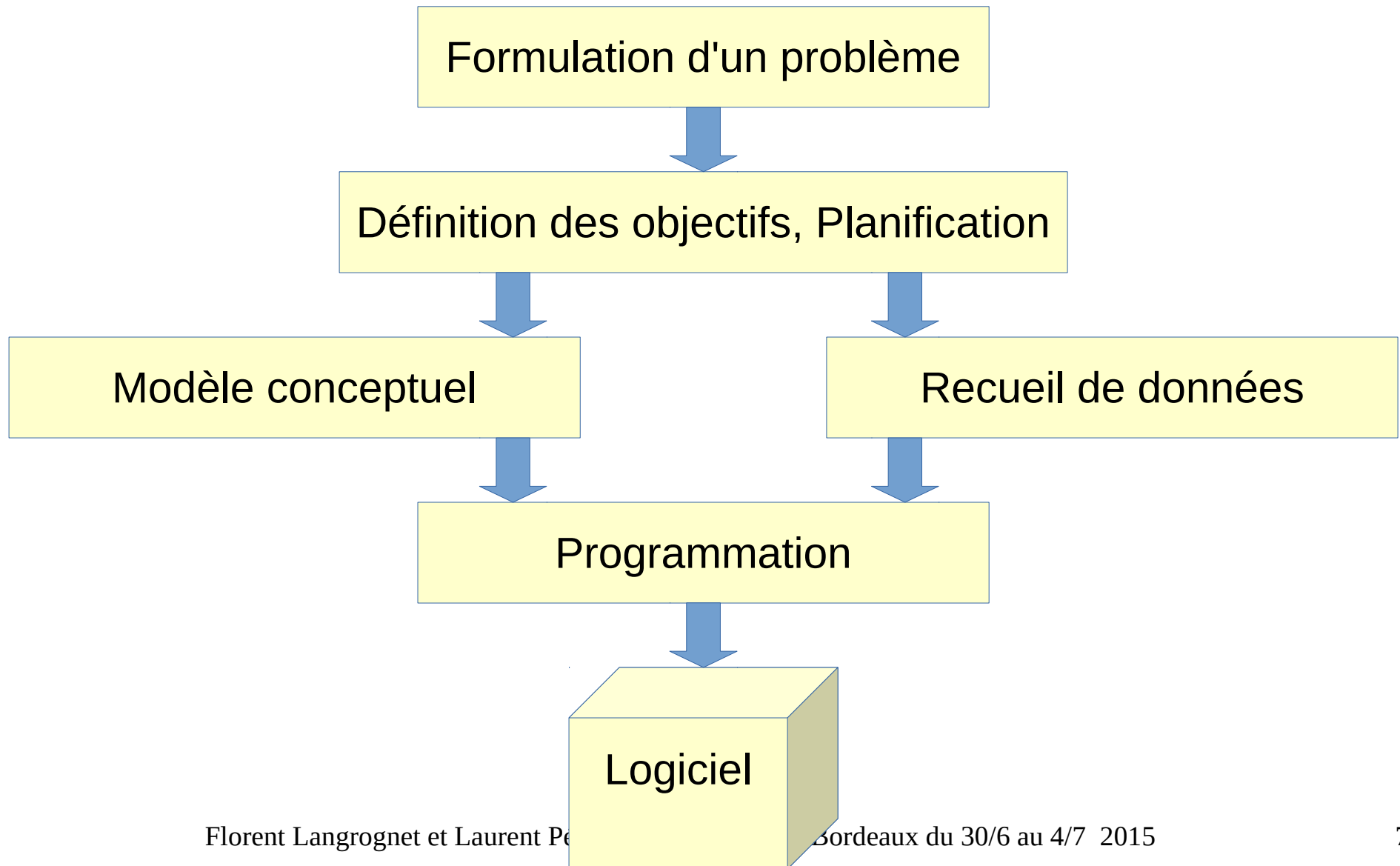


Le logiciel est-il conforme aux attentes ?

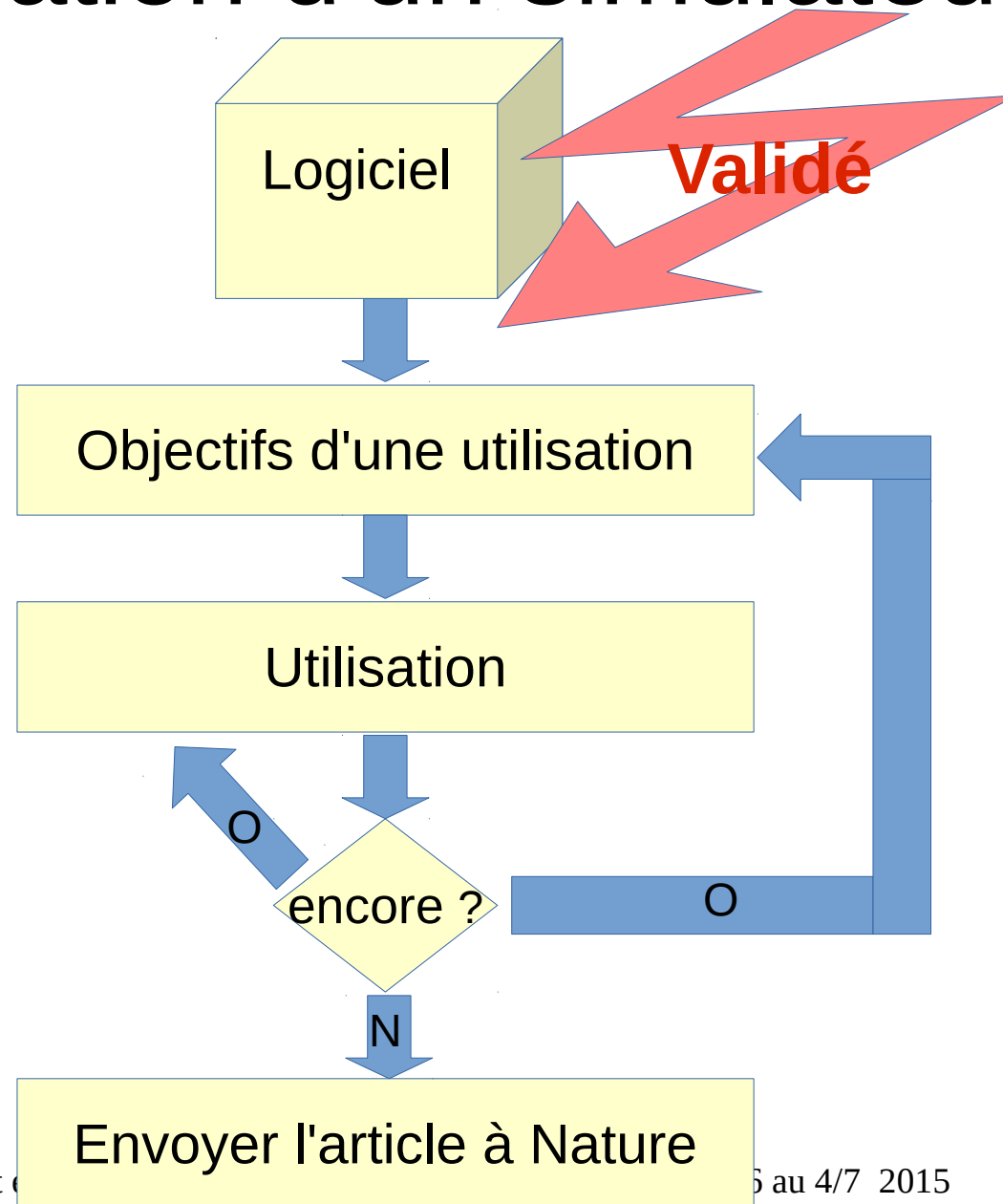
Fonctionne t-il bien ?

Validation - Vérification

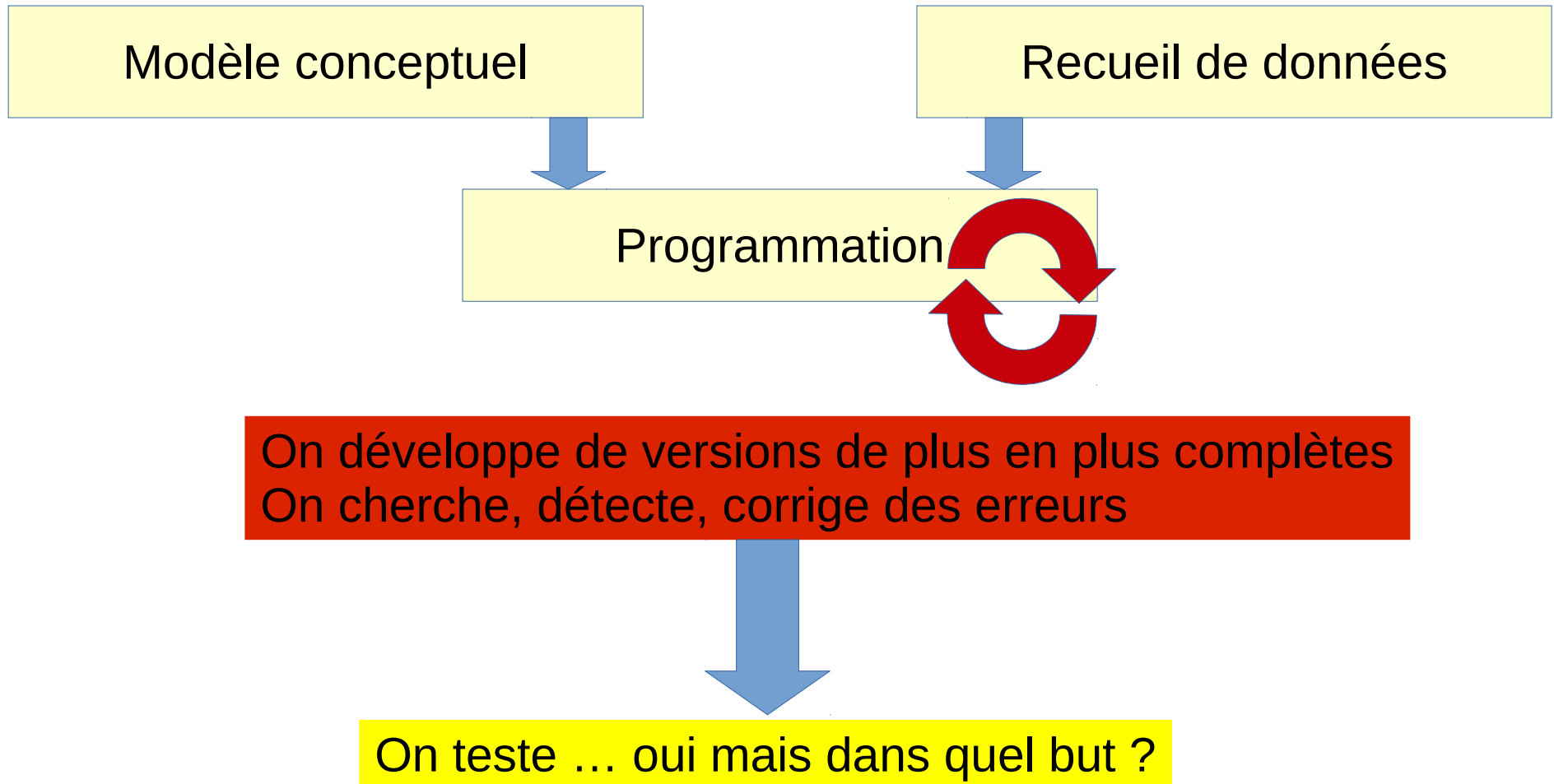
Le développement d'un simulateur



L'utilisation d'un simulateur



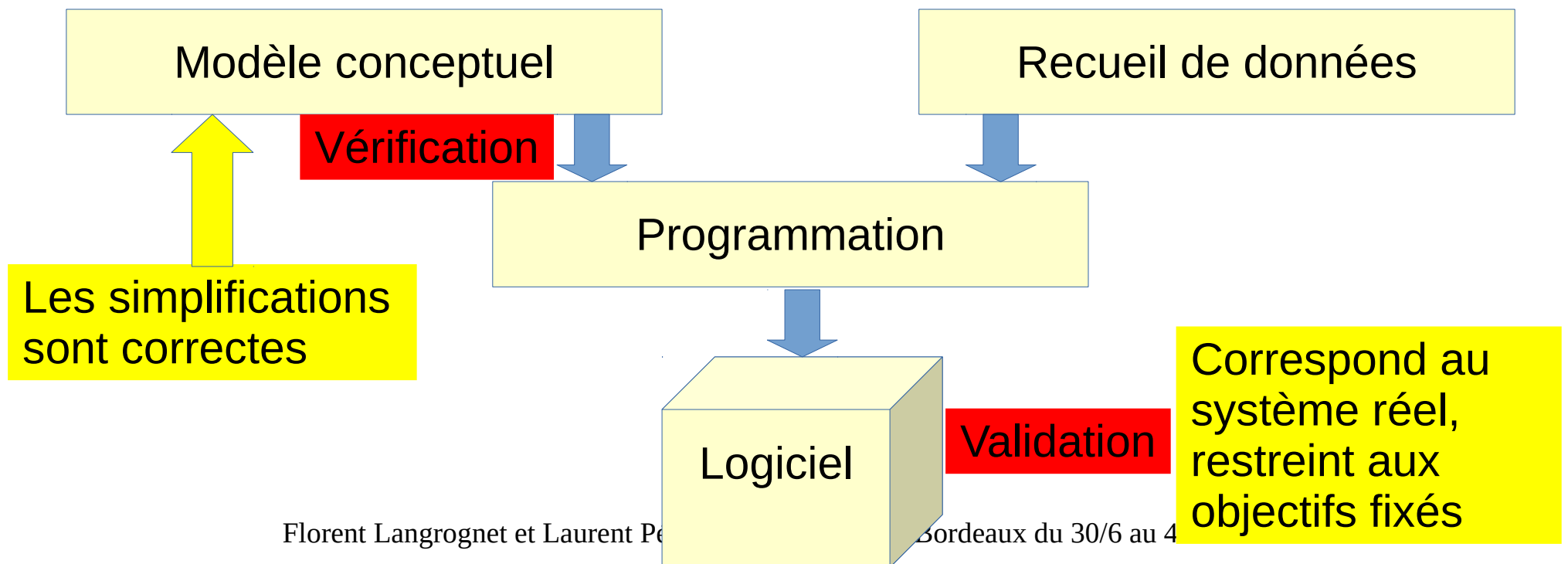
Le cycle du développement



Vérification / Validation / Test (Balci, 2010)

x Verification deals with the assessment of transformational accuracy of the x and addresses the question of "Are we creating the x right ?"

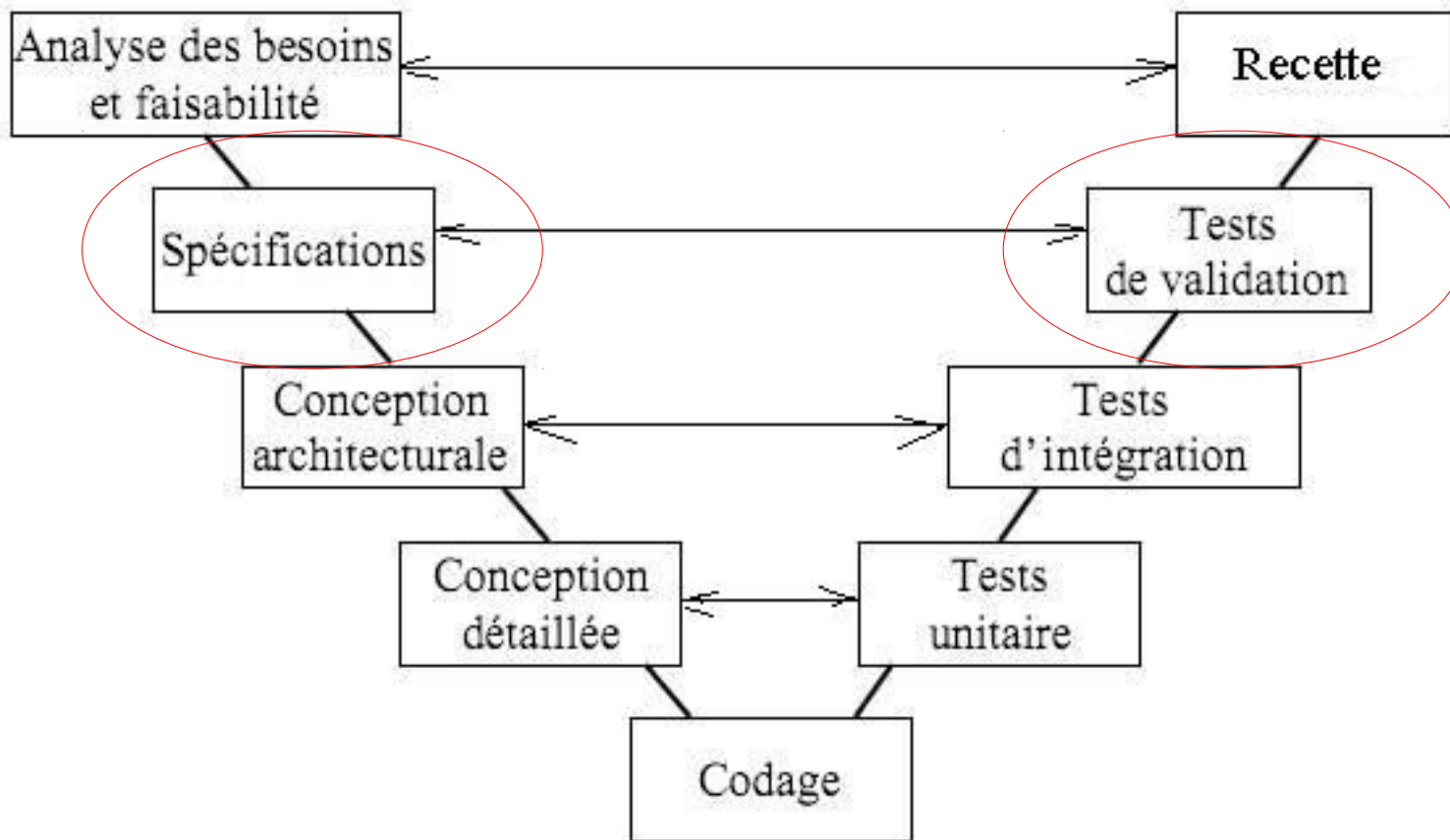
x Validation deals with the assessment of behavioral or representational accuracy of the x and addresses the question of "Are we creating the right x ?"



Validation - Vérification

- Avons-nous construit le **bon** logiciel ?
→ Validé
- Avons-nous **bien** construit le logiciel ?
→ Vérifié

Vues du développeur / testeur



Comment « bien » spécifier ?

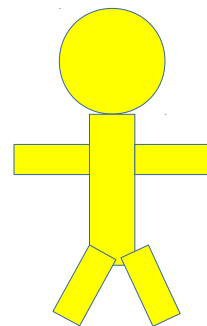
Comment faire le lien avec la vision précédente ?

Acteurs / informations disponibles

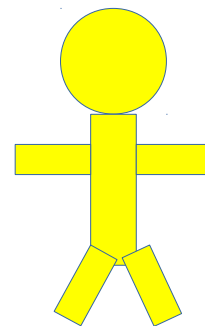
Modèle ... conceptuel, formel ...

Le code

Les sorties fichiers



Développeur



Scientifique

Des rôles différents ... une ou plusieurs personnes ?

Validation

Validation et logiciels scientifiques

- **Dispose t-on de solutions analytiques dans des cas particuliers ?**
 - Permet de valider le logiciel dans des situations particulières
 - Permet d'avoir des estimations de résultats dans des situations proches
- **Dispose t-on des solutions admises par la communauté ?**
 - Validation sur des cas plus complexes

Vérification

Quelles méthodes, quels outils ?

- **Vérification statique**

- Sans exécution

- Ex : méthode formelle (peu/pas utilisé pour les logiciels scientifiques)

- **Vérification dynamique**

- En exécutant le logiciel

Vérification dynamique

Tests

Test Driven Development

Tests d'intégration

Tests unitaires

Tests de non régression

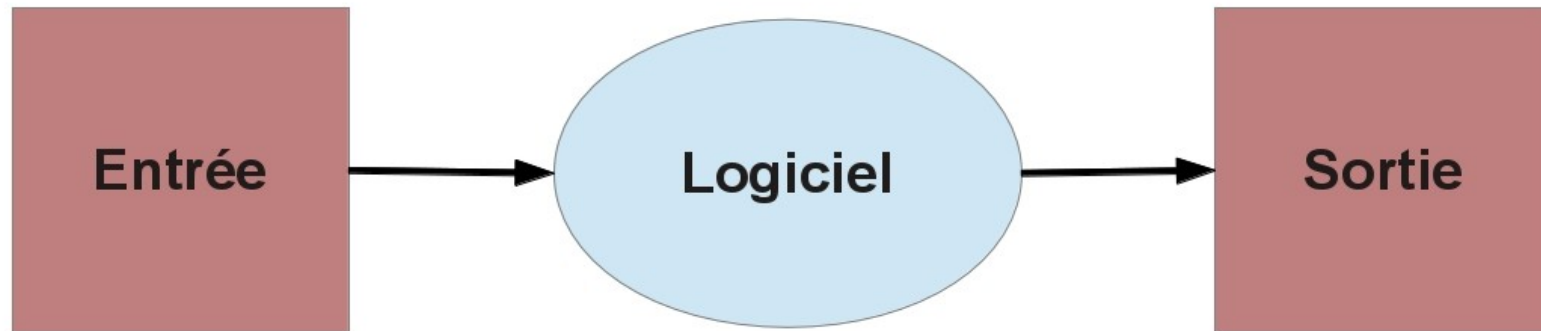
Tests fonctionnels

Tests de validation

Vérification et calculs

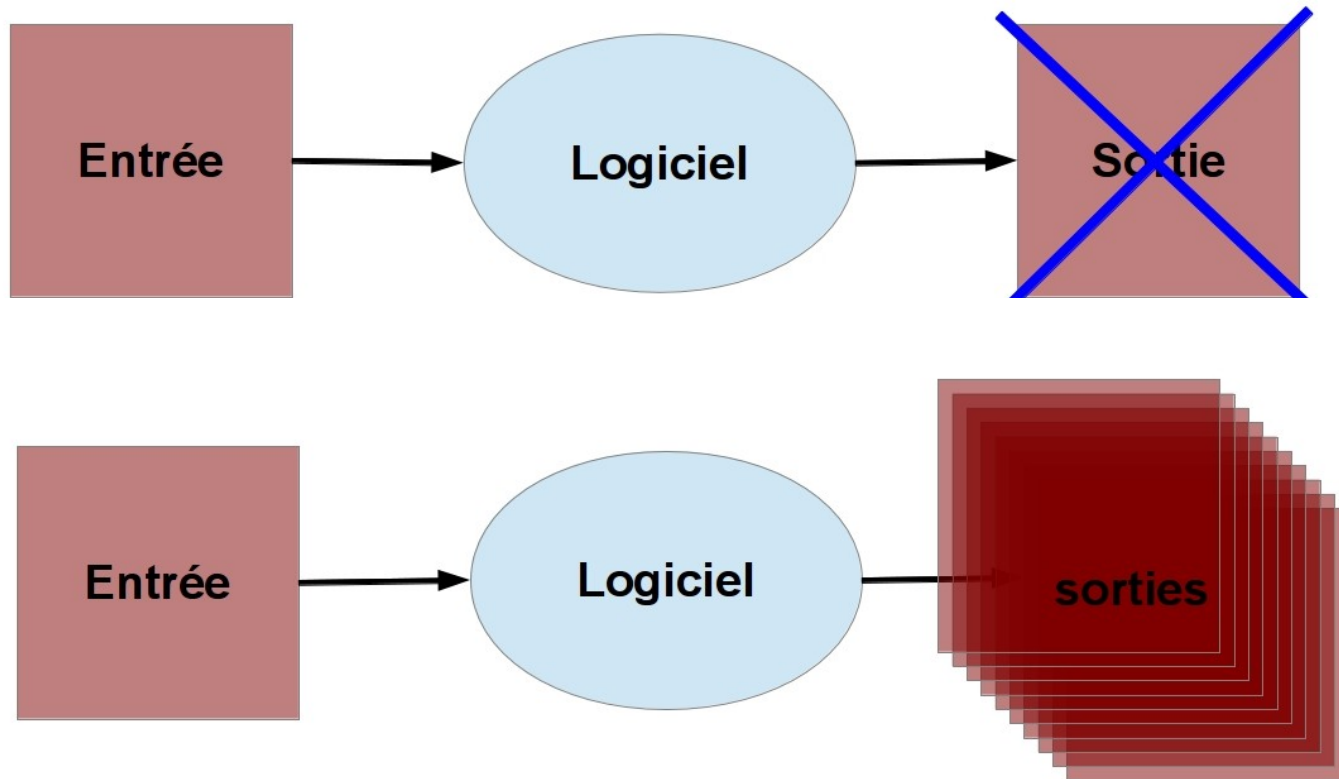
Arithmétique flottante et
vérification/validation

Tests de non régression



Lancer (de manière automatique) des scénarios pour lesquels on connaît la **bonne** valeur de la sortie, sa **valeur de référence**.

Tests de non régression et arithmétique flottante



Il n'y a pas unicité de la sortie numérique

Pourquoi ?

- Le **même code** exécuté dépend
- du langage de programmation du compilateur (et des options)
 - du processeur
 - du système d'exploitation

```
double x ;  
float a, b, c, d ;  
x = a+b+c+d ;
```

Ordre des opérations, précisions ?

Le **même code** peut donc donner **des résultats différents** dans des environnements différents

De plus 2 fonctions mathématiquement égales ne donneront pas forcément le même résultat numérique

Qu'est-ce qu'un résultat de référence ?

- Résultat que l'on peut **reproduire** numériquement (sur une machine)
- Et si possible égal (ou proche) au résultat **exact** ... quand on le connaît.
Sinon : résultat **validé** (experts, approches numériques, ...)



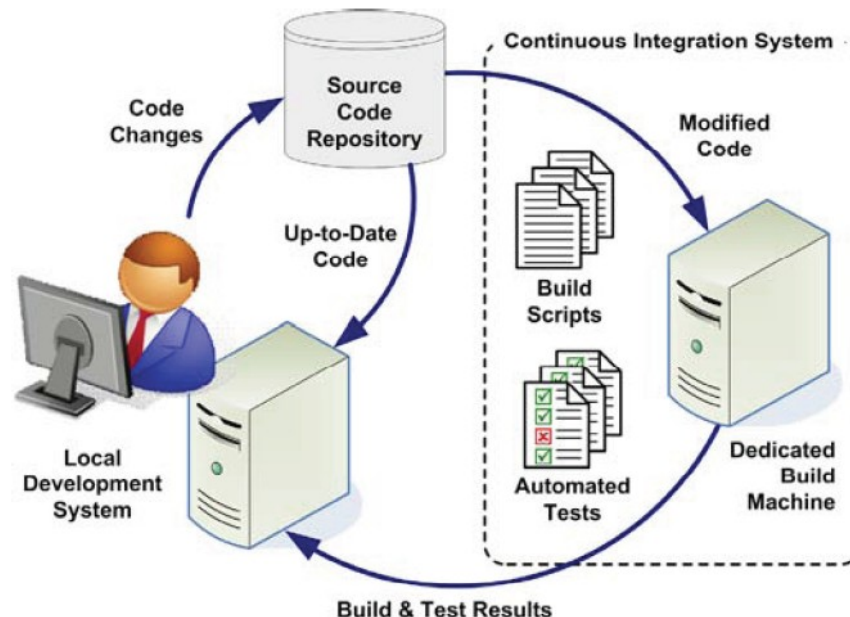
- Le **résultat exact** n'est pas toujours connu, et pas toujours atteignable
- sur une machine
- Le **résultat de référence** n'est pas toujours (rarement ?) le résultat exact.

Intégration continue

- Un **environnement** (compilateur (y compris options), architecture, système d'exploitation) <-> **Un résultat numérique**

Un résultat de référence unique par environnement

- Les **tests de non régression** ne sont pas effectués dans l'environnement informatique du développeur mais dans celui du **serveur d'intégration**



Maintenant la parole est
à vous !