Gestion de Projet Informatique

Partie 3 : Intégration continue et tests

Licence d'Informatique 3e Année
Tianxiao Liu
Université de Cergy-Pontoise

Every large system that works started as a small system that worked.

-- Continuous Integration

Product testers do not make products; they only make them better.

The following is based on a True Story...

In memory to those days and nights on CI

Sommaire

- Principes fondamentaux de l'intégration continue
- Environnement de travail de *CI*
- Les tests suites et le cycle quotidien de *CI*
- Problème des systèmes mal testés
- Principes fondamentaux des tests
- Différents types de test de projet
- Activité de la séance

Principes fondamentaux de *CI*

- Cycle de vie ***Extreme Programming (XP)***
  - *TDD (Test Driven Development)*
- Réduire le risque d'intégration
  - Détecter les problèmes d'intégration le plus tôt possible
  - Le test immédiat des modifications
  - Avoir toujours une version **stable** et **viable**

Environnement de travail de *CI*

- **Un dépôt de source partagée**
  - Logiciel de gestion de version : SVN, Git…
  - Tout le monde travaille sur la branche principale (***trunk***) – Intégration des modifications
  - Chaque développeur fait des commits régulièrement (au moins une fois par **jour**).
- Des ***builds*** et des tests automatisés
  - Serveur de CI (ex. Hudson) : compilation et lancement des tests automatiques (à chaque commit)

Environnement de travail de *CI*

- Logiciel de gestion des tâches
  - Une tâche = un cas (case)
  - Avant que la tâche soit finie, des tests automatisés correspondant sont déjà écrits par le Q.A.
  - Tâches finie par le développeur → tests activés dans la test suite
- Compétence du Q.A.
  - Prédéfinir les tests n'est pas toujours un travail facile : techniques de *mock*, complétude …

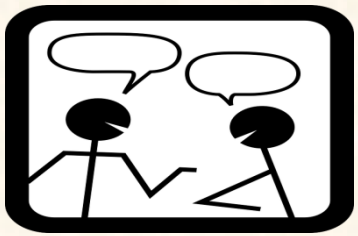
Pour faire un commit

- Les conditions strictes
  - Il faut que le code **compile** …
  - Il faut que tous les tests qui **passaient** avant **passent** encore maintenant
- Commit pour une tâche
  - Lancer **manuellement** les tests correspondants
  - Qualité de code
    - Nettoyer les petits « bricolages »
    - **Cohérence**, convention de codage, design patterns…

Pour faire un commit …

9H – 10H

10H – 11H30



11H30-12H

Test suite failed…

12H-13h30

13H30-14H

Test suite failed…

Bavardé
Pause café prise
Mangé
Fnac fait
Quoi maintenant ?

Integration Test Suite

- L'ensemble de tous les tests
  - Une couverture entière
  - Des tests unitaires + des tests d'intégration entre les modules architecturaux
  - Des **milliers** de tests automatisés
  - Dont l'exécution nécessite **des heures** !

- Bien que idéal, il est **impossible** de lancer cette test suite avant chaque commit…

Submit Test Suite

- Un sous-ensemble de *Integration Test Suite*
  - Le Q.A. choisit soigneusement des tests significatifs et sensibles pour y mettre dans *Submit Test Suite*.
  - Normalement, ceci doit couvrir **>95%** des cas.
  - Son exécution doit être de quelques minutes
  - **Il n'y aura plus bavard, pause café, fnac…**

A l'entreprise : Jour J du projet

Qui a cassé la submit test suite ?!!

AH Ô… mais ça passait chez moi…

Vu le dashboard, c'est lui !

Build de la nuit et régression

- Serveur CI
  - Lancement de ***Integration Test Suite***
  - Tests échoués → (cas) régression → **Priorité N° 1**

T'es sur quoi là ?

Pourquoi tu ne résous pas le cas de régression ?!

Ben, je suis sur la tâche X, comme prévu.

Ah, je n'ai pas vérifié mes emails…

Warning : the following images can be shocking !

Cas 1 : Sonde Mariner 1 – 1962

- Une erreur de *trait d'union* **désastreuse** !

130 millions dollars

Cas 2 : Therac-25 - 1985

- Erreur informatique : bug logiciel

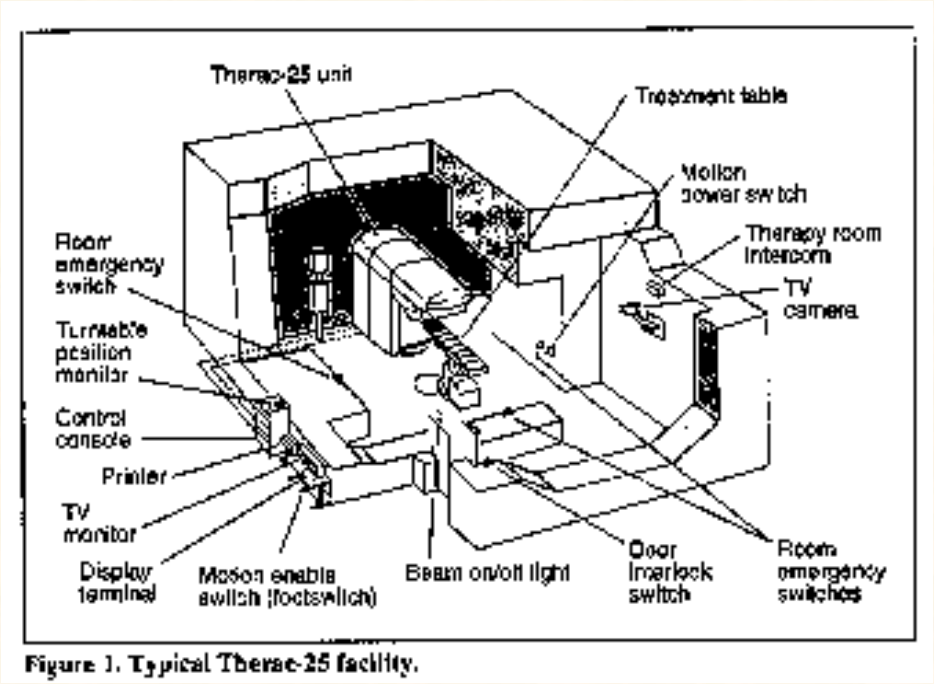


Figure 1. Typical Therac-25 facility.

5 morts !

Cas 3 : Ariane 5 Vol 501 - 1996

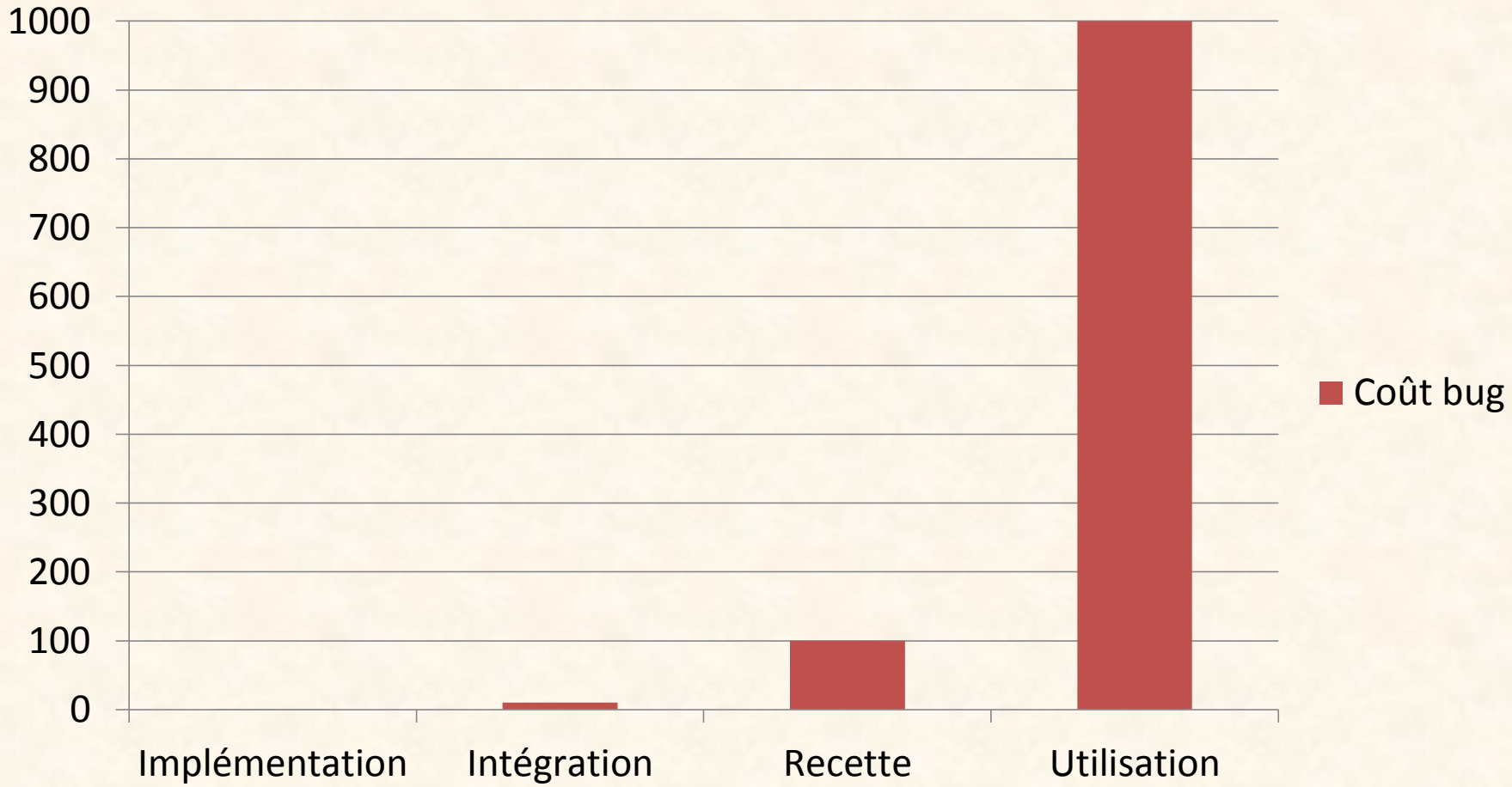
- Explosion à cause d'un dépassement d'entier dans les registres mémoire

370 millions dollars

Question :

Alors, on a besoin d'avoir 20/20 partout pour travailler dans les secteurs sensibles ?

Coûts des bugs



Définition du test

- Le test est l'exécution ou l'évaluation d'un système ou d'un composant, par des moyens automatiques ou manuels, pour vérifier qu'il répond à ses spécifications ou identifier les différences entre les résultats attendus et les résultats obtenus
- **Proverbe connu :** ***Tester peut révéler la présence d'erreurs mais jamais leur absence.***

Les notions de base

- Objectif de test
  - **comportement** du système envisagé
- Données de test
  - données en entrée au système de manière à **déclencher** l'objectif de test
- Résultat de test
  - conséquence ou sortie de l'exécution du test
- **Case de test (*test case*)**
  - Objectif + données + résultat de test

Test : méthodologies

- **Test boîte noire**
  - Spécification (CdC) → tester
  - Sans connaître l'implémentation technique
  - Test pouvant être prédéfini
- **Test boîte blanche**
  - Tester en se basant sur le code source
  - Tester pour du code déjà écrit

Les types de tests

- **Test unitaire**
  - Tester une unité de programme de façon **isolée**
  - Sans appel à d'autres fonctions
- **Test d'intégration**
  - Tester le fonctionnement d'un ensemble de modules (via leur interface)
- **Test de système**
  - D'un point de vue d'utilisateur
  - Conformité du produit fini

Les types de tests

- **Test de robustesse**
  - Support (tolérance) des utilisations imprévues
  - Sans appel à d'autres fonctions
- **Test de sécurité**
  - Le système est-il vulnérable aux attaques ?
- **Test de performance**
  - Avoir un temps de réponse satisfaisant ?
  - Simuler différents niveaux de charges d'utilisateurs

Activités de la séance

QA : Plan de test

- Quels tests prévus pour le système ?
  - Catégories
  - A quel moment effectueront ces tests ?
- A envoyer avant 17H
- Ce document sera mis à jour au fur et à mesure