Avant d'oublier ...

1. Bonjour ! !
2. Je m'appelle Nicolas Stouls. Enchanté.
3. Pour toute question : `nicolas.stouls@insa-lyon.fr`

Sécurité et sûreté de fonctionnement

Introduction

Nicolas Stouls
Université de Lyon
INSA-Lyon, CITI-INRIA, F-69621 Villeurbanne, France
Nicolas.Stouls@insa-lyon.fr

4 octobre 2015

Sûreté de fonctionnement

- Définition proposée par J-C Laprie :
  - Disponibilité : *Accessibilité en continue du système*
  - Fiabilité : *Continuité du service*
  - Confidentialité : *Non-divulgation d'information*
  - Intégrité : *Impossibilité d'altération des informations*
  - Maintenabilité : *Possibilités de réparation et d'évolution*
  - Sécurité-innocuité : *Absence de conséquences catastrophiques (conséquences environnementales, sociétales ou humaines)*

J-C Laprie, Guide de la sûreté de fonctionnement. Cépaduès, 1995

Sécurité

- Définissable par :
  - Disponibilité : *Accessibilité en continue du système*
  - Confidentialité : *Non-divulgation d'information*
  - Intégrité : *Impossibilité d'altération des informations*
  - Attaquant : *Présence d'une entité malveillante*
  - Vie privée : *Non divulgation d'informations personnelles*

Attention : le mot « Sécurité »est très surchargé.

Thématiques et organisation du cours

- Sûreté de fonctionnement
  - Nicolas Stouls (6h) : *méthodes de vérification, preuves*
- Sécurité
  - Yves Caniou (8h) : *Crypto, protocole IPSEC, SSL, VPN, certification, PKI*
  - Marine Minier (4h) : *Sécurité réseaux ad hoc et capteurs*
  - Mathieux Cunche (6h) : *Sécurité et vie privée dans la technologie Wi-Fi*
- Sécurité des Systèmes d'Information
  - François Lesueur (4h) : *Contrôle d'accès, détection/réaction (IDS/Firewall, antivirus, ..)*
- Évaluation : *CC + Examen avec seconde session*

Sûreté de fonctionnement

Avant propos

De quoi va-t-on parler ensemble ?

- De vérification de code

Que faut il retenir dans ce cours ?

- Définitions ? : *Seulement les plus importantes*
- Toutes les formules ? : *Non (Fournies au DS)*
- Leurs applications ? : *Au moins pour les principales*
- Les concepts, les idées ? : *oui*
- Comprendre à quoi ça sert ? : *oui*

Vérifier pour éviter les conséquences dramatiques

- Vérifier quoi ?
  - le système (Tests / Analyse de code)
  - la conception
    - Étude de la méthode de conception
    - Gain de confiance sur la qualité du développement
- Pourquoi le faire ?
  - ~~Pour le fun~~
    - *Quand vous verrez la complexité, vous rayerez ce choix*
  - Parce que c'est obligatoire
    - Aéronautique : DO-178C
    - Transports civils : pas de modification de la courbe de mortalité
  - Pour la reconnaissance clients
    - Évaluation selon les *critères communs*

Critères communs ?

- Depuis l'installation et le manuel jusqu'à la maintenance
- **Exemples :** *Carte de paiement, firewall, hyperviseurs, passeports électroniques ...*

Note : Evaluation Assurance Level

- EAL 1 : tests fonctionnels
- EAL 2 : tests structurels
- EAL 3 : tests et vérifications méthodiques
- EAL 4 : Conception, tests et vérifications méthodiques
- EAL 5 : Conception semi-formelle et tests
- EAL 6 : Vérification semi-formelle de la conception générale
- EAL 7 : Vérification formelle de la conception générale

Ces erreurs logicielles que l'on ne voudrait pas

Mars Climate Orbiter (1999)

- Principe :
  - Calculs d'approche de la sonde faits en livres.seconde
  - Code écrit en newton.seconde
  - 57km n'est définitivement pas une orbite aussi stable que 150km
- Conséquence : 125 millions de dollars

Ces erreurs logicielles que l'on ne voudrait pas

Vol inaugural d'Ariane 5 (2002)

- Principe :
  - Réutilisation du code d'Ariane 4
  - Extinction des sondes inertielles pour cause de valeurs mesurées trop grandes
  - Extinction du programme, car il n'était pas prévu que ce cas de figure arrive
- Conséquence : 8,5 milliards de dollars *(bug le plus coûteux de l'histoire)*

Ces erreurs logicielles que l'on ne voudrait pas

Système antimissiles Patriot (Guerre du Golfe, 1991)

- Principe :
  - Imprécision numérique dans le calcul d'une horloge
  - Nécessité de redémarrer le système toutes les 24h
- Conséquence : 28 morts.

Overflows de compteurs temps : vive l'aéronautique

Compteurs de temps en ms sur trop peu de bits

248j Système de gestion de l'énergie du Boeing 787 *(05/01/15)*

`http://federalregister.gov/a/2015-10066`

50j Système de contrôle aérien de la Caroline du Sud sous Windows XP *(21/09/10)*

`http://www.techworld.com/news/operating-systems/microsoft-server-crash-nearly-causes-800-plane-pile-up-3577711/`

145h Perte/redémarrage de l'ISIS sur A340, A318, A319, A320 et A321 *(21/12/05)*

ISIS (integrated standby instrument system) = altimètre, indicateur de vitesse, et horizon artificiel.

`http://ad.easa.europa.eu/blob/easa_2005_6439_fr_F20041681fb_fr.pdf/AD_F-2004-168R1_2`

Feriez vous mieux ? (1/2)

Ce code Java contient une erreur ... laquelle ?

```
int abs(int v){
  if (v < 0) {
    return -v ;
  }
  return v ;
}
```

Feriez vous mieux ? (1/2)

Ce code Java contient une erreur ... laquelle ?

```
int abs(int v){
  if (v < 0) {
    return -v ;
  }
  return v ;
}
```

Que se passe-t-il pour $-(2^{31})$?

Note : ce n'est pas un bug, mais une caractéristique documentée de l'API.

`http://docs.oracle.com/javase/8/docs/api/java/lang/Math.html`

Feriez vous mieux ? (2/2)

Java : recherche par dichotomie dans un tableau trié

```
public static int binarySearch(int[] a, int key) {
  int low = 0 ;
  int high = a.length - 1 ;
  while (low <= high) {
    int mid = (low + high) / 2 ;
    if (a[mid] < key) { low = mid + 1 ; }
    else if (a[mid] > key) { high = mid - 1 ; }
    else { return mid ; } // key found
  }
  return -1 ; // key not found.
}
```

Feriez vous mieux ? (2/2)

Java : recherche par dichotomie dans un tableau trié

```
public static int binarySearch(int[] a, int key) {
  int low = 0 ;
  int high = a.length - 1 ;
  while (low <= high) {
    int mid = (low + high) / 2 ; // Pb si low+high > 2^31
    if (a[mid] < key) { low = mid + 1 ; }
    else if (a[mid] > key) { high = mid - 1 ; }
    else { return mid ; } // key found
  }
  return -1 ; // key not found.
}
```

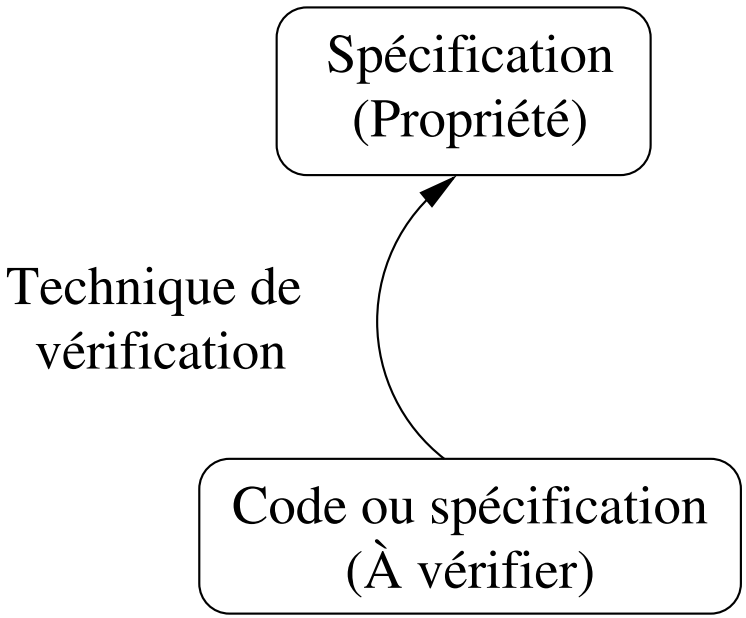
Bug présent dans le JDK de Sun :

`http://googleresearch.blogspot.com/2006/06/extra-extra-read-all-about-it-nearly.html`

Vérifier pour améliorer la qualité

Que vérifier ?

- Un programme **par rapport** à une spécification
- Une spécification **par rapport** à une spécification plus abstraite



Problèmes

- Quoi et quand vérifier ?
- Comment l'exprimer ?
- Comment vérifier ?

Un exemple simple de preuve de programme

Preuve de la fonction `abs` en C

Démo.

Sûreté de fonctionnement : *Quelques questions ....*

- Que veut on vérifier ?
- Comment l'exprimer ?

- À quel moment ? *(Conception, réalisation, runtime)*
- Comment le vérifier ?

Plan de cette partie du cours en 3 séances

- Techniques de GL pour la vérification de programmes :
  - Langages et méthodes de vérification
- Creusons un exemple concret :
  - Preuve de programmes