Vers une meilleure maîtrise des défaillances

L'empaquetage comme moyen de protection

Isabelle Puaut

INSA - IRISA

Problématique

- Connaissance partielle des caractéristiques du logiciel libre
  - Hypothèses de conception
  - Performances
  - Propriétés de Sûreté de Fonctionnement : modes de défaillances
  - Fonctions mal documentées
  - Interopérabilité...
- Evolutions fréquentes, pas toujours contrôlées

Nécessité de principes architecturaux pour les intégrer dans des systèmes critiques

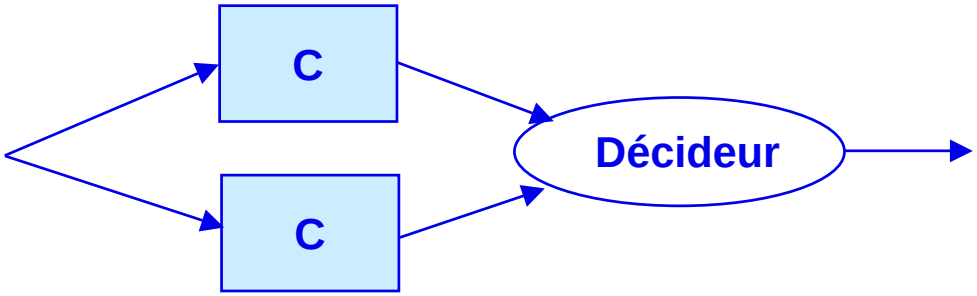
Problématique est similaire à celle d'intégration des COTS

Organisation de l'exposé

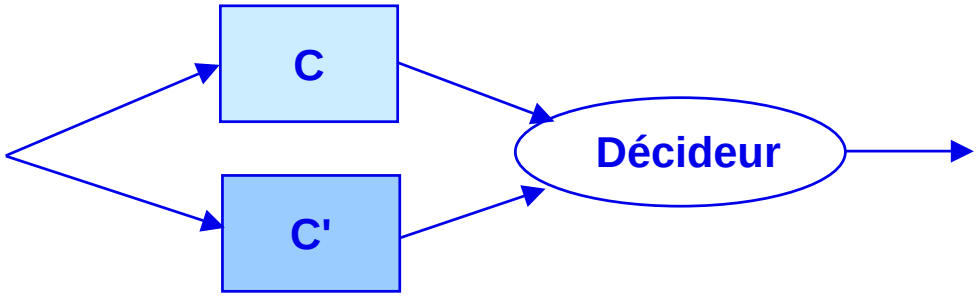
- Solutions architecturales pour l'intégration de LL
- L'empaquetage (wrapping) comme moyen de protection
- Deux expériences d'empaquetage (IRISA - LAAS) : Empaquetages d'exécutifs
  - Objectifs de l'empaquetage
  - Architecture logicielle et contenu des wrappers
  - Validation des wrappers

Principes architecturaux

Décorrélation de l'activation
(= contextes d'activation différents)



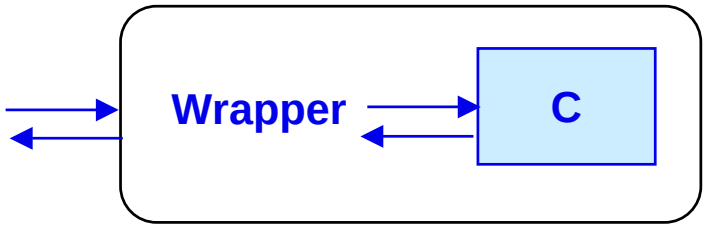
Diversification



Partitionnement

- **Physique**
  Impact sur architecture matérielle
- **Logique**
  Spatiale (ex: mémoire),
  temporelle (ex: ordonnancement)

Empaquetage



.../...

Empaquetage

- **Origine du terme**
  - **Groupe de travail de l'ISAT (Information Science and Technology) de la DARPA**
- **Deux objectifs**
  - **Adaptation des interfaces**
    - Homogénéisation des interfaces face à l'hétérogénéité des composants intégrés (que l'on veut ne pas modifier dans le cas de LL)
  - **Protection**
    - Augmentation des capacités de détection et de confinement d'erreurs
- **Intérêt**
  - **Impact localisé sur l'architecture logicielle du système**
    - à l'interface avec les composants utilisés
  - **Pas d'impact sur l'architecture matérielle**
  - **Coût plus faible que les méthodes de diversification**

Niveaux de protection

- **Faible**
  - Simple adaptation des interfaces (ex: restrictions fonctions API)
  - Complément de fonctionnalités
- **Moyen**
  - Vérification des interactions avec le composant (ex: plages de valeurs)
- **Fort**
  - Assertions plus poussées sur les fonctions (pré et post-conditions)
  - Très dépendantes de la finesse des spécifications du fonctionnement du composant
  - Très dépendantes du niveau d'observabilité
  - Intérêt dans le cadre des LL : haut niveau d'observabilité et commandabilité (accès au code source)

Le contenu d'un wrapper dépend fortement du degré de protection assuré

Wrappers actifs vs passifs

- **Passif**
  - simple notification d'erreur retournée au logiciel applicatif
  - Décision d'engager une procédure de recouvrement est laissée au niveau supérieur
- **Actif**
  - Engagement par le wrapper d'une procédure de recouvrement (reconfiguration dynamique et reprise du service, re-tentative d'invocation, …)
  - Rôle correctif du comportement anormal du composant encapsulé

Organisation de l'exposé

- Solutions architecturales pour l'intégration de LL
- L'empaquetage (wrapping) comme moyen de protection
- Deux expériences d'empaquetage (IRISA - LAAS) : Empaquetages d'exécutifs
  - Objectifs de l'empaquetage
  - Architecture logicielle et contenu des wrappers
  - Validation des wrappers

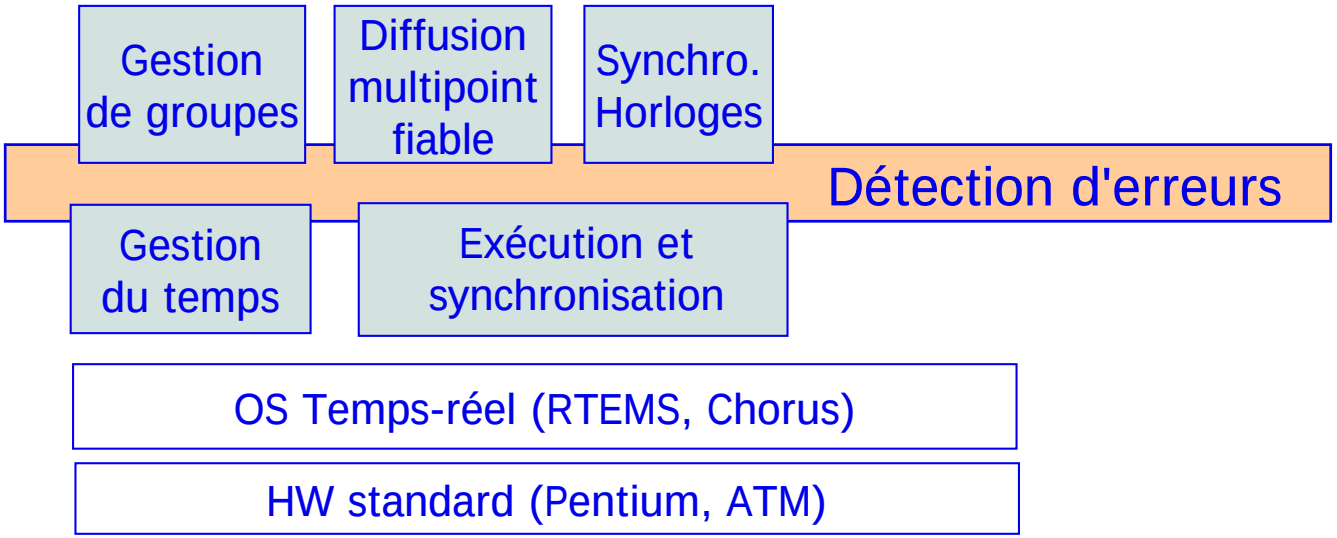
Hades (Highly Available Distributed Embedded Systems) - IRISA

- **Logiciel encapsulé**
  - **exécutif temps-réel (LL ou COTS)**
- **Objectif global**
  - **système temps-réel tolérant les fautes physiques, faible coût**
- **Architecture**
  - **intergiciel encapsulant l'exécutif temps-réel**

(Coopération INRIA - DGA - Dassault-Aviation)

Hades : objectifs de l'empaquetage

- **Homogénéisation des interfaces**
  - **API indépendante de celle des systèmes d'exploitation encapsulés**
- **Ajout de fonctionnalités**
  - **Gestion de la distribution (synchronisation d'horloges, gestion de groupes, multicast, ordonnancement)**
- **Détection d'erreurs**



Prédicats de détection d'erreurs

- **Types de prédicats**
  - Cohérence des structures de données par redondance (ordonnancement, communications, …)
  - Contrôles temporels (durées d'exécution, échéances, arrivées)
  - Vérifications d'exécution (graphes d'appel, transition d'état des tâches)
  - Vérifications diverses sans redondance
- **Wrapper passif**
  - recouvrement d'erreur au niveau supérieur
- **Détection de la propagation des erreurs de l'exécutif vers les wrappers**
  - Aucune modification de l'exécutif encapsulé

Evaluation des wrappers

- **Méthode d'évaluation : injection de fautes par logiciel**
  - Altération de mots mémoires (bit-flip)
  - Fautes physiques temporaires et permanentes
  - Cible = exécutif d'une machine
- **Système cible : micro-noyau Chorus/ClassiX**
- **Effet des fautes**
  - Défaillance de l'application (INCORRECT)
    résultat incorrect, blocage de l'application, dépassement d'échéance
  - Détection d'erreur sans propagation aux autres machines (DETECTE)
  - Application correcte (CORRECT) : erreur masquée ou latente

Evaluation des wrappers

- **Couverture de la détection d'erreurs**

	Pas de wrapper	Avec wrapper
Fautes injectées	3152	3012
CORRECT DETECTE, SANS PROPAG INCORRECT	1839 (58.3%) 703 (22.3%) 610 (19.4%)	1122 (37.3%) 1862 (61.8%) 28 (0.9%)
Couverture Interv. confiance 95%	**80.65%** [79.49 , 81.79]	**99.07%** [98.80 , 99.33]

- **Wrappers efficaces : nombre de comportement incorrects divisé par 22**
- **Redondances entre prédicats de détection d'erreurs**

Organisation de l'exposé

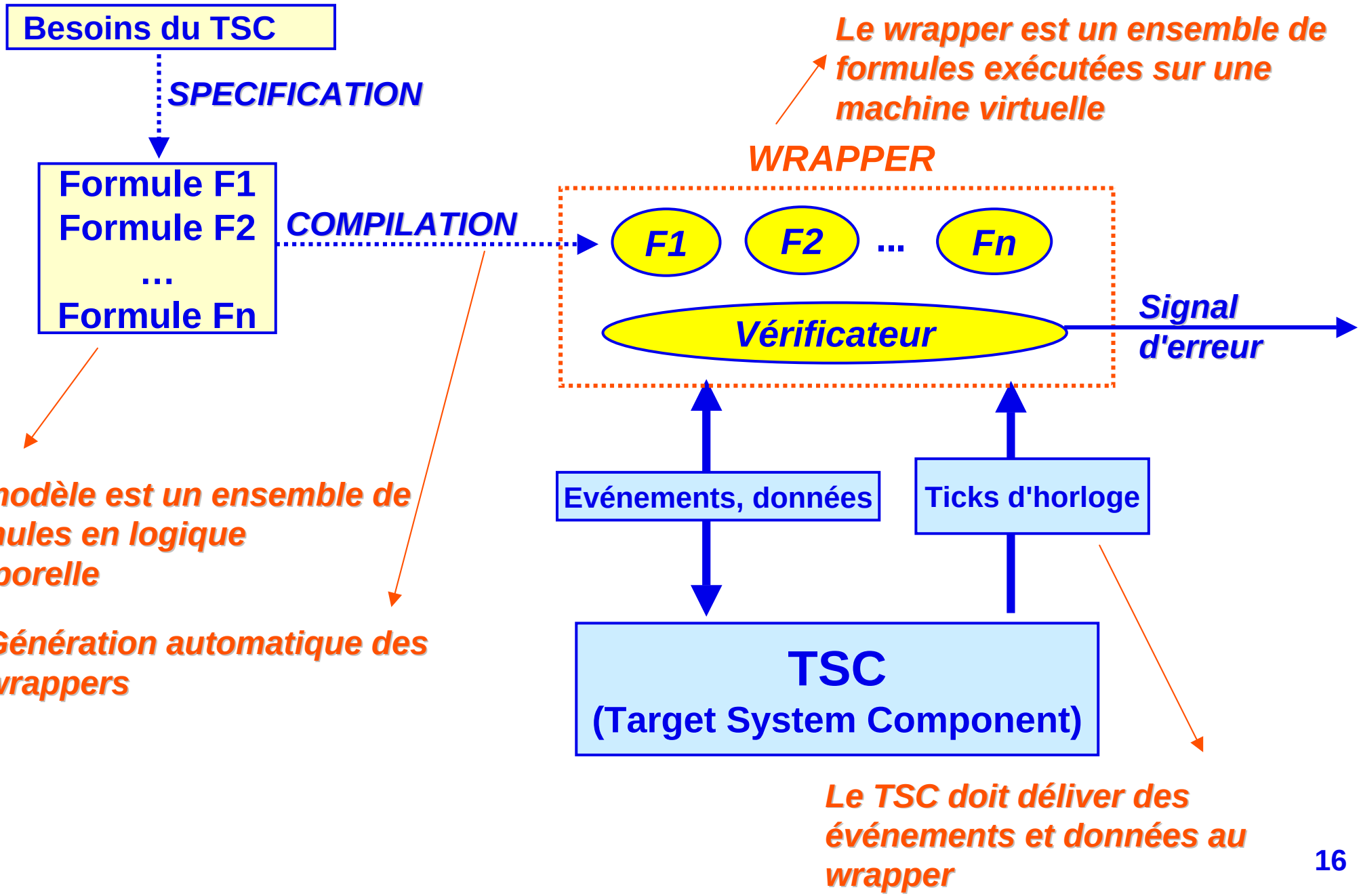
- Solutions architecturales pour l'intégration de LL
- L'empaquetage (wrapping) comme moyen de protection
- Deux expériences d'empaquetage (IRISA - LAAS) : Empaquetages d'exécutifs
  - Objectifs de l'empaquetage
  - Architecture logicielle et contenu des wrappers
  - Validation des wrappers

LAAS - encapsulation d'exécutifs

- **Objectifs**
  - **Détection d'erreurs : exécution de prédicats élaborés sur le fonctionnement du composant cible**
    - Fautes de conception et de réalisation, effets de fautes physiques
    - Augmente la sûreté de fonctionnement en évitant la propagation d'erreurs
  - **Pas d'adaptation des fonctionnalités ou de filtrage d'API (API inchangée)**
  - **A la base, détection d'erreurs seulement (wrapper passifs), étendu au recouvrement d'erreur par poursuite (wrappers actifs)**
- **Composants cibles**
  - **Composants arbitraires**
  - **Evaluation sur les exécutifs temps-réel (suite de l'exposé)**

Coopération LAAS-Thales dans le cadre du LIS
Projet IST DSoS

Cadre général



Expression des prédicats

- **Types de propriétés classiques : sûreté et vivacité**
  - **Sûreté: une situation non désirée ne se produira pas, e.g., interblocage, émission de données incorrectes**
  - **Vivacité: une situation attendue se produira, e.g., réponse attendue à une entrée**
- **Exemples**
  - **Propriété de sûreté concernant l'ordonnancement des tâches**
    La tâche «idle» ne doit pas s'exécuter quand une tâche utilisateur est prête à s'exécuter

    $\text{Always (event=CS} \wedge (\exists\ s \in \text{Tasks: } s \neq \text{idle} \wedge s \in \text{ReadyQueue}) \Rightarrow \text{Running} \neq \text{idle)}$
  - **Propriété de vivacité sur les temporisateurs (timers)**
    un temporisateur se déclenchera de manière certaine

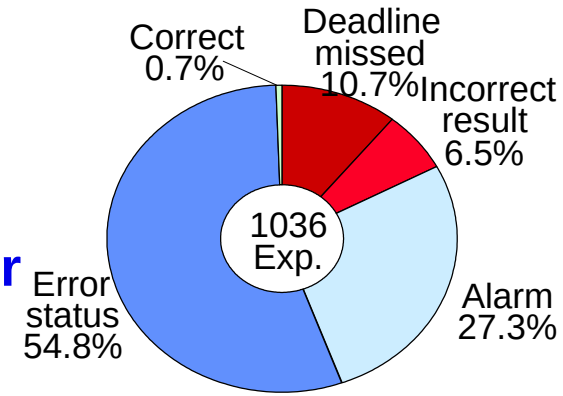
    $\text{Always (event = SetTimeout (t, tempo)} \Rightarrow \text{Nexttempo (t} \notin \text{TimeoutQueue))}$
- **L'expression des propriétés est indépendante de l'implantation du TSC**

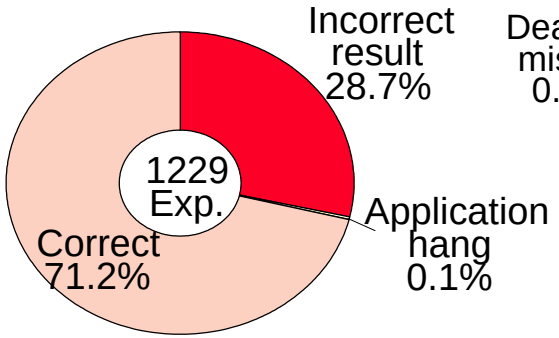
Evaluation des wrappers

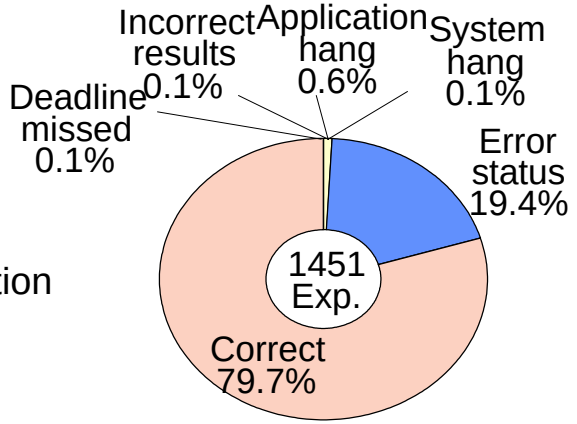
- **Méthode d'évaluation : injection de fautes par logiciel MAFALDA-RT (LAAS)**
  - **Fautes : d'appel + physiques**
  - **Bits-flips dans les paramètres et en mémoire**
- **Composant cible : micro-noyau temps-réel Chorus/ClassiX**
- **31 wrappers dans les modules du noyau**
  - **ordonnancement, synchronisation, gestion du temps**
- **Effets des fautes**
  - **Défaillance de l'application (échéance manquée, résultat incorrect, blocage applicatif, blocage système)**
  - **Erreur détectée (code d'erreur, alarme)**
  - **Erreur masquée ou latente (résultats corrects)**

Evaluation des wrappers

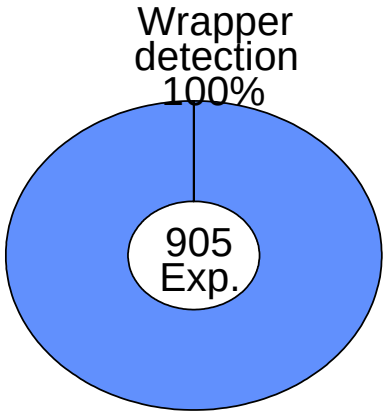
Sans wrapper



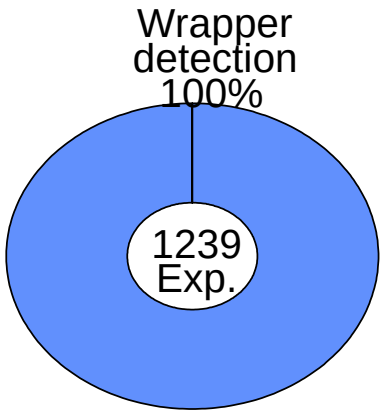




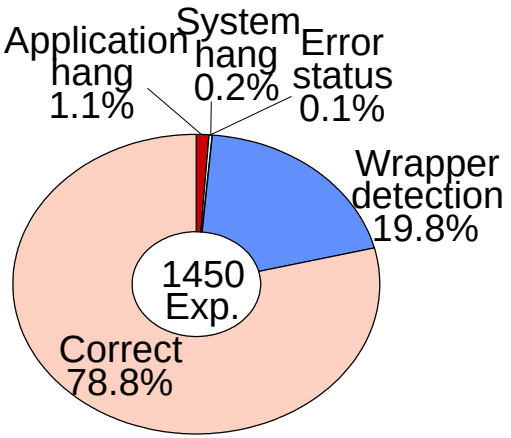
Avec wrapper



Scheduler



Timer



Synchro

- **Bonne capacité de détection des erreurs**
- **Redondance entre wrappers**
  - **(4 wrappers détectent toutes les erreurs)**

Conclusion

- **Solution générale**
  - **Adaptation des interfaces faces aux évolutions des LL**
  - **Détection et confinement des erreurs au niveau du LL intégré**
  - **Différents niveaux de protection**
    - Niveau de protection élevé atteignable sur des LL (commandabilité et observabilité des LL dû à l'accès au code source)
  - **Efficacité de la protection**
- **Applicable à de nombreux LL (systèmes d'exploitation - Linux, Corba, ...)**
- **Modifications localisées dans l'architecture logicielle du système**
- **Coût plus faible que les méthodes de diversification**
  - **Coût localisé à l'interface des composants encapsulés**