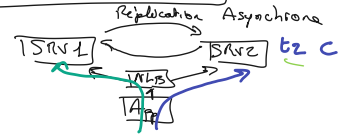
Bonjour Tout le Monde

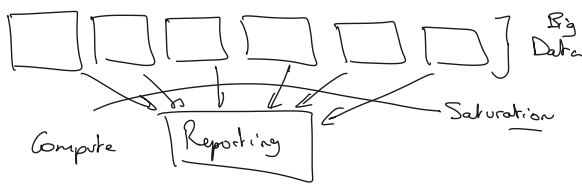
id	t	Valeur
1	∅	A
1	1	B
1	2	C
2		B
3		C

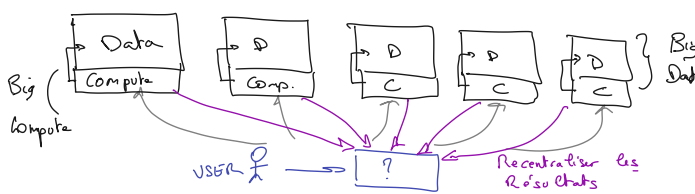
MVCC

t_0 1 → A
t_1 1 → B
t_2 1 → C

t_1 B







Les Styles d'architectures

Années 80

- Monolithiques

→ Excel → Multiplan
→ Foxpro → DBFast
→ Dbase

Android/IOS →
- App → data répliquée en ligne ou uniquement ligne
- App Monolithique aussi (données locales)

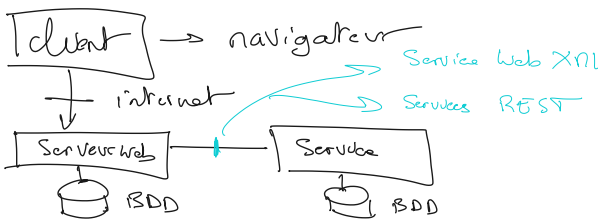
Années 90

- Client/Serveur

- Client lourd
- 2 Tier
  - Outlook
  - Progiciels d'entreprise
- Installer le progiciel → Problème

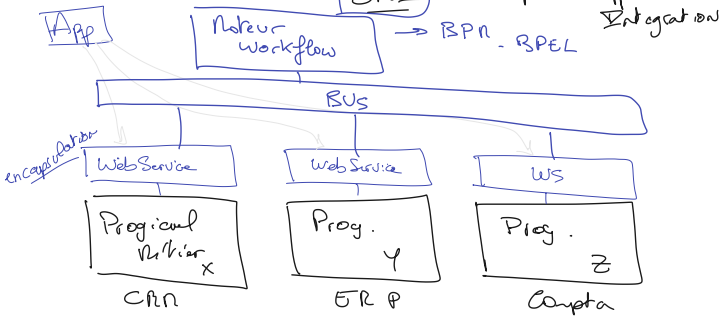
Années fin 90+ (actuellement)

- Architecture n Tier
- Apps Pas trop Complexes -



Architectures SOA Années 2000-2010

EAI — Enterprise Application Integration



2010 et + — Architectures Complexes
- Développés from scratch

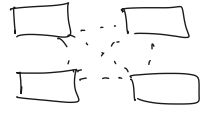
→ Architectures en µServices

si
- Centralisé

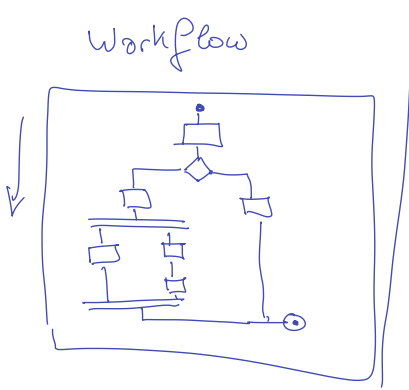
- si une "brique" tombe, tout flux tombe -
architecture actuelle (Dont n Tier !)

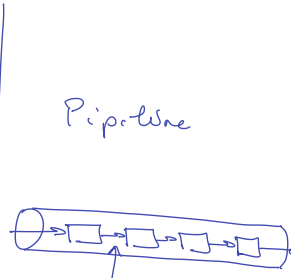
Interdépendance ~~dépendance~~
A éviter !
↳ "Téléphone"

Objectif Final =
fournir une expérience Hautement Disponible



- Indépendance
- Asynchronisme (messages)
↳ "SMS", "Mail"





Test Driven Development ≠ ISaQB

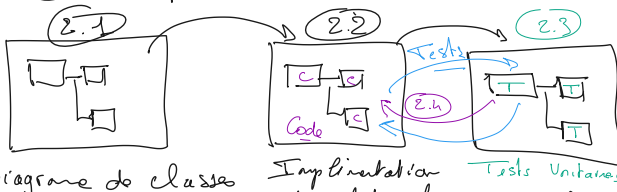
Behavior Driven Development ≠ Sauce

TDD → basé sur les test unitaires
BDD → " sur les objectifs fonctionnels -

Tests Unitaires (TDD)

(1) Analyse fonctionnelle

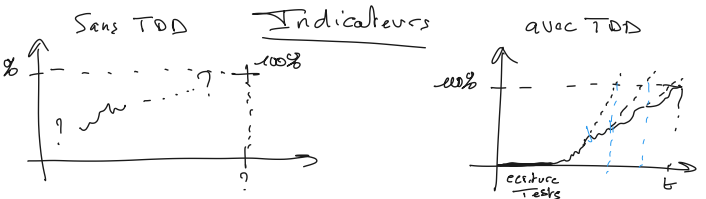
(2) Analyse d'architecture du code (UML)



Types de tests :

• Accessibility testing
• Acceptance testing
• Black box testing : tester un système inconnu.
• End to end testing : tester un workflow fonctionel
• Functional testing : s'assurer qu'il fait exactement ce qui était prévu.
• Interactive testing : eq tests manuels
• Interface tests
• Integration testing : test en environnement équivalent à la prod dans son ensemble.
• Load testing
• Non functional testing : catégorie de tests d'interface (conformité), accessibilité, performance, ...
• Performance testing : recherche des meilleurs métriques, benchmark.
• Regression testing
• Sanity testing : contrôle que les bugs ciblés sont bien corrigés
• Security testing : contrôle face aux menaces
• Single user performance testing
• Smoke testing : valider les fonctions critiques d'un système, envers la stabilité.
• Stress testing : test selon des conditions exceptionnelles de charge, au dela des specs.
• Unit testing
• White-box testing : teste les entrées sorties d'un logiciel bien connu, concernant le design, l'utilisabilité, la sécurité, la stabilité.

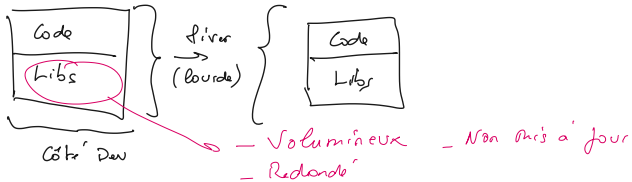
3 Ways to Test
There are 3 ways you can do testing.
• Manual testing is the most hands-on type of testing and is employed by every team at some point. Of course, in today's fast-paced software development lifecycle, manual testing is tough to scale.
• Automated testing uses test scripts and specialized tools to automate the process of software testing.
Continuous testing goes even further, applying the principles of automated testing in a scaled, continuous manner to achieve the most reliable test coverage for an enterprise. Keep reading to learn more about the differences between automated testing vs. manual testing and how continuous testing fits in.



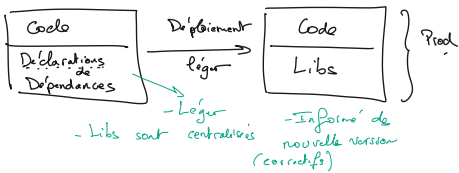
Gestionnaires de Paquets

maven (java) — Composer (php)
Nuget (.net) — Pip (Python) — etc...

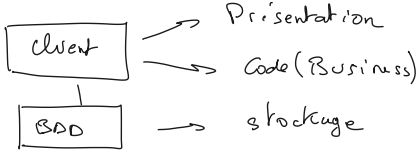
Avant (sans gestionnaire de paquet)



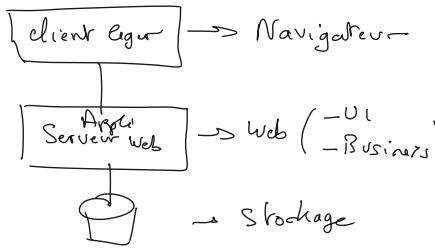
Avec Gestionnaire



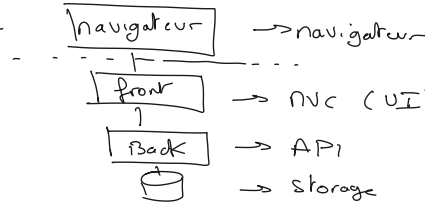
client Serveur

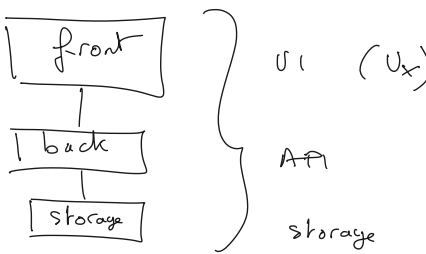


3 Tier

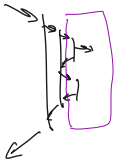


nTier





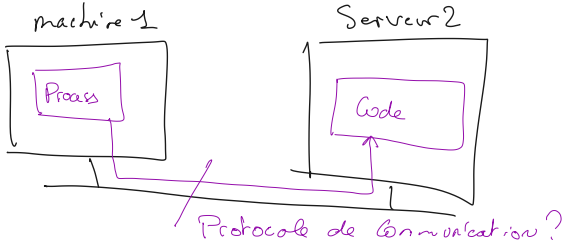
- Appeler du code "LPC"
Local Procedure Call



=> Pas de Réseau

=> bibliothèque

- Appel de code distant: "RPC"
Remote Procedure Call



80 IBM Corba

≃ 2000 - http + XML => (SOAP) Web Service
=> NORME

≃ 2005 + http + JSON => Services Rest (full)
(javascript)

Representational State Transfer
=> bcp + léger pour les mobiles -
=> PAS DE NORME