Validation des logiciels libres

Mythes et solutions

Hélène Waeselynck

LAAS-CNRS

Validation de masse : un mythe

"Given enough eyeballs, all bugs are shallow"

E. Raymond, The Cathedral and the Bazaar

- Croyance que le logiciel a déjà été validé par beaucoup d'autres utilisateurs
- Manque de motivation
- Barrières techniques

⇒ **Validation: démarche volontaire et organisée**
noyau de développeurs, société support, utilisateur

⇒ **LL et sûreté de fonctionnement : l'intégrateur doit apporter les démonstrations requises pour un système cible**
focalisation sur le produit plutôt que sur le processus

Plan

- Vérification de modèle
- Analyse statique
- Test de conformité
- Test de robustesse

- Aspects spécifiques à la sécurité-confidentialité (malveillances)

- Mutualisation des efforts de validation et capitalisation des résultats

Vérification de modèle (model-checking)

- Modèle = composition d'automates
- Propriété vérifiée = formule logique temporelle
- Idéalement: consolidation de la spécification avant d'aller vers le code
- … Mais peut aussi être utilisé a posteriori !

Modèle = abstraction du code source

Approche notamment utilisée pour la vérification de protocoles

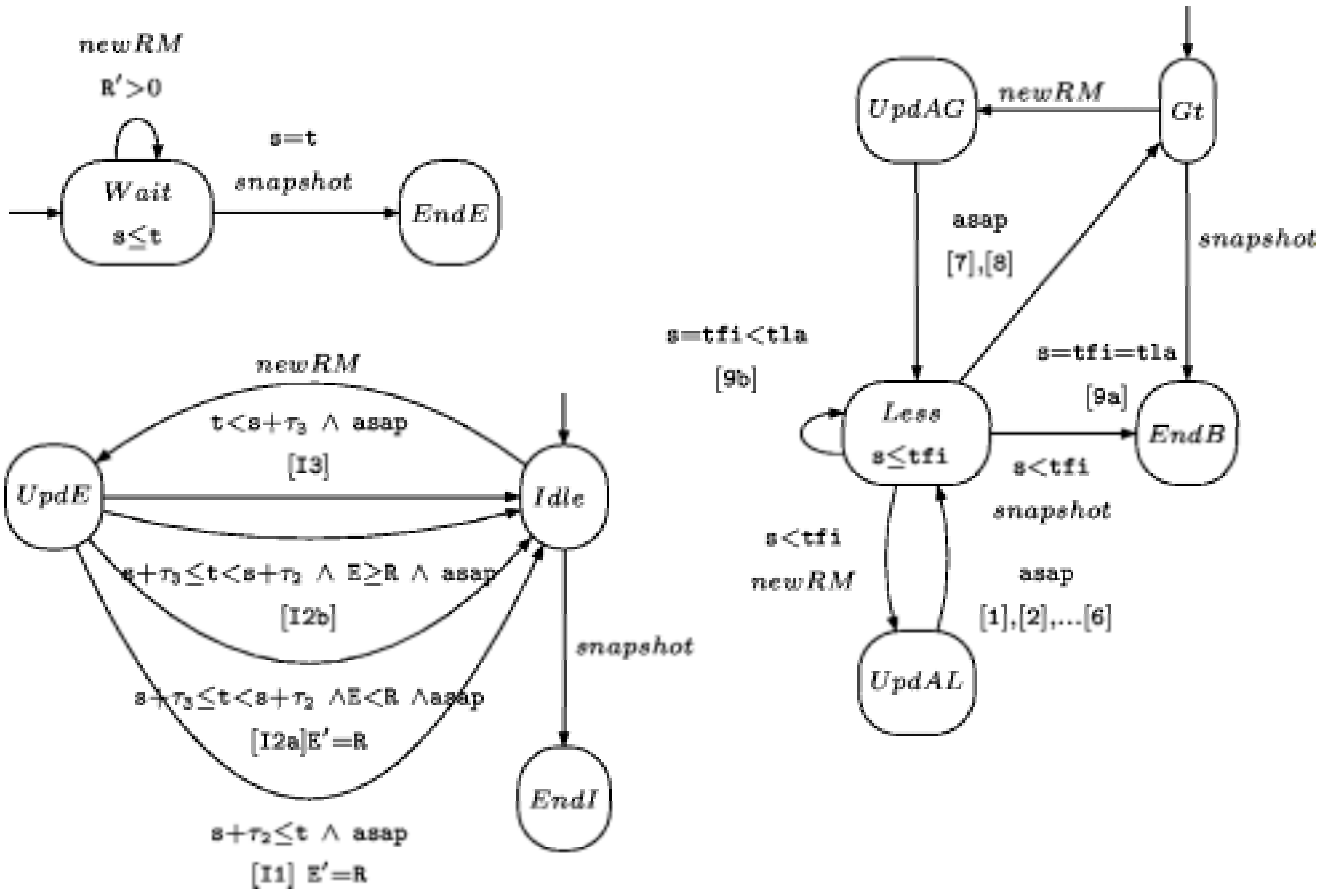
- Protocole de contrôle audio-vidéo de Bang & Olufsen (vérification par univ. Aalborg)
- Protocole ABR de France Telecom (vérification par LSV)

…/…

Algorithme en pseudo-code

```
/* Arrivée d'une nouvelle cellule RM (valeur R) */
if s < tfi then
   if Emx <= R then
      if tfi < s+tau3 then
         if s+tau3 < tla or tfi=tla then
            [1] Emx:= R; Ela:= R; tla:= s+tau3
         else
            [2] Emx:= R; Ela:= R
      else
         if A <= R then
            [3] Emx:= R; Ela:= R; Efi:= R; tfi:= s+tau3; tla:= s+tau3
         else
            [4] Emx:= R; Ela:= R; Efi:= R; tla:= tfi
   else
      if R < Ela then
         [5] Efi:= Emx; Ela:= R; tla:= s+tau2
      else
         [6] Efi:= Emx; Ela:= R
else
   if A <= R then
      [7] Efi:= R; Ela:= R; Emx:= R; tfi:= s+tau3; tla:= s+tau3
   else
      [8] Efi:= R; Ela:= R; Emx:= R; tfi:= s+tau2; tla:= s+tau2
   ...
```

Modèle (LSV)



Vérification d'une propriété de QdS

Analyse statique

- Vérification de propriétés sur le code source, sans exécution de ce code (outils)
- Exemple de propriétés : précision de calculs numériques, débordement de tampons, accès à des variables non initialisées, pire temps d'exécution (WCET), …

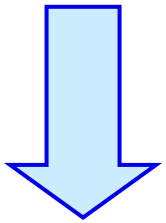
Restrictions sur le code source

Exemple: analyse de WCET sur RTEMS (IRISA)

- Analysé par l'outil HEPTANE (logiciel libre, IRISA)
  - 14KLOC analysées = 12 directives du noyau (gestion de tâches, sémaphores, gestion du temps, interruptions)
  - Analyse code source $\Rightarrow$ pire chemin d'exécution
  - Analyse code objet + modèle micro-architecture $\Rightarrow$ WCET des suites d'instructions correspondantes

Résultats

- **Restrictions HEPTANE réalistes pour les sources analysés**
  - Peu de code non structuré
  - Peu d'appels dynamiques, et fonction toujours connue statiquement
- **Identification automatique des portions de code dont le temps d'exécution est non déterministe**



Modifications du code peu nombreuses et généralement mineures

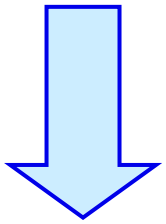
- **Obtention de WCET sûrs et raisonnablement pessimistes**
  - 1.8 x temps mesurés par test
  - 7 x temps mesurés par test si micro-architecture non prise en compte

Modélisation incontournable

☺ Intérêt du matériel libre

Test de conformité

- **Vise à vérifier si un logiciel satisfait ses spécifications**
  - Existence d'un document de spécification ?
  - Existence d'une documentation de test ? Traçabilité / exigences fonctionnelles ?



☹ **Tests fournis (s'ils existent) souvent inexploitables pour validation de conformité**

- 😐 mais permettent de s'assurer que LL fonctionne comme chez les développeurs

☺ **Si LL conforme à standard d'interface (POSIX, CORBA, …):**

- Spécification = document normatif
- Existence de tests de conformité (ex: http://www.opengroup.org, tests utilisés par Airbus lors des expériences de portage de Linux sur l'ATSU)

Test de robustesse

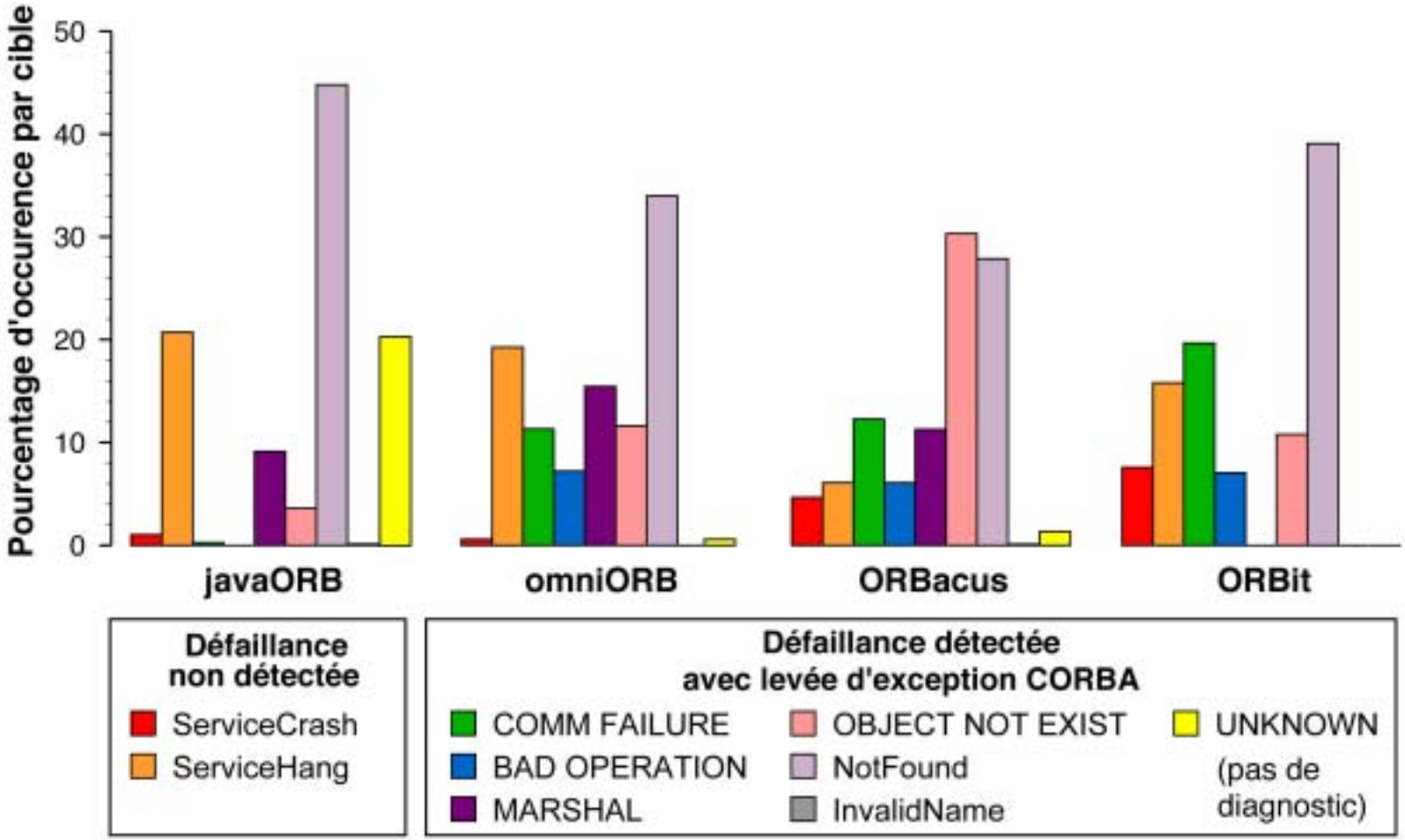
- **Test du comportement en présence d'entrées erronées ou de conditions environnementales stressantes**
  - **Comparaison de la robustesse de produits concurrents**
  - **Caractérisation des modes de défaillance pour la mise en œuvre de parades (ex: empaquetage)**

- **Approche notamment utilisée sur des supports exécutifs COTS ou LL**
  - **micro-noyaux, systèmes d'exploitation, intergiciels**

Exemple: test d'intergiciels CORBA (LAAS-CNRS)

- Test ciblant le « service de noms » de quatre implémentations CORBA (2 COTS + 2 LL)

- Modèle de fautes = corruption de requêtes IIOP entrantes
  - Bit-flip + double-zéro
  - Représentatif de fautes transitoires du système de communication (d'après une étude récente [Stone 2000], 1 erreur non détectée toutes les 80 heures sur un LAN 100 Mb/s saturé)



- **LL aussi robustes que COTS**
  - **les plus robustes : ORBacus (COTS) et omniORB (LL)**
- **Identification d'une faiblesse commune**
  - **Sensibilité à certains bits de l'en-tête IIOP**

Aspects spécifiques à la sécurité-confidentialité

- LL plus sûrs (*secure*) que COTS : une vive controverse
  - source ouvert ⇒ pas de logiques malignes et moins de vulnérabilités ?
  - Mais le source a-t-il réellement été audité en profondeur ? (cf. mythe des “Many eyeballs”)
  - Source ouvert ⇒ plus facile pour les attaquants ?
  - Rq : en pratique, COTS autant attaqués que LL
- Qualité et robustesse du code ⇒ impact positif sur la sécurité
  - Exemples de failles de sécurité dues à des fautes de conception (non-contrôle des débordements de tableaux, des formats de chaînes de caractères, …)
  - Pertinence des méthodes de validation présentées précédemment (notamment : analyse statique, test de robustesse / entrées invalides)

Outils spécifiques

/* Identification de vulnérabilités */

- **Analyseurs de vulnérabilités**
  - État des lieux / vulnérabilités connues
- **Scanners de port**
  - Identification des services réseaux actifs et de leurs vulnérabilités
- **Générateurs de paquets**
  - Test de robustesse / faux paquets

/* Aide à la détection d'intrusion */

- **Analyseurs de logs**
  - Détection du comportement anormal d'un logiciel / système / utilisateur
- **Observateurs du trafic réseau**
- **Vérificateurs de l'intégrité de fichiers**

Mutualisation et capitalisation

- **Ressources offertes par les développeurs**
  - FAQ
  - Forums de discussion
  - Rapports de bogues et base de données des bogues
- **Coté utilisateurs : quelques initiatives**
  - CERT Coordinatination Center (DARPA) ⇒ traitement d'incidents de sécurité, base de données de vulnérabilités
  - Projet Sardonix (DARPA) ⇒ portail dédié aux audits de logiciels libres, avec système de notation des auditeurs
  - Linux Test Project (SGI, IBM, OSDL, Bull, Wipro Technologies)

Conclusion

- Qualité de certains logiciels libres comparable à celle de COTS
- Documentation souvent insuffisante pour démonstration de sûreté de fonctionnement
- Mais des solutions de validation existent …
- … Et gagneraient à être davantage mutualisées, côté utilisateurs !