Introduction :
Pourquoi de la VVT ?

VVT : Validation, Vérification & Test des logiciels

Des bogues, des conséquences désastreuses…

- Banque de New York [21 novembre 1985] : pertes financières énormes
- Le Therac-25 [juillet 1985 ->avril 1986] : 3 morts
- Le crash d'AT&T [15 janvier 1990] : pertes financières énormes + la réputation d'AT&T entachée.
- Le Pentium [juin 1994] : pertes financières énormes + psychose
- Ariane 5-01 [4 juin 1996]

Ariane 5-01 (4 juin 1996)

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 inertielle (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" . . .

Les raisons :

1 Un bout de code d'Ariane IV (concernant le positionnement et la vitesse de la fusée) repris dans Ariane V

2 il contenait une conversion d'un flottant sur 64 bits en un entier signé sur 16 bits

3 pour Ariane V, la valeur du flottant dépassait la valeur maximale pouvant être convertie

4) défaillance dans le système de positionnement

5) la fusée a "corrigé" sa trajectoire

6) suite à une trop grande déviation, Ariane V s'est détruite !

Le coût d'un Bogue ?

- Coût du bogue de l'an 2000 ?
  - Quelques chiffres avancés : 300, 1600 ou même 5 000 milliards de dollars
- Quel impact ?
  - Sécurité des personnes,
  - Retour des produits,
  - Relations contractuelles,
  - Notoriété, image,
  - …

⇒ Nécessité de « vérifier » certains logiciels/systèmes

Nécessité de VVT

Comment effectuer de telles vérifications : Utilisation de méthodes formelles

1. **Test**
   - nécessaire : permet de découvrir des erreurs
   - pas suffisant : prouve la présence d'erreurs, pas leur absence !
2. **Démonstration automatique**
   - exhaustif
   - mise en œuvre difficile
3. **Model-checking**
   - exhaustif, partiellement automatique…
   - mise en œuvre moins difficile (modèle formel+formalisation des propriétés)

1, 2 et 3 sont des méthodes complémentaires :

- Test : non exhaustif mais facile à mettre en œuvre (bon rapport qualité/temps)
- Démonstration automatique : exhaustive mais considérée comme trop coûteux
- Model-checking : un compromis (?)

Sans méthodes formelles :

- Coût des tests : 50 à 60% du coût total, voire 70% !
- Interprétation(s) des termes usuels (-> utilisation d'UML)
- Ambiguïté des méthodes semi-formelles (# sémantiques UML).
- Maîtrise difficile de certains types de programmations
  [événementielle / parallèle / …]
- Maintenance évolutive difficile

Tendances actuelles ~ Méthodes formelles et certification

- Méthodes formelles :
  - **Test**, **Démonstration automatique**, **Model-checking**
- Politique de certification
- Certains niveaux de certification exigent des méthodes formelles
- Obligation de certification
  - Grandes entreprises
  - Application à risques
  - Sous-traitance

Test & Validation dans les méthodes formelles

- Objectif ~ Pouvoir raisonner sur les logiciels et les systèmes afin de :
  - **Connaître** leurs comportements
  - **Contrôler** leurs comportements
  - **Tester** leurs comportements.
- Moyen ~ Les systèmes sont des objets mathématiques.
- Processus :
  1. **Obtenir un modèle formel** du logiciel ou du système. [Si la taille le permet, le modèle peut être le logiciel ou le système]
  2. **Analyser** le modèle formel par une technique formelle.
  3. **Générer des test** par une technique formelle
  4. **Transposer** les résultats obtenus sur les modèles aux logiciels et systèmes réels.
- Problèmes de l'approche :
  - Le modèle est-il fidèle ? **Validition**.
  - Peut-on tout vérifier ? **Décidabilité**.
  - Peut-on tout tester ? **Testabilité**.
  - La transposition des résultats est-elle toujours possible ? **Abstraction**.
  - Le test est-il **correct** ? Le test est-il **exhaustif** ?