

# Test et Maintenance de Logiciel

Ileana Ober

Maître de conférences  
Université Paul Sabatier  
IRIT

<http://www.irit.fr/~Ileana.Ober/>



---

Année Universitaire 2012-2013

©Ileana Ober

# Le besoin de logiciel...

---

- ✿ accroît incessamment



Des problèmes ne cessent pas  
d'apparaître ....

---

# Des problèmes ne cessent pas d'apparaître ....

---

## Toyota Prius Recall: 106,000 Hybrid Cars Pulled Over Steering Problems

THE CANADIAN PRESS -- TOKYO -- Toyota is recalling 106,000 first-generation Prius hybrid cars globally for faulty steering caused by a nut that may come loose. One minor accident suspected of being related to the problem has been reported in the U.S. Toyota Motor Corp. said Wednesday the latest recal...

# Des problèmes ne cessent pas d'apparaître ....

## Toyota Prius Recall: 106,000 Hybrid Cars Pulled Over Steering Problems

THE CANADIAN PRESS -- TOKYO -- Toyota is recalling 106,000 first-generation Prius hybrid cars globally for faulty steering caused by a nut that may come loose. One minor accident suspected of being related to the problem has been reported in the U.S. Toyota Motor Corp. said Wednesday the latest recal...

From Computerwoche 12, 24 March 1995, page 27

CW-report by Walter Mehl (Hamburg)  
"Siemens computer stops switching system (Stellwerk) in Hamburg" From Monday to Wednesday last week all signals the train station in Hamburg-Altona were on stop. Just like it was when track was first invented, the switches could be set only by using a crowbar-sized key to shove them into position. The fault was with a computer from the company Siemens, that had gone on-line the day before.

# Des problèmes ne cessent pas d'apparaître ....

## Toyota Prius Recall: 106,000 Hybrid Cars Pulled Over Steering Problems

THE CANADIAN PRESS -- TOKYO -- Toyota is recalling 106,000 first-generation Prius hybrid cars globally for faulty steering caused by a nut that may come loose. One minor accident

**1985-1987 -- Therac-25 medical accelerator.** A radiation therapy device malfunctions and delivers lethal radiation doses at several medical facilities. Based upon a previous design, the [Therac-25](#) was an "improved" therapy system that could deliver two different kinds of radiation: either a low-power electron beam (beta particles) or X-rays. The Therac-25's X-rays were generated by smashing high-power electrons into a metal target positioned between the electron gun and the patient. A second "improvement" was the replacement of the older Therac-20's electromechanical safety interlocks with software control, a decision made because software was perceived to be more reliable.

What engineers didn't know was that both the 20 and the 25 were built upon an operating system that had been kludged together by a programmer with no formal training. Because of a subtle bug called a "[race condition](#)," a quick-fingered typist could accidentally configure the Therac-25 so the electron beam would fire in high-power mode but with the metal X-ray target out of position. At least five patients die; others are seriously injured.

From Computerwoche 12, 24 March 1995, page 27

CW-report by Walter Mehl (Hamburg)  
"Siemens computer stops switching system (Stellwerk) in Hamburg" From Monday to Wednesday last week all signals the train station in Hamburg-Altona were on stop. Just like it was when track was first invented, the switches could be set only by using a crowbar-sized key to shove them into position. The fault was with a computer from the company Siemens, that had gone on-line the day before.

# Des problèmes ne cessent pas d'apparaître ....

From Computerwoche 12, 24 March 1995, page 27

CW-report by Walter Mehl (Hamburg)

- **November 2000 -- National Cancer Institute, Panama City.** In a series of accidents, therapy planning software created by Multidata Systems International, a U.S. firm, miscalculates the proper dosage of radiation for patients undergoing radiation therapy.

Multidata's software allows a radiation therapist to draw on a computer screen the placement of metal shields called "blocks" designed to protect healthy tissue from the radiation. But the software will only allow technicians to use four shielding blocks, and the Panamanian doctors wish to use five.

The doctors discover that they can trick the software by drawing all five blocks as a single large block with a hole in the middle. What the doctors **don't realize** is that the Multidata software gives different answers in this configuration depending on how the hole is drawn: draw it in one direction and the correct dose is calculated, draw in another direction and the software recommends twice the necessary exposure.

At least eight patients die, while another 20 receive overdoses likely to cause significant health problems. The physicians, who were legally required to double-check the computer's calculations by hand, are indicted for murder.

called a **race condition**, a quick-fingered typist could accidentally configure the Therac-25 so the electron beam would fire in high-power mode but with the metal X-ray target out of position. At least five patients die; others are seriously injured.

# Des problèmes ne cessent pas d'apparaître ....

From Computerwoche 12, 24 March 1995, page 27

CW-report by Walter Mehl (Hamburg)

- **November 2000 -- National Cancer Institute, Panama City.** In a series of accidents, therapy planning software created by Multidata Systems International, a U.S. firm, miscalculates the proper dosage of radiation for patients undergoing radiation therapy.

Multidata's software allows a radiation therapist to stack up to eight metal shields called "blocks" designed to protect the patient. The software will only allow technicians to use four shield

The doctors discover that they can trick the software by putting a hole in one of the metal shields. What the doctors get in return depends on which answers in this configuration depending on how many blocks are used. If the correct dose is calculated, draw in another diagram to show the necessary exposure.

At least eight patients die, while another 201 more have serious problems. The physicians, who were legally required to double-check the computer's calculations by hand, are indicted for murder.

called a [race condition](#), a quick-fingered typist could accidentally configure the Therac-25 so the electron beam would fire in high-power mode but with the metal X-ray target out of position. At least five patients die; others are seriously injured.

## Businesses hit by Bank of Queensland EFTPOS bug

Monday, 04 January 2010 10:34

Patrick Stafford

The Bank of Queensland has confirmed the problem, but says it is continuing to work on the issue with manual fixes currently in place and an investigation underway to find the root of the bug.

Apparently the bank's EFTPOS machines had their internal computer clocks move forward six years on New Year's Day, causing the machines to reject any cards with an expiry date stamped before 2016.

As the current debit and credit cards issued to customers are not valid past 2016, all of these cards were rejected by the machines.

Thousands of customers have been forced to move from EFTPOS to cash, causing many to abandon purchasing some items altogether. It is

signals  
like it  
be set  
position  
done

ock  
erent  
ie

# Ariane - 5 vol de qualification

---



©Ileana Ober, 2012

# Ariane 5 - vol de qualification

---

- ✿ h0+39s autodestruction (500 M€)
- ✿ Le 23 juillet, la commission d'enquête remet son rapport :

*La fusée a eu un comportement nominal jusqu'à la 36ème seconde de vol. Puis les systèmes de référence inertielles (SRI) ont été simultanément déclarés défaillants. Le SRI n'a pas transmis de données correctes parce qu'il était victime d'une erreur d'opérande trop élevée du "biais horizontal" . . .*

- ✿ Une défaillance logicielle a été vite mise en cause

# Raisons (IEEE Computer 01/1997)

---

- ❖ Réutilisation dans Ariane V d'un bout de code d'Ariane IV (concernant le positionnement et la vitesse de la fusée)
- ❖ Une conversion d'un flottant sur 64 bits en un entier signé sur 16 bits-
- ❖ Pour Ariane V, la valeur du flottant dépassait la valeur maximale pouvant être convertie
- ❖ => problème dans le système de positionnement
- ❖ la fusée a “corrigé” sa trajectoire
- ❖ Résultat d'une trop grande déviation, la fusée s'est auto-détruit

# Est-ce de l'incompétence?

---

- ❖ Non.  
Les équipes d'ESA ont suivi à la lettre des processus de développement logiciel largement acceptés dans le monde industriel

# Est-ce un problème de gestion de projet?

---

- ✿ Non.  
Les enquêtes ont identifié un problème clairement technique....

# Est-ce un problème de langage de programmation?

---

- ❖ langage utilisé ADA
- ❖ la conversion des données a été identifiée comme source potentielle de problèmes pendant le développement
- ❖ cependant, tous les conversions n'ont pas été effectuées avec un mécanisme d'exception associé, pour éviter de charger l'ordinateur à plus de 80% de sa capacité
- ❖ sur 7 variables, les conversions de 4 ont été accompagnés de levés d'exceptions
- ❖ le potentiel de problème des conversions a bien été identifié

# Est-ce un problème de conception?

---

- ❖ Pourquoi cette exception n'a pas été monitorée?
- ❖ Les analyses ont montré que le dépassement (un biais horizontal non représentable sur 16 octets) n'était pas possible
- ❖ Est-ce que les analyses ont échoué?
- ❖ Non, l'analyse était correcte au moment où elle a été faite c.à.d. pour **la trajectoire d'Ariane 4**
- ❖ Pour Ariane 5, qui a d'autres paramètres de trajectoire, **l'analyse ne tient plus**

# Est-ce une erreur de programmation?

---

- ✿ Même si on peut reprocher la suppression de la protection par exception pour gagner en performance, le choix a été justifié par les analyses théoriques.
- ✿ Le développement logiciel est aussi une question de compromis...
- ✿ Si on a prouvé que une condition est satisfaite on peut tenir compte de ce résultat dans l'implémentation

# Est-ce un problème de test?

---

- ❖ Pas vraiment. Même si la commission d'enquête a recommandé de faire des tests plus intensifs et de viser le système complet, pas seulement des parties  
Le système central et le logiciel de vol n'avaient pas été testés ensemble auparavant
- ❖ Le seul test réaliste dans ce contexte est un vrai lancement, et c'est vraiment ce qui s'est passé...
- ❖ Même si personne ne l'envisageait comme un test de 500 M€

# C'est quoi alors?

---

- ❖ Une erreur de réutilisation
- ❖ Le module “coupable” a été repris d'un logiciel de 10 ans développé pour Ariane 4
- ❖ Les hypothèses faites dans ce module n'ont pas été clairement spécifiés et vérifiées

# Les défaillances sont une fatalité?



Le 1 octobre 2012 à 10h45

Par Rémy Decourt, Futura-Sciences

## Ariane 5 : 51<sup>e</sup> succès consécutif depuis 2003

**N**ouvelle mission réussie pour ArianeSpace. Après le lancement du satellite de météorologie polaire Metop-B, la société européenne a envoyé sur une [orbite de transfert](#) géostationnaire deux satellites de communications. Avec ce vol, [Ariane 5](#) signe le 51<sup>e</sup> succès consécutif depuis 2003.

Pour le 65<sup>e</sup> lancement de l'année, le quatrième pour une Ariane 5 ECA et le cinquième d'une Ariane 5 si l'on tient compte du lancement de l'[ATV-3 Edoardo Amaldi](#) (dont la séparation avec l'[ISS](#) a été réalisée vendredi soir), le [lanceur](#) européen a mis sur une [orbite](#) de transfert géostationnaire les satellites Astra 2F et GSTA-10.

Le lanceur a décollé du [Centre spatial guyanais](#) vendredi soir à 23 h 18, heure française. Trente et une minutes plus tard, les deux satellites étaient séparés sur leurs orbites respectives avec une très grande précision, au regard des paramètres de la [mission](#). Les mesures indiquent en effet un nôtre

Avec une performance totale de 10.178,7 kg, le [record de performance](#) atteint lors du tir précédent était proche. Le 2 août dernier, les 6.094 kg d'Intelsat 20 et les 3.311 kg d'Hylas-2, représentaient une masse totale à lancer de 10.183 kg ! Ce gain de performance a été rend possible par :

- l'ensemble des travaux réalisés lors du *Performance Improvement Plan* qui ont permis de réduire différents conservatisme initiaux sur l'ensemble du lanceur à partir de l'expérience et de l'analyse des 35 premiers vols Ariane 5 ECA ;
- l'utilisation des caractéristiques spécifiques des étages constituant ce [lanceur](#).

### Ariane 5 : une qualité de service indéniable

Pour Ariane 5, il s'agit du 51<sup>e</sup> succès consécutif depuis 2003, une réussite à souligner. Lancement à

©Ileana Ober, 2012

# Plan du cours

---

- ❖ Généralités sur le test
- ❖ Notions de bogue, erreur, défaillance, etc
- ❖ Gestion des erreurs
- ❖ Combien tester et quoi
- ❖ La maintenance

# Questions

---

- ❖ Comment savez-vous si votre programme fonctionne correctement ?
- ❖ Comment le testez-vous ?
- ❖ Combien y consacrez-vous :
  - ❖ de temps ?
  - ❖ de budget ?
  - ❖ % développement total ?
- ❖ Jusqu'à quand le testez-vous ?
- ❖ Combien vous faudrait-il en plus pour atteindre le "zéro défaut" ?

# Définitions

---

- ⊕ IEEE729 : 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**
- ⊕ Myers : Tester, c'est **exécuter** un **programme** dans l'intention de trouver des **anomalies**
- ⊕ AFCIQ : Le test est une technique de contrôle consistant à s'assurer, **au moyen de son exécution**, que le comportement d'un **programme** est **conforme** à des **données préétablies**

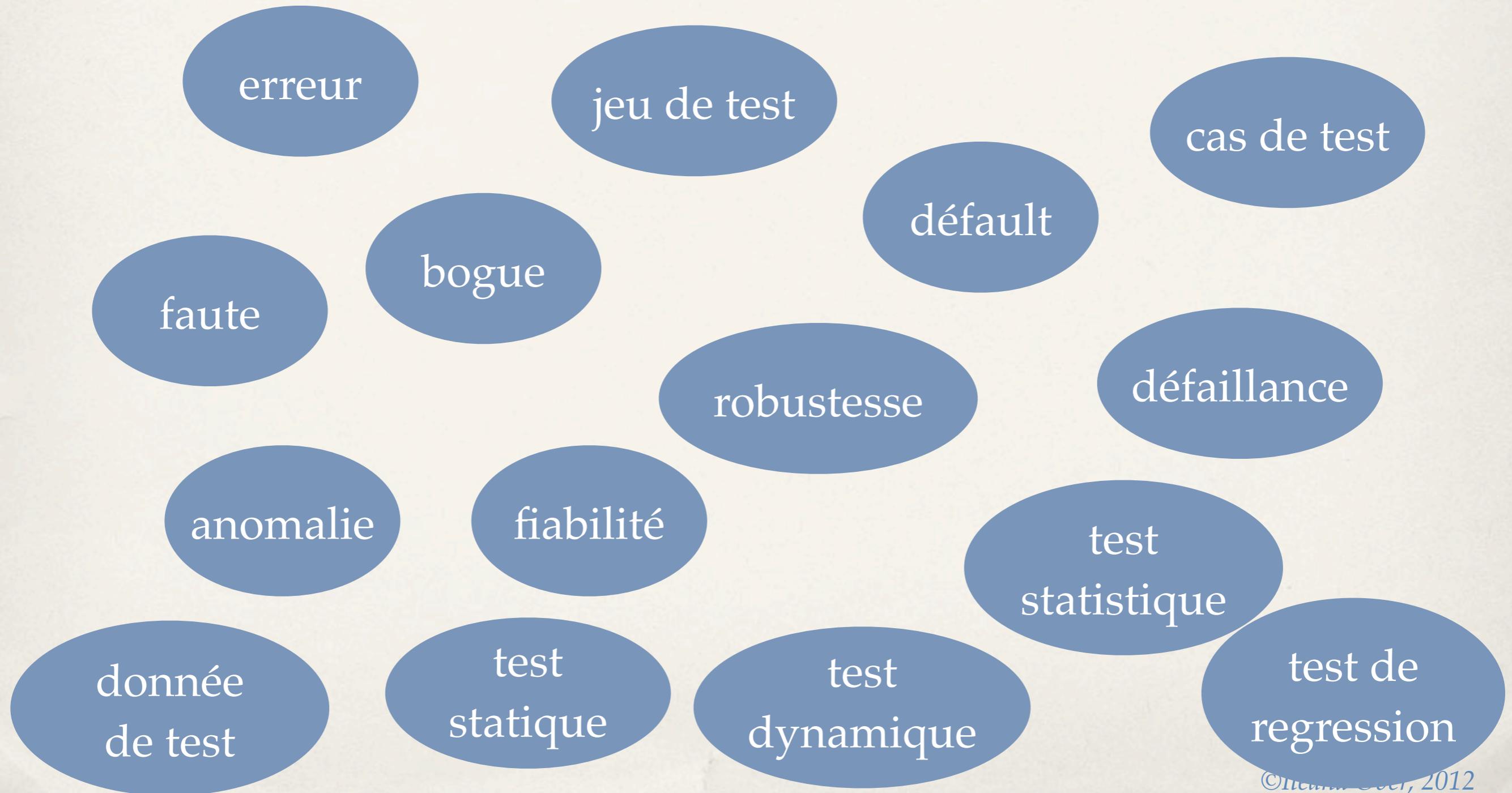
# Plan du cours

---

- ❖ Généralités sur le test
- ❖ Notions de bogue, erreur, défaillance, etc
- ❖ Gestion des erreurs
- ❖ Combien tester et quoi
- ❖ La maintenance

# Vocabulaire

---

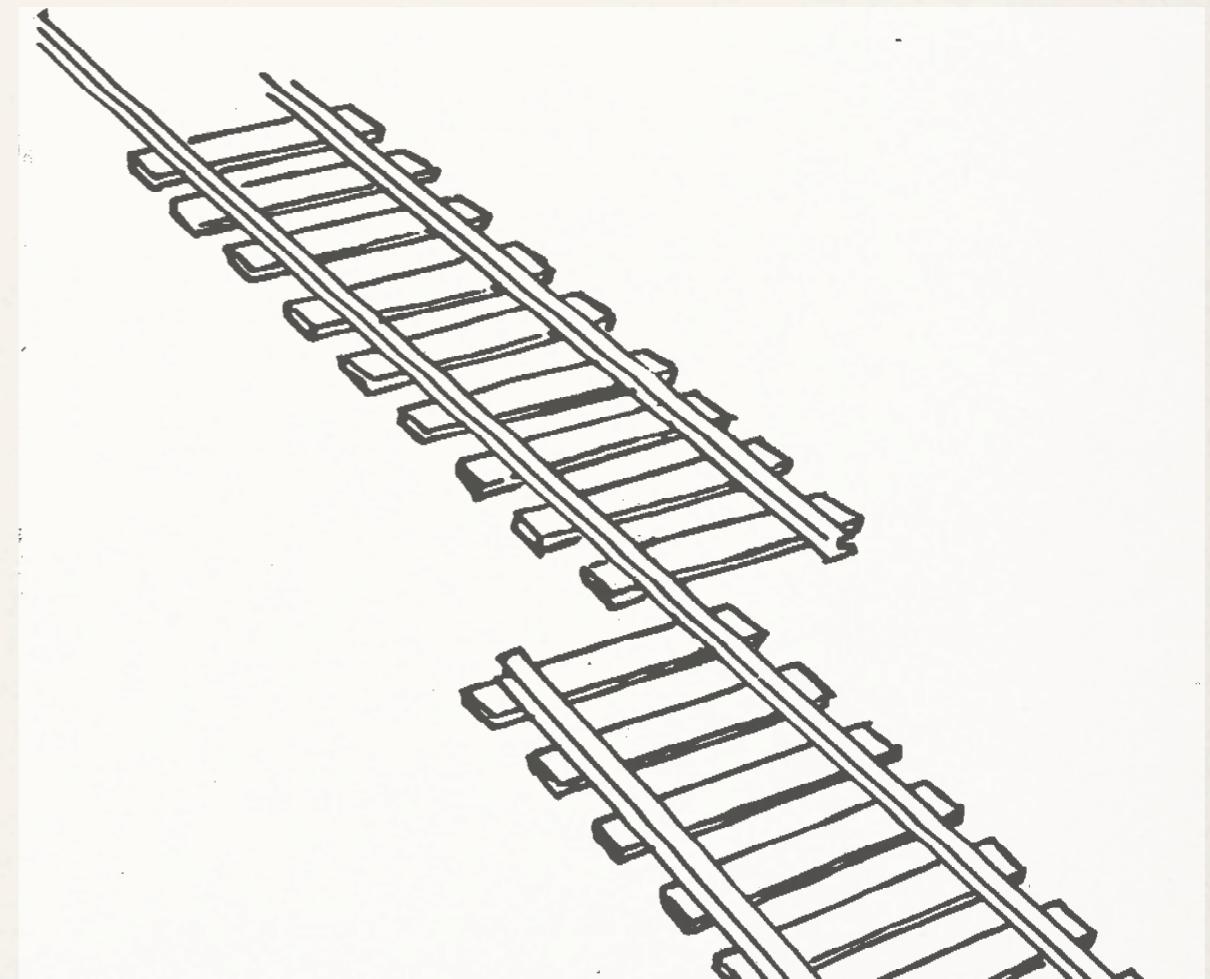


# Qu'est-ce que c'est?

---

- \* Erreur / Défaut / Anomalie / Bogue?

Exemple extrait de  
Bernd Bruegge & Allen H. Dutoit  
Object-Oriented Software Engineering: Using UML, Patterns, and Java

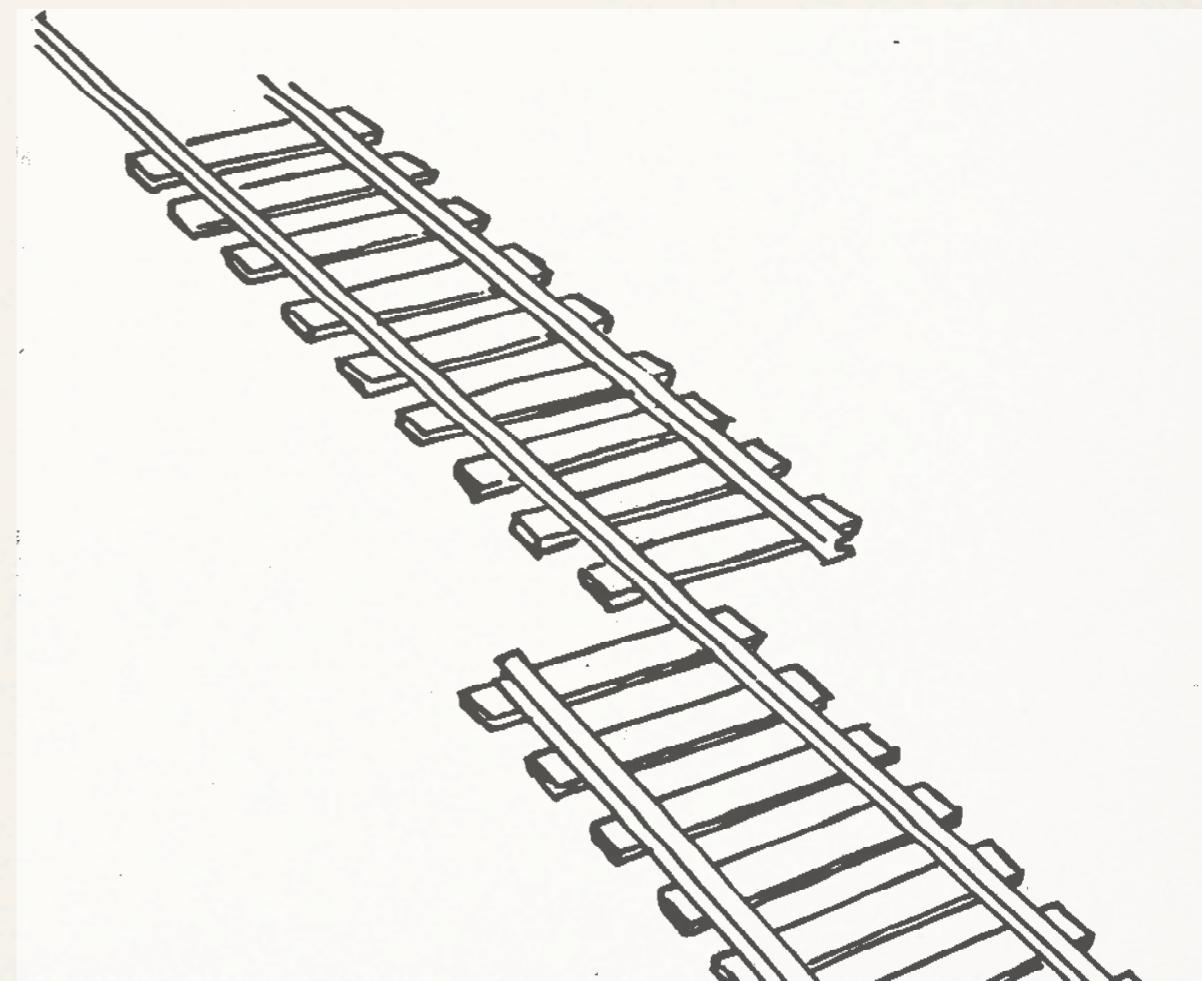


# Qu'est-ce que c'est?

---

- ❖ Il faut d'abord spécifier le comportement attendu
- ❖ On ne peut pas parler d'erreur, bogue, etc. on ne peut pas faire du test sans avoir une spécification

Exemple extrait de  
Bernd Bruegge & Allen H. Dutoit  
Object-Oriented Software Engineering: Using UML, Patterns, and Java



# Erreur

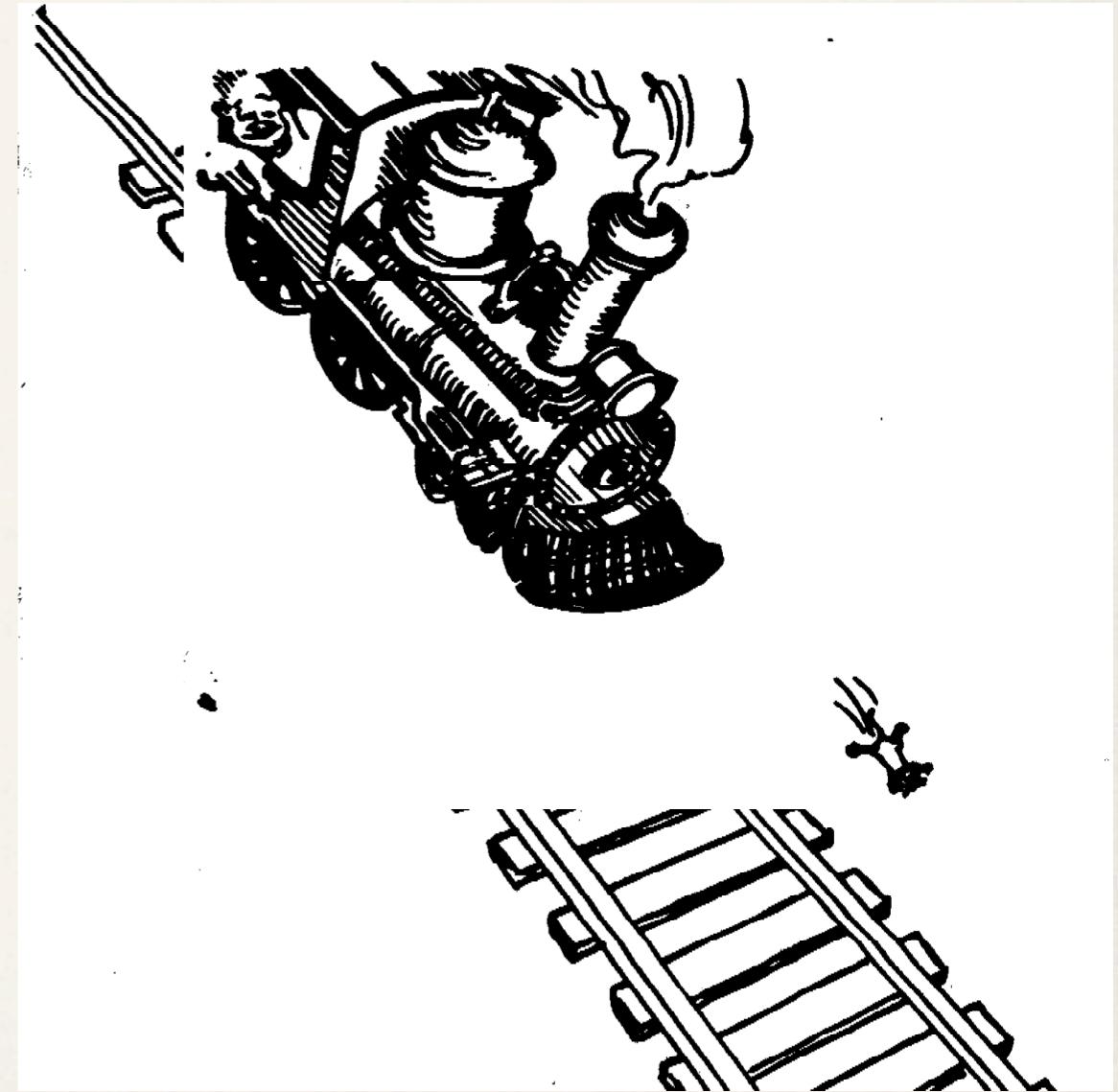
---

# Erreur

---

*si l'exécution mène à un état « erreur »*

Exemple extrait de  
Bernd Bruegge & Allen H. Dutoit  
Object-Oriented Software Engineering: Using UML, Patterns, and Java



# Défaut

---

*déviation du comportement **observé** par rapport au comportement **attendu***



Exemple extrait de  
Bernd Bruegge & Allen H. Dutoit  
Object-Oriented Software Engineering: Using UML, Patterns, and Java

# Définitions

---

- ✿ Erreur : le système arrive dans un état « *erreur* »
- ✿ Défaut : *déviation du comportement observé par rapport au comportement attendu*
- ✿ Anomalie : incapacité du système à remplir une fonction
- ✿ Bogue : cause (algorithmique, mécanique) d'une erreur

# Classification des défauts

---

- ❖ Calcul
- ❖ Logique
- ❖ E/S
- ❖ Interface
- ❖ Définition des données
- ❖ Traitement des données
- ❖ etc.

# Plan du cours

---

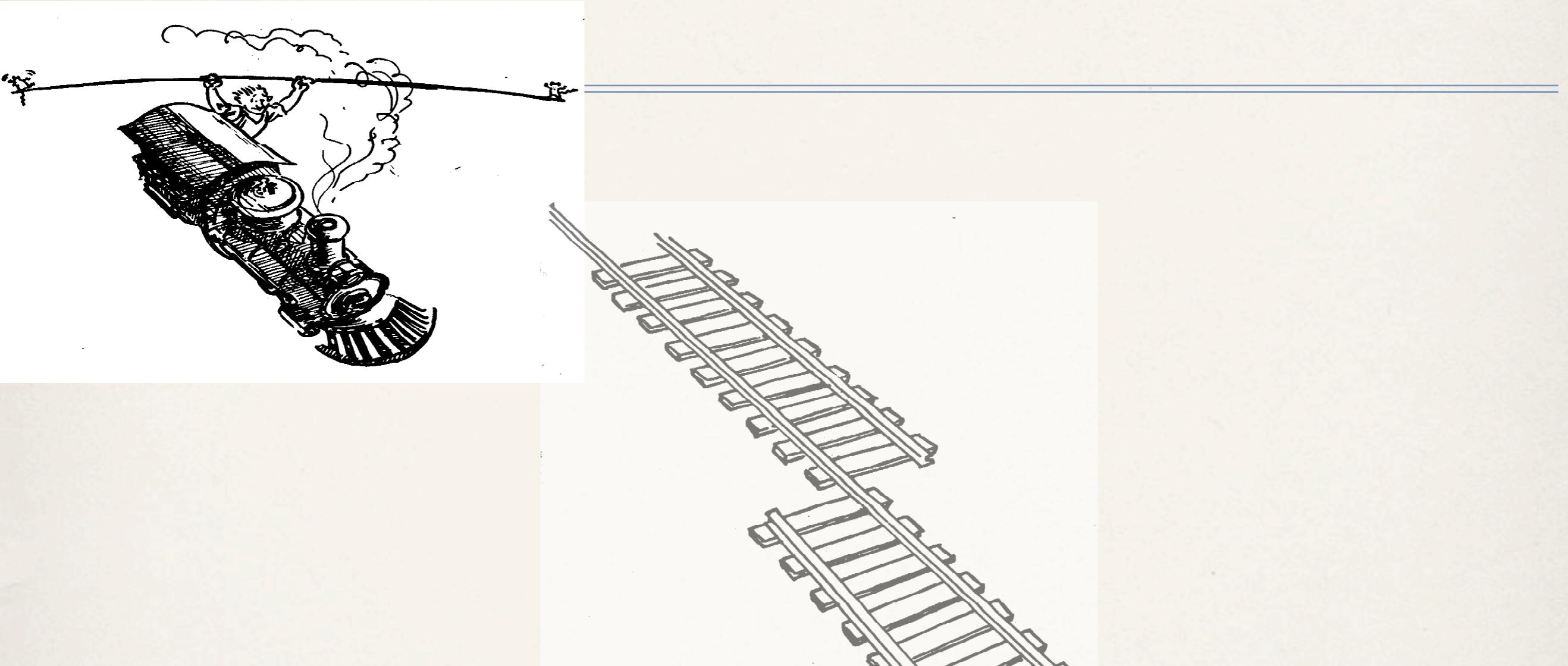
- ❖ Généralités sur le test
- ❖ Notions de bogue, erreur, défaillance, etc
- ❖ Gestion des erreurs
- ❖ Combien tester et quoi
- ❖ La maintenance

# Gérer les erreurs/défaux ....

---

Différentes stratégies ....

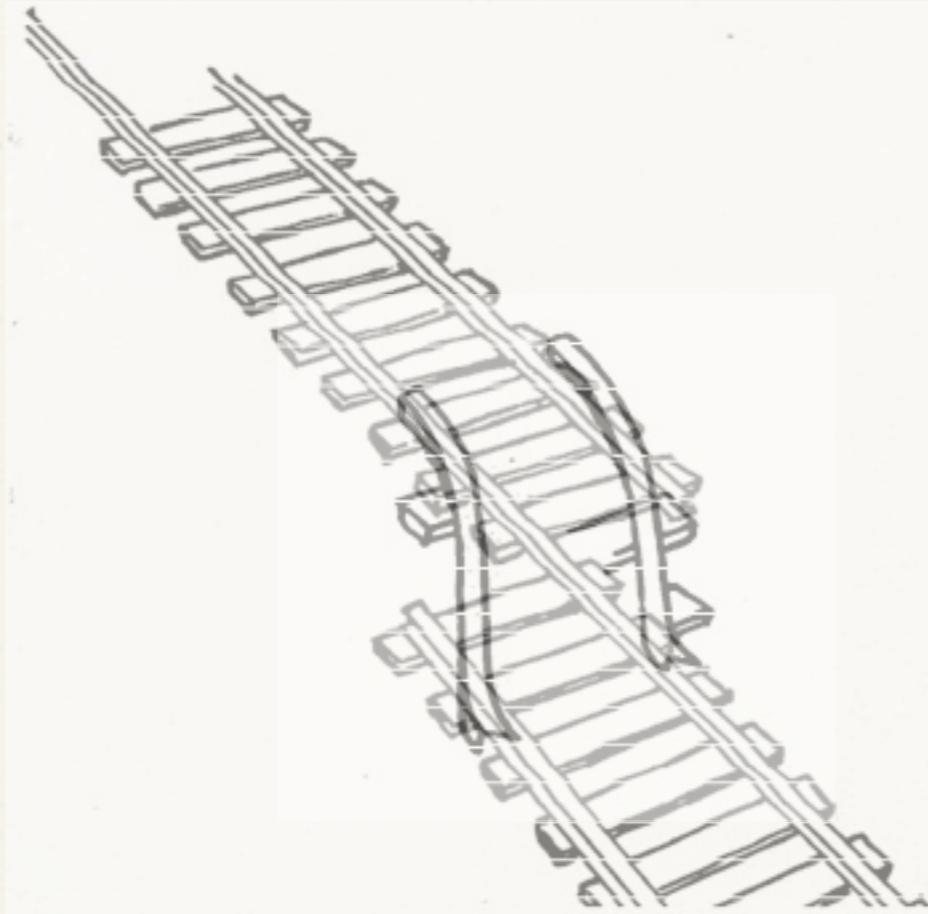
# 1ère stratégie: changer la spéc



Exemple extrait de  
Bernd Bruegge & Allen H. Dutoit  
Object-Oriented Software Engineering: Using UML, Patterns, and Java

# 2ème stratégie: patcher

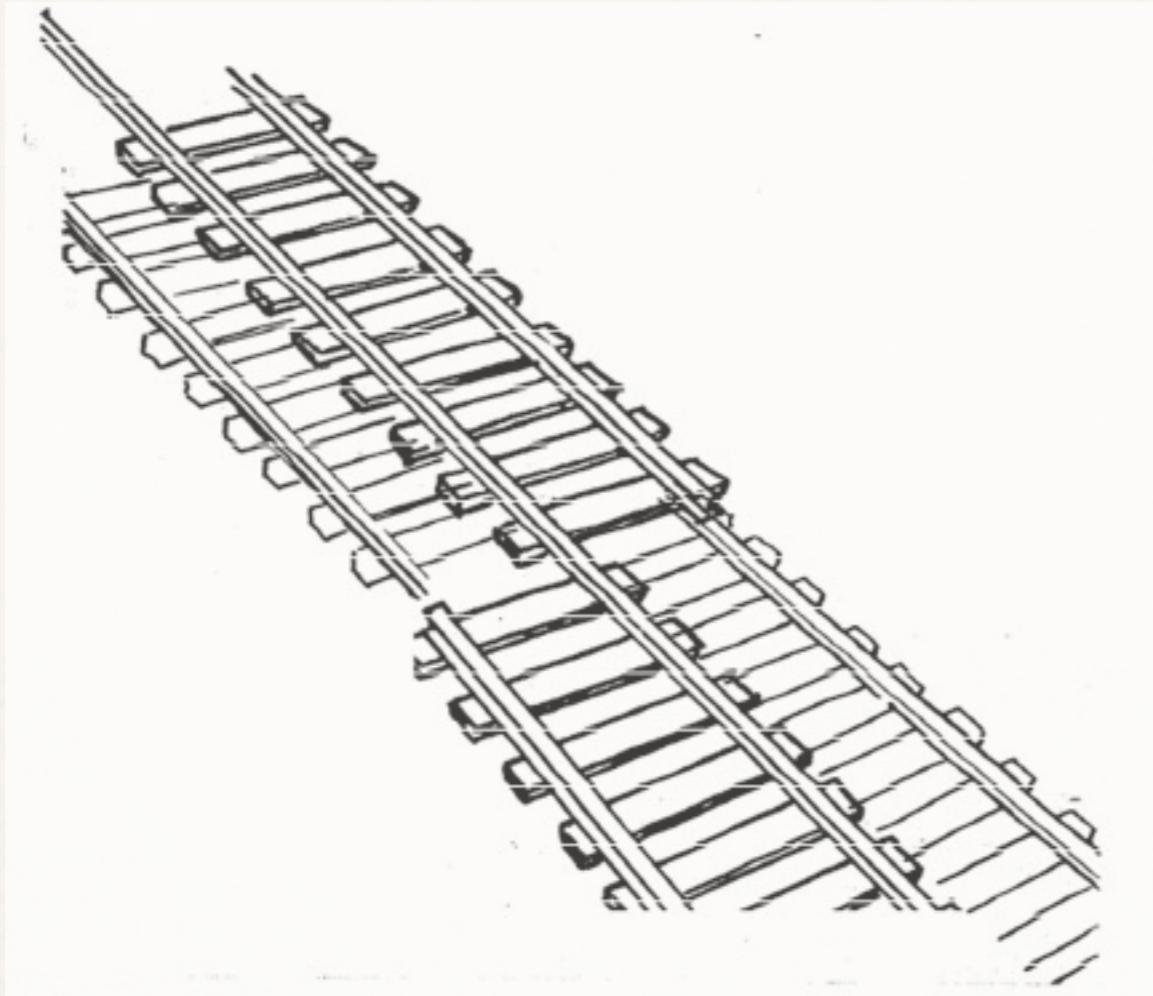
---



Exemple extrait de  
Bernd Bruegge & Allen H. Dutoit  
Object-Oriented Software Engineering: Using UML, Patterns, and Java

# 3ème stratégie: redondance modulaire

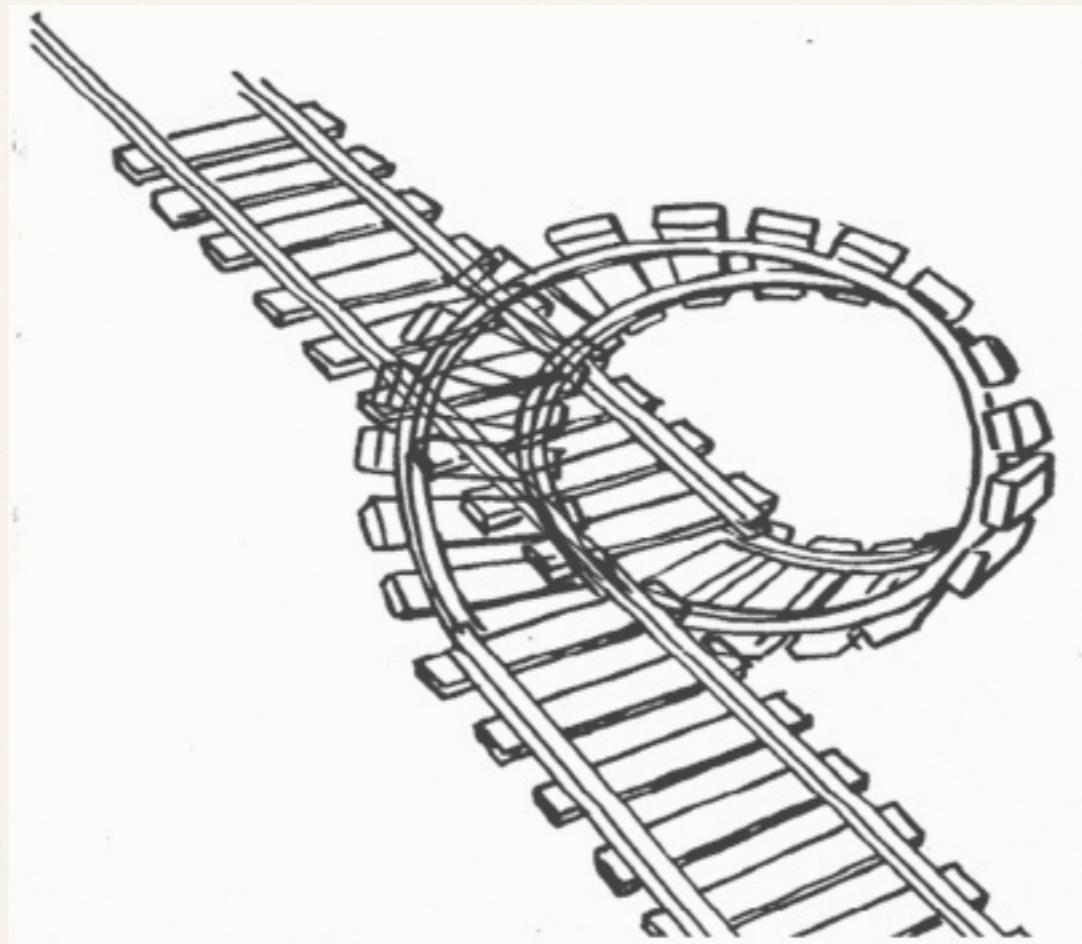
---



Exemple extrait de  
Bernd Bruegge & Allen H. Dutoit  
Object-Oriented Software Engineering: Using UML, Patterns, and Java

# 4ème stratégie: vérification

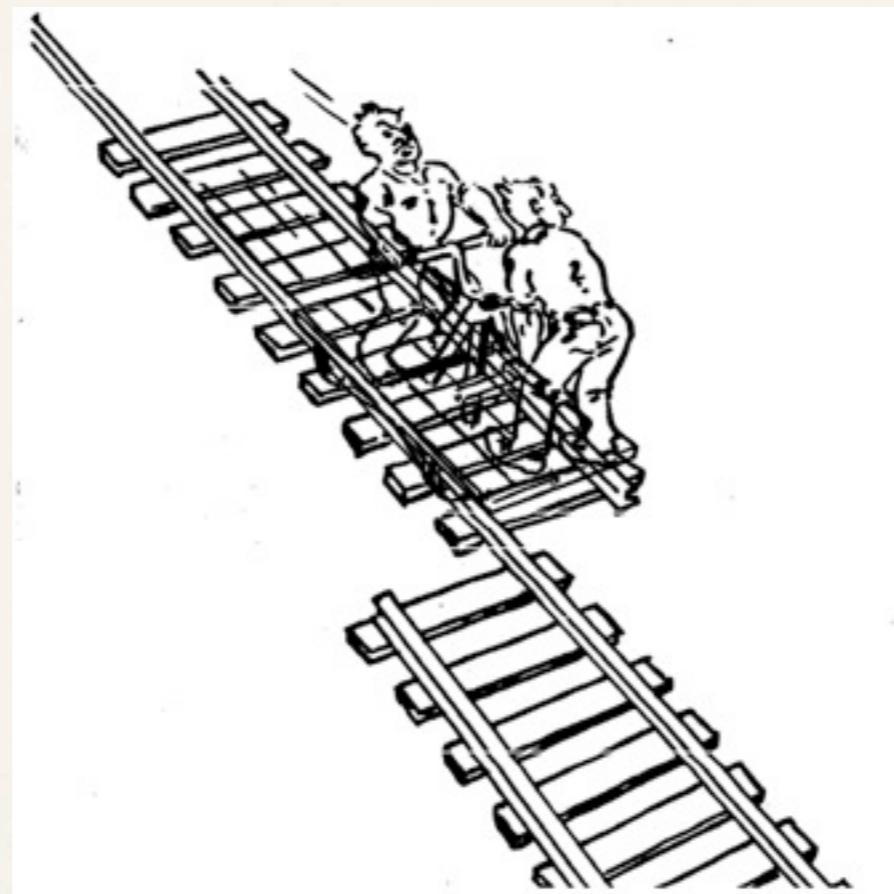
---



Exemple extrait de  
Bernd Bruegge & Allen H. Dutoit  
Object-Oriented Software Engineering: Using UML, Patterns, and Java

# 5ème stratégie: le test

---



Exemple extrait de  
Bernd Bruegge & Allen H. Dutoit  
Object-Oriented Software Engineering: Using UML, Patterns, and Java

# Gérer les erreurs/défaux

---

- ✿ changer la spécification
  - pas vraiment une solution
- ✿ patcher
  - réduit la performance
- ✿ redondance modulaire
  - cher
- ✿ vérification
  - se fait dans un environnement hypothétique (très?) différent de celui réel
- ✿ test
  - n'offre pas de certitude

# Plan du cours

---

- ❖ Généralités sur le test
- ❖ Notions de bogue, erreur, défaillance, etc
- ❖ Gestion des erreurs
- ❖ Combien tester et quoi
- ❖ La maintenance

# Différents aspects liés au test ...

---

- ❖ pourquoi tester?
- ❖ tester quoi?
- ❖ par qui?
- ❖ jusqu'à quand?

# Pourquoi tester?

---

Tester, c'est exécuter le programme dans l'intention d'y trouver des anomalies ou des défauts - G. J. Myers (*The Art of Software Testing*, 1979)

- ❖ Atteindre des niveaux de confiance et de qualité satisfaisants
- ❖ tout en limitant l'effort de test ( niveaux préétablis, raisonnables )

# Combien de temps il faut tester?

---

- ❖ Y-a-t-il un pourcentage des cas de test qu'il faut essayer?
- ❖ Y-a-t-il une durée qu'il faut prévoir (genre 40% du temps de développement)?
- ❖ avant de répondre, il faut savoir que ....

# Tester quoi?

---

# Tester quoi?

---

Un programme avec 70 branchements ( if, for, etc) ...

# Tester quoi?

---

Un programme avec 70 branchements ( if, for, etc) ...

... génère plus de cas de test ...

# Tester quoi?

---

Un programme avec 70 branchements ( if, for, etc) ...

... génère plus de cas de test ...



... qu'il y a des cuillères d'eau ....

# Tester quoi?

---

Un programme avec 70 branchements ( if, for, etc) ...

... génère plus de cas de test ...



... qu'il y a des cuillères d'eau ....



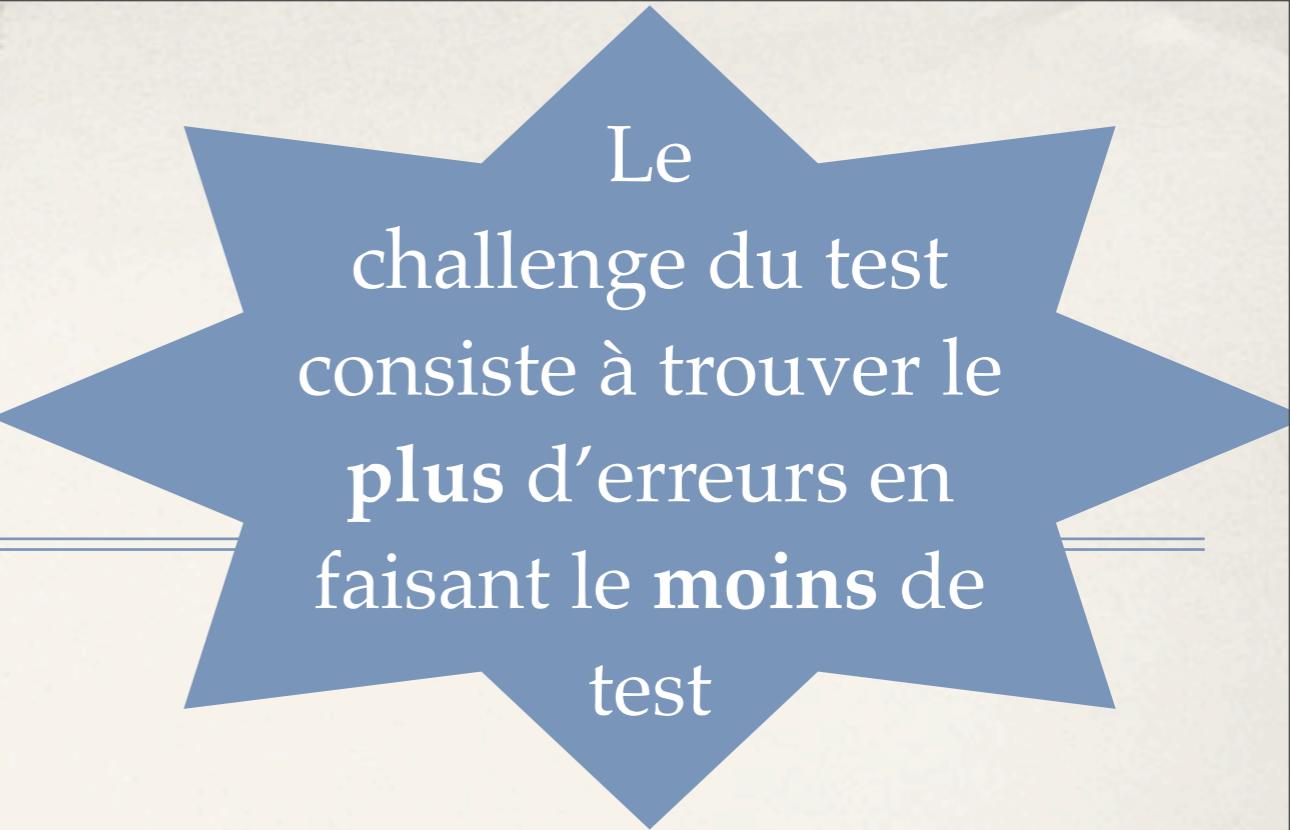
... dans l'Océan Pacifique.  
(Bob Stahl)

# Combien tester?

---

- ❖ Une bataille perdue d'avance
  - ❖ un programme a un nombre infini (ou gigantesque) d'exécutions possibles
  - ❖ un jeu de tests n'examine qu'un nombre fini (petit) d'exécutions possibles
- ❖ Trouver des heuristiques :
  - ❖ approcher l'infini (ou le gigantesque) avec le fini (petit)
  - ❖ → tester les exécutions les plus « représentatives »

# Combien tester?



Le challenge du test consiste à trouver le plus d'erreurs en faisant le **moins** de test

- ✿ Une bataille perdue d'avance
  - ✿ un programme a un nombre infini (ou gigantesque) d'exécutions possibles
  - ✿ un jeu de tests n'examine qu'un nombre fini (petit) d'exécutions possibles
- ✿ Trouver des heuristiques :
  - ✿ approcher l'infini (ou le gigantesque) avec le fini (petit)
  - ✿ → tester les exécutions les plus « représentatives »

# Observation

---

- ❖ Le test peut seulement montrer la **présence** des bogues, non pas leur **absence** (Dijkstra)
- ❖ Par contre, le test peut “augmenter notre confiance” dans le bon fonctionnement d'un programme
- ❖ Un bon jeu de tests doit exercer un maximum de “comportements différents” du programme

# Test

---

- ❖ activité de processus de développement
- ❖ 50%-60% du temps de développement
  - ❖ élaboration du matériel de test
  - ❖ déroulement des tests
  - ❖ analyse des résultats, décisions
  - ❖ documentation
- ❖ répartie entre plusieurs phases
- ❖ dépend de la phase

# Qui effectue le test?

---

“ On ne peut pas tester ses propres programmes ” (Myers)

- ❖ Le test doit toujours être fait par quelqu'un d'autre que le développement
- ❖ Il doit être un processus déstructif par rapport au code existant - pour des raisons psychologiques (et incontrôlables) on ne peut pas être déstructifs envers nos créations
- ❖ Le métier de testeur est un métier en soi
- ❖ Métier “challenging and fun”
- ❖ Les offres d'emplois regorgent de demandes de testeurs

# Les rôles de l'équipe de test

---

- ✿ Trouver les bogues importants vite
- ✿ Fournir une évaluation de la qualité du produit logiciel
- ✿ S'assurer que le produit satisfait certains standards
- ✿ Travailler avec les programmeurs
- ✿ Aider dans la prédition et le contrôle des couts du support
- ✿ Aider à bien évaluer le coût de la maintenance et des évolutions

# Plan du cours

---

- ❖ Généralités sur le test
- ❖ Notions de bogue, erreur, défaillance, etc
- ❖ Gestion des erreurs
- ❖ Combien tester et quoi
- ❖ Oracle
- ❖ La maintenance

# Oracle / Verdict

---

- ❖ Oracle = expression booléenne caractérisant la détection d 'une erreur
- ❖ Généralement = (resultat\_attendu = résultat\_obtenu)
- ❖ Très difficile de connaître le résultat attendu
- ❖ Limite fortement certaines méthodes de test (ex : aléatoire)
- ❖ Point le plus mal maitrisé pour l'automatisation

# Oracles parfaits

---

- comparer à une référence : logiciel existant, tables de résultats
- résultat simple à vérifier (ex : solution d'une équation)
- disponibilité d'un logiciel similaire : test comparatif

# Oracles partiels

---

- \* oracle le plus basique : le programme ne plante pas
- \* instrumentation du code (assert)
- \* plus évolué : programmation avec contrats (Eiffel, Jml pour Java)

```
public class IntegerSet {  
    ...  
    byte[ ] a; /* The array a is sorted */  
    /*@ invariant  
        (\forall int i; 0 <= i && i < a.length-1;  
         a[i] < a[i+1]);  
     @*/  
    ...
```

# Oracles difficiles à obtenir

---

- ❖ IHM
- ❖ spécifications complexes
- ❖ certains logiciels de calcul scientifique (e.g. prédictions météo, séismologie etc.)
- ❖ propriétés non-fonctionnels

# Plan du cours

---

- ❖ Généralités sur le test
- ❖ Notions de bogue, erreur, défaillance, etc
- ❖ Gestion des erreurs
- ❖ Combien tester et quoi
- ❖ Oracle
- ❖ La maintenance

# La maintenance

---

- ❖ Le développement ne s'arrête pas le jour de la livraison
- ❖ Une ligne de produits s'étale souvent sur plus de 10 ans
- ❖ Des nouveaux besoins apparaissent durant cette période (au delà des besoins de correction d'erreurs)
- ❖ Age moyen des systèmes 7,5 ans
- ❖ 5% de systèmes ont des durées de vie > 20 ans  
(problèmes de renouvellement des équipes)

# Maintenance

---

- 50% des informaticiens affecté à la modification des applications existantes, plutôt qu'au développement de nouvelles applications

# Analyse et prévisions du métier d'informaticien aux US

Year	Development Personnel	Maintenance Personnel	Total Personnel	Maintenance Percent
1950	1,000	100	1,100	9.09%
1955	2,500	250	2,750	9.09%
1960	20,000	2,000	22,000	9.09%
1965	50,000	10,000	60,000	16.67%
1970	125,000	25,000	150,000	16.67%
1975	350,000	75,000	425,000	17.65%
1980	600,000	300,000	900,000	33.33%
1985	750,000	500,000	1,250,000	40.00%
1990	900,000	800,000	1,700,000	47.06%
1995	1,000,000	1,100,000	2,100,000	52.38%
2000	750,000	2,000,000	2,750,000	72.73%
2005	775,000	2,500,000	3,275,000	76.34%
2010	800,000	3,000,000	3,800,000	78.95%
2015	1,000,000	3,500,000	4,500,000	77.78%
2020	1,100,000	3,750,000	4,850,000	77.32%
2025	1,250,000	4,250,000	5,500,000	77.27%

Ober, 2012

# Bilan

---

- ❖ L'omni-présence du logiciel entraîne la prolifération des bogues
- ❖ Le test - technique qui permet de chercher des anomalies
- ❖ On ne peut pas tout tester => il faut être astucieux et trouver les bonnes tests
- ❖ A travers le test on ne put jamais être sûrs que le logiciel fonctionne correctement
- ❖ Le test est une première étape vers une maintenance réussie