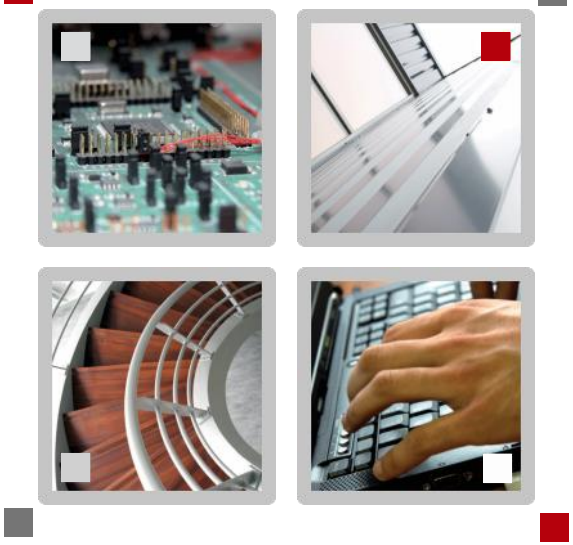
Gestion de la variabilité dans les projets de test logiciel



Bertrand BATOGE
bertrand.batoge@kereval.com

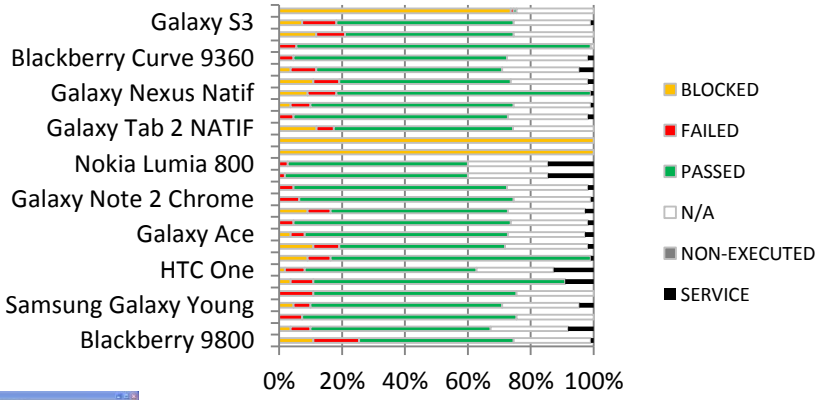
La variabilité dans le logiciel

- La variabilité, dans le logiciel, prend différentes formes :
  - Dues à l'assemblage de multiples composants

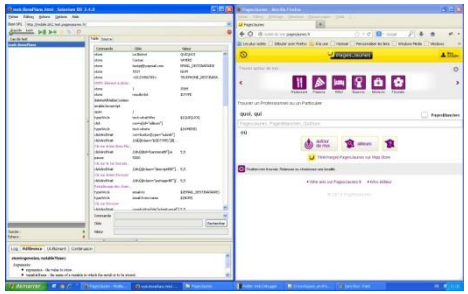
  - Dues à des contextes d'exécution hétérogènes
    - Un logiciel doit fonctionner sur une multitude d'ordinateurs différents
    - Une appli mobile doit fonctionner sur les smartphones et tablettes du marché
  - Dues à de nombreuses façons de configurer une application :
    - Compilateur GCC :
      - 199 paramètres, 40 contraintes, 4.6 * 1061 configurations

Un problème transverse aux projets de test

- Ces défis se retrouvent sur de multiples projets :
  - La validation des applications destinées aux téléphones mobiles

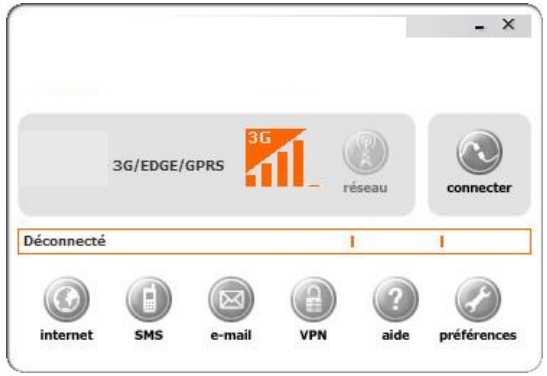


  - Validation de sites web



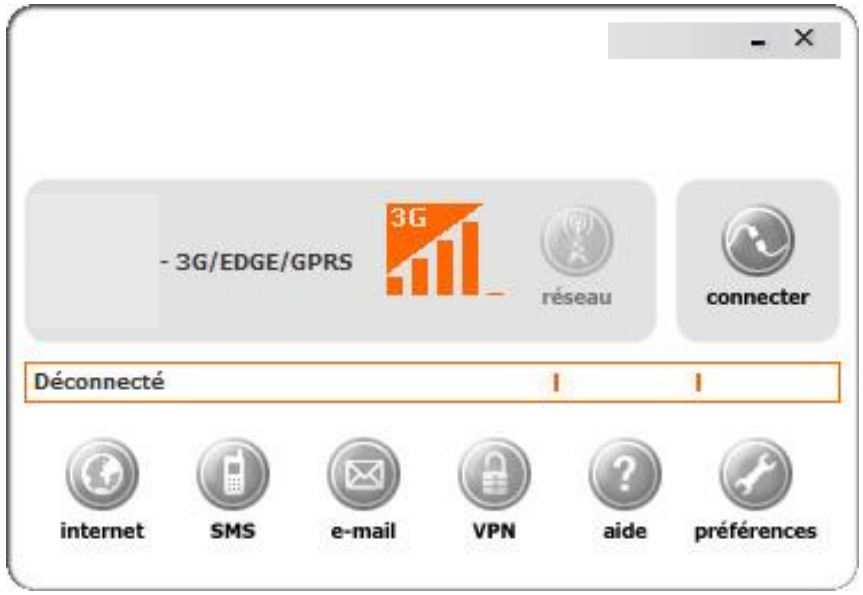
  - Validation d'applications lourdes :

Kits de connexion réseau 3G



Exemple : Projet KIT

- Kit de connexion pour les professionnels en mobilité
- Gère des connexions 3G - Wifi – Ethernet
- Fournit un grand nombre de fonctionnalités:
  - SMS
  - Email
  - VPN…



Les spécifications du projet KIT

Un grand nombre d'environnements d'exécution:

OS (5): Win. 2000, Win. XP 32 bits, Win. XP 64 bits, Win. Vista 32 bits,Win. Vista 64 bits

Mobile (25): Novatel Xu870, GT Max GX0301, Lucent Merlin U530, Huawei, E870...

Wifi internal (5): intel centrino 2100, 2200, 2915, 3945,

Wifi external(8): Sagem 706 A, Sagem 703...

Modem (8): Sagem F@st 800 USB, Thomson ST330, Siemens A100, ZTE ZXDSL 852...

VPN (4): Safenet, Cisco, Avasy, empty

Mail Client (4): Outlook, Outlook Express, Windows Live Mail, empty

Browser (4): Firefox 2.0, Firefox 1.5, Internet Explorer 5.5, empty

Une configuration est une sélection d'un élément de chaque catégorie :

Les spécifications du projet KIT

Un grand nombre d'environnements d'exécution:

OS (5): Win. 2000, Win. XP 32 bits, Win. XP 64 bits, Win. Vista 32 bits,Win. Vista 64 bits
Mobile (25): Novatel Xu870, GT Max GX0301, Lucent Merlin U530, Huawei, E870...
Wifi internal (5): intel centrino 2100, 2200, 2915, 3945,
Wifi external(8): Sagem 706 A, Sagem 703...
Modem (8): Sagem F@st 800 USB, Thomson ST330, Siemens A100, ZTE ZXDSL 852...
VPN (4): Safenet, Cisco, Avasy, empty
Mail Client (4): Outlook, Outlook Express, Windows Live Mail, empty
Browser (4): Firefox 2.0, Firefox 1.5, Internet Explorer 5.5, empty

2.560.000 environnements différents

La mise en place d'un environnement de test prend du temps et coûte en argent et en ressources.

Problèmes :

- Ces **2.560.000** environnements d'exécution contiennent des configurations invalides
  - Comment identifier et supprimer ces configurations invalides ?
- Comment tester **1461** exigences sur **2.560.000** environnements ?
- Une approche exhaustive :
  - **1461 * 2.560.000 = $3.7*10^9$ TC** à exécuter pour vérifier chaque exigence dans chacune des configurations

➔ Une autre approche est nécessaire

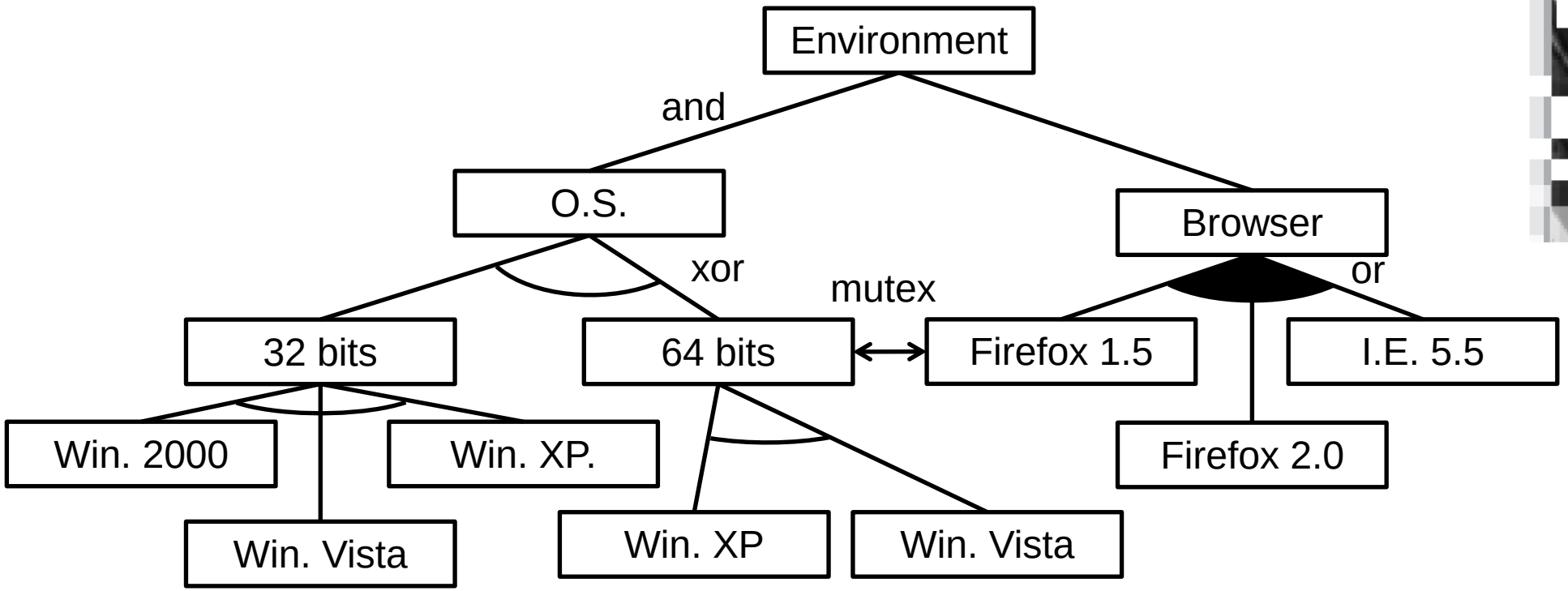
Défis posés par la variabilité

- Comment représenter la variabilité dans le logiciel ?
  - **Les modèles de features** permettent de capturer les relations entre les éléments de l'environnement
- Comment identifier les configurations à tester ?
  - **Le Test Pairwise permet d'extraire un sous-ensemble pertinent et raisonnable de configurations**

Contexte

MODÈLES DE FEATURES

Modèle de features



	Environment	O.S.	32 bits	64 bits	Win. 2000	Win. Vista	Win. XP.	Win. XP (x64)	Win. Vista (x64)	Browser	Firefox 1.5	Firefox 2.0	I.E. 5.5
Configuration valide	1	1	1	0	1	0	0	0	0	1	1	1	1
Configuration valide	1	1	0	1	0	0	0	1	0	1	0	1	1
Configuration invalide	1	1	0	1	0	0	0	1	0	1	1	0	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Contexte

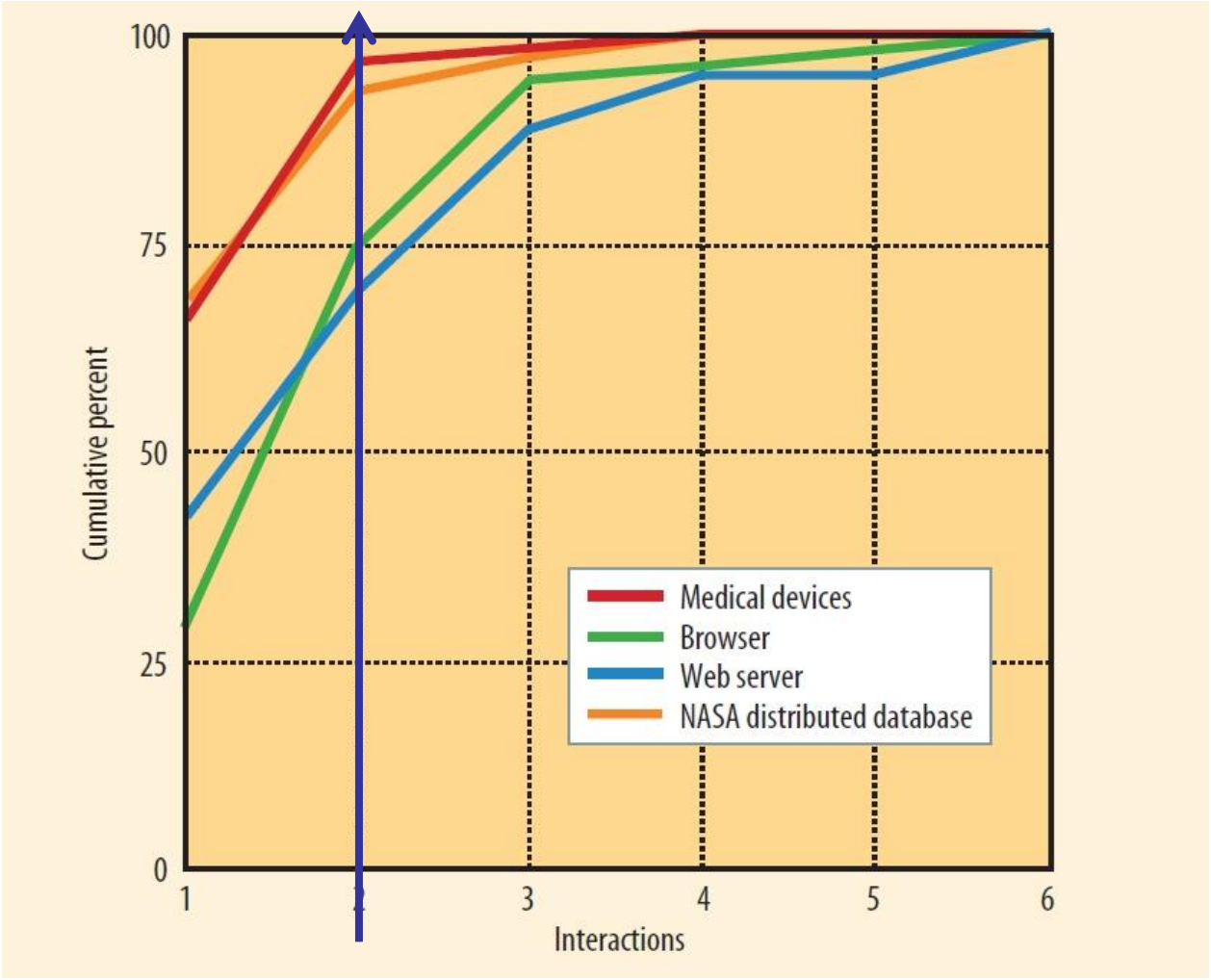
TEST PAIRWISE

Test pairwise

- Définition
  - Critère de test combinatoire basé sur l'hypothèse *réaliste* que les *défauts* sont causés par l'interaction d'au plus deux facteurs (paramètres ou composants).
  - Permet de réduire l'explosion combinatoire pour la sélection des cas de tests lorsqu'un comportement dépend de plusieurs variables dont le domaine est **fini**.

Pertinence du test Pairwise

Pourcentage de défauts détectés en fonction du degré de test



« Test Pairwise »

[KHUN, OWASP 2005,SOFTWARE FAULT INTERACTIONS]

Navigateur : Mozilla
Serveur : Apache

Test pairwise, un exemple

Browser	FF	CHROME	IE
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

TEST EXHAUSTIF

4 features :
{Browser, FF, Chrome, IE}

2 valeurs possible : 0 ou 1

Browser	FF	CHROME	IE
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	0

TEST PAIRWISE

Test pairwise, un exemple

4 features : {Browser, FF, Chrome, IE}

2 valeurs possible : 0 ou 1

Test Pairwise : **5 configurations**

Browser	FF	CHROME	IE
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	0

TEST PAIRWISE

Test pairwise, un exemple

4 features : {Browser, FF, Chrome, IE}

2 valeurs possible : 0 ou 1

Test Pairwise : **5 configurations**

Browser	FF	CHROME	IE
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	0

TEST PAIRWISE

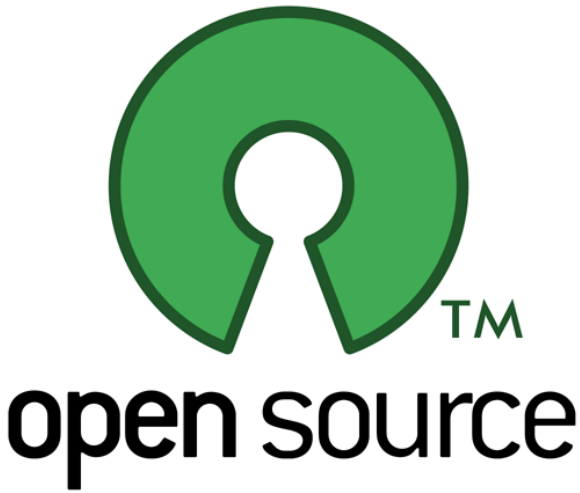
Background

TEST PAIRWISE ET LES MODÈLES DE FEATURES

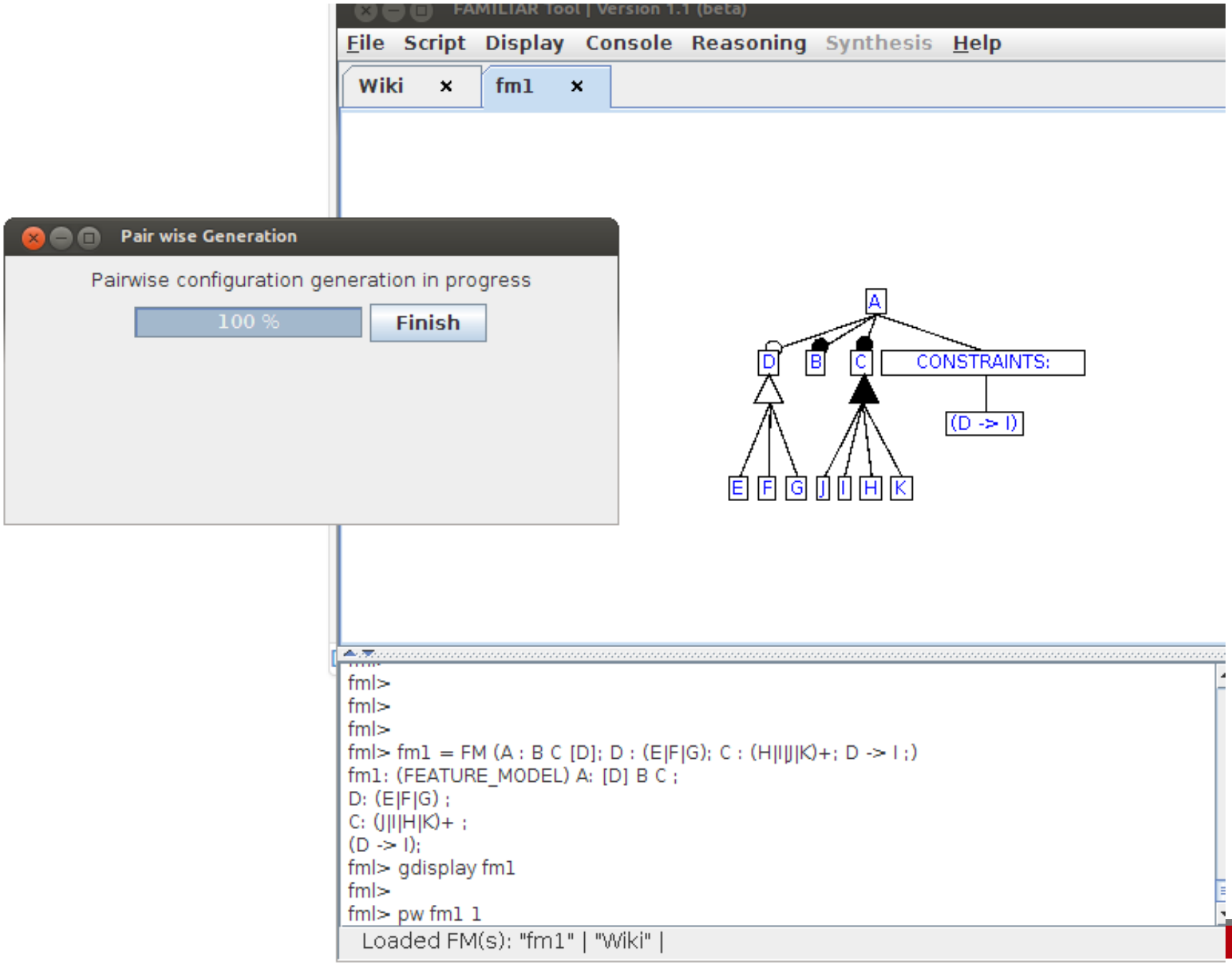
Vue globale

Des outils pour gérer la variabilité

- Familiar : outil open source pour gérer la variabilité et créer des modèles de features.
  - http://familiar-project.github.io/
- Pacogen : outil pour sélectionner les configurations de test.
- Pacogen est intégré à Familiar.



Des outils pour gérer la variabilité



Des outils pour gérer la variabilité

Configuration de test du modèle de feature fm1

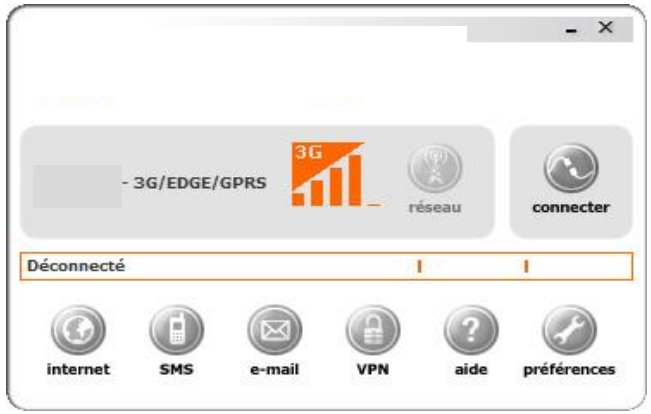
A	B	C	D	J	I	H	K	E	F	G
1	1	1	1	1	1	1	1	0	1	0
1	1	1	1	1	1	1	1	0	0	1
1	1	1	1	1	1	1	1	1	0	0
1	1	1	0	0	0	0	1	0	0	0
1	1	1	0	0	1	1	0	0	0	0
1	1	1	1	0	1	0	0	0	0	1
1	1	1	1	0	1	0	0	0	1	0
1	1	1	1	0	1	0	0	1	0	0
1	1	1	0	1	0	1	0	0	0	0
1	1	1	0	1	0	0	0	0	0	0

Download CSV File

Application dans le cadre du projet KIT

Rappel : le projet KIT

- Kit pour les Professionnels en mobilité



Comment tester **1461** exigences sur **2.560.000** environnements ?

Une approche exhaustive :

1461 * 2.560.000 = $3.7*10^9$ TC à exécuter pour vérifier chaque exigence dans chacune des configurations

Quelques chiffres

- **5** années de test
- **7** testeurs
- Plusieurs versions de **KIT** validées:
  - Pour une version :
    - de 100 à 400 homme-jour
    - En moyenne **300** homme-jour
- Beaucoup de fonctionnalités:
  - **1461** exigences

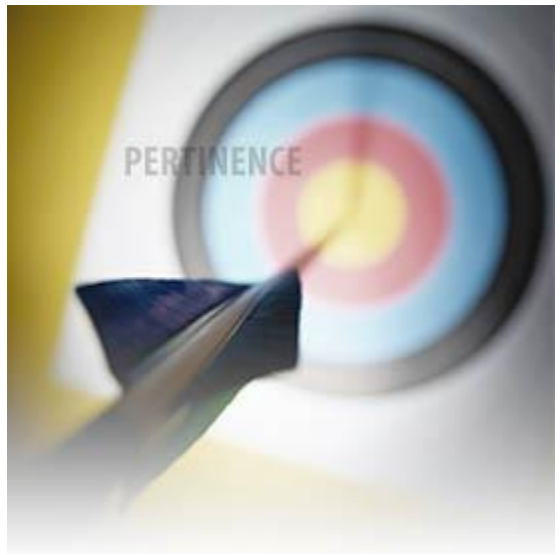
Approche de test originale

- Configuration de test partielles :
  - En très grande majorité les configurations de test ne considéraient qu'un système d'exploitation et une autre dimension de l'environnement :
    - OS et navigateur
    - OS and clé 3g...
  - 390 environnements dont 149 redondants du point de vue Pairwise
- Pour chaque configuration partielle un ensemble de cas de test étaient exécutés

Avantages	Inconvénients
Façon pratique de tester l'application	**Manque de diversité**
Bon fonctionnement	**Configurations incomplètes**

Approche de test pairwise

- Création du modèle de feature
  - Réalisé à l'aide de la documentation disponible
  - Discussion avec les testeurs du projet
  - Formalisation des connaissances :
    - Incompatibilité entre win 2000 et ZTE Rhos
      -> invalide plus de 200 000 environnements de test.
  - 20 heures pour réaliser le modèle.
  - 75 features
- Configurations de test extraites :
  - 254 configurations de test



Comparaison des approches

	Approche originale	Approche Pairwise
Nombre de configurations de test	390	254
Pourcentage de paires couvertes	4 %	100 %
Configurations	Configurations partielles	Configurations complète
Processus	Manuel	Automatisé

COMPARAISON DES DEUX APPROCHES DE TEST DU PROJET KIT

CONCLUSION ET PERSPECTIVES

Conclusion:

- Les modèles de features permettent
  - de représenter les relations entre les éléments qui composent l'environnement
  - d'identifier les configurations invalides
    - Dépendances
    - Exclusions

- Le test PairWise permet de sélectionner les configurations de test pour
  - couvrir les exigences
  - conserver un nombre raisonnable de cas de test à exécuter

Perspectives: la gestion des attributs

- Evolutions des outils :
  - Quantifier le coût d'une configuration,
  - Son poids,
  - Sa probabilité d'erreur,
  - …

Merci,
Questions ?

Publications

- Journal
  - **Optimal Minimization of Pairwise-covering est Configurations Using Constraint Programming** (in revision), TOSEM, Aymeric Hervieu, Dusica Marijan, Arnaud Gotlieb, and Benoit Baudry
- International conferences
  - **Pracical pairwise testing for software product lines.** Dusica Marijan, Arnaud Gotlieb, Sagar Sen and Aymeric Hervieu in SPLC 2013
  - **Managing Execution Environment Variability during Software Testing: An Industrial Experience.** Aymeric Hervieu, Benoit Baudry and Arnaud Gotlieb in ICTSS 2012
  - **PACOGEN: Automatic Generation of Pairwise Test Configurations from Feature Models**. Aymeric Hervieu, Benoit Baudry and Arnaud Gotlieb in ISSRE 2011
- Workshop
  - **Minimum Pairwise Coverage Using Constraint Programming Techniques.** Arnaud Gotlieb, Aymeric Hervieu and Benoit Baudry in ICST 2012