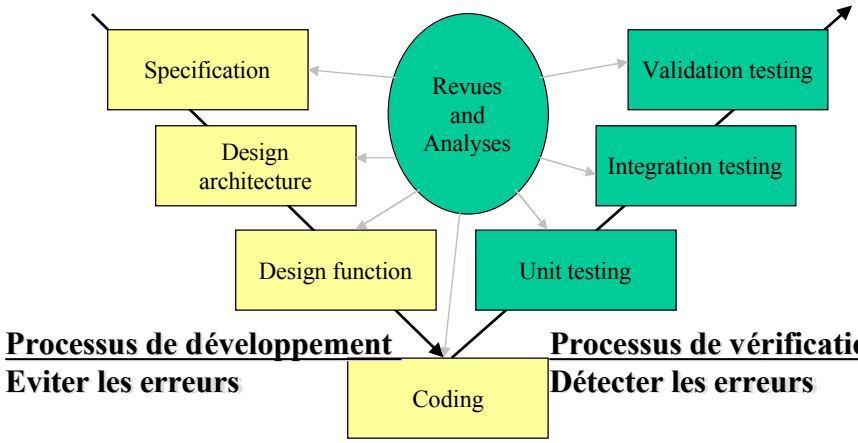
Vérification de logiciels par analyse statique

- Contexte et motivations
- Les techniques envisagées
- Evolution des processus
- Conclusion

AIRBUS

Contexte et motivations



Processus de développement
Eviter les erreurs

Processus de vérification
Détecter les erreurs

Le cycle de vie du logiciel

AIRBUS

Caractéristiques notables du processus de vérification

- •Objectifs déterminés par la réglementation
- •Basé sur les tests + revues, analyses manuelles
- •Coût élevé : plus de 50% du cycle de vie
- •Logiciels de tests => [2 - 10] x module
- •Moyens de tests spécifiques, dédiés à chaque phase

Les facteurs de changement

Implémentation des fonctions de plus en plus par logiciel

- •Nouvelles fonctions
- •Extension de fonctions
- •Migration hardware => software

=>Logiciel plus complexe à vérifier

Technologies Hardware toujours plus performantes

- •Processeurs
  - •FPU
  - •Pipeline, caches internes, MMU, superscalaire
- •Calculateurs
  - •Hiérarchie de mémoires, NUMA
- •Compilateurs
  - •Optimisations

=>Comportements plus difficiles à maîtriser

Evolution des exigences de vérification

- •Exigences réglementaires
- •Exigences industrielles
- •Exigences induites par les technologies

=>
Approche Tests x Analyses « manuelles »

- •De plus en plus limitée
- •Sans issue à terme

Techniques envisagées

Analyse statique outillée (ou preuves de propriétés)

- exhaustivité
- analyse du source (binaire dans certains cas)
- Les seuls moyens matériels sont des stations de travail de type courant
- degré d'automatisation
- la phase « intellectuelle » d'identification des cas de test n'a pas d'équivalent en preuve

But = Amélioration coût x efficacité de la vérification

⇒ Allégement des moyens mis en œuvre

⇒ Diminution du temps de vérification

⇒ Minimiser le coût de détection des erreurs.

Deux approches techniques

⇒Preuve exacte (logique de Hoare)

⇒Preuve approchée (interprétation abstraite)

Stratégie cible

⇒Preuves x tests x revues

Les outils

CAVEAT

- •Preuve exacte - R&D en collaboration avec le CEA
- •Relations logiques exprimées sur les variables
- •Programme source C
- •Cibles:

  ⇒ Sûreté de fonctionnement : car exigence d'exhaustivité, lourdeur des moyens de tests (parfois)

  ⇒ **Vérification unitaire : remplacer le TU pour la vérification algorithmique**.

Les outils

ABSINT/WCET

- •Preuve approchée - Projet IST DAEDALUS
- •Borne optimale pour le pire cas du temps d 'exécution
- •Binaire exécutable (Coldfire, PPC)
- •Cible:

  ⇒Vérification d 'intégration : remplacer test + analyse

Les outils

ABSINT/STACK

- •Preuve approchée - R&D ABSINT
- •Borne optimale pour la pile d ’exécution
- •Binaire exécutable (PPC)
- •Cible:

⇒Vérification d ’intégration : remplacer test + analyse

Les outils

Polyspace Verifier

- •Preuve approchée - Projet IST DAEDALUS
- •Propriétés de run-time error
- •Programme source C
- •Cibles:

⇒ Vérification unitaire/intégration/analyses manuelles

⇒Certains mécanismes de tolérance aux fautes

Les outils

CEA/Floating Point

- •Preuve approchée - Projet IST DAEDALUS
- •Précision des calculs en flottant
- •Cibles:

  ⇒Vérification unitaire/intégration : remplacer les analyses manuelles « assistées »

Evolution des processus

L ’introduction des techniques de preuves:

- •Nouvelles méthodes
- •Aménagements du processus de vérification
- •Intégration dans la gestion des processus du cycle de vie
- •Facilité d ’apprentissage et d ’usage par les équipes de développement

Conclusion

Preuve exacte

- phase R&D quasiment terminée
- prochain objectif : introduction effective sur un logiciel réel

Preuve approchée

- phase R&D
- en règle générale, les outils basés sur l'interprétation abstraite permettront d'automatiser assez largement des analyses qui sont aujourd'hui essentiellement intellectuelles