

Principes, techniques et outils de test

**Outils de test  
et d'analyse logiciels  
pour la productivité et la qualité**

IUT Lyon, 25 mai 2009  
Klaus LAMBERTZ (Verifysoft)

Verifysoft et ses partenaires

Coût de l'erreur logiciel

Les outils de test dans le cycle de développement

Complexité du code / Analyse statique

Test unitaire et couverture de test

Test fonctionnel

Questions / Discussion

## Klaus LAMBERTZ

Co-fondateur et dirigeant

Verifysoft Technology GmbH



Né en 1962 à Cologne (Allemagne)

Etudes en économie (Marketing, Commerce international) à  
Fachhochschule Cologne et ISG Paris

A travaillé dans une banque, responsable export: pour des entreprises  
Allemandes et Françaises (PKL, Partenaires-Livres, Maury Imprimeur)

Depuis 1999 travaille dans le domaine du test logiciel

Directeur des ventes Allemagne pour Parasoft Inc.

2003 Création de Verifysoft Technology GmbH, Offenburg (Allemagne)

## Verifysoft Technology GmbH

Créée en 2003 – par des investisseurs privés

Offenburg (Allemagne)

150+ clients en Europe

Distributeur et développeur des outils de test logiciel  
support, conseils et séminaires



**Testwell**

Testwell, Tampere (Finlande)

**CONFORMIQ**

Conformiq, Saratoga (USA)

 coverity

Coverity, San Francisco (USA)

# Testwell (Finlande)

The logo for Testwell, featuring the word "Testwell" in a bold, blue, sans-serif font on a white rectangular background.

Fondée en 1992 afin de continuer le développement des outils de test initialement développés au sein du groupe Nokia

Tampere (Finlande)

„Spinout“ of Nokia group

Premier outils de test:

1984 Test unitaire et de couverture pour ADA

1989 Couverture de code pour C/C++

Couverture de test, mesure de complexité de code, test unitaire



Fondée en 1998

Espoo (Finlande)

Siège social: Saratoga (USA)

Spécialiste „model driven testing Tools“

Générateur automatique de cas de test:  
Conformiq Qtronic



fondée en 2002

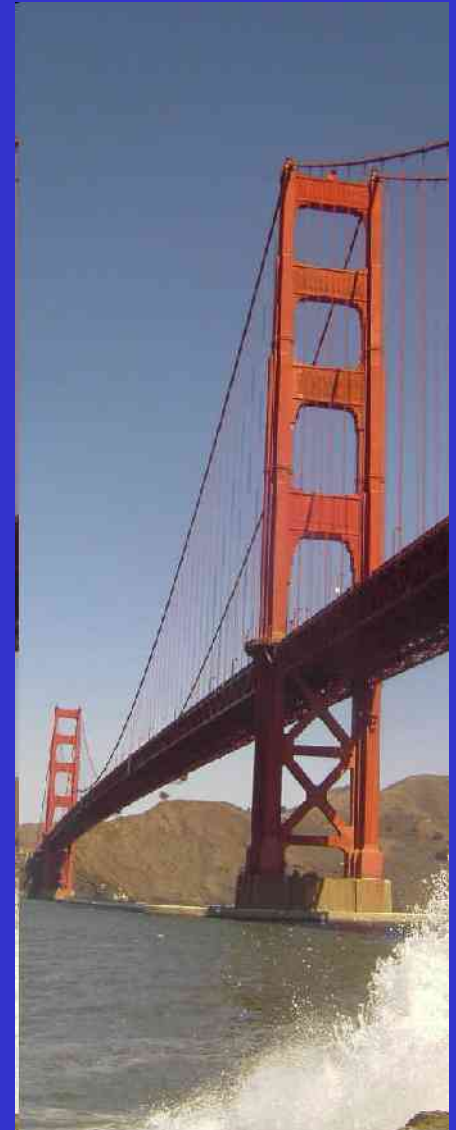
San Francisco (USA)

par des chercheurs scientifiques

de l'Université de Stanford

Coverity Prevent

(analyse statique de code)



Séminaires sur le test logiciel :  
par ex. Test de systèmes embarqués  
Certification de logiciels

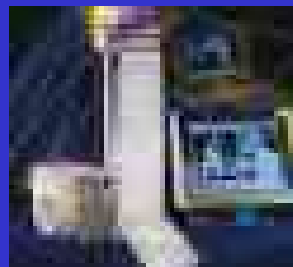
...



Domaines principaux d'application :

Développement des applications „critiques“

- Aéronautique
- Automotive
- Médical



# Nos références



et beaucoup d'autres ...

L'erreur logicielle

coûte jusqu'à 14.000.000 Euro

par entreprise et par an

50% des sociétés trouvent dans la première année d'utilisation jusqu'à dix erreurs critiques

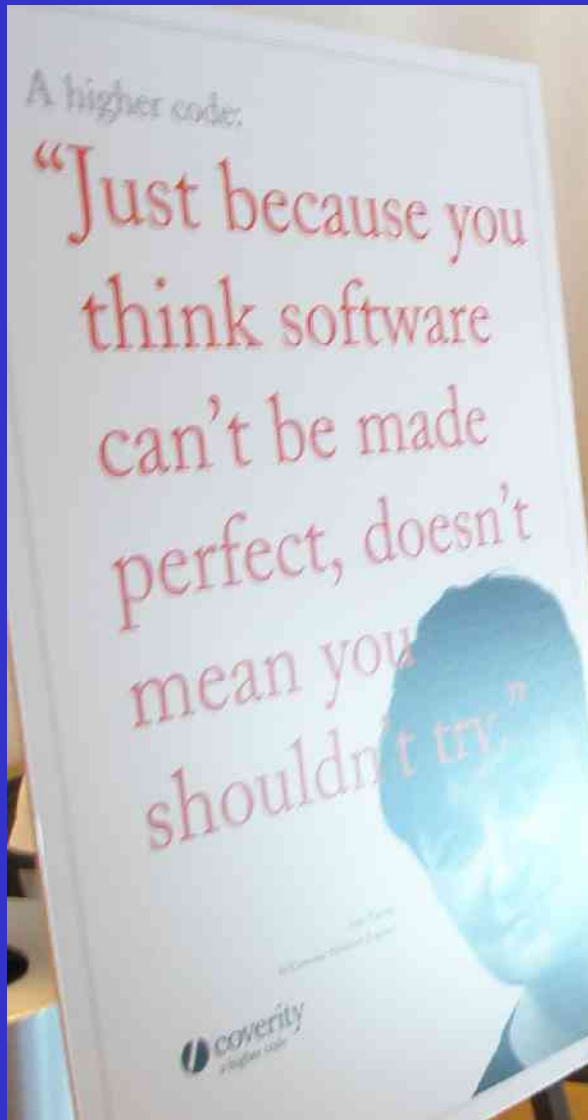
IDC-Studie „Improving Software Quality to Drive business Agility“,  
sondage 2008 sur des sociétés américaines comptant 250-10.000 employés [www.idc.com](http://www.idc.com)

Les pertes liées aux erreurs de programmation augmentent chaque année

Estimation: 100-150.000.000.000 Euros/an en Europe

-> Pour y pallier, le meilleur moyen est:  
le „savoir-faire“ des développeurs !!!

# Coût de l'erreur logicielle



40-60% du budget global est généralement consacré au test et à la correction.

Le but du test est d'arriver à un produit „zéro défaut“

Le test est nécessaire  
pour obtenir des certifications

IEC 61508

EN 50128 (ferroviaire)

IEC 62304 (médical)

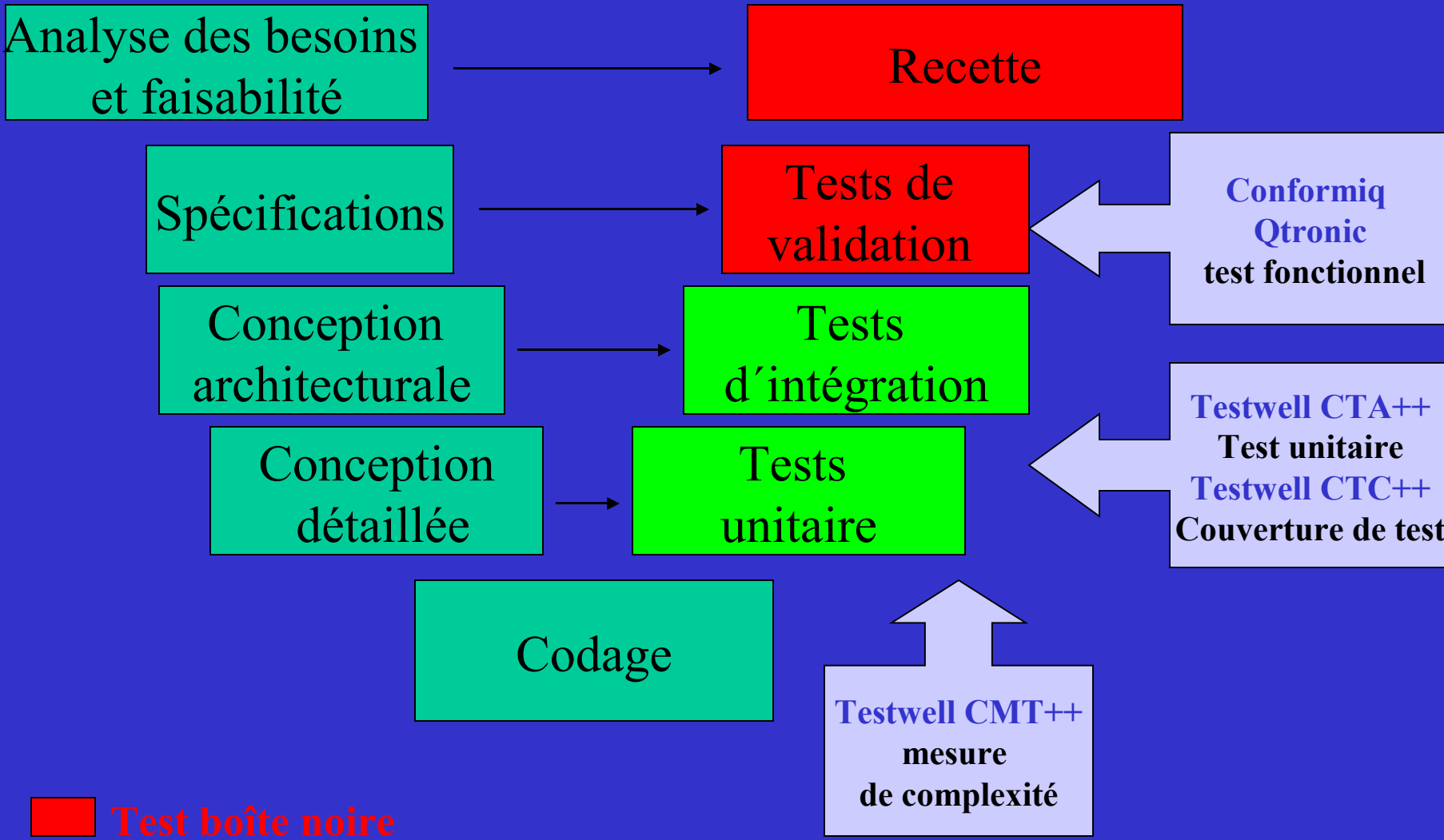
EN 62138 (nucléaire)

DO-178B (aéronautique)

Le test: indispensable  
mais ... très coûteux  
activité répétitive

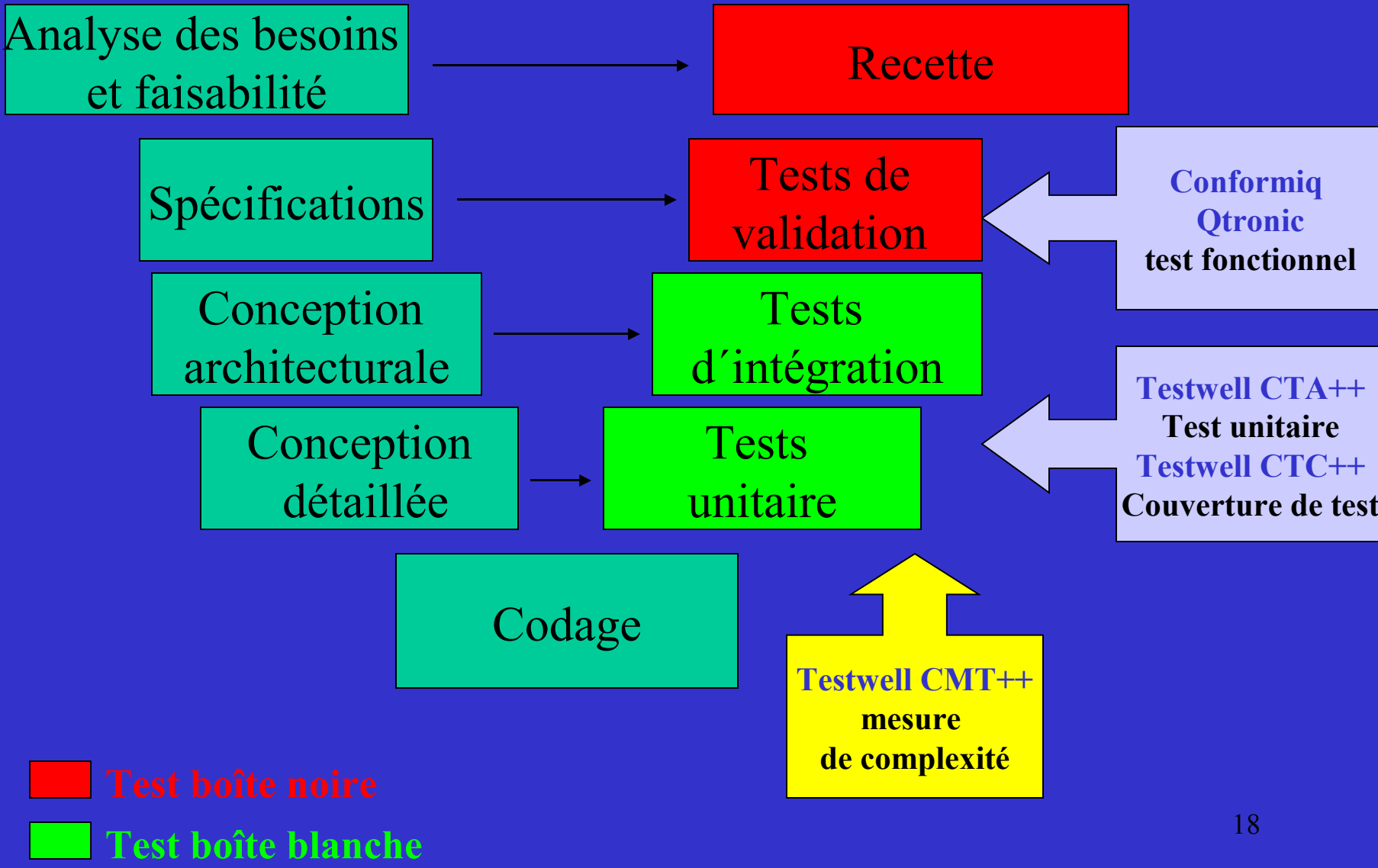
-> Automatisation du test  
utilisation d'outils de test !

# Cycle de développement



**Test boîte noire**  
**Test boîte blanche**

# Cycle de développement

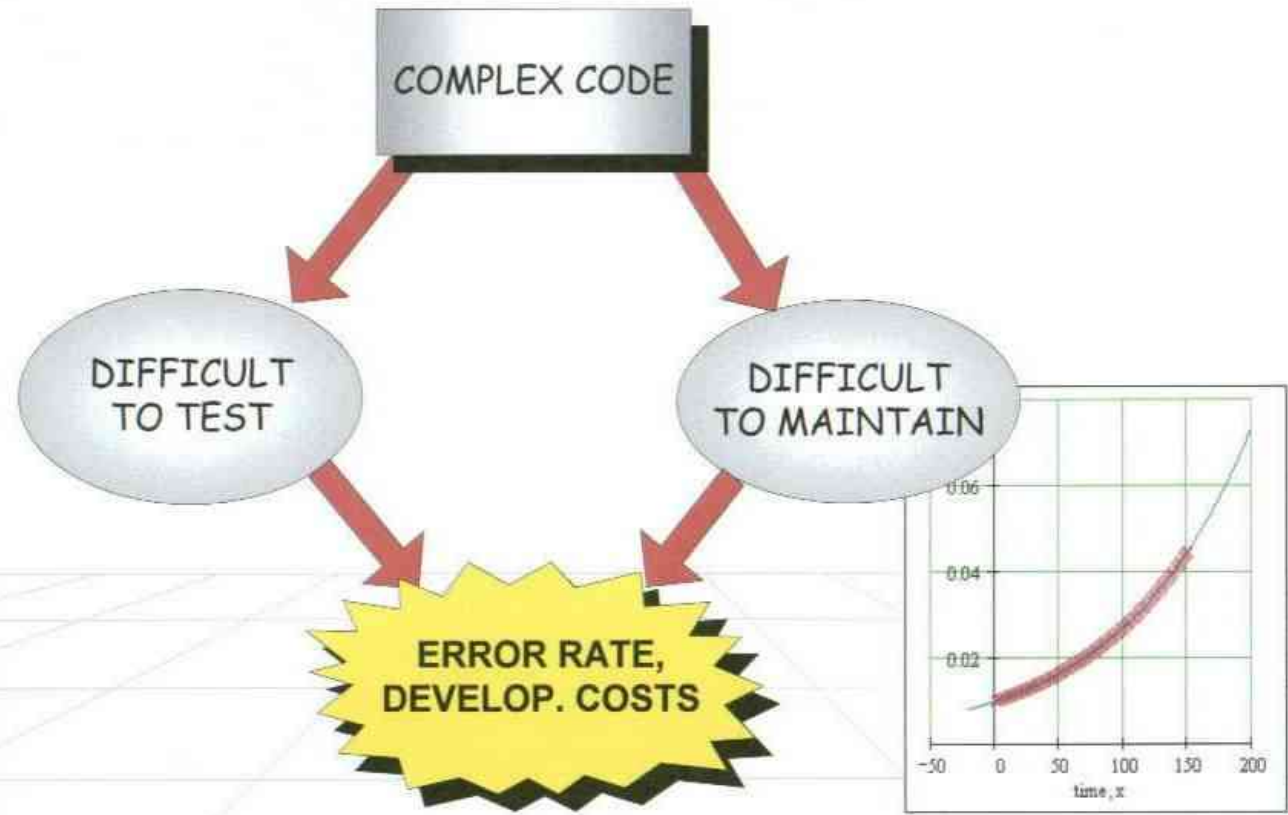


## Testwell CMT++ Testwell CMTJava



## Mesure de complexité du code pour C/C++ / Java

Wish to locate complex code



## Pourquoi analyser la complexité du code?

La complexité du code est en corrélation avec le taux d'erreur  
et la robustesse de l'application

- Un code complexe est difficile à tester
  - > plus d'erreurs dans l'application finale
- Un code complexe est difficile à maintenir
- La complexité du code est souvent la raison d'erreur
- Testwell CMT++ et CMTJava permettent donc de faire des économies.

## Testwell CMT++ et CMTJava

analysent les métriques suivantes:

- \* Métriques de lignes de code (LOC)
- \* Métriques d'Halstead
- \* Nombre cyclomatique de McCabe
- \* Indice de maintenabilité

## Métriques des lignes de code (LOC)

- **LOCphy**

nombre de lignes (number of physical lines)

- **LOCpro**

nombre de  
lignes avec du code programme

- **LOCbl**

nombre de lignes vides

- **LOCcom**

nombre de lignes avec commentaires

## Nombre cyclomatique de Mc Cabe $v(G)$

Le nombre cyclomatique  $v(G)$

décrit la complexité d'écoulement

(Control flow complexity)

d'un programme

## Halstead-Metrics

- B nombre d'erreurs estimé
- D niveau de difficulté, prédisposition d'erreurs
- E effort pour implémenter
- L niveau du programme (représente le niveau du programme)
- N longueur du logiciel
- N1 nombre d'opérateurs
- N2 nombre d'opérandes
- n taille de vocabulaire ou nombre d'opérateurs uniques et d'opérandes
- n1 nombre d'opérateurs uniques
- n2 nombre d'opérandes uniques
- T temps nécessaire pour l'implémentation (temps nécessaire pour comprendre)
- V volume: taille de l'implémentation d'un algorithme

## Index de maintenabilité (MI)

Indique quand il est moins coûteux et risqué  
de re-écrire le code

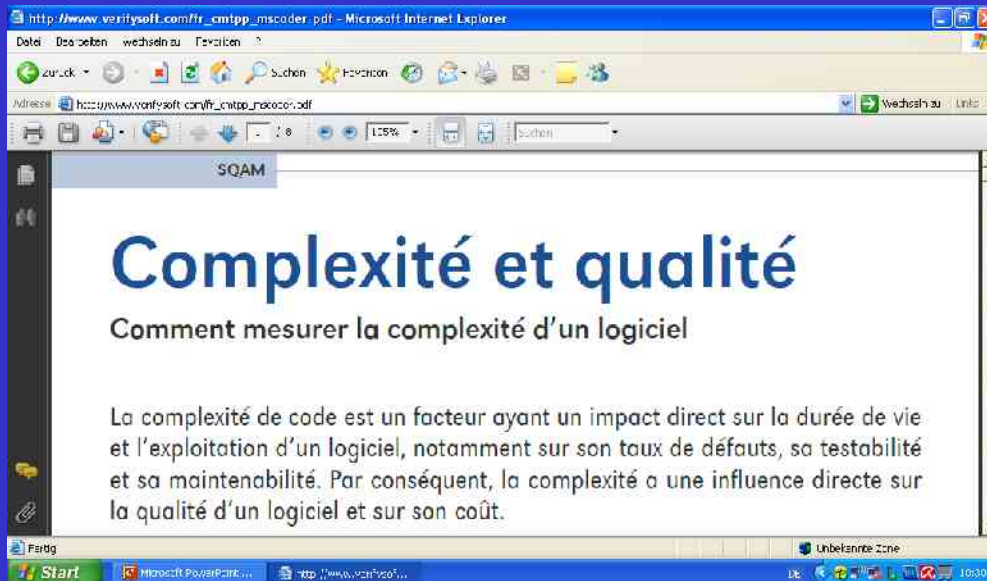
au lieu de garder des parties

complexes du code

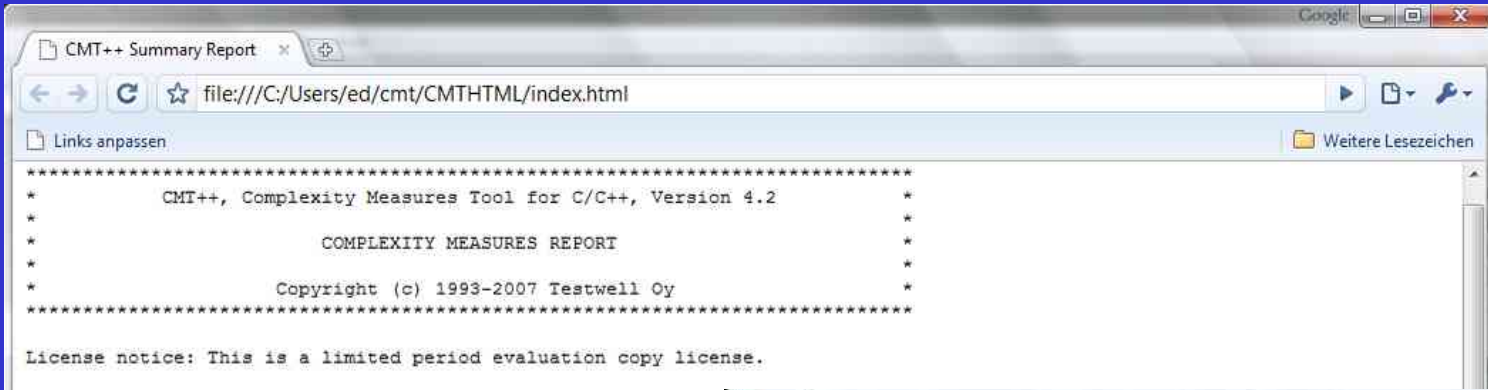
## Pour plus d'information



[www.verifysoft.com/fr\\_cmtx.html](http://www.verifysoft.com/fr_cmtx.html)



[www.verifysoft.com/  
fr\\_cmtpp\\_mscoder.pdf](http://www.verifysoft.com/fr_cmtpp_mscoder.pdf)



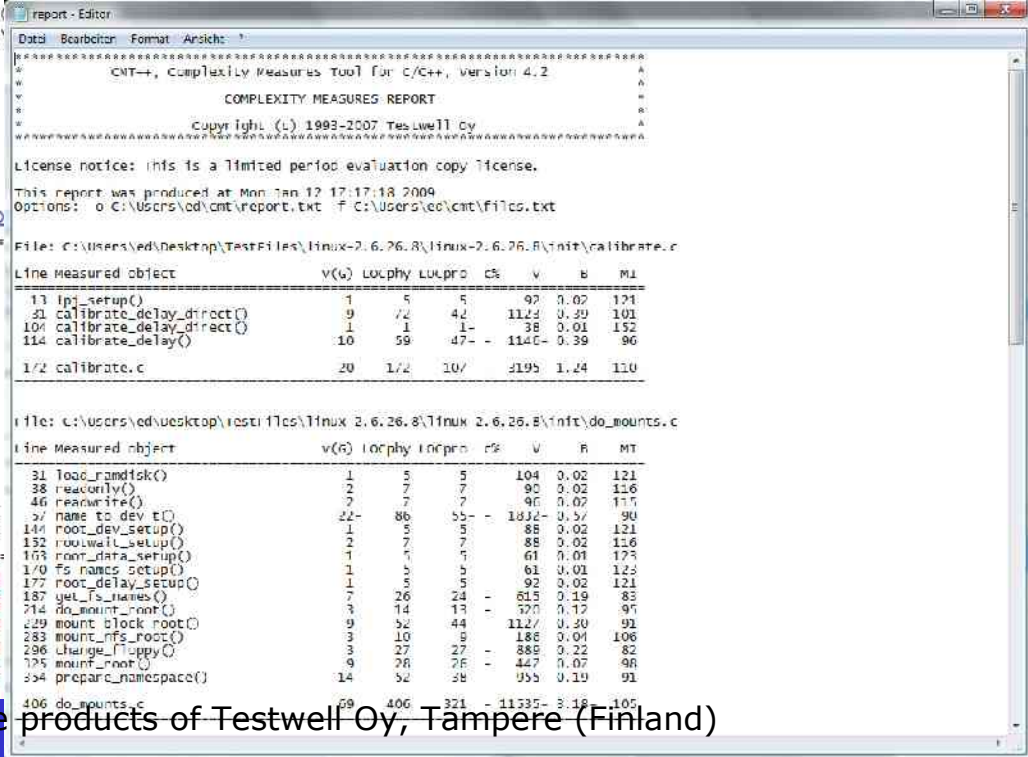
This is SUMMARY view. Go to [DETAILED](#) view. See [instructions](#).

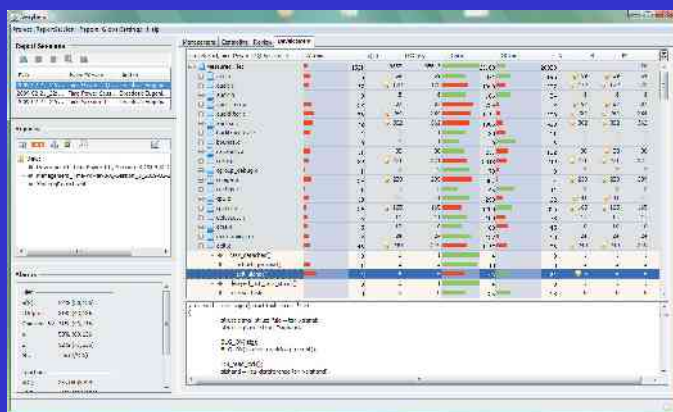
Alarms-%	Measured object	v(G)	LOCphy	LOCpro
	calibrate.c	20	172	
	do_mounts.c	69	406	
	do_mounts_md.c	47	280	
	do_mounts_rd.c	46	429	
	msgutil.c	15	127	
	OVERALL (24 %)			

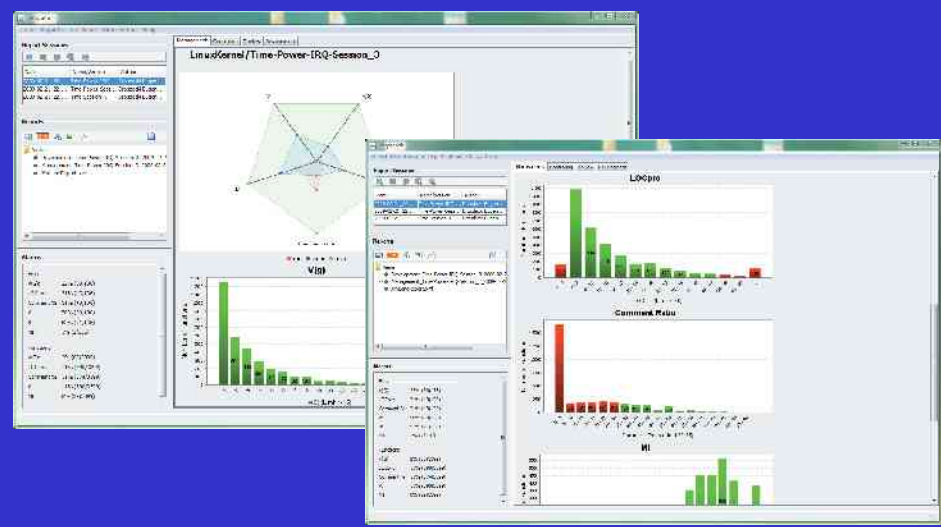
OVERALL SUMMARY:

Measure	5 Files		Limits	Alarmed
	Alarmed	%		
Cyclomatic number v(G)	0	0	1-100	4
Program lines LOCpro	0	0	4-400	12
Comment %	5	100	30-75	19
Volume V	3	60	100-8000	9
Estimated number of bugs B	3	60	0-2	0
Maintainability index MT	0	0	65-	1

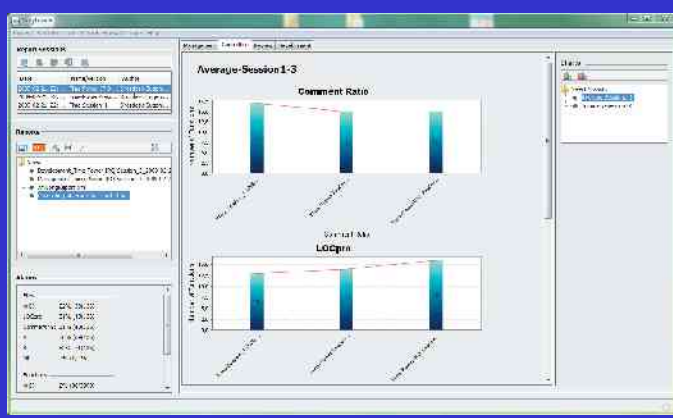




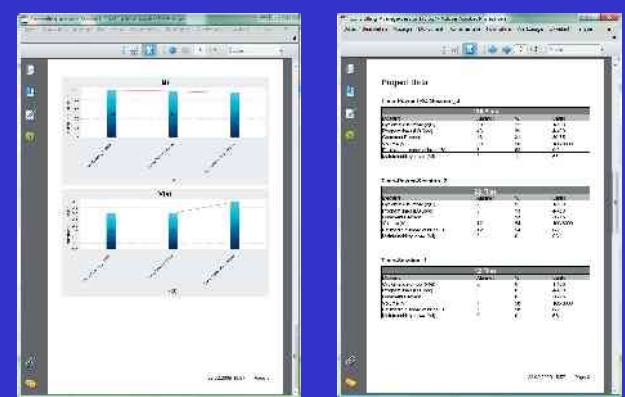
Point de vue  
„Développeur“



Point de vue „Manager“



Point de vue „Audit“



**Tous ses rapports sont disponibles en pdf**

Testwell CMT++ /

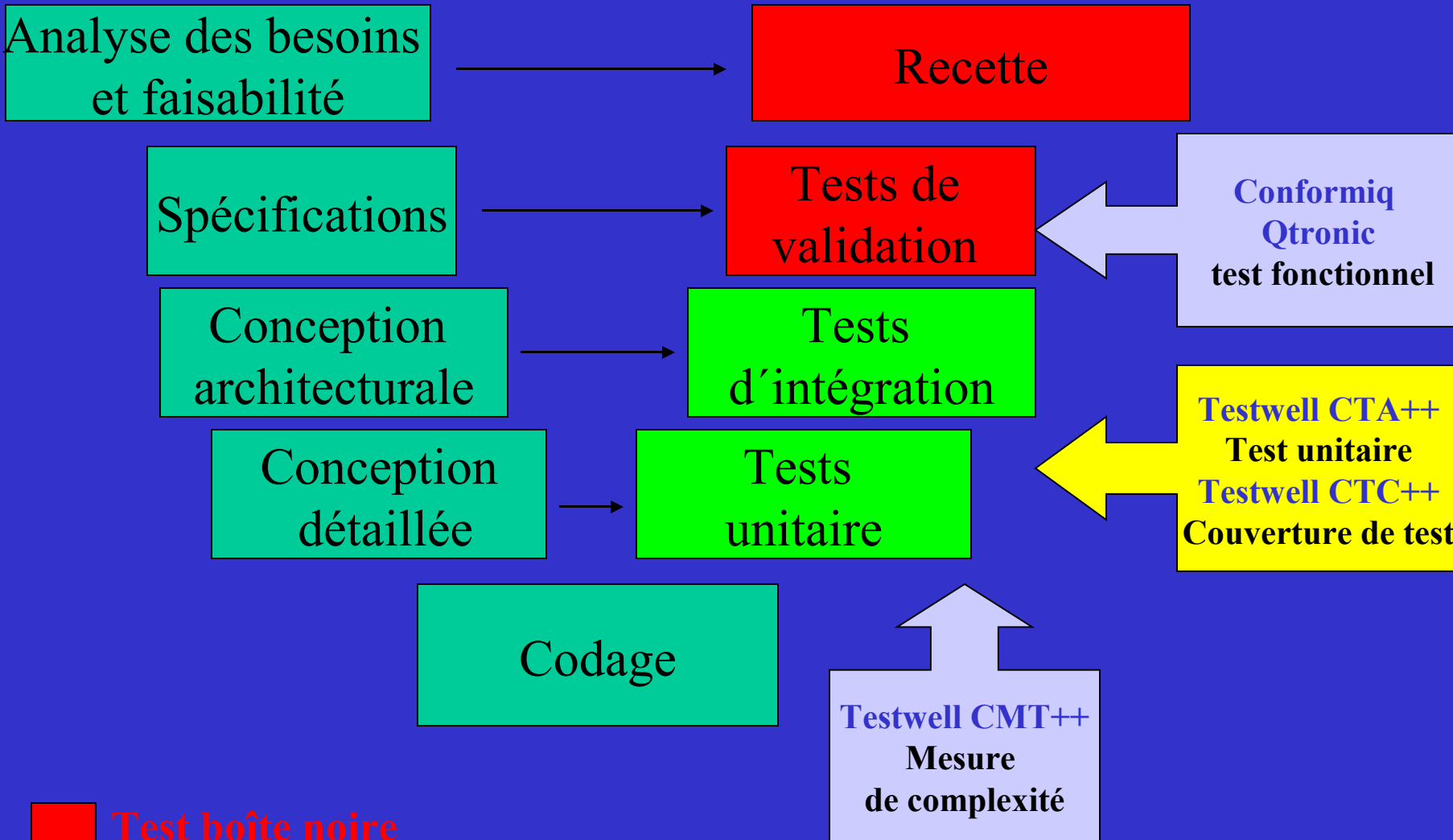
Testwell CMTJava

Sont inclus dans

Notre programme académique

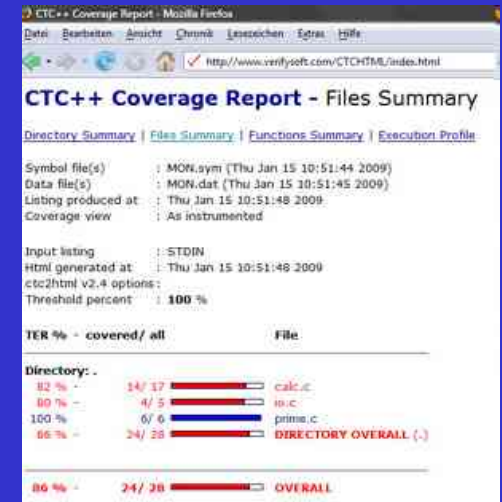


# Cycle de développement



**Red box** Test boîte noire  
**Green box** Test boîte blanche

## Testwell CTC++ Couverture de test pour C and C++



## CTC add-on pour Java et C#

## Testwell CTC++ montre la couverture de test

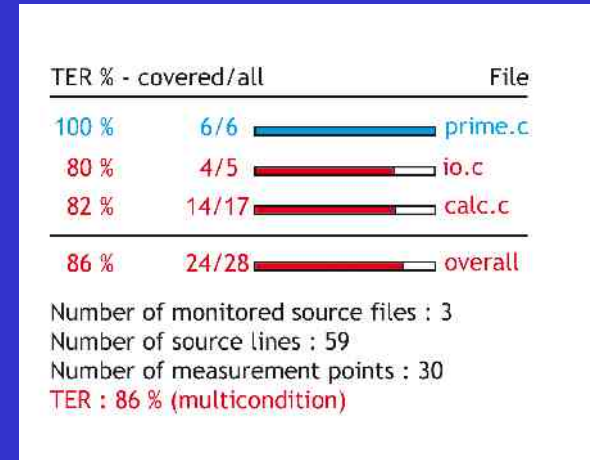
pour:

chaque fonction

chaque fichier source

des parties choisies de l'application

le projet complet



## Bénéfices :

### Testwell CTC++ montre les parties du code testées/non testées

- Vous pourrez écrire de meilleurs tests/cas de tests (plus adaptés)
- Vous évitez de passer du temps à écrire des cas de tests redondants
- Vous savez quand vous pouvez arrêter de tester
- Vous pouvez prouver à vos clients que les codes ont été testés  
conformément à leurs attentes
- En tant que clients, vous pouvez être certains que les codes délivrés par votre sous traitant sont conformes à vos attentes
- Obligation pour obtenir une certification



## Bénéfices:

Testwell CTC++

- aide à localiser du code mort
- montre le temps d'exécution pour chaque fonction
- trouve des goulots d'étranglement

Analyse de Test pour tous les niveaux (- C3):

- Couverture de fonction
- Couverture de décision
- Couverture de condition/de branche  
(Condition/Branche Coverage)
- Couverture de condition/décision modifiée  
(Modified Condition/Decision Coverage, MC/DC)
- Couverture de condition multiple  
(Multicondition Coverage, MCC)

Testwell CTC++ vérifie tous les niveaux du code et peut être utilisé pour des Certifications

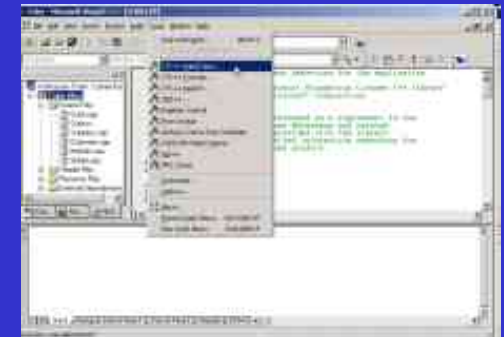
Aéronautique, Automotive, Médical,  
...

par ex. DO178-B (tous les niveaux)



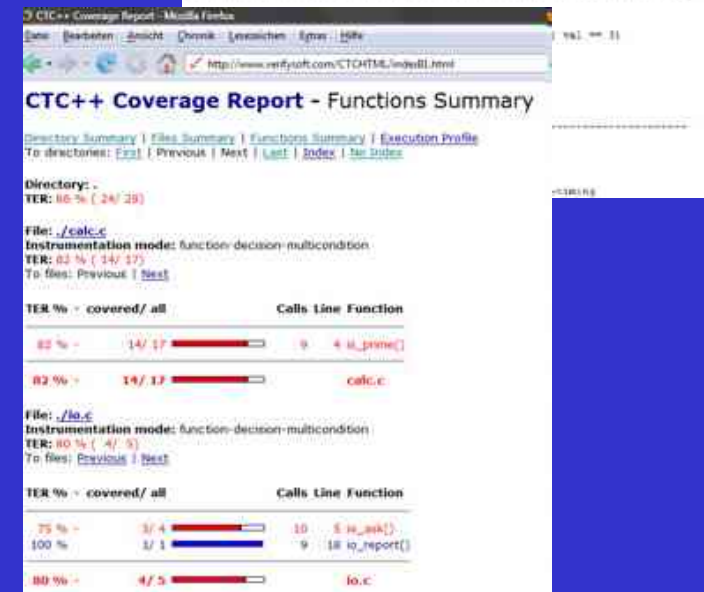
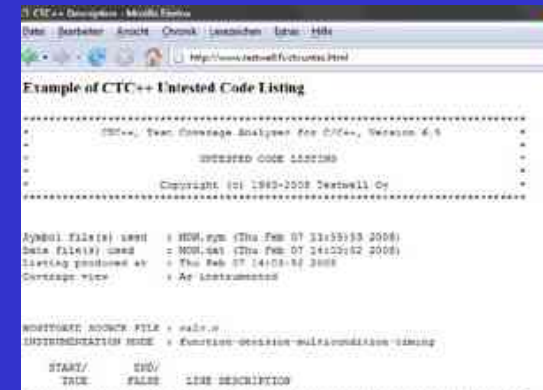
## Testwell CTC++ est simple à l'emploi:

- \* Aucune modification de code nécessaire
- \* Support existant makefiles
- \* GUI integration in important IDEs
  - Microsoft Visual Studio .NET
  - Microsoft Visual Studio 5.0/6.0
  - WindRiver Tornado
  - Borland C++ 5.02
  - Metrowerks CodeWarrior
  - Eclipse



## Sorties: text, XML ou HTML:

- montre les codes non testés  
(mise en évidence)
- montre combien de fois la partie  
du code à été testée
- différents rapports de couverture
  - summary-levels for files
  - functions and whole application
  - Execution Profile Listing



## Testwell CTC++ outil idéal pour les systèmes embarqués :

- \* très faible coût d`instrumentation
- \* fonctionne avec toutes les cibles / microcontrôleurs  
"host target add-on" est fourni en code source

-> peut être facilement adapté à de nouvelles cibles

- \* fonctionne avec les plus petites cibles
- \* fonctionne avec tous les compilateurs



## CTC++ Add-on for Java et C#

extension Testwell CTC++ pour Java et C#

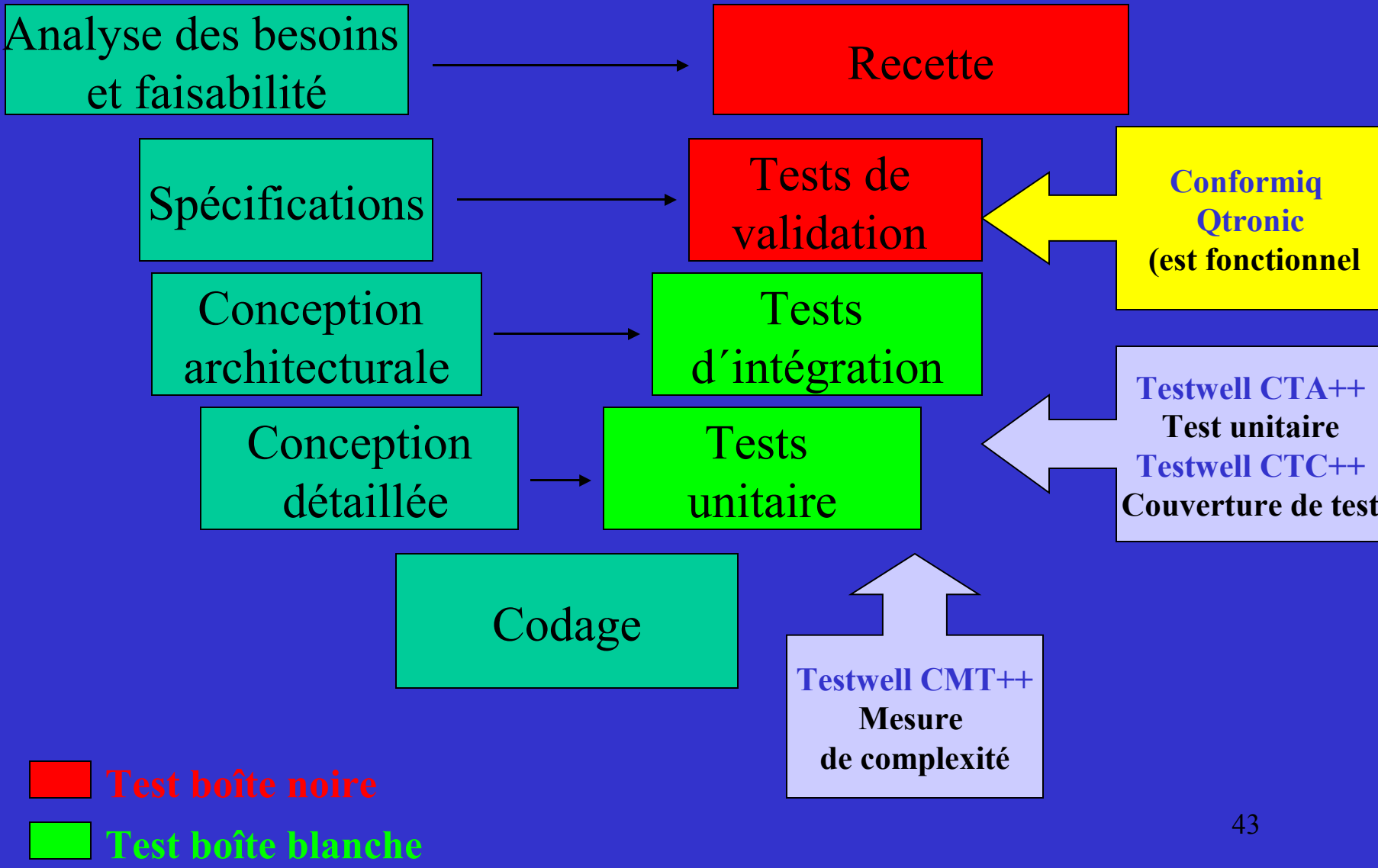
-> Vous avez seulement besoin d'un outil de couverture pour C, C++, Java, C#, ...



Testwell CTC++ et  
CTC++ Add-on pour Java et C#  
et Testwell CMT++ /  
Testwell CMTJava  
sont inclus dans notre  
programme académique



# Cycle de développement



# Conformiq Qtronic

Générateur automatique de cas de test  
pour les test fonctionnels

(tests de boîte noire)



Les cas de tests manuels prennent du **temps** ...

et entraînent des **risques**:

- 💣 tests incorrects
- 💣 tests oubliés

🕒 tests redondants

🕒 la maintenance pour les scripts prend du temps

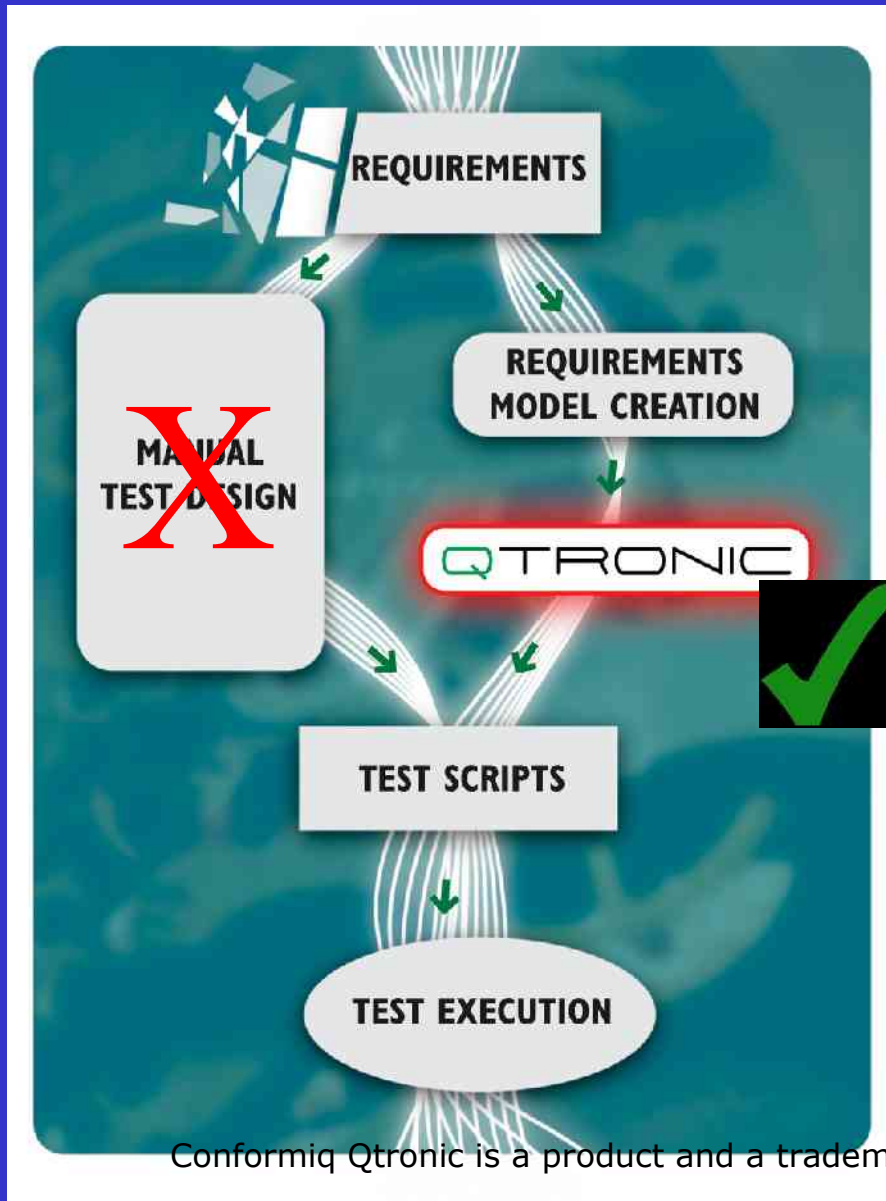


## Notre solution: Automated Test Design

- > model driven testing, model based testing,  
specification based testing,  
specification driven testing,  
...



# Manuel vs. Automatique

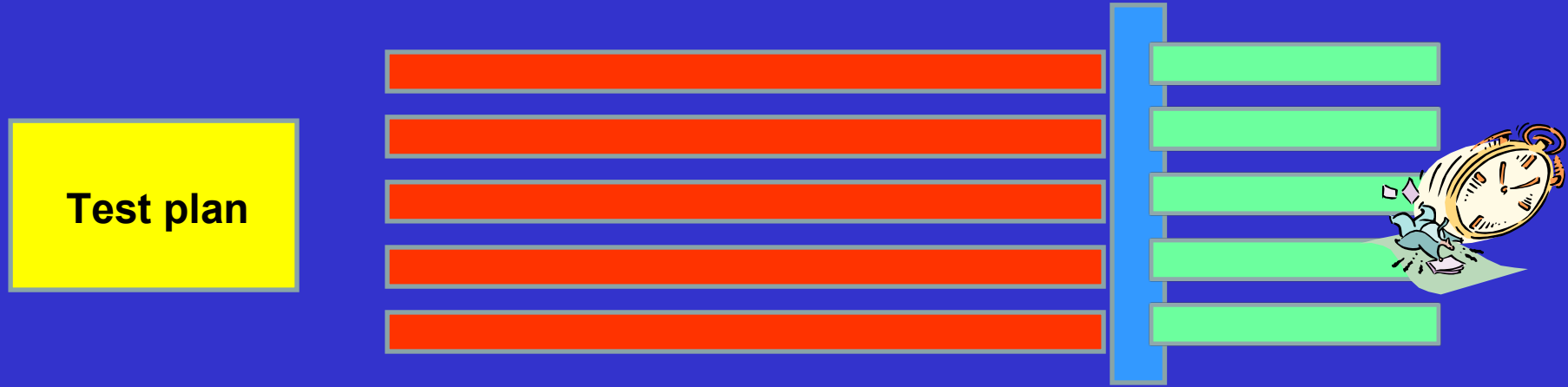


Génération automatique de cas de test basée sur des modèles  
au lieu

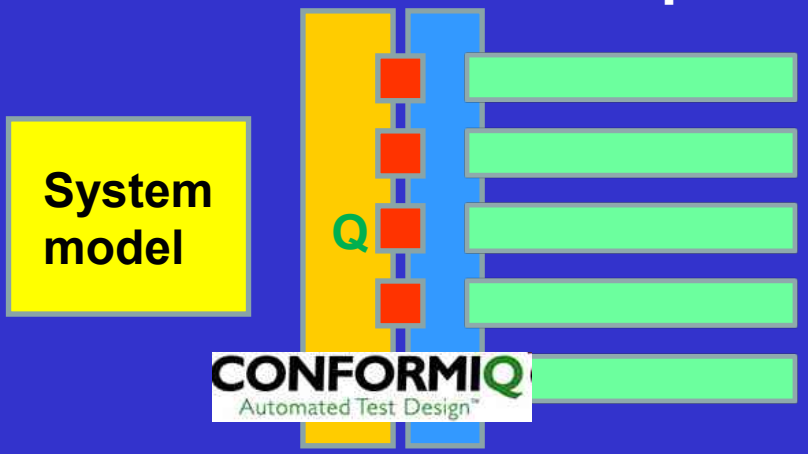
de perdre du temps à écrire les cas de tests manuellement

# Manuel vs. Automatique

## Cas de tests écrits manuellement



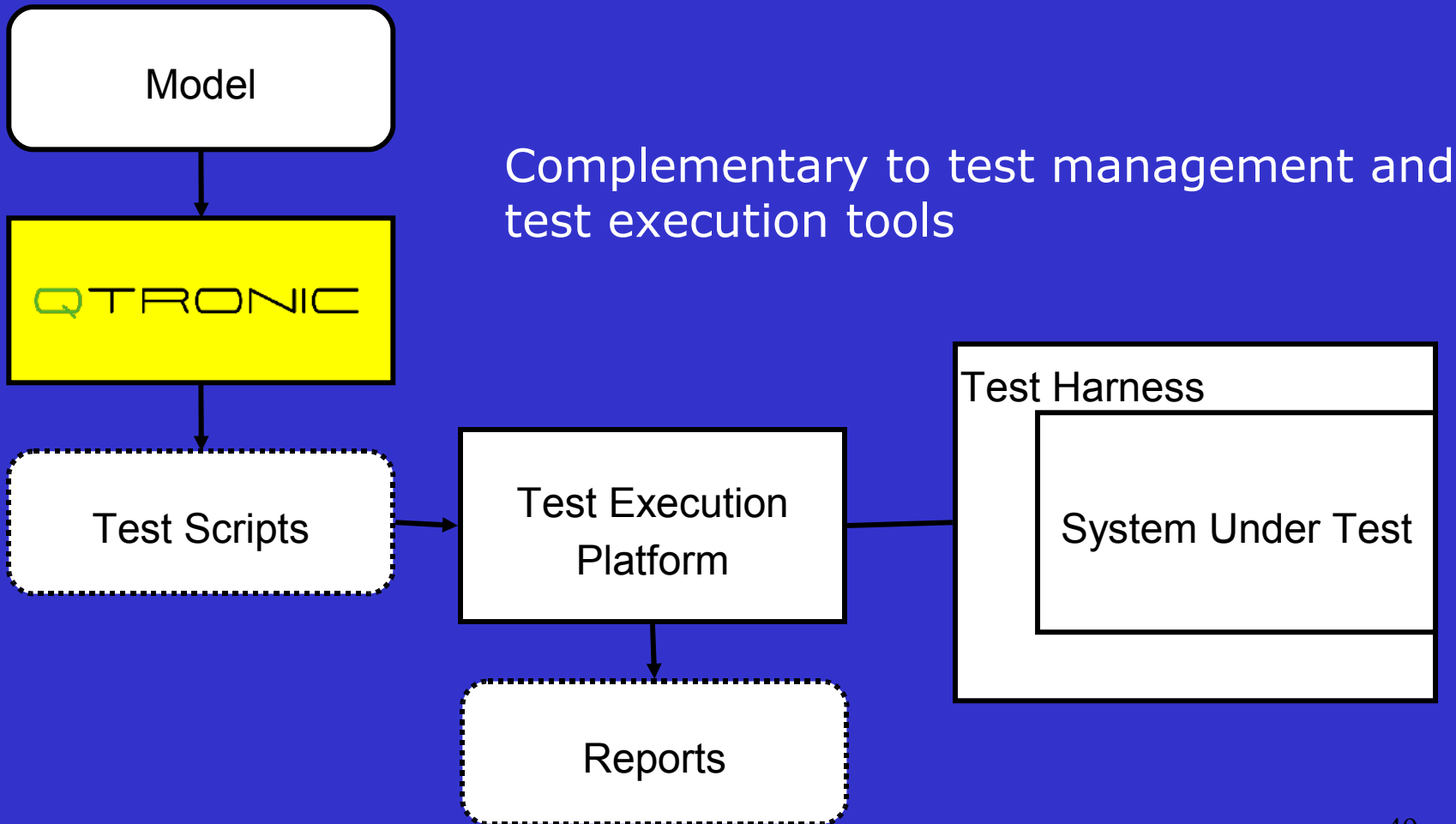
## Génération automatique



Exécution des scripts de tests

Exécution des scripts de tests

# Conformiq Qtronic







- textuellement en Java (avec des élément C#-)
  - > „Qtronic Modelling Language“ (QML)
- graphiquement: UML State Charts (optional)
  
- Le modèle peut être réalisé:
  - **text editor** („Java“)
  - **Qtronic Modeller** (UML State Charts)
  - **Third Party Modeling (UML) Tools**

## Comment générer des tests ?

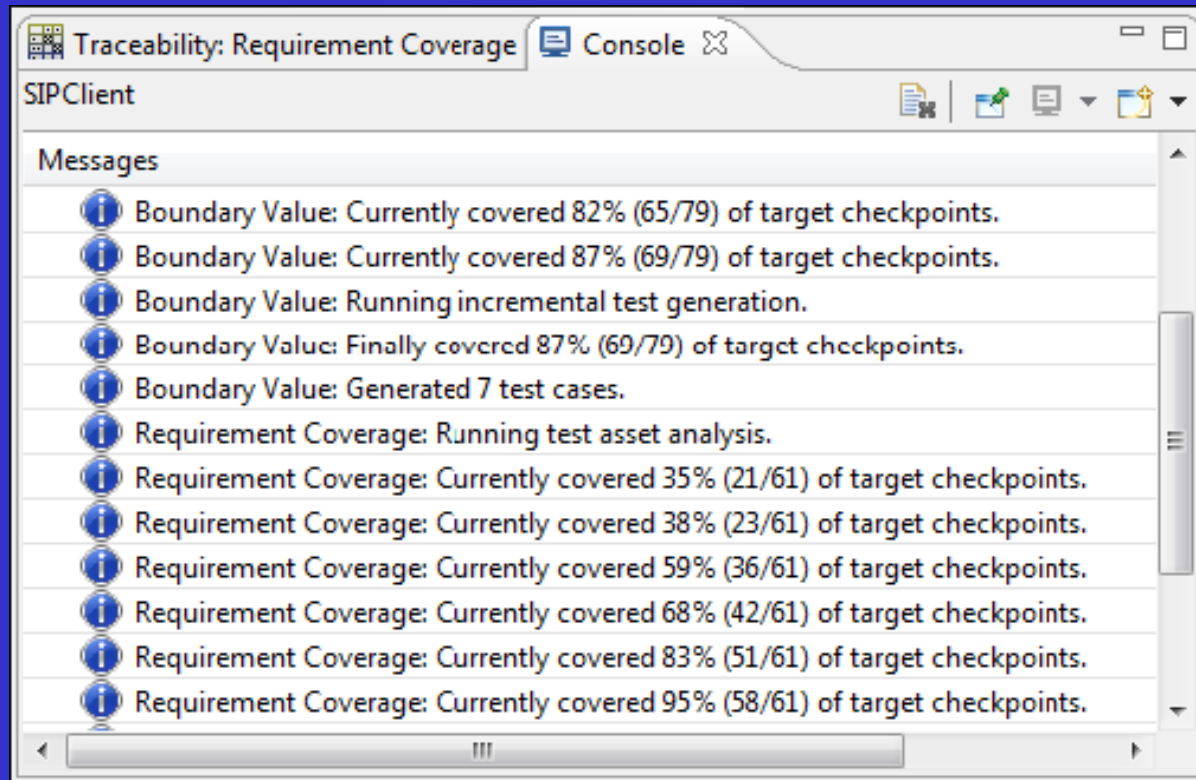
une fois que le modèle est importé vers Conformiq  
Qtronic

et les critères de couverture de test ont été choisis

il suffit de sélectionner

„Update Test Set“

dans Conformiq Qtronic



## Statut de la génération du test montré par Eclipse:

-Après génération des cas de tests, la couverture final sera montré (pour toutes et chaque exigeances et caractéristiques.

# Génération de cas de test

The screenshot displays the Qtronic Eclipse SDK interface with several key components:

- Test Cases: SIPClient**: A list of 7 test cases, with 'Test Case 4' selected. The list includes:
  - Test Case 1
  - Test Case 2
  - Test Case 3
  - Test Case 4
  - Test Case 5
  - Cancel Timeouted
  - Test Case 7
- Requirement Coverage**: A table showing 100% coverage for all listed requirements.
- Traceability: Requirement Coverage**: A table mapping testing goals to requirements.
 

Testing Goals	1	2	3	4	5	6
<b>Requirements</b>						
17.1.2.2 Non-INVITE timers						
Resends CANCEL after E timeout						X
Terminates CANCEL cycle after F timeout						X
Resends BYE after E timeout			X			
Terminates BYE cycle after F timeout			X			
<b>Terminating</b>						
Wait for OK in response to BYE			X			
Send OK in response to BYE	X					
17.1.1.2 INVITE timers						
- Steps: Test Case 4**: Shows the execution trace for the selected test case.
- Console**: Displays a series of messages including coverage statistics for Boundary Values and DC 2, and a final message: "DC 2: Generated 7 test cases."
- Sequence Diagram**: A partial diagram showing a SIP UAC component with messages:
  - UserInput -> userIn
  - SIPReq <- netOut
  - SIPReq <- netOut

Liste des cas de test

**Traceability Matrix**

Testing Goals	1	2	3	4	5	6
<b>Requirements</b>						
<b>17.1.2.2 Non-INVITE timers</b>						
Resends CANCEL after E timeout						X
Terminates CANCEL cycle after F timeout						X
Resends BYE after E timeout				X		
Terminates BYE cycle after F timeout				X		
<b>Terminating</b>						
Wait for OK in response to BYE						
Send OK in response to BYE						
<b>17.1.1.2 INVITE timers</b>						
Terminates INVITE cycle after B timeout						
Resends INVITE after A timeout						
Acknowledge established call with ACK		X	X	X		
<b>State Chart</b>						
<b>2-Transitions</b>						
<b>Transitions</b>						
<b>Implicit Consumption</b>						
<b>States</b>						
SIPClient-Init	X	X	X	X	X	X
SIPClient-Calling-initial-0	X	X	X	X	X	X
SIPClient-Calling-Wait	X	X	X	X	X	X
SIPClient-Calling-junction-2						
SIPClient-Ready		X	X	X		

**Tableau de Tracabilité :**  
Montre pour chaque cas de test ce que ça couvre

covered 48% (29/61) of target checkpoints,  
covered 50% (31/61) of target checkpoints,  
covered 72% (44/61) of target checkpoints,  
covered 83% (51/61) of target checkpoints,  
covered 90% (55/61) of target checkpoints,  
covered 100% (61/61) of target checkpoints.  
incremental test generation.  
coverage reached, stopping.  
covered 100% (61/61) of target checkpoints.  
17 test cases.

Test

# Génération de cas de test

The screenshot displays a software development environment with a sequence diagram for 'Test Case 4' and a coverage matrix. The sequence diagram shows interactions between a 'Tester' and 'SIP UAC' components. The messages are as follows:

- t=0.0: Tester sends 'UserInput -> userIn' to SIP UAC.
- t=0.0: SIP UAC sends 'SIPReq <- netOut' to Tester.
- t=0.0: Tester sends 'SIPResp -> netIn' to SIP UAC.
- t=0.0: SIP UAC sends 'SIPReq <- netOut' to Tester (highlighted with a dashed box).
- t=0.0: Tester sends 'UserInput -> userIn' to SIP UAC.
- t=0.0: SIP UAC sends 'SIPReq <- netOut' to Tester.
- t=0.5: SIP UAC sends 'SIPReq <- netOut' to Tester.

The coverage matrix on the right shows the relationship between test cases and requirements. The matrix is as follows:

	1	2	3	4	5	6
Requirement 1				X		X
Requirement 2				X		X
Requirement 3			X			
Requirement 4		X				
Requirement 5						
Requirement 6						
Requirement 7						
Requirement 8						
Requirement 9						
Requirement 10						
Requirement 11						
Requirement 12						
Requirement 13						
Requirement 14						
Requirement 15						
Requirement 16						
Requirement 17						
Requirement 18						
Requirement 19						
Requirement 20						
Requirement 21						
Requirement 22						
Requirement 23						
Requirement 24						
Requirement 25						
Requirement 26						
Requirement 27						
Requirement 28						
Requirement 29						
Requirement 30						
Requirement 31						
Requirement 32						
Requirement 33						
Requirement 34						
Requirement 35						
Requirement 36						
Requirement 37						
Requirement 38						
Requirement 39						
Requirement 40						
Requirement 41						
Requirement 42						
Requirement 43						
Requirement 44						
Requirement 45						
Requirement 46						
Requirement 47						
Requirement 48						
Requirement 49						
Requirement 50						
Requirement 51						
Requirement 52						
Requirement 53						
Requirement 54						
Requirement 55						
Requirement 56						
Requirement 57						
Requirement 58						
Requirement 59						
Requirement 60						
Requirement 61						
Requirement 62						
Requirement 63						
Requirement 64						
Requirement 65						
Requirement 66						
Requirement 67						
Requirement 68						
Requirement 69						
Requirement 70						
Requirement 71						
Requirement 72						
Requirement 73						
Requirement 74						
Requirement 75						
Requirement 76						
Requirement 77						
Requirement 78						
Requirement 79						
Requirement 80						
Requirement 81						
Requirement 82						
Requirement 83						
Requirement 84						
Requirement 85						
Requirement 86						
Requirement 87						
Requirement 88						
Requirement 89						
Requirement 90						
Requirement 91						
Requirement 92						
Requirement 93						
Requirement 94						
Requirement 95						
Requirement 96						
Requirement 97						
Requirement 98						
Requirement 99						
Requirement 100						

Test Case List

Message graphique: séquence pour un cas de test

Quality Matrix: for each test that it covers

fully covered 100% (61/61) of target checkpoints.  
generated 7 test cases.

# Génération de cas de test

The screenshot displays a test management interface with three main components:

- Test Case List:** A list of test cases for 'SIPClient', with 'Test Case 4' selected.
- Tester Timeline:** A vertical timeline for 'Tester' showing time points (t=0.0, t=0.5) and arrows indicating the sequence of test steps.
- Steps: Cancel Timeouted:** A detailed log of test steps with columns for Message / Field, Port / Field value, and Time.
 

Step	Message / Field	Port / Field value	Time
1	UserInput	to userIn	0.0
	input1	invite	
	input2	sip:127.0.0.1:5061	
2	SIPReq	from netOut	0.0
	op	INVITE	
	param	sip:127.0.0.1:5061	
3	SIPResp	to netIn	0.0
	status	180	
	cseq		
4	UserInput	to userIn	0.0
	input1	cancel	
	input2		
5	SIPReq	from netOut	0.0
	op	CANCEL	
	param	sip:127.0.0.1:5061	
6	SIPReq	from netOut	0.5
7	SIPReq	from netOut	1.5
8	SIPReq	from netOut	3.5
9	SIPReq	from netOut	7.5
- Matrix:** A grid showing test coverage for each step across six test cases.
 

	1	2	3	4	5	6
Step 1						
Step 2				X		X
Step 3				X		X
Step 4			X			
Step 5		X				
Step 6	X	X	X	X	X	X
Step 7	X	X	X	X	X	X
Step 8	X	X	X	X	X	X
Step 9	X	X	X	X	X	X

Test Case List

Messa  
for a t

Vue des étapes du test avec des informations plus détaillées sur le message

Matrix:  
each test  
it covers



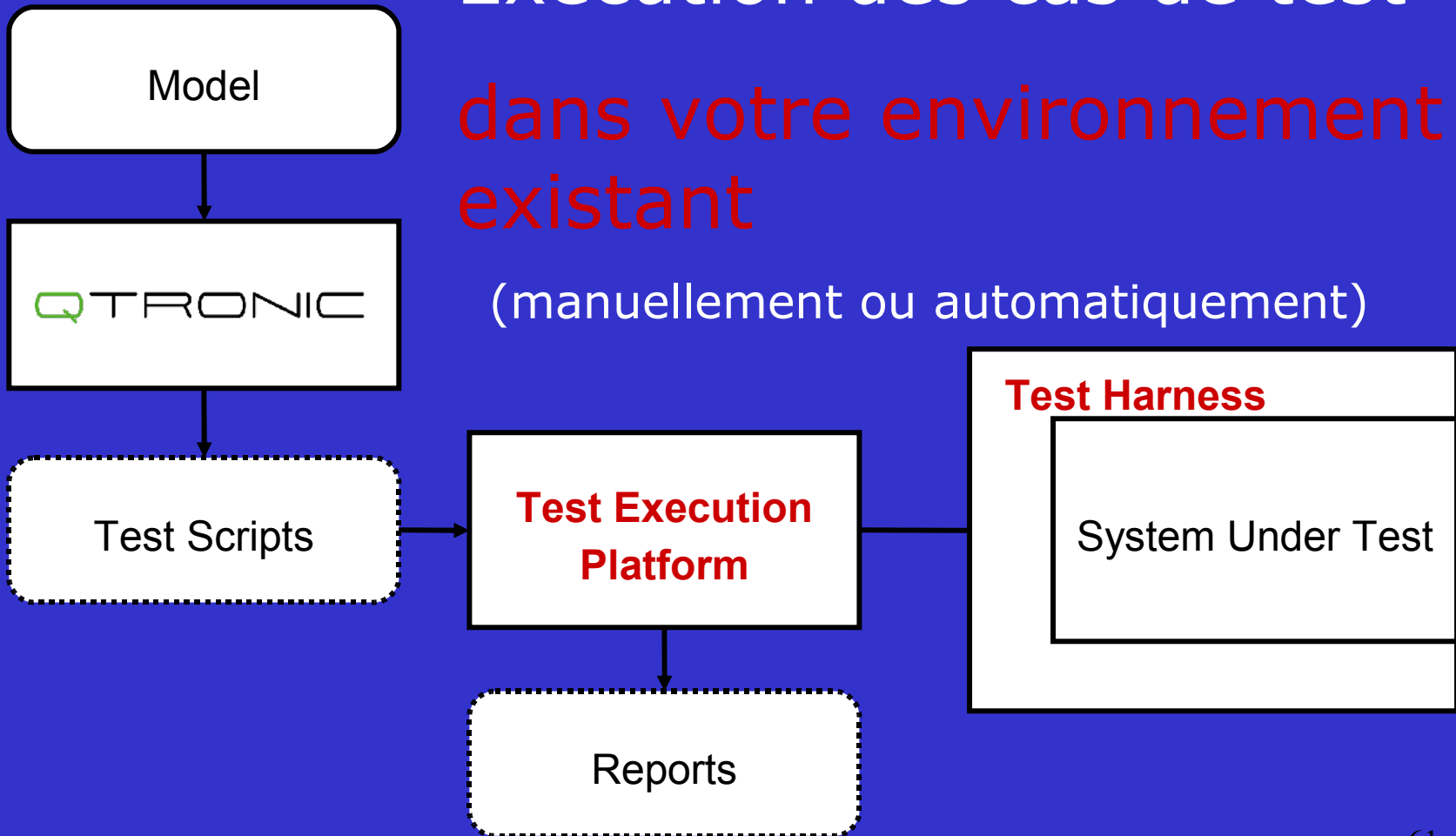
Export de scripts de test:

Trois scripteurs par défaut sont fournis avec Conformiq Qtronic:

- HTML TTCN-3 TCL
- d'autres formats peuvent être écrits

## Exécution des cas de test dans votre environnement existant

(manuellement ou automatiquement)



conditions spéciales  
pour Conformiq Qtronic

dans le cadre  
de notre  
programme académique



## Bibliothèque de test en ligne

depuis début mai 2009

vous trouverez sur notre site Internet des publications portant sur les thèmes de la qualité logiciel mais également le test logiciel.



4 documents

en Français et Anglais

au 20 mai 2009



9 documents

en Allemand et Anglais

au 20 mai 2009



Vous pourrez  
télécharger ses  
documents



mais  
également  
y soumettre  
vos propres  
travaux

[www.verifysoft.com/  
papers.html](http://www.verifysoft.com/papers.html)



## Resumé



- Evaluation gratuite de nos outils avec accès à notre support:
  - Testwell CMT++/CMTJava
  - Testwell CTC++/CTC pour Java et C#
  - Testwell CTA++
  - Conformiq Qtronic
- Accès à nos tarifs très préférentiels
- Bibliothèque en ligne :

Accès et participation

- Stagiaires

Verifysoft Technology GmbH  
Technologiepark - In der Spoeck 10  
77656 OFFENBOURG (Allemagne)

Contact: Klaus LAMBERTZ

Tél. FRANCE: (+33) 03 68 33 58 84  
[france@verifysoft.com](mailto:france@verifysoft.com)

[www.verifysoft.com](http://www.verifysoft.com)

Merci !