Automobile : l'intégration et le test pour la norme ISO 26262 nécessitent de l'automatisation

Garder le cap sur le standard ISO 26262 pour respecter les normes de sécurité fonctionnelle automobiles exige une automatisation tout au long du cycle de vie du logiciel. Du cahier des charges jusqu'au déploiement et à la vérification, l'automatisation permet l'analyse d'impact, l'évaluation de la qualité du logiciel, la génération de code, la simulation et les tests unitaires et d'intégration pour répondre aux exigences de la norme. Le résultat est l'assurance d'une production rentable de logiciels, comme l'explique ici LDRA.

AUTEUR

Mark Pitchford, Technical Marketing Engineer, LDRA.

Les automobiles actuelles sont un véritable labyrinthe de systèmes électromécaniques interactifs. Beaucoup d'entre eux, tels que les freins, la direction, les airbags, le groupe motopropulseur et les systèmes adaptatifs d'assistance au conducteur (Adaptive Driver Assistance Systems, ADAS), sont critiques en termes de vie humaine et de sécurité. D'autres, comme les systèmes d'info-divertissement, le sont moins. Cependant, ils dépendent tous d'une quantité pléthorique de logiciels et, dans de nombreuses conceptions, ces systèmes partagent également la même infrastructure de communication interne. Ce qui signifie que la sécurité fonctionnelle du point de vue des systèmes doit être assurée pendant le développement et le test de ces logiciels.

Dans ce cadre, la norme ISO 26262 est une norme de sécurité fonctionnelle pour les véhicules routiers. Elle définit les exigences et les processus afin d'assurer la sécurité selon une gamme de niveaux de classification des dangers appelés niveaux d'intégrité de la sécurité automobile (Automotive Safety Integrity Levels, ASIL). Ceux-ci spécifient des mesures de sécurité fonctionnelle pour les niveaux A (les moins dangereux) à D (les plus dangereux). La norme ISO 26262 établit un processus qui commence par des exigences générales, puis des spécifications relevant concrètement de la sécurité, de la conception de l'architecture du logiciel et in fine du codage effectif et de l'implémentation des unités fonctionnelles. Il existe également des étapes pour tester et vérifier chacune de ces phases.

L'établissement impératif des spécifications détaillées concernant la conception du système et les exigences de sécurité peut être effectué à un niveau assez abstrait à l'aide de tableurs, d'outils de traitement de texte et autres gestionnaires de contraintes plus formels. Cependant, il faut ensuite que ces exigences soient fidèlement respectées jusqu'au niveau de chaque composant logiciel qui les implémente et de chaque routine de vérification qui les prouve. En vertu de la norme ISO 26262, la traçabilité bidirectionnelle est essentielle pour assurer un cycle de développement transparent et ouvert. De même, si le code doit être réécrit, il est important de comprendre à partir de quelle exigence en amont il a été dérivé.

Conception d'architecture du logiciel et testabilité

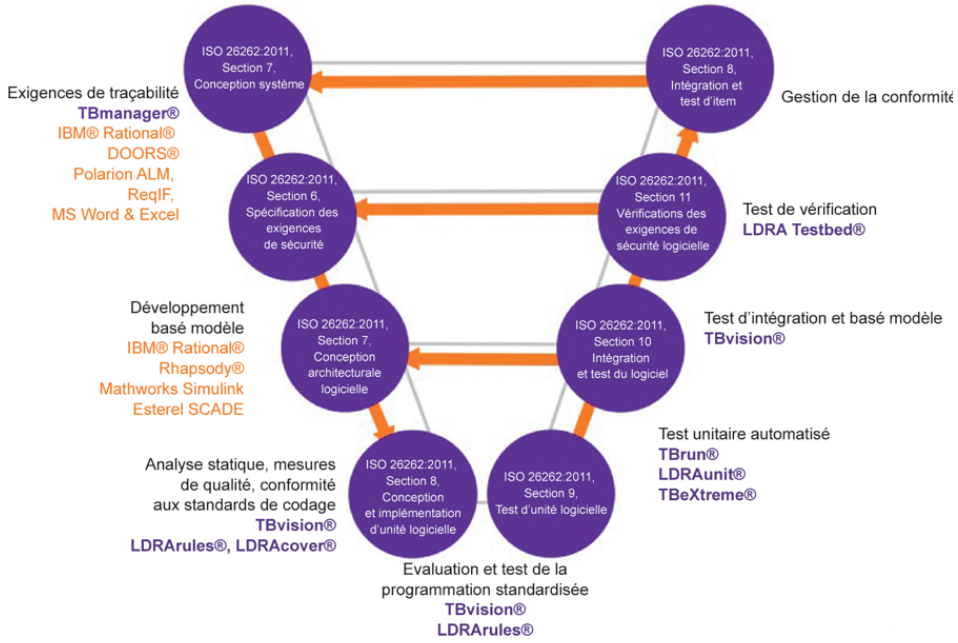
Le Design-for-Testability (conception pour la testabilité) est un terme qui peut s'avérer très large, mais le concept est clair dans le cadre de la norme ISO 26262. La conception du logiciel, telle que décrite dans la section 7 de la norme, vise spécifiquement à produire une architecture logicielle conforme aux exigences de sécurité du logiciel. Des outils de modélisation sont souvent utilisés pendant cette phase initiale afin d'explorer plusieurs solutions pour l'architecture du logiciel. Certaines entreprises s'appuient toujours sur des méthodes manuelles de conception de haut niveau, en utilisant des documents ou même un codage de haut niveau (c'est-à-dire sans le comportement détaillé). Néanmoins, quelle que soit la méthode adoptée tant durant la conception que pour l'implémentation, la conception architecturale doit être vérifiée. Pour les niveaux inférieurs de sécurité, c'est-à-dire les niveaux A et B, des techniques telles que les procédures informelles et les inspections peuvent suffire. Pour les niveaux plus élevés d'intégrité de la sécurité tels que C et D, les techniques d'automatisation permettent aux développeurs d'effectuer de manière "rentable" l'analyse et la vérification de l'architecture, incluant l'analyse en profondeur du flux de contrôle et du flux de données. Au final, le respect de la norme ISO 26262 doit conduire à une architecture logicielle bien définie qui est vérifiable et facilement traçable concernant ses exigences de sécurité fonctionnelle.

La norme ISO 26262 requiert également une structure hiérarchique de composants logiciels pour tous les niveaux d'intégrité de sécurité. Du point de vue de la qualité, la norme comprend des directives telles que :

- Les composants logiciels doivent être restreints en taille et faiblement couplés avec d'autres composants.
- Toutes les variables doivent être initialisées.
- Il ne devrait pas y avoir de variables globales ou leur utilisation doit être justifiée.

1 CORRESPONDANCE ENTRE OUTILS D'UNE CHAÎNE AUTOMATISÉE ET DIRECTIVES DE L'ISO 26262

L'application des principes de l'ISO 26262 au niveau du codage de l'unité assure que ces dernières sont correctes et pourront travailler ensemble dans l'architecture définie. Une suite d'outils intégrée rassemble ces moyens automatisés afin qu'ils soient appliqués dans un modèle de processus en V.



- Il ne devrait pas y avoir de conversions de type implicite, de sauts inconditionnels ou de flux cachés de données ou de contrôle.
- Les objets ou variables dynamiques doivent être vérifiés avant leur utilisation.

Sans automatisation, le processus de vérification de toutes ces règles et recommandations pour chaque unité en cours d'implémentation serait fastidieux, coûteux et propice aux erreurs.

Normes et directives de codage

Conformément aux spécifications établies par la norme ISO 26262, l'implémentation de l'unité logicielle conduit à une application plus performante et de haute qualité. Bien que cette norme ISO 26262 n'impose pas un standard particulier de codage, il exige cependant un codage standard, quel qu'il soit. Des normes et des méthodes appropriées telles que MISRA C: 2012, MISRA C++: 2008, SEI CERT C, CWE et autres partagent ce même objectif d'éliminer les problèmes potentiels de sécurité et de sûreté et sont supportées par des suites d'outils automatisées. Les directives de codage peuvent être vérifiées et appliquées régulièrement à l'aide d'un outil d'analyse statique intégré qui examine le code source et souligne les écarts par rapport à la norme sélectionnée.

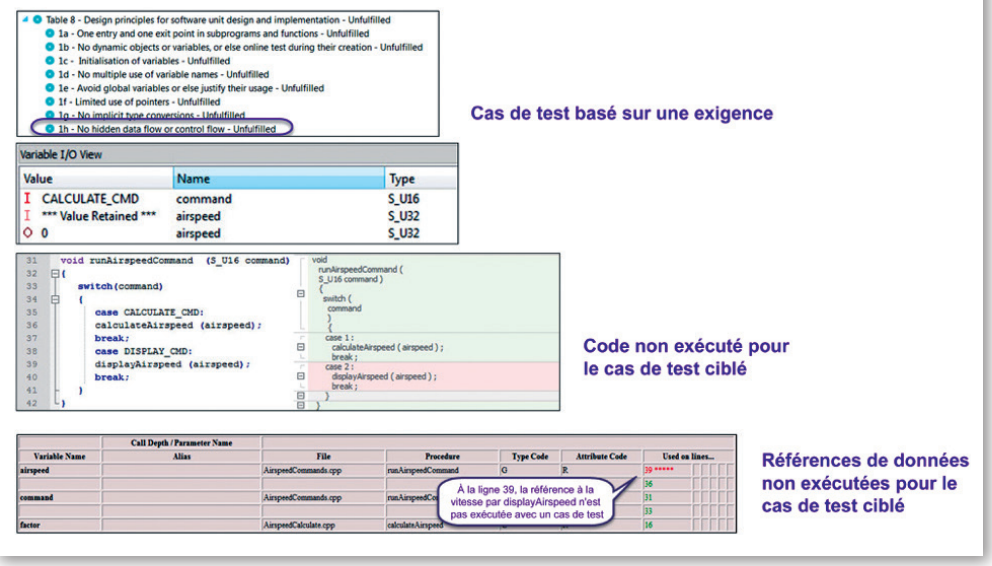
Au-delà de la conformité aux standards de codage, les outils logiciels entièrement intégrés peuvent vérifier et appliquer des techniques pour une conception de qualité des unités logicielles et faciliter leur intégration et leurs tests en fonction de l'architecture logicielle définie et des contraintes du système. L'application obligatoire de tels principes au niveau du codage de l'unité assure que ces unités seront correctes et pourront travailler ensemble dans l'architecture définie. Idéalement, une suite d'outils intégrée devrait rassembler ces moyens automatisés afin qu'ils puissent être appliqués à tous les stades de développement dans le modèle de processus en «V» standard (figure 1), permettant de coordonner la traçabilité, l'analyse et les tests des exigences sur toutes les étapes du développement du produit .

Analyse statique et test unitaire logiciel

Globalement, les directives de la norme ISO 26262 visent à rendre le code plus compréhensible, plus fiable, moins propice à l'erreur et plus facile à tester et à maintenir. Par exemple, la restriction de la taille des composants logiciels et de leurs interfaces les rend plus faciles à lire, à maintenir et à tester, et donc moins susceptibles de contenir des erreurs. L'analyse statique peut vérifier une multitude de directives pour s'assurer que les variables sont initialisées, que les variables globales ne sont pas utilisées et que la récursivité est évitée. L'existence de variables globales, par exemple, peut causer de la confusion au sein d'un programme trop grand et rendre les tests plus difficiles. L'application d'une analyse statique tout au long de la phase

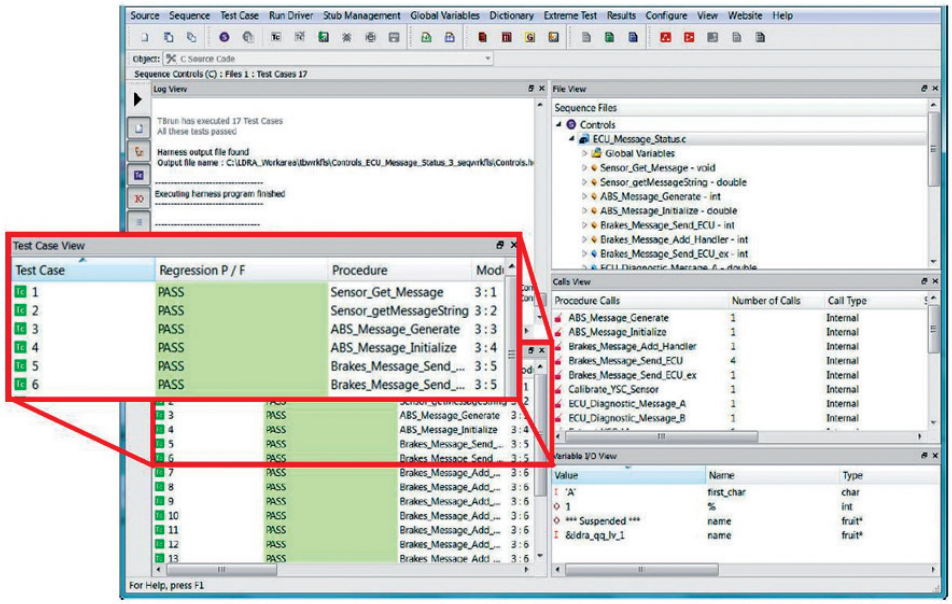
2 ANALYSE STATIQUE PENDANT LA PHASE D'IMPLANTATION DU CODE

L'analyse statique examine le couplage du contrôle et des données au sein d'une unité logicielle et relie cette dernière à l'architecture du système.



3 TESTS REPOSANT SUR LES EXIGENCES

Dans la suite d'outils de LDRA, les tests reposant sur les exigences permettent au développeur d'indiquer les entrées et les sorties attendues. Les sorties sont capturées avec des données de couverture structurelle et comparées aux résultats attendus.



d'implémentation du code met en évidence les violations au fur et à mesure qu'elles se produisent et donc assure au final qu'il n'y aucune violation. En outre, les outils peuvent générer des évaluations de la complexité pour permettre de mesurer et contrôler la taille, la complexité, la cohésion et le couplage des composants logiciels (figure 2).

Le test logiciel unitaire démontre à ce niveau que chaque unité logicielle (fonction ou procédure) remplit les spécifications de conception de l'unité et n'inclut aucun comportement indésirable. Une fois que cela est prouvé, les tests d'intégration d'unité démontrent que ce comportement reste correct lorsque ces unités sont déployées au sein d'un système, et que cette intégration réalise réellement la conception architecturale du logiciel telle que définie au niveau supérieur. On peut envisager le test d'intégration ultime mettant en œuvre le système complet, avec tous les logiciels fonctionnant comme un ensemble cohérent.

Les tests unitaires et les tests d'intégration d'unités s'appuient sur cette infrastructure pour exécuter des parties du code et vérifier que la fonctionnalité des interfaces logicielles est conforme aux spécifications et exigences du cahier des charges. Ce qui inclut de veiller à ce que seules ces exigences soient respectées et que le logiciel n'inclut aucune fonctionnalité non requise (c'est-à-dire indésirable). Cette même capacité de test unitaire peut être utilisée pour créer des tests d'injection de pannes pour la sécurité fonctionnelle, mesurer l'utilisation des ressources et, le cas échéant, s'assurer que le code généré automatiquement se comporte conformément au modèle à partir duquel il a été dérivé.

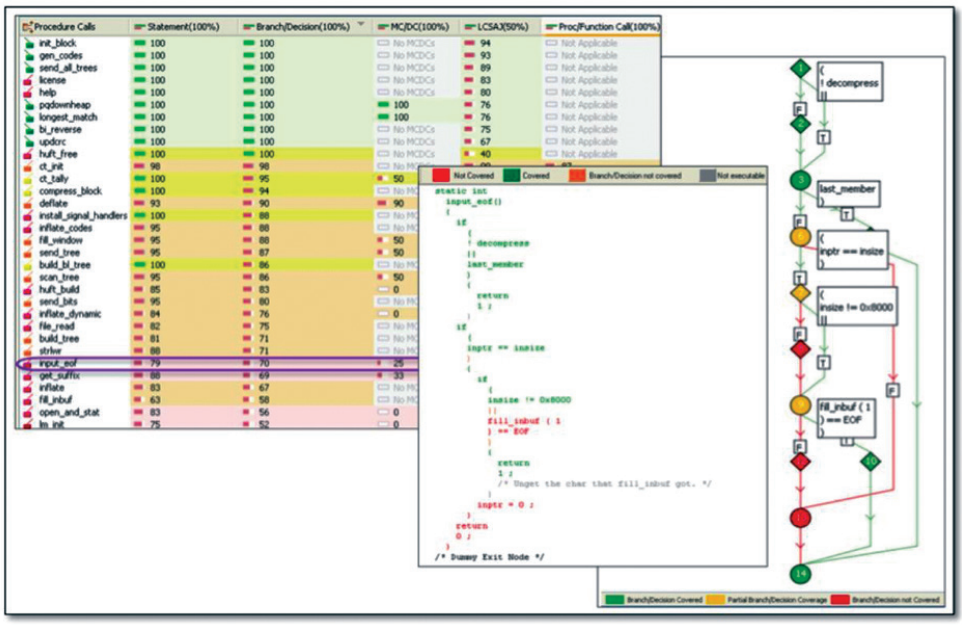
Alors que l'analyse statique peut effectuer une « inspection » automatisée du code source pour vérifier l'adhésion aux directives de la norme ISO 26262 pour le codage et l'implémentation de l'unité, au-delà de cela, les informations dérivées de cette analyse statique peuvent être utilisées pour fournir un cadre pour l'analyse dynamique – l'analyse de l'exécution du code. Idéalement, toutes les analyses dynamiques devraient être mises en œuvre à l'aide du matériel cible afin que tout problème résultant de ses limitations soit mis en évidence le plus tôt possible. Si le matériel cible n'est pas disponible dans les premières phases d'un projet, le code doit être exécuté dans un environnement de simulation reposant sur les spécifications de vérification. Ainsi le développement peut se poursuivre, tout en sachant qu'au final les tests sur le matériel cible réel seront nécessaires.

Assurer la traçabilité des exigences de sécurité du logiciel

Les tests d'intégration sont donc conçus pour s'assurer que toutes les unités fonctionnent ensemble et conformément à la conception architecturale et aux exigences. Dans le cas d'un projet compatible avec la norme ISO 26262, cela implique la vérification des fonctions relatives aux exigences de sécurité du logiciel ISO 26262 ainsi qu'aux contraintes

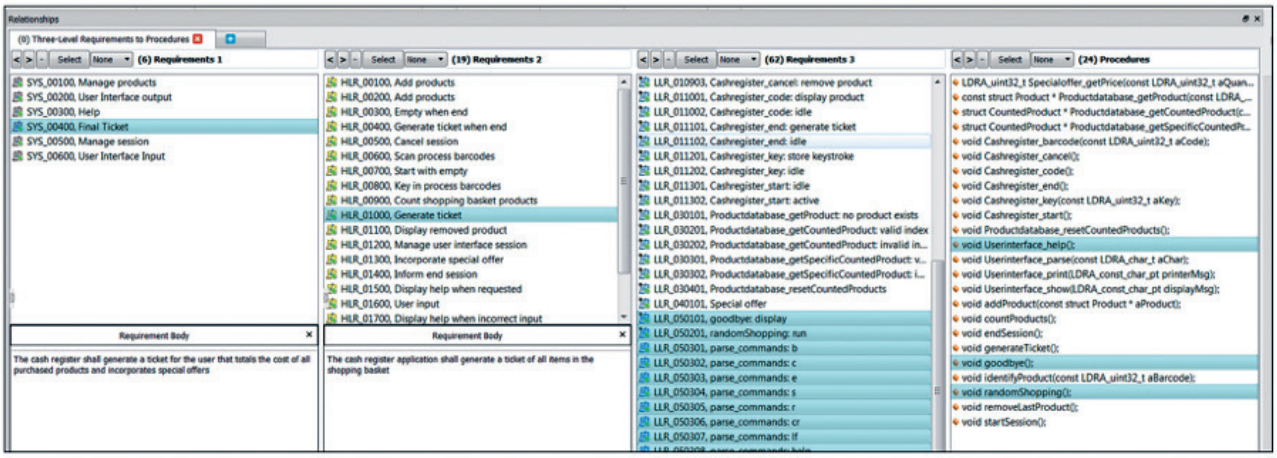
4 TESTS UNITAIRES, D'INTÉGRATION ET SYSTÈME POUR LE DEGRÉ DE COUVERTURE D'UN PROJET

L'analyse de la couverture structurelle de l'application corrèle les fonctions internes des unités et leurs interfaces avec la conception architecturale du système.



5 FONCTION DE TRAÇABILITÉ

La fonction de traçabilité dans les outils de LDRA présente une conception détaillée liée en amont aux exigences logicielles et en aval aux unités logicielles.



Exigences du système | **Exigences de haut niveau pour le logiciel** | **Exigences de faible niveau pour le logiciel** | **Code source**

fonctionnelles plus génériques. Encore une fois, ces tests peuvent initialement utiliser un environnement simulé, mais ISO 26262 impose alors l'analyse des différences entre le code source et le code objet et entre l'environnement de test et celui de la cible afin de spécifier des tests supplémentaires pour une utilisation finale dans l'environnement cible. Dans tous les cas, les tests sur le matériel cible doivent être validés avant la certification.

Lorsque les tests échouent, il est probable que le code devra être révisé. De même, les exigences peuvent changer au cours du développement d'un projet. Dans les deux cas, toutes les unités affectées doivent être identifiées et tous les tests d'unité et d'intégration associés doivent être ré-exécutés. Heureusement, de tels tests de non-régression peuvent être automatisés et systématiquement appliqués pour s'assurer que toute nouvelle fonctionnalité n'affecte pas les actions déjà implémentées et prouvées.

Cette attention constante à la représentation fidèle des exigences par le code est particulièrement pertinente pour le test d'intégration et les tests unitaires. Les entrées et sorties attendues de ces tests découlent des exigences, tout comme les tests par défaut et les tests de robustesse (figure 3). Lorsque les unités sont intégrées dans le contexte, les mêmes données de test peuvent être réutilisées.

Les tests unitaires et d'intégration avec analyse dynamique garantissent alors que le logiciel fonctionne correctement, à la fois comme unité et « qu'il s'accorde bien avec les autres » lorsqu'il est connecté à d'autres unités dans le programme global. Cependant, il est également nécessaire d'évaluer l'exhaustivité de ces tests ainsi que de s'assurer qu'il n'y a pas de fonctionnalité parasite. Les tests d'analyse fonctionnelle et de la couverture des appels vérifient que tous les appels ont été effectués et toutes les fonctions ont été appelées. Il est cependant nécessaire d'examiner plus en profondeur la structure en exécutant la déclaration et la couverture de chacun des branchements, ce qui garantit que chaque instruction a été exécutée au moins une fois et que chaque branche possible à chaque point de décision est prise en compte au moins une fois.

Lorsqu'il y a plusieurs conditions à prendre en considération à un point de décision, le nombre de combinaisons possibles peut rapidement conduire à une situation où le test de chacun d'entre eux n'est pas pratique.

Dans ce cadre Modified Condition/Decision Coverage (MC/DC) est une technique qui réduit le nombre de cas de test requis dans de telles circonstances, appelant uniquement à tester que chaque condition peut affecter indépendamment le résultat (figure 4). L'analyse de la couverture au niveau de l'unité vérifiera ici les conditions au sein de cette unité mais évidemment ne fera pas appel à l'extérieur de cette unité.

Afin de mettre tout cela dans le contexte, les fonctions des unités logicielles sont déterminées par la conception architecturale du logiciel, qui est à son tour déterminée par les exigences. Les exigences et l'architecture, qui définissent les unités, établissent également les tests requis par chacune de ces unités. A leur tour, lorsque les unités sont intégrées, elles sont testées pour vérifier leur interaction fonctionnelle ainsi que leur conformité à la conception architecturale du logiciel et – pour la norme ISO 26262 – aux exigences fonctionnelles et de sécurité.

Les exigences en haut du modèle en V (voir figure 1 en page 3) sont souvent définies à l'aide d'outils de contraintes spécialisés tels que l'outil Rational DOORS d'IBM, ou des outils de modélisation tels que Simulink de MathWorks. Avoir une suite d'outils logiciels qui s'interface avec de tels outils peut être un avantage pour vérifier la traçabilité bidirectionnelle requise par la norme ISO 26262, même lorsque les outils de modélisation sont utilisés pour générer automatiquement le code source (figure 5). Ces unités

générées automatiquement sont soumises aux mêmes procédures rigoureuses de test, de vérification et d'intégration que les unités codées à la main, le code hérité et le code source ouvert.

Vers des outils automatisés de test et de vérification pour l'ISO 26262

Il est utile de considérer le développement comme un processus étape par étape, avec un test qui intervient après le codage. Cependant, les tests réguliers lors du développement, complétés par un suivi des exigences bidirectionnelles, sont essentiels car plus un défaut apparaît tardivement, plus le coût en temps et financier est élevé.

Avec toutes ces actions simultanées, le maintien d'une gestion actualisée du projet et de son état de traçabilité par des moyens traditionnels est un « cauchemar » logistique. Par exemple, la cause possible d'une faute au niveau du test d'intégration pourrait être une contradiction dans les exigences, ce qui est beaucoup plus facile à traiter si on le détecte très tôt. Si elle intervient plus tard et implique que les exigences doivent être modifiées, cela aura des répercussions inévitables sur tout le projet. Quelles autres parties du logiciel sont affectées ? Jusqu'où faut-il remonter pour modifier et tester pour savoir si le changement est couvert ? Un scénario similaire correspond à une erreur de codage détectée tard dans la journée. Quelles autres unités dépendent de ce code ? Que faire s'il y a une spécification incorrecte dans l'une des exigences mais que les tests unitaires ont déjà été exécutés et deviennent donc pour le moins suspects ? Comment déterminer la meilleure manière de s'assurer que tout a été corrigé ?

Dans de telles situations, le traçage des exigences manuelles deviendra impossible à partir d'un certain niveau. Au mieux, cela laissera toujours un sentiment d'incertitude. Selon la manière de recenser ces besoins, quelle que soit l'approche de conception adoptée – code généré par le modèle ou à la main –, des outils de test et de vérification automatisés peuvent faire plus que simplement donner le statut d'une étape de développement particulière. Une suite d'outils intégrée peut assurer la traçabilité entre les exigences, la conception et le code source – des fonctions du plus bas niveau, avec leurs résultats de test, jusqu'aux exigences spécifiées. In fine, rester en accord avec la norme ISO 26262 quand on veut concrétiser une excellente idée en un système fiable et sécurisé nécessite une attention constante avec un sens du détail que seuls les outils automatisés peuvent fournir. ■