Mesurer la couverture du code :

10 critères pour sélectionner un outil de couverture de code

Afin de développer des logiciels sûrs et fiables, les tests sont une partie indispensable de l'assurance qualité. Sans tests suffisants et documentés, il est impossible de déterminer si un logiciel est sûr et fonctionnellement correct. La mesure de la couverture de code (couverture des tests) est particulièrement importante dans ce contexte. En effet, elle peut être utilisée pour déterminer dans quelle mesure un logiciel a déjà été testé. La couverture de code indique le rapport entre le code testé et le code total. En termes simplifiés, par exemple, la couverture de code est de 75 %, si trois des quatre options possibles sont exécutées pendant le test.

En particulier dans le développement de logiciels critiques pour la sécurité, les normes industrielles prescrivent des exigences précises en matière de couverture de code, de sorte que les produits ne peuvent être certifiés ici sans la preuve d'une couverture de test suffisante. Mais également dans d'autres projets de développement, les entreprises attachent de plus en plus d'importance à la qualité des logiciels et mesurent la couverture du code.

Différents analyseurs de couverture de code sont disponibles sur le marché pour mesurer la couverture de code. Ils diffèrent considérablement en termes de manipulation et de qualité. Pour cette raison, nous présentons dix critères de base pour la sélection d'un outil de couverture de code :

1 Indépendance par rapport au compilateur

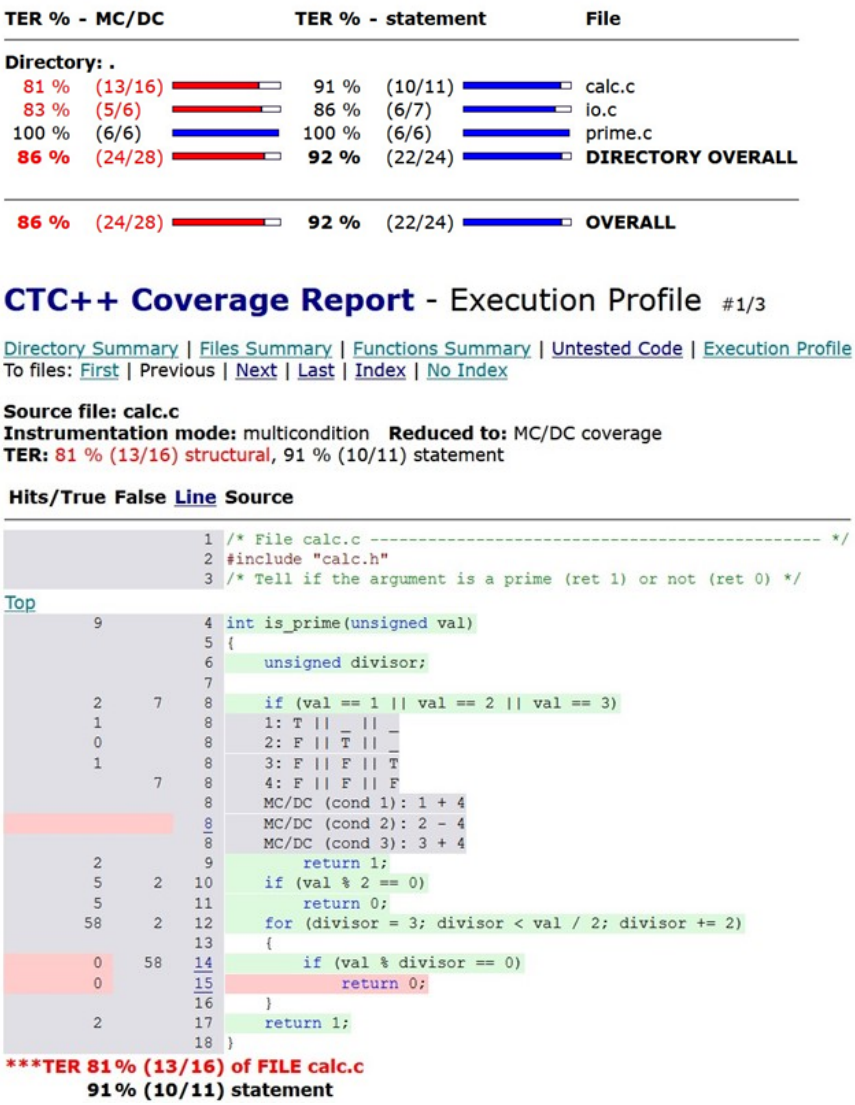
Bien sûr, un outil de couverture de code doit fonctionner avec le compilateur utilisé dans le projet. Cependant, il est très utile de s'appuyer sur un outil qui peut être utilisé indépendamment du compilateur dès le départ. Ainsi de tels outils peuvent ensuite être utilisés dans tous les projets et également dans le projet en cours en cas de changement de compilateur. Un outil de couverture qui peut être utilisé indépendamment du compilateur peut être utilisé de manière beaucoup plus diversifiée et constitue donc un investissement rentable.

2 Facilité d'utilisation

Le meilleur logiciel est utilisé à contrecœur (et donc rarement) s'il est inutilement compliqué ou mal conçu. Une manipulation simple, en revanche, peut augmenter de manière significative l'acceptation par l'utilisateur de l'utilisation d'un outil de couverture de tests. Idéalement, l'outil fonctionne en arrière-plan et ne génère pas de travail supplémentaire pour l'utilisateur pendant les tests.

3 Compréhensibilité des rapports de couverture

Lors de l'évaluation des rapports de couverture, il devrait être visible en un coup d'œil, quelles parties du code ont déjà été testées et où la couverture est encore insuffisante. Avec de bons outils de couverture, le testeur peut facilement identifier au niveau du code source les cas de test encore en suspens. En exécutant ces tests manquants, la couverture de code peut alors être augmentée de manière ciblée. En même temps, cela évite le travail inutile qui résulterait de tests redondants.



TER % - MC/DC		TER % - statement		File
Directory: .				
81 %	(13/16)	91 %	(10/11)	calc.c
83 %	(5/6)	86 %	(6/7)	io.c
100 %	(6/6)	100 %	(6/6)	prime.c
86 %	(24/28)	**92 %**	(22/24)	**DIRECTORY OVERALL**
86 %	(24/28)	**92 %**	(22/24)	**OVERALL**

CTC++ Coverage Report - Execution Profile #1/3

Directory Summary | Files Summary | Functions Summary | Untested Code | Execution Profile
To files: First | Previous | Next | Last | Index | No Index

Source file: calc.c
Instrumentation mode: multicondition **Reduced to:** MC/DC coverage
TER: 81 % (13/16) structural, 91 % (10/11) statement

Hits/True	False	Line	Source
		1	/* File calc.c ---------- */
		2	#include "calc.h"
		3	/* Tell if the argument is a prime (ret 1) or not (ret 0) */
Top			
9		4	int is_prime(unsigned val)
		5	{
		6	unsigned divisor;
		7	
2	7	8	if (val == 1 || val == 2 || val == 3)
1		8	1: T || _ || _
0		8	2: F || T || _
1		8	3: F || F || T
	7	8	4: F || F || F
		8	MC/DC (cond 1): 1 + 4
		8	MC/DC (cond 2): 2 - 4
		8	MC/DC (cond 3): 3 + 4
2		9	return 1;
5	2	10	if (val % 2 == 0)
5		11	return 0;
58	2	12	for (divisor = 3; divisor < val / 2; divisor += 2)
		13	{
0	58	14	if (val % divisor == 0)
0		15	return 0;
		16	}
2		17	return 1;
		18	}

***TER 81 % (13/16) of FILE calc.c
91 % (10/11) statement

Fig. 1 et 2 : En plus d'une vue d'ensemble de la couverture du code des différentes parties du code, Testwell CTC++ affiche également des informations détaillées qui montrent exactement dans quelle mesure le code source est couvert par les tests, même pour les niveaux de couverture les plus élevés.

4 Prise en charge de niveaux de couverture plus élevés pour les développements critiques en matière de sécurité

Pour le test des logiciels critiques en matière de sécurité, les normes (par exemple, ISO 26262 dans le secteur automobile, DO-178C dans l'aviation et EN-50128 dans le transport ferroviaire) stipulent des niveaux de couverture élevés, jusqu'à la couverture MC/DC. Il est donc impératif de s'assurer que l'outil de couverture prend en charge tous les niveaux de couverture requis. Pour pouvoir utiliser une solution à long terme, il faut tenir compte non seulement des

exigences actuelles, mais aussi des exigences futures. Important à savoir : de nombreux outils de couverture n'offrent qu'une couverture de décision ou une couverture de condition (Decision Coverage ou Branch Coverage en anglais) et sont donc insuffisants pour le développement de logiciels critiques pour la sécurité.

5 Intégration flexible

Même au sein d'une entreprise, les environnements de développement et les chaînes d'outils sont souvent très hétérogènes. Un outil de couverture doit pouvoir s'adapter facilement à tous ces environnements différents. L'intégration dans le processus de construction et dans l'exécution des tests doit être possible de manière transparente et sans grand effort. Si l'outil peut également être utilisé via la ligne de commande, il y a des avantages pour faire des builds automatisés.

6 Faible coût d'instrumentation

La plupart des outils de couverture mesurent la couverture de code en instrumentant le code source. Le code source est enrichi par l'outil de couverture avec des "compteurs", qui comptent où et combien de fois les parties de code pertinentes ont été exécutées pendant les tests. Cependant, cela augmente la taille du code original. Lors de tests sur des cibles embarquées disposant d'une mémoire limitée, il convient donc de veiller à ce que cette "surcharge d'instrumentation" soit aussi faible que possible. Les différences de besoins en mémoire entre les différents outils de couverture de code sont parfois considérables. L'analyseur de couverture de code Testwell CTC++ de Verifysoft Technology, par exemple, est très économe en ressources à cet égard. Avec Testwell CTC++, il est même possible de réduire encore l'espace mémoire nécessaire en utilisant l'option du « bit coverage » (de couverture binaire). Avec cette option, Testwell CTC++ mesure alors uniquement si une partie du code a été testée, mais pas combien de fois elle l'a été.

7 Prise en charge de différents langages de programmation

Les entreprises travaillent souvent avec différents langages de programmation ou prévoient d'en introduire d'autres à l'avenir. Il est donc judicieux de choisir dès le départ un outil qui prend en charge tous ces langages ou le plus grand nombre possible d'entre eux.

8 Prise en charge de la programmation « créative »

Certains outils de couverture rencontrent des problèmes lorsqu'ils analysent des constructions de langage qui s'écartent des normes communes ou qui ont une profondeur d'imbrication élevée. Cependant, un bon outil de mesure de la couverture des tests devrait également être capable de faire face à un style de programmation "créative".

Figure 3 L'analyseur de couverture de test CTC++ de Testwell est souvent comparé à un véhicule tout-terrain : l'outil fonctionne toujours de manière fiable, même sur un " terrain " difficile...

9 Adéquation au développement de logiciels critiques pour la sécurité

Lors du développement de logiciels critiques pour la sécurité, les normes en vigueur exigent que l'ensemble de la chaîne d'outils soit qualifié. L'objectif est de prouver que l'analyseur de couverture et les autres outils utilisés dans la chaîne d'outils fonctionnent de manière fiable. Les fabricants d'outils professionnels de couverture de code soutiennent les projets logiciels avec des kits de qualification et des recommandations sur la qualification des outils. Dans ce contexte, il convient également de vérifier si l'outil de couverture sélectionné est déjà utilisé avec succès dans des projets critiques pour la sécurité.

10 Licences d'évaluation, support technique et références clients

L'adéquation d'un outil de couverture à ses propres projets doit être vérifiée lors d'une évaluation de l'outil dans toutes ses fonctionnalités.
Pendant cette période, vous aurez déjà une impression de la performance du support technique. L'assistance est-elle également disponible par téléphone ou uniquement par e-mail ? Quelle est l´expertise du personnel d'assistance ? Quels sont les délais de réponse ? Le manuel d'utilisation est-il de bonne qualité et pratique ? Le fabricant propose-t-il également des formations ? Enfin, il est également conseillé de jeter un coup d'œil aux références clients du fabricant. Celles-ci peuvent fournir des informations supplémentaires sur la qualité de l'analyseur de couverture et les performances du fournisseur.

" La couverture de code est obligatoire pour le développement de logiciels critiques pour la sécurité, et ce pour une bonne raison. C'est aussi une bonne méthode pour quiconque souhaitant améliorer la qualité de ses logiciels en général, de mesurer et d'augmenter la couverture et la valeur des tests ", explique Klaus Lambertz, directeur général de Verifysoft Technology GmbH. " Lors de la sélection d'un analyseur de couverture de code, il faut veiller à ce que l'outil réponde aux exigences fixées. En outre, des facteurs tels que la facilité d'utilisation et le support professionnel jouent un rôle important. Utilisé correctement, un bon outil de couverture de test permet d'améliorer sensiblement la qualité, d'accroître la motivation des développeurs et des testeurs, et d'effectuer les tests de manière plus économique. "

Couverture de code en un coup d'œil : Différents niveaux de couverture de test

Couverture des fonctions (Function Coverage)
La couverture des fonctions mesure si toutes les fonctions du programme ont été appelées. La couverture des fonctions est le plus "faible" des niveaux de couverture de test habituels.

Couverture des instructions (**Statement Coverage)**
La couverture des instructions mesure le pourcentage des instructions testées par rapport à l'ensemble des instructions.

Couverture des decisions / Couverture des branches (Decision Coverage / Branch Coverage)
À ce niveau de couverture, chaque décision doit être testée au moins une fois comme vraie et une fois comme fausse. Pour les instructions if normales, cela correspond à la couverture des branches, où chaque branche doit avoir été exécutée.

Couverture des points de tests / couverture des conditions (Condition Coverage)
La couverture des conditions (également nommé couverture de points de tests) examine en détail les décisions composées. Pour les décisions qui consistent en de multiples conditions atomiques composées via des opérateurs booléens, chacune de ces conditions doit être testée individuellement comme "vraie" et comme "fausse".

Couverture des Conditions/Décisions Modifiée (MC/DC) et Couverture des conditions multiples (Multicondition Coverage, MCC)

Pour la couverture des conditions multiples, toutes les combinaisons vrai-faux possibles doivent être vérifiées pour les décisions composites. Dans le cas de conditions multiples au sein d'une décision, cela nécessite un nombre de cas de test souvent très (et trop) élevé. Dans la pratique et dans les normes, la couverture condition/décision modifiée (MC/DC) est donc pertinente, le nombre de cas de test étant réduit et la valeur informative de la couverture de test restant suffisamment élevée.

Liens utiles :

Couverture de test : https://www.verifysoft.com/fr_code_coverage.html

Testwell CTC++ : https://www.verifysoft.com/fr_ctcpp.html

Auteur : Klaus Lambertz, Verifysoft Technology GmbH