Règlement sur les dispositifs médicaux (MDR)

Premiers secours pour vieux code

En plus de la norme internationale CEI/EN 62304 (Logiciels de dispositifs médicaux – Processus du cycle de vie du logiciel) traitant des activités nécessaires pour assurer la fiabilité et la sécurité des logiciels de dispositifs médicaux ; un nouveau règlement sur le « medical device software (MDR) », valable depuis mai 2021, est entré en vigueur dans l'Union européenne. Ce n'est pas la première fois que l'assurance qualité des logiciels de dispositifs médicaux est mise en avant. La norme CEI/EN 62304, le MDR et d'autres normes stipulent que le fabricant doit garantir l'assurance qualité tout au long du cycle de vie d'un produit. Dans le cas des systèmes actuels, cela ne pose généralement guère de problème. La situation est tout autre lorsque de nouvelles fonctions doivent être ajoutées à des appareils plus anciens : souvent la documentation est insuffisant ou même inexistante et plus personne ne connaît le code.

Bien que les normes IEC 62304 et MDR ne concernent que le secteur médical, le problème est néanmoins connu dans d'autres industries également.

Dans ce cas, des outils tels que Imagix 4D, qui analysent la structure d'un programme et aident ainsi les développeurs, peuvent être utiles.

Par Klaus Lambertz, directeur général de Verifysoft Technology GmbH

La transformation numérique est également de plus en plus présente dans la technologie médicale. Presque tous les nouveaux appareils sont dotés de logiciels, de connexions sans fil et de la capacité de lire les données de capteurs. Cela crée de nouvelles opportunités importantes en matière de thérapie et de diagnostic, mais aussi de nouveaux risques. Les scénarios dans lesquels des pirates informatiques attaquent des dispositifs médicaux n'appartiennent plus à la catégorie de la science-fiction mais deviennent réels. Il est donc essentiel de minimiser les risques pour les patients et le personnel médical, en assurant la meilleure qualité possible des appareils. C'est l'objectif des nombreuses normes pertinentes en matière de technologie médicale. Avec le règlement européen 2017/745 relatif aux dispositifs médicaux (MDR), les exigences en matière d'assurance qualité deviennent encore plus centrales.

Les normes IEC 62304, MDR ou ISO 14971 exigent à l'unisson, mais généralement sans aide concrète, qu'un fabricant mette en œuvre des processus d'assurance qualité et de gestion des risques. Dans le cas des systèmes embarqués et des logiciels autonomes, il faut distinguer deux domaines dans le cadre de l'assurance qualité. D'une part, lors du développement, l'objectif est d'éviter les erreurs de code (vérification) et de garantir les fonctionnalités requises (validation). D'autre part, les systèmes doivent être considérés tout au long de leur cycle de vie. Les produits qui reposent essentiellement sur des logiciels peuvent être soumis à des modifications importantes au cours de leur évolution, par exemple lors de mises à jour ou de l'ajout de nouvelles fonctions. Des changements sont également effectués lorsqu'une bibliothèque utilisée pendant le développement est remplacée par une version plus récente. Les normes et règlements pertinents tiennent

compte de ces deux aspects. Par exemple, le MDR indique clairement : " Pour les produits dont les composants comprennent des logiciels, ou pour les produits sous forme de logiciels, les logiciels doivent être développés et fabriqués conformément à l'état de l'art, en tenant compte des principes du cycle de vie des logiciels, de la gestion des risques, y compris la sécurité des informations, de la vérification et de la validation."

Les problèmes viennent avec l'âge

Les nouveaux produits ne posent en général aucun problème en matière d'assurance qualité tout au long de leur cycle de vie, si des processus appropriés ont été ancrés dans l'entreprise. En particulier avec les méthodes de développement agiles actuelles telles que l'intégration continue/le déploiement continu, la documentation joue un rôle majeur. Le principe du "clean code", c'est-à-dire d'un code propre et débarrassé de toutes les circonvolutions superflues, est également devenu une composante importante de nombreux départements de développement. Un élément de ce principe est le refactoring, qui vise à améliorer le code. Le code n'est pas parfait dès le départ, toutes les parties doivent être soumises à des révisions permanentes. La tâche du refactoring consiste à amener le code dans une forme souhaitable pour les développeurs, c'est-à-dire facilement compréhensible. Le refactoring a deux objectifs principaux : Rendre l'extensibilité du code aussi facile que possible et en même temps assurer la maintenabilité. En outre, il faut parvenir à ce que le code puisse être réutilisé en totalité ou en partie dans des projets ultérieurs. Contrairement au débogage, cependant, le refactoring n'affecte pas le comportement du programme. Le code n'est pas modifié sur le plan fonctionnel. À proprement parler, les erreurs ou les problèmes de sécurité découverts lors du refactoring ne sont pas éliminés, mais seulement marqués pour être nettoyés.

Si le remaniement est effectué en tant que mesure d'intégration dans le processus de développement continu, l'effort est gérable. Toutefois, ce n'est pas le cas avec les systèmes plus anciens : Plus longtemps une application est utilisée de manière opérationnelle sans optimisation du code, plus ce code devient difficile à comprendre. En effet, au cours du cycle de vie d'une application, des modifications et des adaptations doivent être apportées encore et encore, ce qui influence le comportement du logiciel. L'effort requis pour la maintenance et la modernisation augmente rapidement à mesure que la connaissance de l'architecture et des fonctionnalités s'amenuise.

Le cas extrême est le code hérité : le code hérité est un défi en termes de maintenance ou d'amélioration. La plupart du temps, le code est extrêmement confus. Souvent, la documentation la plus élémentaire fait défaut. Et les développeurs responsables à l'époque sont à la retraite ou dispersés aux quatre vents. Pourtant, le logiciel doit être enrichi de nouvelles fonctionnalités actualisées. Souvent, il faut éliminer des erreurs qui n'ont pas été découvertes jusqu'à présent. Pour les développeurs chargés de cette tâche, commence alors une recherche de traces dans l'ancienne structure, qui exige alors des qualités archéologiques.

Le remaniement du code existant nécessite une préparation

Afin de préparer l'ancien code de telle sorte qu'un remaniement planifié puisse être effectué avec un effort raisonnable, les aspects suivants du logiciel doivent d'abord être examinés :

- Les fichiers : Dans le contexte du C, les fichiers sont soit des en-têtes, soit des fichiers compilables, c'est-à-dire les composants physiques du logiciel. Il est important de connaître les relations qu'ils entretiennent entre eux - par exemple, quels sont leurs en-têtes communs ?
- Les sous-systèmes : Quels sont les sous-systèmes et quelles relations entretiennent-ils entre eux ? Quelle architecture sous-tend les sous-systèmes ?
- Types de données : Les types sont généralement des pointeurs, des enums (types énumérés), des classes, des structs et autres. Ici, les relations entre les types et les variables sont d'un intérêt particulier.
- Les fonctions : La hiérarchie d'appel des fonctions dans un projet est d'une importance élémentaire pour comprendre le code. Les appels entrants et sortants doivent être pris en compte. Le flux de contrôle entre les fonctions est également pertinent, à savoir à quel moment un saut est effectué vers une autre fonction. De même, les branches et les boucles du programme doivent être connues car elles influencent le flux de contrôle.

Ces analyses ne peuvent pas être effectuées manuellement au-delà d'une certaine complexité du projet. La simple identification des relations entre les différents fichiers d'un projet est une tâche sujette aux erreurs. L'utilisation d'outils adaptés est inévitable pour effectuer des analyses qui peuvent être automatisées dans la mesure du possible. Une aidel efficace pour l'analyse du code source en C/C++ et Java, est Imagix 4D, un outil développé par Imagix Corp. USA et distribué en Europe par Verifysoft Technology.

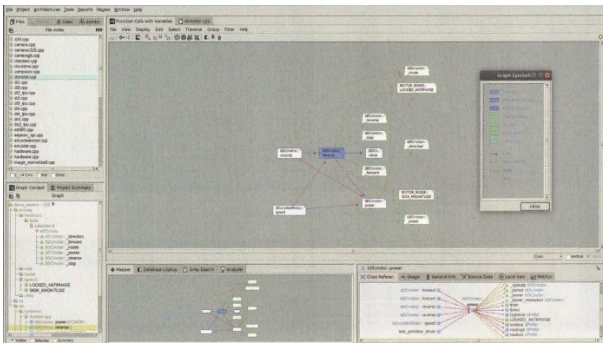

Fig.1: Les diagrammes d'appel de fonction montrent la séquence des fonctions appelées et des informations complémentaires.

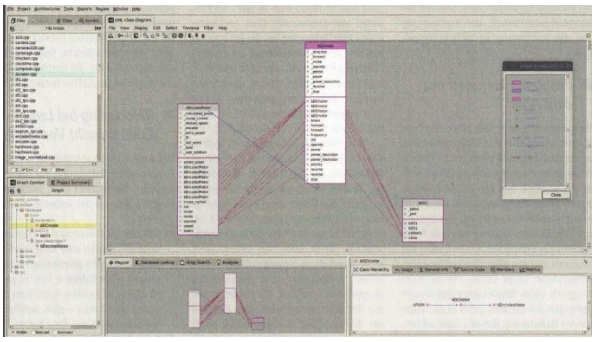

Fig.2: Le diagramme de classe UML exprime la propriétés des classes et les relations entre elles en notation UML .

Préparation graphique de la structure

Imagix 4D analyse le code source d'un logiciel et prépare graphiquement les informations pertinentes pour le refactoring. Les développeurs disposent ainsi d'une représentation de l'ensemble du projet, montrant toutes les relations avec le niveau de détail requis. En fonction de la question posée, l'outil dispose de différents modes d'affichage. Pour fournir un aperçu des dépendances de tous les sous-systèmes présents dans un projet, les informations sont préparées sous la forme d'une matrice de structure de conception. Cela permet, par exemple, de décomposer la granularité des sous-systèmes depuis le répertoire racine jusqu'au niveau des fonctions individuelles. Pour une meilleure compréhension de l'architecture des sous-systèmes, celle-ci peut à son tour être affichée sous forme de diagramme. Dans le cas d'architectures peu claires, par exemple avec un grand nombre de fichiers directement dans le répertoire racine, des filtres aident à trouver le bon centre d'intérêt. De nombreuses autres vues, par exemple pour afficher les dépendances de fonctions ou les flux de contrôle, fournissent aux développeurs des informations plus détaillées. Une autre fonctionnalité importante est la recherche d'anomalies dans le code pour augmenter spécifiquement la qualité de l'application. Il s'agit notamment des récursions, des blocages, des variables inutilisées ou des conversions de type inappropriées.

Grâce à ces connaissances, il est possible de comprendre le code existant et de retracer sa fonctionnalité. Dans l'étape suivante, le code peut alors être nettoyé et mis sous une forme cohérente dans le cadre d'un refactoring. En outre, l'utilisation d'Imagix 4D permet de créer une documentation complète avec un effort raisonnable, indispensable pour toute certification éventuelle. Sur cette base, l'application peut ensuite être dotée de nouvelles fonctions. En outre, la base est préparée pour continuer à exploiter l'ancien projet avec les approches actuelles du développement agile.

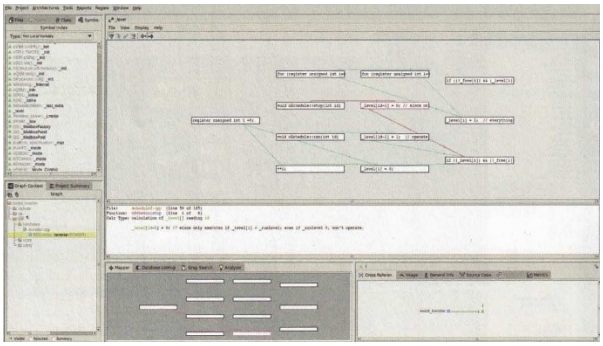

Fig. 3 : L'arbre de calcul d'une variante montre quelles valeurs et autres variables contribuent à une variable et quelles autres variables sont affectées

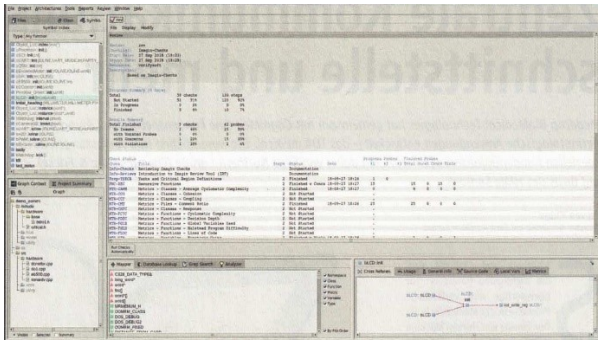

Fig. 4: Grâce à la fonction de révision, Imagix 4D prend en charge des processus autrement purement manuels comme un outil central semi-automatique

Conclusion

L'assurance qualité tout au long du cycle de vie n'est pas nouvelle dans le domaine de la technologie médicale, mais elle prend de l'ampleur en raison des récents développements technologiques et du MDR. Pour assurer la qualité, même pour les produits à longue durée de vie ou les produits largement utilisés sur le terrain, il faut d'abord connaître le code - un véritable problème dans de nombreux cas. Ce n'est pas pour rien que le refactoring s'est imposé comme une partie intégrante des méthodes de développement agiles. Les applications patrimoniales en bénéficient également - surtout si elles n'ont pas encore atteint la fin de leur cycle de vie. Pour cela, toutefois, le code source existant doit être analysé en détail, car le refactoring ne doit en aucun cas modifier le comportement du logiciel. Les approches par essai-erreur n'ont pas leur place ici.

Sans outils appropriés, l'analyse d'applications complexes est difficilement réalisable, et les erreurs ne peuvent être exclues au prix d'un effort économiquement justifiable. Une présentation graphique des architectures et des structures sous-jacentes permet aux développeurs de bien comprendre comment une application est structurée et où se trouvent les bons points d'entrée pour le remaniement. Cela signifie que même les anciens systèmes peuvent être amenés à un niveau permettant de répondre aux exigences de qualité et de gestion des risques. Tout le monde en profite : une plus grande sécurité pour les patients et une plus longue durée de vie des produits pour les fabricants.

De plus amples informations sont disponibles à l'adresse suivante

https://www.verifysoft.com/fr_imagix4d.html