Verifysoft TECHNOLOGY

La couverture de code presque un jeu d’enfant ?

Les tests de logiciels sont un élément indispensable de l‘assurance qualité. Sans tests suffisants et documentés, il est impossible de déterminer si un logiciel est sûr et fonctionnellement correct.

Pour vérifier si le code complet a été testé, la mesure de la couverture de code est nécessaire.

Cela permet de déterminer dans quelle mesure un logiciel a déjà été testé et quelles parties de code doivent encore être testées pour atteindre la couverture de code définie. La couverture de code indique le rapport entre le code testé et le code total. Pour simplifier, on peut dire, par exemple, que la couverture de code est de 75% si le test porte sur trois des quatre options possibles. Une couverture complète du code signifie d’abord que 100 pour cent du niveau de test spécifié a été atteint.

Pour les applications logicielles « non critiques », il n‘est souvent pas utile de viser une couverture de code de 100 %. Il est toutefois important de savoir pourquoi vous exécutez certains cas de test et quelles parties de code sont ainsi contrôlées. Sans cette information, il existe d‘une part un risque élevé de tests redondants qui passent et repassent sur le code déjà testé, et d‘autre part le risque que des parties importantes du code ne soient pas vérifiées. Dans les domaines d‘application non critiques, les tests ne devraient donc être ni « trop peu » ni « complets », mais « suffisants » pour atteindre un équilibre raisonnable entre les coûts et les avantages.

La situation est différente pour les logiciels critiques en termes de sécurité. Dans le secteur automobile, ferroviaire et aéronautique, un logiciel défectueux peut entraîner des conséquences catastrophiques. Dans la technique médicale, les patients comptent sur des stimulateurs cardiaques et des pompes à insuline intelligentes qui fonctionnent correctement. Avec la criticité croissante des différents appareils ainsi que la possibilité de les attaquer de l‘extérieur, les exigences en matière de tests logiciels augmentent.

C‘est pourquoi les normes sectorielles prescrivent des exigences précises en matière de couverture de code dans le développement de logiciels critiques pour la sécurité. Les produits ne peuvent pas y être certifiés sans la preuve d‘une couverture de test suffisante.

La couverture de code est donc obligatoire pour les logiciels critiques, mais elle est également utile pour les logiciels moins critiques. Toutefois, la mesure de la couverture de code semble souvent complexe et longue, surtout pour les débutants. Pour les logiciels embarqués, il existe des défis supplémentaires tels que l‘espace mémoire limité. Cependant, si l‘on tient compte de quelques aspects fondamentaux, la mesure de la couverture de code devient rapidement plus simple et, si l‘on utilise un outil approprié, presque un jeu d‘enfant.

Les analyseurs de couverture de code disponibles sur le marché se distinguent toutefois nettement par leur maniement et leur qualité. Certains critères aident à choisir un outil de couverture de code approprié.

Exigences relatives aux outils de couverture de code

Convivialité

Il en va de même pour les outils de couverture de code : les meilleurs logiciels sont employés à contrecœur (et donc rarement), si son utilisation est inutilement compliquée ou mal pensée. Une manipulation simple peut en revanche augmenter l‘acceptation de l‘utilisation d‘un outil de couverture de test par l‘utilisateur et contribue ainsi à économiser du temps et de l‘argent. Idéalement, l‘outil doit fonctionner en arrière-plan, être éventuellement intégré dans un processus d‘intégration continue et ne pas générer de travail supplémentaire pour l‘utilisateur lors du test.

Traçabilité des rapports de couverture

L‘analyse de couverture de code génère généralement de grandes quantités de données qui doivent être traitées par l‘outil de couverture d‘une manière pertinente et interprétable.

Les informations fournies doivent être disponibles sous une forme lisible par une machine pour un traitement automatique ultérieur (par ex. XML) et sous une forme lisible par l‘homme (par ex. HTML) pour un aperçu rapide et de qualité.

Pour le testeur ou le responsable qualité, l‘examen des rapports de couverture devrait permettre de voir, si possible en un coup d‘œil, quelles parties du code ont déjà été testées et où il manque encore des tests. Avec de bons outils de couverture, le testeur peut très facilement voir, au niveau du code source, quels cas de test sont encore en attente. En exécutant ces tests manquants, la couverture de code peut alors être augmentée de manière ciblée. Cela évite en même temps le travail inutile qui résulterait de tests redondants.

Des rapports de couverture compréhensibles sont également importants dans le contexte des certifications ou des audits qui peuvent être exigés.

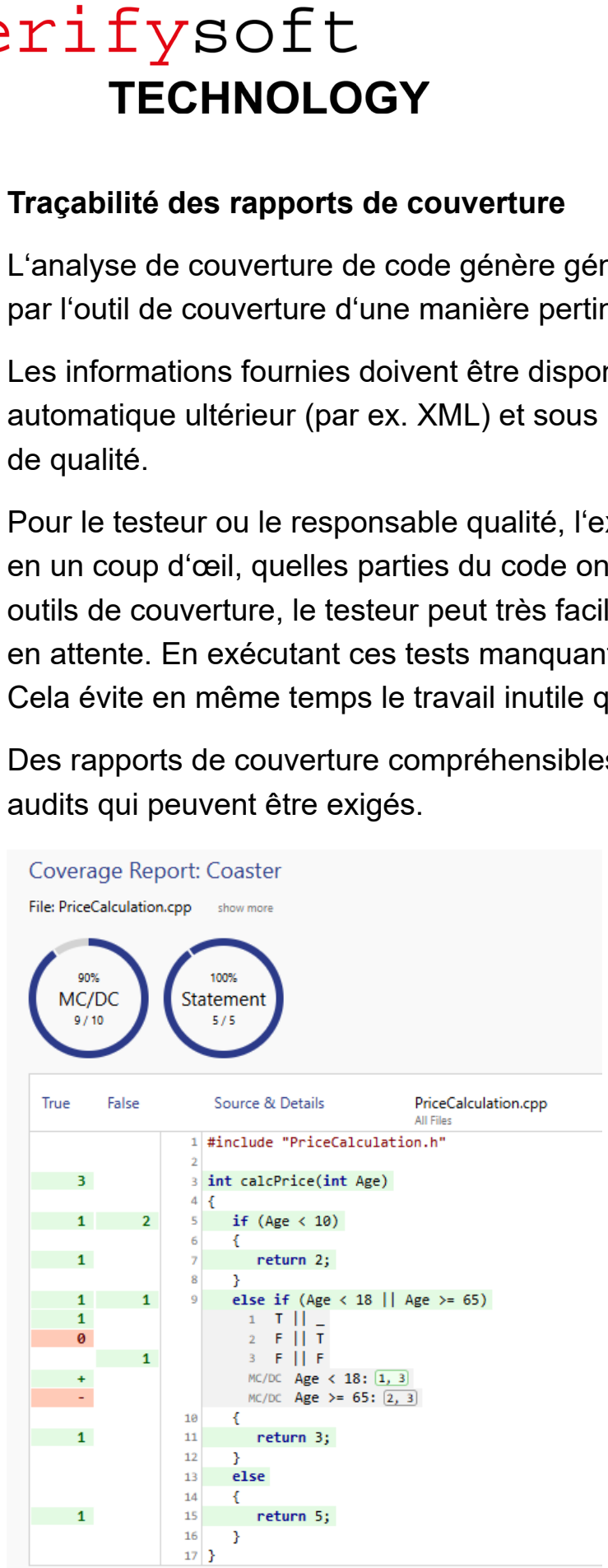

Image 1 : Informations détaillés concernant la couverture de code dans le code source

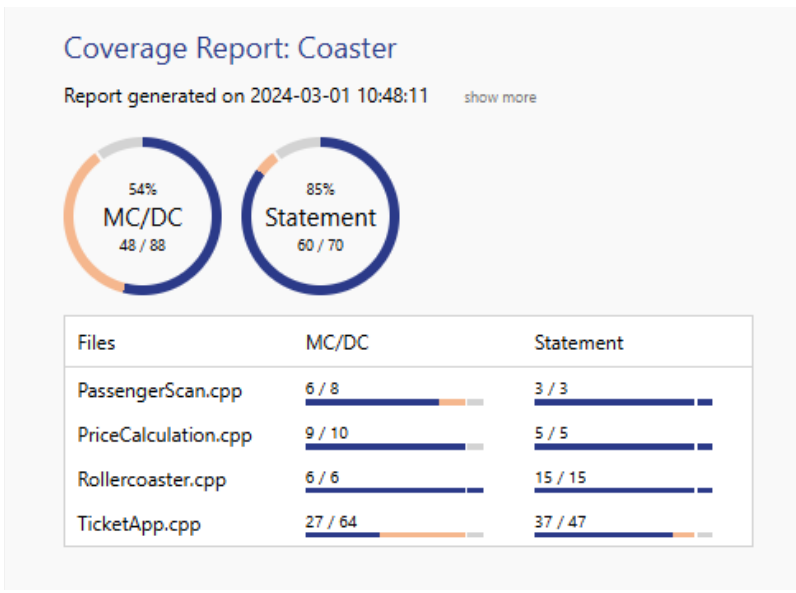

Image 2 : Aperçu des résultats de la couverture de code

Possibilité de justifications

Souvent, certaines parties de code ne peuvent pas être atteintes par des cas de test, par exemple en raison d‘une programmation défensive. Dans ce cas, l’analyseur de couverture de code devrait offrir la possibilité d’expliquer et de documenter la couverture manquante via des justifications. L’analyseur de couverture de code Testwell CTC++ offre la possibilité de déposer des justifications directement dans le code source via des commentaires ou, alternativement, de les documenter dans des fichiers en dehors du code source. Dans les rapports de couverture, il est possible de voir de manière transparente quelles parties de code ont été réellement testées et lesquelles ont été « expliquées » par des justifications.

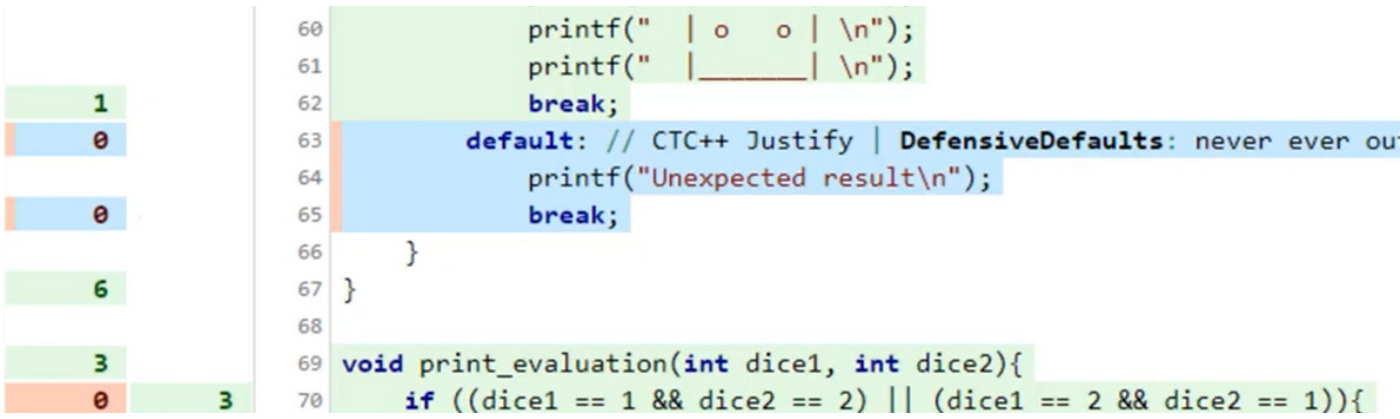

Image 3 : Les justifications insérées en tant que commentaire dans le code source apparaissent en bleu dans le rapport.

Image 4 : Dans l‘aperçu, les justifications sont marquées en bleu clair, le code effectivement couvert en bleu foncé.

Soutien à la programmation « créative »

Certains outils de couverture rencontrent des problèmes lors de l‘analyse de constructions linguistiques qui s‘écartent des standards habituels ou ont une profondeur d‘imbrication élevée. Un bon outil pour la mesure de la couverture de test devrait cependant aussi s‘accommoder d‘un style de programmation « créatif ».

Indépendance vis-à-vis du compilateur utilisé

Bien entendu, un outil de couverture de code doit fonctionner correctement avec le compilateur utilisé dans le projet. Toutefois, il est très judicieux de miser dès le début sur un outil qui peut être utilisé indépendamment du compilateur. De tels outils peuvent ensuite être employés dans tous les projets et il est également possible de changer de compilateur en cours de projet si nécessaire. Un outil de couverture utilisable indépendamment du compilateur peut être utilisé de manière beaucoup plus variée et constitue donc un investissement rentable.

Intégration flexible

Même au sein d‘une entreprise, les environnements de développement et les chaînes d‘outils sont souvent très hétérogènes. En outre, de plus en plus de tests sont automatisés afin de réduire les coûts et d‘éviter les sources d‘erreurs dues aux actions manuelles.

Il est difficile de se passer de tests automatisés et de la saisie automatique de la couverture de code, surtout dans les grands projets de développement. L‘automatisation des tests sur dans le processus de build est surtout nécessaire pour les développementsagiles avec intégration continue/déploiement continu (CI/CD). C‘est pourquoi l‘intégration de l’outil de couverture dans le processus de build doit être possible de manière transparente et sans grand effort. Des outils tels que Testwell CTC++, qui peuvent également être utilisés en ligne de commande, simplifient l’intégration et offrent des avantages lors de la création de Builds automatisés.

Prise en charge de mesures de couverture plus élevées / aptitude au développement critique en matière de sécurité

Pour le test de logiciels critiques pour la sécurité, les normes comme ISO 26262 dans ledomaine automobile, DO-178C dans l‘aviation et EN 50716, la norme qui a succédé à EN 50128 dans le domaine ferroviaire, prescrivent des mesures de couverture élevées qui, selon le niveau de sécurité, vont jusqu‘à la couverture MC/DC. Il est donc impératif de veiller à ce que l‘outil de couverture à choisir supporte effectivement le niveau de couverture requis.

Pour pouvoir utiliser une solution à long terme, il convient de prendre en compte non seulement les exigences actuelles, mais aussi les exigences futures, déjà prévisibles. Il est important de savoir que de nombreux outils de couverture ne couvrent que la couverture de branches ou la couverture des décisions et sont donc insuffisants pour le développement de logiciels critiques en termes de sécurité.

Selon le niveau d‘intégrité de sécurité (SIL) pour IEC 61508 ou Automotive Safety Level(ASIL) pour ISO 26262, la couverture des instructions, la couverture des branchements ou la couverture MC/DC (Modified Condition/Decision Coverage) est nécessaire. Pour toutes les normes, plus les conséquences d‘éventuelles erreurs sont dangereuses, plus le degré de couverture exigé est strict. C‘est ce qui ressort du tableau de la norme ISO 26262.

Methods		ASIL			
		A	B	C	D
1a	Statement coverage	++	++	+	+
1b	Branch coverage	+	++	++	++
1c	MC/DC (Modified Condition/Decision Coverage)	+	+	+	++

Image 5 : Spécifications de la norme ISO 26262 pour la couverture de code au niveau des tests unitaires

Le niveau de sécurité le plus élevé, ASIL D, exige le degré de couverture le plus élevé, MC/DC. Dans le tableau, ++ signifie « très recommandé », + signifie « recommandé ».

Pour les logiciels pour lesquels il n‘existe pas de prescriptions sous forme de normes, il est également important de déterminer à l‘avance quel code doit être testé et selon quels critères. Pour ce faire, il peut être utile d‘utiliser les normes susmentionnées comme guide et d‘orienter l‘étendue des tests en fonction de la criticité du système.

Certification et qualification

La plupart des normes de sécurité exigent également que l‘ensemble de la chaîne d‘outils soit qualifié. Il s‘agit ici de prouver que tant l‘analyseur de couverture que les autres outils utilisés fonctionnent de manière fiable dans l‘ensemble de la chaîne d‘outils.

Si l‘outil est certifié (par exemple par le TÜV), il peut en général être utilisé sans autres mesures de qualification dans des projets selon les normes IEC 61508, ISO 26262, EN 50716 ainsi que IEC 62304.

Exigences spécifiques pour le développement de logiciels embarqués

En règle générale, l‘espace mémoire disponible pour les logiciels des appareils embarqués est limité. De plus, les processeurs sont souvent limités pour des raisons de coûts. Il en résulte des défis particuliers concernant la mesure de la couverture de code, qui doivent être pris en compte lors du choix d‘un analyseur de couverture.

Faible overhead d‘instrumentation

La plupart des outils de couverture mesurent la couverture de code via l‘instrumentation du code source. Le code source est alors enrichi par l‘outil de couverture avec des « compteurs »,

qui comptent où et combien de fois les parties de code concernées ont été exécutées lors du test. En outre, une bibliothèque doit être implémentée sur la cible à tester, qui se charge entre autres du transfert de données vers un hôte. Cela augmente la taille du code original.

Lors de tests sur des cibles embarquées disposant d‘un espace mémoire limité, il faut donc veiller à ce que cet « overhead d´instrumentation » reste le plus faible possible.

Les différences entre les différents outils de couverture de code sont parfois considérables en ce qui concerne l‘utilisation de la mémoire. L’analyseur de couverture de code Testwell CTC++ est très peu gourmand en ressources et offre en outre la possibilité de réduire encore sensiblement l’espace mémoire grâce à des compteurs plus petits. La fonction « Bit Coverage » réduit la taille des compteurs de 32 bits à 16 ou 8 bits, mais ne mesure plus le nombre de fois qu’une partie de code a été testée, mais seulement si elle a été testée. Cela suffit pour répondre aux exigences des normes.

Si l‘overhead d‘instrumentation est malgré tout trop élevé, la couverture du code peut être analysée séparément pour certaines parties du code (instrumentation partielle).

Vérifier les effets possibles sur le processeur

Pour réduire les coûts, il n‘y a pas que les mémoires de nombreux appareils embarqués qui sont petites - les processeurs aussi ont souvent des restrictions. Malheureusement, l‘instrumentation a également un impact sur le processeur. Ainsi, il peut arriver dans certains cas que le timing défini soit dépassé, ce qui peut éventuellement conduire à des exécutions de programme erronées. C‘est pourquoi les analyseurs de couverture de code devraient également être évalués en fonction de leurs effets sur le timing.

Ici aussi, la fonction Bit-Coverage mentionnée ci-dessus peut apporter une solution.Il est également utile d‘effectuer les tests une fois avec et une fois sans l‘analyse de couverture afin de s‘assurer que la mesure de la couverture n‘a pas entraîné de modifications du comportement du programme.

Conclusion

La couverture de code est obligatoire pour le développement de logiciels critiques pour la sécurité, et ce pour de bonnes raisons. Mais la mesure de la couverture de test est également une bonne méthode pour tous ceux qui souhaitent améliorer la qualité de leurs logiciels en général, afin de mesurer et d‘augmenter la qualité et la pertinence des tests logiciels.

Lors du choix d‘un analyseur de couverture de code, il faut veiller à ce que l‘outil réponde aux exigences posées. En outre, des facteurs tels que la facilité d‘utilisation et le soutien professionnel en cas de questions ou d‘assistance jouent un rôle important.

L‘adéquation d‘un outil de couverture à ses propres projets devrait en tout cas être appréciée dans le cadre d‘une évaluation de l‘outil. Déjà à ce stade, on pourra se faire une idée de la qualité et de l´efficacité du support technique. Enfin, il est également recommandé de jeter un coup d‘œil sur les références clients du fabricant.

Utilisé correctement, un bon outil de couverture de test permet d‘améliorer considérablement la qualité, d‘augmenter la motivation des développeurs et des testeurs et de réaliser des tests à moindre coût.

Aperçu des niveaux de couverture

Il existe de nombreux niveaux de couverture de test différents. En général, plus la couverture est stricte plus un niveau de couverture de test est élevé et détaillé, plus les efforts et les coûts nécessaires pour l‘atteindre sont importants.

Function Coverage

La « Function Coverage » (ou la couverture de fonction) mesure si toutes les fonctions du programme ont été appelées. C´est le niveau le « plus faible » des niveaux de couverture de test habituels. Comme elle ignore le fonctionnement interne du logiciel, son utilité est assez limitée.

Statement Coverage

Le « Statement Coverage » (ou la couverture des instructions) détermine quelles instructions ont été exécutées par les tests. Cela permet également de détecter le code mort.

Decision Coverage / Branch Coverage

Avec ce niveau de couverture (couverture des décisions / couverture des branches)chaque décision doit être testée au moins une fois comme « true » et une fois comme « false ». Pour les instructions « if » normales, cela signifie que chaque ramification doit avoir été exécutée.

Condition Coverage

La couverture des conditions prend en compte les décisions composées de manière détaillée. Pour les décisions constituées de plusieurs conditions atomiques composées d´opérateurs booléens, chacune de ces conditions doit être testée individuellement comme « true » et comme « false ».

Verifysoft TECHNOLOGY

Multicondition Coverage et Modified Condition/Decision Coverage (MC/DC)

Dans le cas de la couverture des conditions multiples (Multicondition Coverage), il s‘agit de vérifier toutes les combinaisons vrai/faux possibles pour des décisions composées. Dans le cas de plusieurs conditions au sein d‘une décision, cela nécessite généralement un très grand (voire trop grand) nombre de cas de test. C‘est pourquoi, dans la pratique et dans les normes, la couverture condition / décision modifiée (Modified Condition/Decision Coverage, MC/DC) est judicieuse, car elle permet de réduire le nombre de cas de test tout en conservant une couverture de test suffisamment pertinente. Pour chacune des conditions atomiques, une paire de cas de test est testée, ce qui entraîne une modification du résultat global de la condition composée. Cependant, seule la valeur de vérité de la condition atomique considérée change, tandis que la valeur de vérité des autres conditions atomiques doit rester constante. Dans la pratique, on obtient ainsi le même objectif que pour la couverture des conditions multiples - mais avec moins de cas de test.

Auteur:

Klaus Lambertz

Fondateur et Directeur Général

Verifysoft Technology GmbH

Cet article a été publié le 23/05/2025 dans le magazine programmez!

https://www.programmez.com/magazine/programmez-269

Pour plus d‘informations:

Verifysoft Technology GmbH : https://www.verifysoft.com

Testwell CTC++ Code Coverage Analyzer: https://verifysoft.com/fr_ctcpp.html

E-Mail: info@verifysoft.com