



Projet AMC+ / MK1+ : réalisation de la partition embarquée Vehicle Management System (VMS) de l'hélicoptère COUGAR dans le cadre de sa certification

Lionel Sassolas

► To cite this version:

Lionel Sassolas. Projet AMC+ / MK1+ : réalisation de la partition embarquée Vehicle Management System (VMS) de l'hélicoptère COUGAR dans le cadre de sa certification. Informatique [cs]. 2013. dumas-01160176

HAL Id: dumas-01160176

<https://dumas.ccsd.cnrs.fr/dumas-01160176>

Submitted on 4 Jun 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Mémoire
présenté en vue d'obtenir le
DIPLÔME D'INGÉNIEUR C.N.A.M.

Par Lionel SASSOLAS
12 Juillet 2013

PROJET AMC+/MK1+

**Réalisation de la partition embarquée
Vehicle Management System (VMS)
de l'hélicoptère COUGAR dans le cadre de
sa certification**

Membres du jury :

M. Yves LALOUM (Président du jury, CNAM)
M. Bastien PESCE (Professeur, CNAM)
M. Noël QUESSADA (Tuteur, CNAM)
M. Frederic WILHELM (Collaborateur, STERIA)

Contenu

| | |
|---|----|
| Remerciements | 4 |
| Glossaire | 6 |
| Entreprise et rôle dans l'entreprise | 8 |
| Présentation de l'entreprise | 8 |
| STERIA, une référence dans le monde des SSII | 8 |
| Evolution du chiffre d'affaire | 8 |
| Effectifs | 9 |
| Mon rôle au sein de cette entreprise | 10 |
| Présentation Client | 10 |
| Présentation Equipe | 10 |
| Projet | 11 |
| Contexte | 11 |
| Planification | 11 |
| Architecture AMC | 12 |
| Enjeu | 14 |
| Norme DO-178B/ED-12B | 15 |
| But | 15 |
| Catégories des conditions de panne | 16 |
| Définition des niveaux logiciels | 17 |
| Objectifs à atteindre | 18 |
| Documents à fournir | 19 |
| Certification MK1+ | 21 |
| Etude du projet AMC+ | 25 |
| Recensement des problèmes | 25 |
| Propositions | 27 |
| Evaluation du travail et répartition des tâches | 28 |
| Evaluation de la complexité des modifications | 28 |
| Planification et répartition des tâches | 28 |
| Tableau de suivi | 29 |
| Champs du tableau | 29 |
| Exemple | 30 |
| Gestion de configuration | 31 |
| Développement | 33 |
| SCADE | 33 |
| Exemples | 34 |
| Problèmes et solutions | 35 |
| Simulation | 35 |
| Traçabilité | 35 |
| Commentaires | 36 |
| Règles de codage | 36 |
| Fiche de relecture fonctionnelle SCADE (Annexe 1) | 37 |
| Fiche de relecture formelle SCADE (Annexe 3) | 39 |
| Fiche de relecture de code manuel (Annexe 4) | 40 |
| Tests unitaires | 42 |
| IBM Rational Test RealTime (RT/RT) | 42 |
| Exemples | 43 |
| Fiche de relecture Plan de Tests Unitaires (Annexe 2) | 44 |
| Tests d'intégration logiciel | 46 |
| Méthode d'intégration de l'AMC+ | 46 |
| Exemple ARTIST | 47 |
| Nouvelle approche | 48 |
| Vue d'ensemble | 48 |
| Description fiche de test Excel | 49 |
| Exemples | 50 |
| Gestion des problèmes | 51 |
| Problem Report | 51 |
| Origine des PR | 51 |
| Caractéristiques d'un PR | 52 |

| | |
|---|----|
| Cycle de vie d'un PR | 54 |
| Software Change Note | 56 |
| Caractéristiques d'une SWCN | 56 |
| Cycle de vie d'une SWCN | 57 |
| Rôle de chaque entité | 58 |
| Cas particulier | 58 |
| Livraison | 59 |
| Conclusion sur le projet | 60 |
| Références bibliographiques | 61 |
| Annexe 1 : Fiche de relecture fonctionnelle SCADE | 62 |
| Annexe 2 : Fiche de relecture Plan de Tests Unitaires | 63 |
| Annexe 3 : Fiche de relecture formelle SCADE | 64 |
| Annexe 4 : Fiche de relecture de code manuel | 70 |
| Résumé | 79 |
| Mots clés | 79 |
| Abstract | 79 |

Remerciements

En préambule à ce mémoire, je souhaitais adresser mes remerciements les plus sincères aux personnes qui m'ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire.

Je tiens à remercier sincèrement mes professeurs et tuteurs du CNAM, Messieurs Bastien PESCE et Noël QUESSADA, qui se sont montrés à l'écoute et disponibles tout au long de la réalisation de ce mémoire, ainsi que pour l'aide et le temps qu'ils ont bien voulu me consacrer et sans qui ce mémoire n'aurait jamais vu le jour.

Mes remerciements vont également à M. Frédéric WILHELM, qui m'a soutenu pendant ces longs mois et a accepté de représenter notre société Steria lors de ma soutenance.

Enfin, je voulais remercier mon directeur de projet, M. Gérard OLIVER qui m'a fait confiance et m'a positionné sur la mission sur laquelle se base mon mémoire.

Introduction

Après 6 ans dans l'armée de l'air en tant que développeur informatique sur des projets expérimentaux, je suis revenu dans le milieu civil en février 2011. C'est en préparant ma reconversion que j'ai commencé l'enseignement au Conservatoire National des Arts et Métiers. J'ai quitté l'armée en ayant obtenu le diplôme de concepteur architecte informatique, et ainsi pu avoir assez facilement un emploi au sein d'une société de services informatique.

Lors de ma première mission de plus d'un an et demi en tant qu'intégrateur système au sein d'Eurocopter sur leur nouvelle avionique, j'ai décidé de poursuivre mon cursus et de viser le diplôme d'ingénieur.

Pour mon mémoire, la mission dans laquelle j'évoluais ne me permettant pas de détenir un thème intéressant, et ma hiérarchie ne souhaitant pas m'aider à changer de poste, j'ai décidé de changer de société en cherchant un sujet adéquat.

Grâce à mes expériences professionnelles passées, j'ai pu appréhender les différentes étapes d'un projet. Le sujet que j'ai choisi regroupe non seulement ces étapes, mais ajoute aussi un aspect, la qualité. Bien que je connaisse déjà les normes et les processus mis en place dans mes anciens projets, je ne faisais qu'appliquer les directives. Le challenge ici sera de créer les processus pour respecter des normes données, et de s'assurer que les différents intervenants y adhèrent et s'y soumettent. Bien évidemment, j'aurai ensuite à réaliser les phases d'analyse, de conception et de test.

Glossaire

AMC (Aircraft Management Computer) : Le but de l'AMC est de fournir un calculateur de gestion de l'information extensible capable d'être installé sur les types différents d'hélicoptères fabriqués par Eurocopter. Cette capacité est réalisée tant via le matériel que via le logiciel. D'abord le matériel contient toutes les interfaces de signal nécessaires pour supporter une large variété de capteurs d'hélicoptère standardisés. La seconde, le logiciel est reconfigurable et rechargeable permettant à l'AMC d'être adapté aux exigences particulières de chaque hélicoptère.

ARINC (Aeronautical Radio, Incorporated) : Société détenue par les principales compagnies aériennes et des constructeurs aéronautiques américains qui définit les principaux standards de communications à l'intérieur des aéronefs et entre les aéronefs et le sol.

ARINC 429 : Standard le plus courant, de par son utilisation dans tous les avions récents d'Airbus et Boeing. Il fournit la description des fonctions et des interfaces du support physique et électrique du système de transmission de données numériques.

ARINC 653 : Standard de partitionnement temporel et spatial de ressources informatiques. Ce standard définit également des interfaces de programmation qui permettent d'assurer l'indépendance de l'application vis-à-vis du logiciel et du matériel sous-jacent.

EID (Electronic Instrument Display) : Ecran d'affichage du VMS installé dans le cockpit.

EUROCAE (EUROpean Organisation for Civil Aviation Equipment) : Organisme européen qui fédère les acteurs du domaine de l'aviation civile afin d'établir des règles de standardisation des systèmes utilisés par l'aviation civile en Europe.

IRS (Interface Requirements Specification) : Document qui recense les interfaces d'entrées et de sorties d'un logiciel.

Lustre : Lustre est un langage de programmation synchrone, déclaratif, et par flots. Il possède une définition formelle, et est utilisé pour la programmation des systèmes réactifs. Son développement a commencé au début des années 1980, dans le cadre d'un projet de recherche. Il est entré dans le monde industriel en 1993, lorsque la société Esterel Technologies a publié l'environnement commercial SCADE, dont il constitue le cœur. Lustre est désormais utilisé pour la conception de logiciel critique dans l'aéronautique, le ferroviaire et les centrales nucléaires.

PR (Problem Report) : Il sert à décrire un problème. Il est le résultat d'un écart constaté entre le résultat attendu par le testeur et le résultat obtenu lors du test.

SCADE (Safety Critical Application Development Environment) : SCADE est un environnement de développement intégré, destiné à la conception de systèmes critiques. Il est basé sur le langage Lustre, et permet de générer du code en langage C. Il peut être qualifié DO-178B niveau A par ses utilisateurs, ce qui explique sa popularité en aéronautique.

SRS (Software Requirements Specification) : Document qui recense les exigences fonctionnelles d'un logiciel.

SWCN (SoftWare Change Note) : Elle sert à décrire les corrections nécessaires à la résolution d'un PR. Un PR peut avoir plusieurs SWCN, mais une SWCN ne doit être liée qu'à un seul PR. La description doit être claire et précise pour permettre la vérification.

VMS (Vehicle Management System) : La partition VMS installée dans l'AMC gère les données moteur et véhicule, en vérifiant leur validité et leur cohérence.

Entreprise et rôle dans l'entreprise

Présentation de l'entreprise

STERIA, une référence dans le monde des SSII

Société de services informatiques leader en Europe, au service des entreprises qui placent les nouvelles technologies au cœur de leurs stratégies, Steria est focalisé sur la mise en œuvre de partenariats stratégiques avec ses clients, sur chacun de ses marchés clés : les services publics, la finance, les télécommunications, l'énergie, les transports et l'industrie.

Steria propose à ses clients des services intégrés alliant conseil pour leurs processus métiers, mais aussi développement et exploitation de leurs systèmes d'information. En octobre 2007, Steria a racheté l'entreprise britannique Xansa.

Le nouveau groupe compte plus de 18 000 collaborateurs dans 16 pays. Au 31 décembre 2006, le chiffre d'affaires pro forma de Steria s'élevait à 1,8 milliard d'euros. Le Groupe, dont le siège est implanté à Paris, est coté sur Euronext Paris, Eurolist.

Evolution du chiffre d'affaire

En 2007, Steria a récolté le fruit de quatre années de transformation : initiée en 2002 par la reprise des activités de services de Bull en Europe, cette transformation s'est poursuivie avec le rachat de Mummert Consulting en Allemagne, début 2005.

Elle s'exprime aujourd'hui par des résultats tangibles qui confortent la pertinence de la stratégie de développement du Groupe en Europe.

Fin 2007, Steria fait l'acquisition de Xansa et confirme ainsi sa position de leader des services informatiques au Royaume Uni.

Sa position dans le top 10 en Europe se voit renforcée, ainsi que son positionnement dans le BPO et l'offshore.

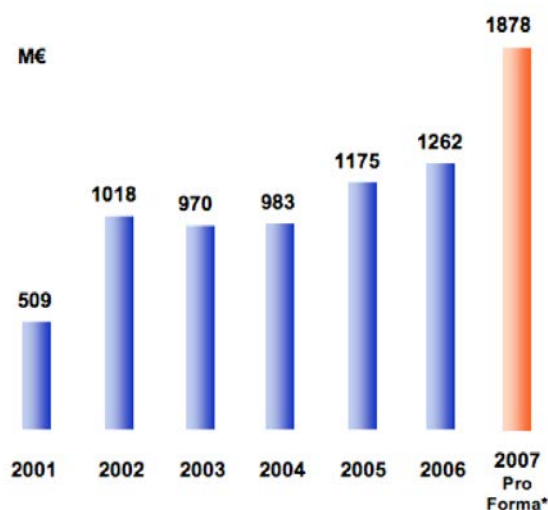


Figure 1 - Chiffre d'affaire

Effectifs

Steria, une des dix premières SSII européennes, reste néanmoins une entreprise à taille humaine. La culture d'entreprise de Steria, fondée sur la valeur humaine, encourage les collaborateurs dans leurs ambitions et leur volonté de progresser.

Par ailleurs, Steria reste également une société pionnière dans le domaine de l'actionnariat salarié, avec 15% du capital, détenu par ses salariés. L'effectif global s'élève désormais à 18700 collaborateurs, dont 27% en offshore.

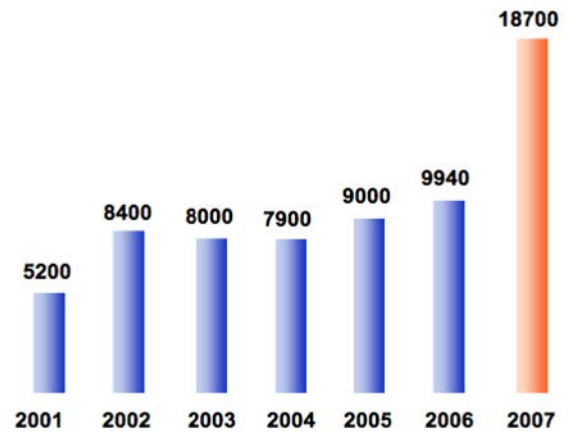


Figure 2 - Effectifs

Mon rôle au sein de cette entreprise

Je travaille en tant que prestataire de service de l'entreprise Eurocopter. Je suis tous les jours dans ses locaux, utilise ses outils et respecte ses normes. Le projet sur lequel porte mon mémoire est lui aussi intimement lié à son corps de métier : le développement d'un calculateur embarqué.

Présentation Client

Eurocopter est le premier exportateur mondial d'hélicoptères, premier constructeur mondial dans le domaine des hélicoptères civils, et deuxième constructeur mondial dans le domaine des hélicoptères militaires.

Depuis 2000, Eurocopter est membre du groupe EADS, l'un des leaders mondiaux sur le marché de l'aérospatiale. En 2011, le chiffre d'affaire annuel d'Eurocopter dépasse les 5,4 milliards d'euros pour environ cinq cents avions livrés. Ce chiffre représente les trois domaines principaux de l'entreprise : le civil, le parapublic (Santé ou Gendarmerie, par exemple) et le militaire.

Eurocopter, c'est également plus de treize mille employés répartis entre la France et l'Allemagne. Ce chiffre ne couvre pas les différentes filiales du groupe présentes dans près de trente pays, de l'Espagne au Japon, des Etats-Unis à l'Australie. Il faut donc rajouter près de cinq mille collaborateurs.

Eurocopter est le leader incontesté sur le marché civil et parapublic, et détient une part solide sur le marché militaire.

Présentation Equipe

J'ai intégré l'équipe Steria présente sur le site d'Eurocopter à l'automne 2012. Sous l'autorité d'un chef de projet, elle est composée alors de cinq ingénieurs, avec des spécialisations en développement ou en test. Je suis le sixième, et je constate rapidement que le turn-over de l'équipe sur le projet est plus élevé que la moyenne.

Projet

Contexte

En mars 2010, l'équipe avec laquelle je travaille actuellement s'est vu confier le projet AMC+. Le calculateur embarqué du super puma qui gère les données moteur et véhicule doit être remplacé. L'AMC fabriqué par General Electric Company laisse sa place à l'AMC fabriqué par Elbit Systems. Le matériel devant donc changer, Eurocopter décide de modifier aussi la partition VMS installée (codée en C) pour la ré-implémenter au nouveau standard en vigueur : SCADE.

Deux ans et demi plus tard, le projet ayant beaucoup de retard, j'ai rejoint l'équipe pour démarrer le projet MK1+ sur l'hélicoptère Cougar en parallèle. Le MK1+ est à l'AMC+ ce que le Cougar est au Super Puma, c'est-à-dire son équivalent militaire.

Planification

Avant mon arrivée, notre client et mon chef de projet se sont mis d'accord sur le découpage en versions intermédiaires et leurs dates de livraison :

| Version | % Développement | % Tests unitaires | Date de livraison |
|---------|-----------------|-------------------|-------------------|
| 0.1 | 20% | 0% | 26/10/2012 |
| 0.2 | 25% | 0% | 16/11/2012 |
| 0.3 | 40% | 10% | 21/12/2012 |
| 0.4 | 45% | 20% | 11/01/2013 |
| 0.5 | 50% | 30% | 01/02/2013 |
| 0.6 | 60% | 50% | 22/02/2013 |
| 0.7 | 75% | 75% | 22/03/2103 |
| 0.8 | 100 % | 75% | 26/04/2013 |
| 1.0 | 100 % | 100% | 31/05/2013 |

Tableau 1 - Plan de livraison

Chaque livraison fera l'objet d'une recette par le client sur le banc d'intégration logiciel, avec le calculateur réel.

Au mois de juin, la V1.0 doit être testée sur le banc d'intégration système par une autre équipe, avant de passer la certification.

Architecture AMC

L'AMC comprend deux parties identiques et indépendantes désignées comme les côtés gauche et droit (côté gauche = AMC1; Côté droit = AMC2).

La partition VMS est hébergée sur la plateforme (PTF) de l'AMC, développée par le constructeur, qui met en œuvre le partitionnement ARINC 653. Cette mise en œuvre garantit l'indépendance de chaque partition et l'impact qu'il pourrait y avoir sur ou à partir d'autres partitions. Le VMS est installé à l'identique dans chacune des deux parties de l'AMC.

L'AMC est donc fonctionnellement divisée en deux parties, les fonctions de bas niveau telles que les acquisitions de signaux sont gérées par la plateforme, et les fonctions de haut niveau sont gérées par la partition VMS. Le but est de séparer les fonctions orientées matériel de bas niveau et les fonctions spécifiques à l'aéronef de niveau supérieur pour rendre plus facile la maintenabilité et l'évolutivité.

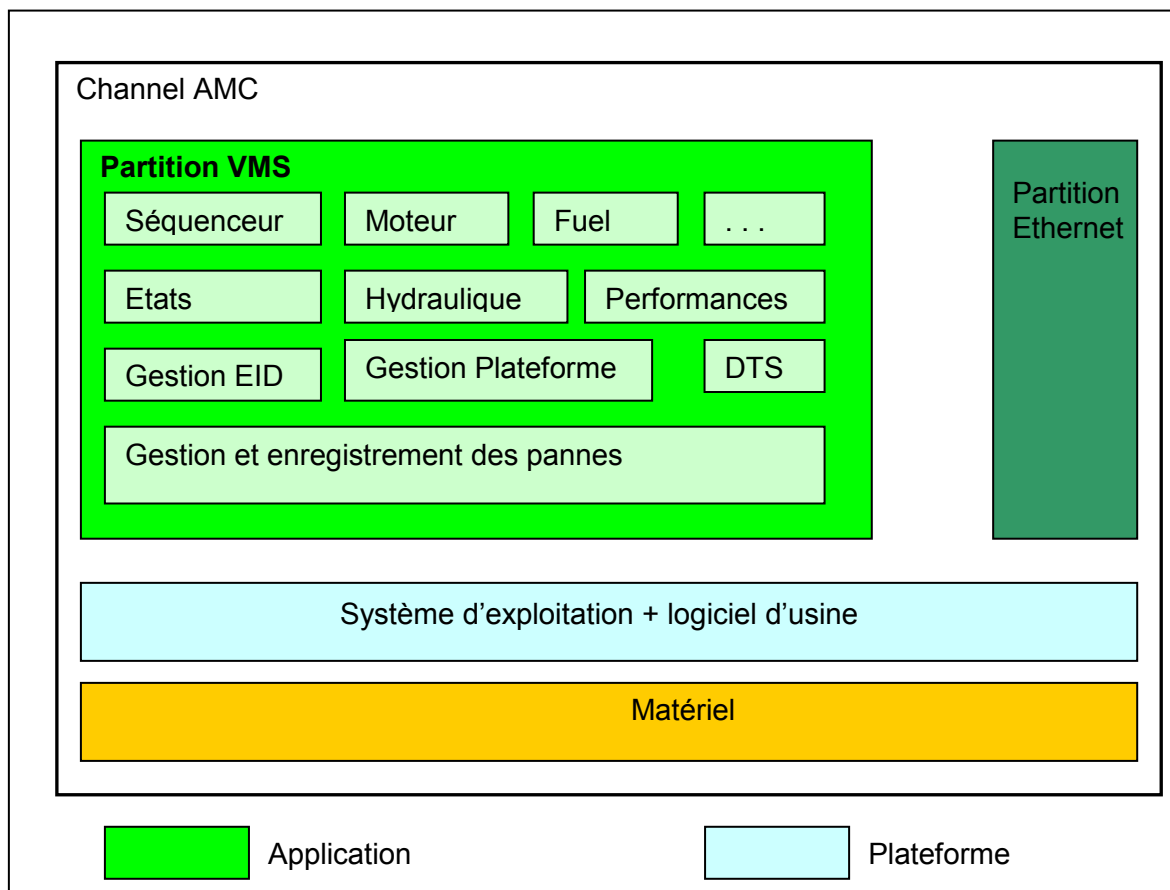


Figure 3 - Architecture AMC

Les fonctions affectées à la plateforme sont l'acquisition du signal, le filtrage, et la vérification (analogique, discrets, ARINC, Ethernet), le transfert des données des signaux à la partition VMS, la transmission des sorties formatées par la partition VMS, et le formatage d'usine.

En ce qui concerne la partition VMS, elle s'occupe de la réception des données des signaux de la plate-forme et de la conversion des données des signaux en paramètres spécifiques aux aéronefs (par exemple, la conversion d'une fréquence de rotation moteur en numérique).

Elle effectue également une vérification croisée (Cross check) des paramètres entre les côtés gauche et droit AMC.

Le traitement des touches, le statut et le mode d'affichage des EID 1 & 2 sont également pris en charge par cette partition.

Elle gère aussi le formatage et la livraison à la plateforme des données ARINC 429.

De plus, elle s'occupe de la gestion du mode hélicoptère, à savoir si l'hélicoptère est en vol, au sol, moteur tournant, tous les moteurs fonctionnels ou pas...

Enfin, l'interface de maintenance au sol ainsi que la gestion et l'enregistrement des pannes est réalisée par notre partition.

Enjeu

L'enjeu du projet est de réaliser la partition VMS du MK1+ sans refaire les mêmes erreurs qui ont conduit au grand retard de l'AMC+. Il faut donc identifier les problèmes rencontrés tout au long du projet, aussi bien au niveau des outils, des méthodes ou des processus, et apporter des solutions.

Le but final est la certification de l'hélicoptère par les autorités compétentes en juillet 2013. Dans ce cadre, j'ai reçu une formation à la norme DO-178B/ED-12B. Celle-ci fixe les conditions de sécurité applicables aux logiciels critiques de l'aviation. Elles précisent notamment les contraintes de développement liées à l'obtention de la certification d'un logiciel d'avionique. Tout mon travail devra donc respecter ces consignes.

Norme DO-178B/ED-12B

But

L'accroissement rapide de l'utilisation des logiciels dans les systèmes et équipements de bord utilisés sur les aéronefs et les moteurs s'est traduit au début des années 1980 par un besoin de directives acceptées par l'industrie pour satisfaire les exigences de navigabilité. Le document DO-178B (DOcument) en version anglaise, ou son équivalent, le document ED-12B (EUROCAE Document) en version française a été rédigé pour répondre à ce besoin.

Ces documents ont donc pour but de fournir des recommandations pour la réalisation de logiciels destinés aux systèmes et équipements de bord, logiciels qui doivent remplir une fonction définie avec, en ce qui concerne la sécurité, un degré de confiance en accord avec les exigences de navigabilité.

Ces recommandations se présentent sous la forme d'objectifs pour les processus de cycle de vie du logiciel, de descriptions des activités et des considérations de conception pour satisfaire ces objectifs, et des descriptions des preuves qui indiquent que les objectifs ont été satisfaits.

Ces documents sont destinés à être utilisés par la communauté aéronautique internationale. Afin d'aider à une telle utilisation, les références à des réglementations et procédures nationales spécifiques ont été réduite au minimum, et remplacé par des termes génériques.

Catégories des conditions de panne

La norme définit différentes catégories de panne selon leur criticité :

- Catastrophique : Conditions de panne susceptibles d'empêcher la poursuite en toute sécurité d'un vol et d'un atterrissage.

- Dangereuse : Conditions de panne susceptibles de réduire les possibilités de l'aéronef ou la capacité de l'équipage à faire face à des conditions hostiles pouvant aller jusqu'à :

1. Une réduction importante des marges de sécurité ou des capacités fonctionnelles.

2. Des problèmes physiques ou un accroissement de charge de travail tels que l'équipage ne soit plus en mesure d'accomplir sa tâche de manière précise ou complète, ou

3. Des effets négatifs sur les occupants tels que des blessures graves ou potentiellement mortelles pour un petit nombre d'entre eux

- Majeure : Conditions de panne susceptibles de réduire les possibilités de l'aéronef ou la capacité de l'équipage à faire face à des conditions hostiles d'une gravité se traduisant, par exemple, par une réduction significative des marges de sécurité ou des capacités fonctionnelles, un accroissement significatif de la charge de travail de l'équipage ou des conditions affectant son efficacité, ou un inconfort pour les occupants, comportant éventuellement des blessures.

- Mineure : Conditions de panne n'engendrant pas de réduction significative de la sécurité de l'aéronef, et susceptibles d'entraîner pour l'équipage des actions se situant tout à fait dans le domaine de leurs capacités. Les conditions de panne mineures peuvent comprendre, par exemple, une légère réduction des marges de sécurité ou des capacités fonctionnelles, une légère augmentation de la charge de travail de l'équipage telle que des modifications ordinaires de plans de vol, ou une certaine gêne pour les occupants.

- Sans effet : Conditions de panne n'affectant pas la capacité opérationnelle de l'aéronef ou n'entraînant pas une augmentation de la charge de travail de l'équipage.

Définition des niveaux logiciels

Cette norme présente 5 niveaux logiciels de criticité (aussi appelés niveaux DAL *Design Assurance Level*), de A à E, définis comme suit :

- Niveau A : Un défaut du système ou sous-système étudié peut provoquer une panne catastrophique : Sécurité du vol ou atterrissage compromis / Crash de l'aéronef.

- Niveau B : Un défaut du système ou sous-système étudié peut provoquer une panne dangereuse entraînant des dégâts sérieux voire la mort de quelques occupants.

- Niveau C : Un défaut du système ou sous-système étudié peut provoquer une panne majeure entraînant un dysfonctionnement des équipements vitaux de l'appareil.

- Niveau D : Un défaut du système ou sous-système étudié peut provoquer une panne mineure pouvant perturber la sécurité du vol.

- Niveau E : Un défaut du système ou sous-système étudié peut provoquer une panne sans effet sur la sécurité du vol. Une fois que l'autorité de certification a confirmé qu'un logiciel étant de niveau E, aucune recommandation de cette norme n'est applicable.

Les Niveaux sont établis par les études de sûreté de fonctionnement. Ces études fixent alors le niveau DAL pour le matériel et le logiciel conformément aux normes de sécurité ou directives de l'avionneur. Le niveau DAL d'un sous-système peut être différent du niveau système à la condition que le niveau DAL du système soit atteint par une architecture matérielle/logicielle adéquate.

Objectifs à atteindre

Plus le niveau de criticité est proche du niveau A, plus le nombre d'objectifs à tenir est élevé. Comme les projets AMC+ et MK1+ sont DAL B, nous allons nous concentrer sur ce niveau. Ainsi pour le niveau B, plusieurs objectifs sont à atteindre.

Le logiciel doit être documenté, la liste de documents à fournir est fixée plus bas. Préalablement au développement, des plans doivent être établis pour fixer les méthodes de développement, de vérification, de gestion de configuration, d'assurance qualité.

Il faut assurer et vérifier la traçabilité entre les spécifications du système, les spécifications de haut niveau du logiciel, et les vérifications. Tout ce qui est spécifié doit être formellement vérifié : la couverture fonctionnelle doit être assurée. Les documents doivent aussi être formellement vérifiés.

Le logiciel doit être géré en configuration, par exemple toutes les évolutions du code source doivent être justifiées.

Un service Qualité indépendant doit assurer le respect de la norme en inspectant les sorties du cycle de vie du logiciel.

Des règles de développement doivent être fixées au préalable (sur les phases de spécification, conception et codage).

Les contraintes de traçabilité et de vérification s'appliquent également aux phases de conception et de codage du logiciel.

Les exigences de bas niveau (ou exigences de conception) doivent être formellement vérifiées.

La couverture structurelle, ou couverture de code, doit être vérifiée et analysée : toutes les instructions du code doivent avoir été exécutées et testées, tous les écarts doivent être justifiés. Ces contraintes amènent souvent à devoir passer des tests unitaires.

Les activités de développement et de vérification doivent être confiées à des équipes indépendantes.

Documents à fournir

Les processus liés au développement de logiciels embarqués aéronautiques soumis à la norme ED-12B / DO-178B s'accompagnent de l'édition de plusieurs documents.

Avant le développement, lors de la phase de planification, il faut rédiger le plan des aspects logiciels relatifs à la certification pour garantir une bonne compréhension et communication entre le postulant et l'autorité de certification.

Le plan de développement logiciel doit aussi être édité. Il a pour objectif de recueillir toute information nécessaire au contrôle du projet. Il décrit l'approche choisie pour développer le logiciel et il est le plan de haut niveau généré et utilisé par les gestionnaires pour diriger l'effort de développement.

Un autre document est le plan de vérification du logiciel. Son but est de planifier les efforts mis en œuvre pour assurer la conformité du logiciel en fonction des besoins du client. Pour s'en assurer, diverses mesures seront mises en œuvre telles que des périodes de rétroaction et de tests, et ce durant les différentes phases de la conception du logiciel. Chacun des modules devra être testé et validé afin de le rendre conforme à ce qui a été demandé.

Il faut aussi réaliser le plan de gestion de configuration du logiciel, qui rassemble l'ensemble des règles et des moyens destinés à gérer et garantir la cohérence de la configuration (i.e. des différents logiciels, sous-ensembles logiciels, modules, composants et documents) à travers les évolutions.

Le plan d'assurance qualité du logiciel doit aussi être fourni. Il sert à décrire l'ensemble des dispositions spécifiques prises pour assurer la qualité du produit fourni dans le cadre d'un projet ainsi que la qualité du processus de développement.

Il faut aussi transmettre les règles de conception du logiciel qui permettent la conception, l'écriture et la mise au point d'un logiciel jusqu'à sa livraison au demandeur.

Des règles de codage doivent également être établies. C'est un ensemble de règles à suivre pour uniformiser les techniques de développement logiciel, diffuser les bonnes pratiques et éviter les erreurs de développement "classiques" au sein d'une équipe.

En ce qui concerne le processus de développement proprement dit, il faut produire les spécifications des exigences du logiciel ainsi que la description de la conception du logiciel.

Le premier document spécifie le besoin en détaillant le plus précisément possible les exigences du client.

Le second document consiste en une description de l'architecture du logiciel, c'est à dire obtenir à la fois une décomposition du système d'information en un ensemble de modules et de structures de données et une description du rôle de chaque module individuellement, et en interaction avec les autres.

Concernant le processus de vérification, la norme requiert la documentation recensant les cas et procédures de vérification du logiciel, ainsi que les résultats de la vérification du logiciel. Ces résultats comprennent les résultats de tests unitaires, les résultats de tests d'intégration, les résultats de tests, l'analyse de la couverture de code et la revue de chaque document du logiciel.

Concernant le processus de gestion de configuration il faut rédiger un Index de la configuration du logiciel et un index de l'environnement du cycle de vie du logiciel.

Enfin, au niveau du processus d'assurance qualité, il faut répertorier tous les enregistrements relatifs à la qualité ainsi que le compte-rendu de revue de conformité.

Certification MK1+

Ma participation dans le cadre de la certification du projet se borne aux processus de développement et de vérification. Les étapes en amont ont déjà été réalisées par le chef de projet et la gestion de configuration est réalisée par l'outil d'Eurocopter certifié (Marcel Impact).

Sur les dix tableaux issus du document officiel qui recensent les différents points de contrôle nécessaires à chaque étape, je n'ai sélectionné que les trois correspondants à mon périmètre. Les deux premiers concernent la conception logicielle et le second la vérification (Tableaux A-4, A-5 et A-7).

Il est nécessaire de préciser que les planches SCADE ne sont pas considérées comme du code, mais comme de la conception (ou des exigences de bas niveau correspondantes aux exigences de haut niveau de la spécification fonctionnelle). Comme ce produit est qualifié DO-178B, le code source généré en C par SCADE l'est aussi. De ce fait, aucune vérification n'est nécessaire sur ce code, et je n'ai pas besoin de prendre en compte les tableaux traitant du code source ou du code exécutable (Tableaux A-5 et A-6).

Néanmoins, comme je le montrerai plus tard, il est parfois nécessaire de recourir à du code manuel à utiliser dans des modèles SCADE. Pour ces cas particuliers, il faut bien sûr vérifier ce code en utilisant le tableau approprié (Tableau A-5). Le code exécutable n'est lui toujours pas à vérifier, c'est la chaîne de production qui le certifie.

Je détaillerai les différents contrôles plus tard dans ce mémoire, lors des phases de relecture des planches SCADE, du code manuel et des tests unitaires.

Réalisation de la partition embarquée
Vehicle Management System (VMS)
de l'hélicoptère COUGAR dans le cadre de sa certification

| | Objective | | Applicability by SW level | | | | Output | | Control category by SW level | | | |
|----|---|--------|---------------------------|---|---|---|-------------------------------|-------|------------------------------|---|---|---|
| | Description | Ref. | A | B | C | D | Description | Ref. | A | B | C | D |
| 1 | Low-level requirements comply with high-level requirements. | 6.3.2a | ● | ● | ○ | | Software Verification Results | 11.14 | ② | ② | ② | |
| 2 | Low-level requirements are accurate and consistent. | 6.3.2b | ● | ● | ○ | | Software Verification Results | 11.14 | ② | ② | ② | |
| 3 | Low-level requirements are compatible with target computer. | 6.3.2c | ○ | ○ | | | Software Verification Results | 11.14 | ② | ② | | |
| 4 | Low-level requirements are verifiable. | 6.3.2d | ○ | ○ | | | Software Verification Results | 11.14 | ② | ② | | |
| 5 | Low-level requirements conform to standards. | 6.3.2e | ○ | ○ | ○ | | Software Verification Results | 11.14 | ② | ② | ② | |
| 6 | Low-level requirements are traceable to high-level requirements. | 6.3.2f | ○ | ○ | ○ | | Software Verification Results | 11.14 | ② | ② | ② | |
| 7 | Algorithms are accurate. | 6.3.2g | ● | ● | ○ | | Software Verification Results | 11.14 | ② | ② | ② | |
| 8 | Software architecture is compatible with high-level requirements. | 6.3.3a | ● | ○ | ○ | | Software Verification Results | 11.14 | ② | ② | ② | |
| 9 | Software architecture consistent. | 6.3.3b | ● | ○ | ○ | | Software Verification Results | 11.14 | ② | ② | ② | |
| 10 | Software architecture is compatible with target computer. | 6.3.3c | ○ | ○ | | | Software Verification Results | 11.14 | ② | ② | | |
| 11 | Software architecture is verifiable. | 6.3.3d | ○ | ○ | | | Software Verification Results | 11.14 | ② | ② | | |
| 12 | Software architecture conforms to standards. | 6.3.3e | ○ | ○ | ○ | | Software Verification Results | 11.14 | ② | ② | ② | |
| 13 | Software partitioning integrity is confirmed. | 6.3.3f | ● | ○ | ○ | ○ | Software Verification Results | 11.14 | ② | ② | ② | ② |

| | | |
|---------------|-------|--|
| LEGEND | ● | The objective should be satisfied with independence. |
| | ○ | The objective should be satisfied. |
| | Blank | Satisfaction of objective is at applicant's discretion. |
| | ① | Data satisfies the objectives of Control Category 1 (CC1). |
| | ② | Data satisfies the objectives of Control Category 2 (CC2). |

Figure 4 - Table A-4 : Contrôle du processus de conception logiciel

| Objective | | | Applicability by SW level | | | | Output | | Control category by SW level | | | |
|-----------|---|--------|---------------------------|---|---|---|-------------------------------|-------|------------------------------|---|---|---|
| | Description | Ref. | A | B | C | D | Description | Ref. | A | B | C | D |
| 1 | Source Code complies with low-level requirements. | 6.3.4a | ● | ● | ○ | | Software Verification Results | 11.14 | ② | ② | ② | |
| 2 | Source Code complies with software architecture. | 6.3.4b | ● | ○ | ○ | | Software Verification Results | 11.14 | ② | ② | ② | |
| 3 | Source Code is verifiable. | 6.3.4c | ○ | ○ | | | Software Verification Results | 11.14 | ② | ② | | |
| 4 | Source Code conforms to standards. | 6.3.4d | ○ | ○ | ○ | | Software Verification Results | 11.14 | ② | ② | ② | |
| 5 | Source Code is traceable to low-level requirements. | 6.3.4e | ○ | ○ | ○ | | Software Verification Results | 11.14 | ② | ② | ② | |
| 6 | Source Code is accurate and consistent. | 6.3.4f | ● | ○ | ○ | | Software Verification Results | 11.14 | ② | ② | ② | |
| 7 | Output of software integration process is complete and correct. | 6.3.5 | ○ | ○ | ○ | | Software Verification Results | 11.14 | ② | ② | ② | |

| | | |
|--------|-------|--|
| LEGEND | ● | The objective should be satisfied with independence. |
| | ○ | The objective should be satisfied. |
| | Blank | Satisfaction of objective is at applicant's discretion. |
| | ① | Data satisfies the objectives of Control Category 1 (CC1). |
| | ② | Data satisfies the objectives of Control Category 2 (CC2). |

Figure 5 - Table A-5 : Contrôle du processus de codage

Réalisation de la partition embarquée
Vehicle Management System (VMS)
de l'hélicoptère COUGAR dans le cadre de sa certification

| Objective | | | Applicability by SW level | | | | Output | | Control category by SW level | | | |
|-----------|---|----------------------|---------------------------|---|---|---|--|-------|------------------------------|---|---|---|
| | Description | Ref. | A | B | C | D | Description | Ref. | A | B | C | D |
| 1 | Test procedures are correct. | 6.3.6b | ● | ○ | ○ | | Software Verification Cases and Procedures | 11.13 | ② | ② | ② | |
| 2 | Test results are correct and discrepancies explained. | 6.3.6c | ● | ○ | ○ | | Software Verification Results | 11.14 | ② | ② | ② | |
| 3 | Test coverage of high-level requirements is achieved. | 6.4.4.1 | ● | ○ | ○ | ○ | Software Verification Results | 11.14 | ② | ② | ② | ② |
| 4 | Test coverage of low-level requirements is achieved. | 6.4.4.1 | ● | ○ | | | Software Verification Results | 11.14 | ② | ② | ② | |
| 5 | Test coverage of software structure (modified condition/decision) is achieved. | 6.4.4.2 | ● | | | | Software Verification Results | 11.14 | ② | | | |
| 6 | Test coverage of software structure (decision coverage) is achieved. | 6.4.4.2a 6.4.4.2b | ● | ● | | | Software Verification Results | 11.14 | ② | ② | | |
| 7 | Test coverage of software structure (statement coverage) is achieved. | 6.4.4.2a 6.4.4.2b | ● | ● | ○ | | Software Verification Results | 11.14 | ② | ② | ② | |
| 8 | Test coverage of software structure (data coupling and control coupling) is achieved. | 6.4.4.2c | ● | ● | ○ | | Software Verification Results | 11.14 | ② | ② | ② | |

| | | |
|--------|-------|--|
| LEGEND | ● | The objective should be satisfied with independence. |
| | ○ | The objective should be satisfied. |
| | Blank | Satisfaction of objective is at applicant's discretion. |
| | ① | Data satisfies the objectives of Control Category 1 (CC1). |
| | ② | Data satisfies the objectives of Control Category 2 (CC2). |

Figure 6 - Table A-7 : Contrôle du processus de vérification

Etude du projet AMC+

Afin de ne pas réitérer les erreurs de ce projet, la première étape de ma démarche a été de passer un certain temps à observer les manières de travailler de chacun et à m'entretenir avec l'équipe AMC+. Il en ressort un certain nombre de problèmes ou dysfonctionnements listés ci-dessous.

Recensement des problèmes

Le premier problème qui existe sur ce projet est le turn-over assez important de l'équipe. En effet, seul notre chef de projet est là depuis le début et donc il est celui qui connaît le mieux le projet et qui doit répondre à toutes les questions. Il perd donc beaucoup de temps à former et diriger ses ingénieurs, qu'il pourrait utiliser à meilleur escient. Je ne peux malheureusement rien faire sur ce sujet-là, en espérant que la solution des autres problèmes créera une ambiance sereine sur le projet qui ne donne plus envie de partir rapidement.

Etant nouveau sur le projet, le second problème m'est apparu assez rapidement. En effet, aucune procédure ni aucune fiche n'existe pour prendre en main les outils de développement ou de test qui sont assez complexes : la capitalisation de l'expérience est inexistante.

Le troisième problème est la principale plainte de l'équipe. Notre outil de développement SCADE ne permet pas d'exécuter le code développé. De ce fait, le développeur code à l'aveugle, sans pouvoir vérifier qu'il n'a pas fait d'erreur à part avec une simple relecture. Le test de la partie développée est réalisé souvent plusieurs semaines plus tard par un autre membre de l'équipe, avec un autre outil. Si des problèmes sont découverts, le temps que le développeur perd à se replonger dans son code est du temps qui aurait pu être gagné en amont.

Le quatrième problème est le manque de rigueur ou de connaissance de l'équipe au niveau de la traçabilité. Chaque développement et chaque test sont réalisés à partir d'une exigence écrite dans la spécification. En théorie, chaque développeur ou testeur doit inscrire dans son code ou son test, le ou les exigences associées. En pratique, en utilisant un outil de traçabilité, il n'est pas rare de trouver des exigences orphelines, qui sont censées être couvertes et testées, mais l'on ne sait pas dire où.

Le cinquième problème est le manque de commentaires dans les planches SCADE. Alors que la norme nous impose que chaque modèle doit être vérifiable (c'est à dire testable en mode boîte noire), très peu de commentaires sont inscrits. Il faudrait au moins décrire ce que fait le modèle, ainsi que ses entrées et ses sorties.

Un sixième problème concernant encore SCADE, est qu'il existe des règles de codage à respecter. Or, le document qui recense ces règles n'est pas connu de la plupart des développeurs, et donc un gros point de la norme ne peut pas être respecté.

Le septième problème est la gestion des bugs. Faute de processus écrits, et de fiches d'exemples pour rédiger un 'Problem Report' ou une 'Software Change Note', les membres de l'équipe ne savent pas quoi reporter, et comment corriger un problème proprement. Il arrive même que celui qui découvre un problème, le corrige dans la foulée, sans le tracer nul part. Chose impensable en sachant qu'en respectant la norme DO-178B, il faut que celui qui trouve et reporte le problème ne soit pas le correcteur. Et surtout il faut absolument une trace du problème et de ce qui a été changé dans le code pour étudier les effets de bords potentiels et vérifier la correction.

Un autre problème est la disponibilité du banc d'intégration logiciel, que l'on doit se partager entre plusieurs projets, et plusieurs sociétés sous-traitantes. Non seulement le temps est réduit, mais comme l'AMC+ est prioritaire sur le MK1+, le temps affecté à mon projet est plus que raccourci.

Une partie de ces problèmes vient du fait que la priorité absolue a été donnée au fonctionnel. Il fallait avoir à tout prix un logiciel capable de partir aux essais en vol. La grosse erreur a été de reléguer la certification loin derrière. Pour preuve, aucun processus de vérification n'est démarré et aucune relecture n'est faite.

Propositions

J'expliquerai au cours de ce mémoire mes propositions au cas par cas dans les différentes phases du projet. Par exemple, pour palier au problème de la capitalisation de l'expérience, j'ai créé des fiches à chaque étape, en même temps que je prenais en main les outils. J'en détaillerai certaines le moment venu. En voici tout de même la liste :

- Démarche à suivre pour les nouveaux arrivants : document qui répertorie tous les points administratifs à effectuer en arrivant chez le client Eurocopter et éviter de perdre des semaines en attendant un compte, des droits ou des accès.
- Création/Modification d'une planche SCADE : document qui explique le processus à respecter pour créer ou modifier des modèles SCADE, avec la prise en main de Marcel Impact, et de SCADE.
- Création d'un PTU : document décrivant le processus à respecter pour créer un plan de test unitaire, avec la prise en main du logiciel RT/RT.
- Cycle de vie d'un PR : document expliquant le processus de gestion des bugs, de sa découverte à sa résolution, en prenant en main l'outil Mantis Bug Tracker.
- Création d'un PR : document montrant comment renseigner les champs de la fiche lors de la création d'un problème
- Création d'une SWCN : document qui décrit comment renseigner les champs de la fiche lors de la création d'une Software Change Note.
- Génération de la matrice de traçabilité : document décrivant la manière de réaliser la matrice et de vérifier la couverture fonctionnelle de notre développement.
- Officialisation et livraison d'une version : document qui explique le processus pour livrer une version du projet (générer le code), et l'officialiser (figer la version, et passer à la suivante).
- Création d'un test pour EASIPlayer : document qui montre la démarche pour créer un test automatique, ainsi que la prise en main des logiciels ARTIST et EASIPlayer.
- Chargement d'une version sur un AMC : document décrivant la méthode pour charger notre logiciel sur un calculateur AMC réel, et la prise de main de l'outil NetLoader.

Evaluation du travail et répartition des tâches

La seconde phase du projet a été d'évaluer la quantité de travail à réaliser. Au lieu de recommencer le développement depuis le début, j'ai choisi de m'appuyer sur la version la plus à jour de la partition VMS de l'AMC+. En ayant les spécifications fonctionnelles des deux projets fournies par les architectes, j'ai pu tracer les différences.

Evaluation de la complexité des modifications

J'ai ensuite classé les modifications par catégories de difficulté auxquelles sont associés des délais de réalisation. En plus du développement, j'ai alloué à chaque tâche du temps pour la relecture et les tests unitaires. Par expérience, le temps de vérification (relectures + tests) est égal à deux fois le temps de développement.

| Complexité | Charge de développement | Charge de vérification |
|------------|-------------------------|------------------------|
| Standard | 0 | 0.5 |
| Simple | 0.5 | 1 |
| Moyen | 1 | 2 |
| Complexe | 2 | 4 |
| Hors Norme | 5 | 10 |

Tableau 2 – Complexité des tâches

Planification et répartition des tâches

J'ai enfin classé les 575 exigences de la spécification selon leur complexité.

| Complexité | Nombre | % dev | Charge dev (JH) | Charge test (JH) |
|------------|------------|-------------|-----------------|------------------|
| Standard | 288 | 50,1% | 0 | 144 |
| Simple | 195 | 33,9% | 97,5 | 195 |
| Moyen | 74 | 12,9% | 74 | 148 |
| Complexe | 16 | 2,8% | 32 | 64 |
| Hors Norme | 2 | 0,3% | 10 | 20 |
| | 575 | 100% | 213,5 | 571 |

Tableau 3 – Charge de travail

Avec le temps qui nous est alloué pour la réalisation, il est aisé de constater qu'il faut au moins cinq personnes pour tenir le planning. Nous commençons néanmoins le projet à deux, avec l'assurance du chef de projet que des renforts arriveront en cours de réalisation pour respecter notre engagement.

Tableau de suivi

Pour avoir un indicateur d'avancement, j'ai créé un tableau Excel comme défini ci-après.

Champs du tableau

| Colonnes du tableau | Description |
|---------------------------|---|
| Exigences | Liste des exigences issues de la SRS du MK1+. |
| Évaluation | Evaluation de ma part des différences entre les exigences de l'AMC+ et du MK1+, classée en différentes catégories : <ul style="list-style-type: none"> • Supprimer (l'exigence n'existe pas dans le MK1+) • Nouvelle (l'exigence est nouvelle dans le MK1+) • Strictement identique (l'exigence est la même sur les 2 projets) • Légère différence • Moyenne différence • Grande différence • Hors catégorie |
| Complexité | Complexité des exigences classée en différents critères : <ul style="list-style-type: none"> • Standard • Simple • Normal • Complexe • Hors norme |
| Charge Développement | Charge en jour de l'exigence à développer. |
| Charge Vérification | Charge en jour de l'exigence à vérifier (relectures + tests). |
| Commentaires | Commentaires utiles sur les différences entre exigences. |
| Version | Version cible du développement. |
| Responsable Développement | Personne en charge du développement. |
| Temps passé Développement | Durée en jour du temps passé à développer l'exigence. |
| Nom Nœud SCADE | Nom du nœud ou modèle SCADE sur laquelle l'exigence a été développée. |
| Avancement Développement | Remplir cette colonne par : <ul style="list-style-type: none"> • Terminé lorsque l'exigence a été développée • En cours (Le reste à faire doit alors être renseigné) Laisser la colonne vide si le développement n'est pas commencé. |
| Responsable Test | Personne en charge de tester l'exigence. |
| Temps passé Test | Durée en jour du temps passé à tester l'exigence. |
| Reste à faire Test | Durée en jour du reste à faire à tester de l'exigence. |
| Nom PTU | Nom du nœud de test de l'exigence. |
| Etat Test | Résultat du test à remplir lorsqu'il est terminé (OK/KO). |
| Responsable | Personne en charge de la relecture du modèle SCADE sur |

| | |
|------------------------------|--|
| Relecture SCADÉ | lequel l'exigence est développée. |
| Temps passé Relecture SCADÉ | Durée en jour du temps passé à relire le modèle. |
| Nom fiche de relecture SCADÉ | Nom de la fiche de relecture (Modèle à respecter : SW_Review_NomDuModèle.doc). |
| Etat Relecture SCADÉ | Résultat de la relecture (OK/KO). |
| Responsable Relecture PTU | Personne en charge de la relecture du nœud de test sur lequel l'exigence est testée. |
| Temps passé Relecture PTU | Durée en jour du temps passé à relire le test. |
| Nom fiche de relecture PTU | Nom de la fiche de relecture (Modèle à respecter : PTU_Review_NomDuNoeudTest.doc). |
| Etat Relecture PTU | Résultat de la relecture (OK/KO). |

Tableau 4 – Champs du tableau de suivi

Exemple

| EXIGENCES | Evaluation | Complexité | charge dev-cougar | charge verif-cougar |
|---|-----------------------|------------|-------------------|---------------------|
| <SWRDA-031> Startup program pin failure mode | Strictement identique | Standard | 0 | 0,5 |
| <SWRDA-032> Determination of Flight/Ground state | Presque identique | Simple | 0,5 | 1 |
| <SWRDA-033> Maintenance of time by the AMC | Strictement identique | Standard | 0 | 0,5 |
| <SWRDA-034> Flight number management | Strictement identique | Standard | 0 | 0,5 |
| <SWRDA-035> Relative flight time | Strictement identique | Standard | 0 | 0,5 |
| <SWRDA-036_b> AMC failure management and recording | Complexe | Complexe | 2 | 4 |
| <SWRDA-036> AMC time of operation | Strictement identique | Standard | 0 | 0,5 |
| <SWRDA-037> Test identification | Hors catégorie | Hors Norme | 5 | 10 |
| <SWRDA-038> AMC failure log | Strictement identique | Standard | 0 | 0,5 |
| <SWRDA-039> Contents of the AMC failure log | Moyen | Moyen | 1 | 2 |
| <SWRDA-040> Multiple occurrences of a failure in flight | Strictement identique | Standard | 0 | 0,5 |
| <SWRDA-041> VMS central failure log | Moyen | Moyen | 1 | 2 |

Figure 7 - Tableau de suivi (1)

| Commentaires | Version | Resp. Dév | Temps Passé Dév | Modèle SCADÉ | Avanc. Dév. |
|---|---------|-----------|-----------------|-----------------------------|-------------|
| | 3/0/06 | LSA | | 0 StartupConfigurationCheck | Terminé |
| Suppression de la partie the weight_on_wheels | 3/0/04 | LSA | | 0,5 ManageFlightGroundState | Terminé |
| | 3/0/04 | LSA | | 0 ManageFlightGroundState | Terminé |
| | 3/0/04 | LSA | | 0 ManageFlightGroundState | Terminé |
| | 3/0/04 | LSA | | 0 ManageFlightGroundState | Terminé |
| Suppression des FADES, ACEMB, AVCC | 3/0/07 | ACO | | 1,5 MonitorVmsB | Terminé |
| | 3/0/06 | LSA | | 0 VMS_B_NormalMain | Terminé |
| Suppression des FADES, ACEMB, AVCC et le range diffère 603 au lieu de 605 | 3/0/07 | LSA | | 4 TestIdentifier | Terminé |
| | 3/0/06 | LSA | | 0 LogFailure | Terminé |
| Gestion de la mémoire non volatile différente | 3/0/06 | LSA | | 0,5 LibVmsTypeEC225 | Terminé |
| | 3/0/06 | LSA | | 0 ProcessFaultLog | Terminé |
| Suppression ACMB, AVCC, MFDAU et FADEC | 3/0/07 | ACO | | 1 LogFailure | Terminé |

Figure 8 - Tableau de suivi (2)

Gestion de configuration

Avant de passer à la phase de développement, il me semble opportun d'étudier d'abord la gestion de configuration du projet.

La gestion de configuration des projets développés à Eurocopter est réalisée par l'outil MARCEL (Mécanismes d'Aide à la Réalisation et au Contrôle d'Etats Logiciels) récupéré chez Airbus.

Il peut être utilisé sous Unix ou sous Windows. Sur nos projets, nous utilisons la version Windows, via l'outil IMPACT (Integrated Management Process And Configuration Toolkit).

IMPACT est un plug-in de l'environnement de développement intégré (EDI) Eclipse. Cette interface, plus moderne que celle présente sous Unix, permet de piloter MARCEL depuis Eclipse tout en permettant aux utilisateurs de bénéficier des services fournis nativement par cet EDI.

Il permet naturellement de couvrir l'ensemble du processus des développements avioniques qui sont assurés par l'outil MARCEL.

L'interface modulable contient trois fenêtres importantes :

(1) L'affichage de tous les projets présents sur le serveur MARCEL, avec les différentes versions pour chacun. Nous retrouvons nos versions officialisées et livrées du MK1+ jusqu'à la version 0.07 (en orange), ainsi que la version de développement courante, la version 0.08 (en vert).

(2) L'affichage de tous les objets du projet. Il existe trois grands types d'objets. Les SCADE_MODEL qui sont les planches SCADE du projet, les SCADE_NODE_TEST qui sont les plans de tests unitaires associés à une planche SCADE, et les SCADE_IMPORTED_NODE qui sont du code manuel en langage C utilisables sous SCADE.

(3) L'affichage de l'intérieur des objets, comme un explorateur de fichiers. C'est par cet affichage que l'on a accès à la modification d'une planche SCADE.

Outre la gestion de configuration, MARCEL est aussi un outil de gestion de processus. Par exemple, pour modifier un modèle SCADE, l'outil importe la planche voulue dans un dossier privé et en verrouille l'accès à tous les autres utilisateurs le temps des modifications. Mon document Création/Modification d'une planche SCADE en explique les principes.

Aussi, une livraison de version ne peut avoir lieu que si tous les objets compilent correctement. Mon document Officialisation et livraison d'une version explique pas à pas les étapes à respecter pour mener à bien le processus.

Réalisation de la partition embarquée Vehicle Management System (VMS) de l'hélicoptère COUGAR dans le cadre de sa certification

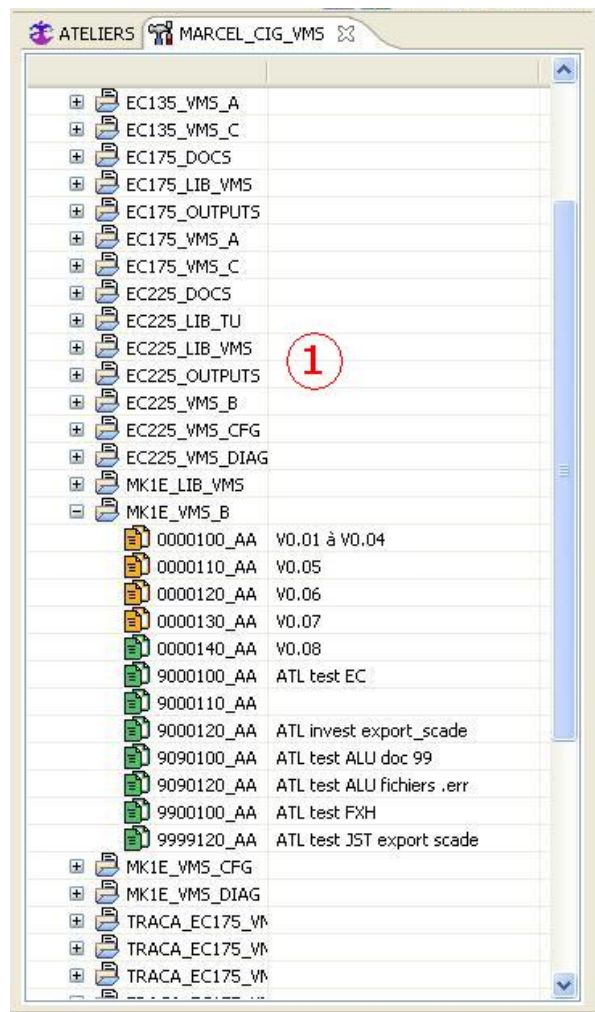


Figure 8 -IMPACT (1)

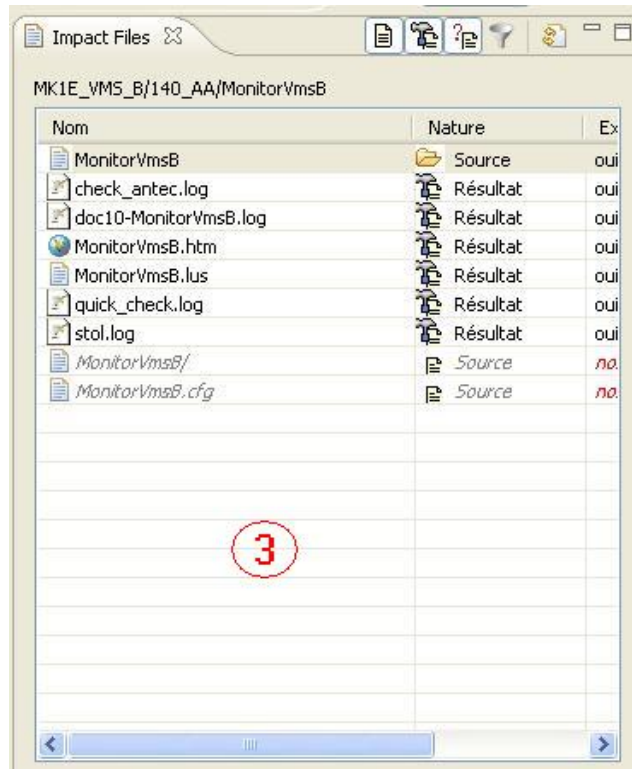


Figure 10 - IMPACT (3)

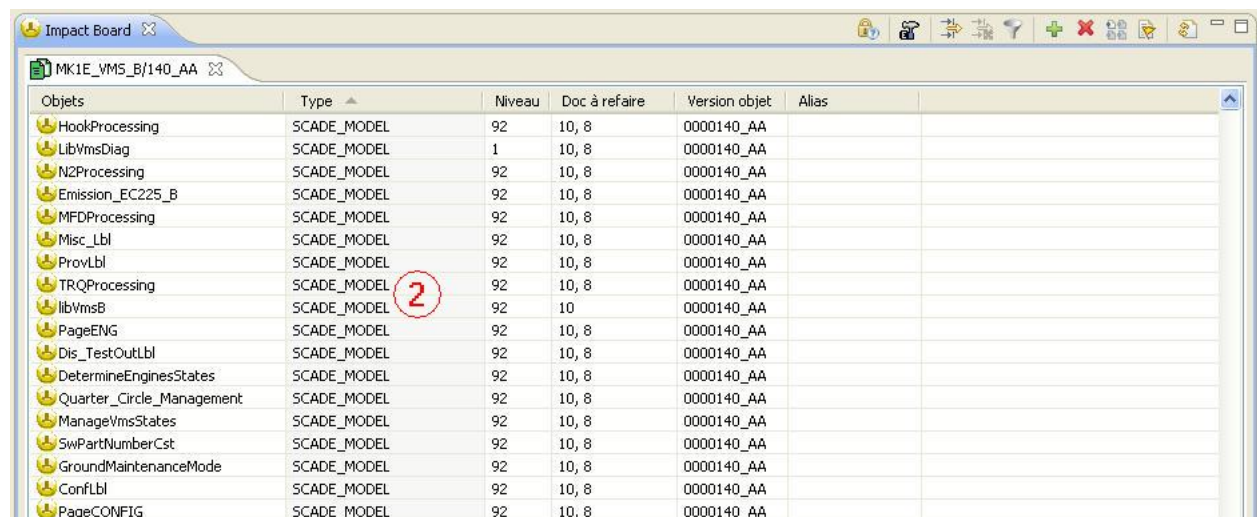


Figure 9 - IMPACT (2)

Développement

SCADE

SCADE (Safety-Critical Application Development Environment) est un environnement de développement pour les systèmes critiques, en particulier dans le domaine de l'embarqué avionique, nucléaire ou automobile. Il est basé sur le langage Lustre qui est un langage de programmation synchrone, déclaratif, et par flots. Lustre possède une définition formelle, et est utilisé pour la programmation des systèmes réactifs.

SCADE fournit l'ensemble des fonctionnalités réclamées par les standards pour le codage d'applications critiques :

- Modélisation de l'application.
- Vérification du modèle.
- Génération de code certifié.

Il propose un langage rigoureux pour saisir les besoins en matière de calcul et de logique :

- Equations de flot de données pour le calcul.
- Machines à états pour la logique.
- La fonction principale de SCADE est appelée de façon cyclique.
- Code développé manuellement pour l'interfaçage avec les couches logicielles de bas niveau.

Il propose un outil de vérification fonctionnelle « SCADE Suite SimulatorTM ». SCADE garantit que les résultats de la simulation du modèle sont identiques à ceux qui seront obtenus plus tard par le code généré à partir du modèle sur la cible.

La phase de codage suivant la conception est entièrement automatisée en utilisant le générateur de code certifié « SCADE SuiteKCGTM ».

Le code produit est :

- Traçable par rapport au modèle fourni en entrée
- Lisible
- Indépendant de la cible
- Comportement déterministe

KCG a été développé avec le même niveau d'exigence que celui attendu pour le code produit :

- Certifié niveau A de DO-178B
- Pas de nouvelle vérification nécessaire au niveau du code

Exemples

Les planches SCADÉ peuvent être très simples, comme sur la figure 11 où l'on vérifie seulement si la variable d'entrée se situe entre les deux valeurs données en paramètres...

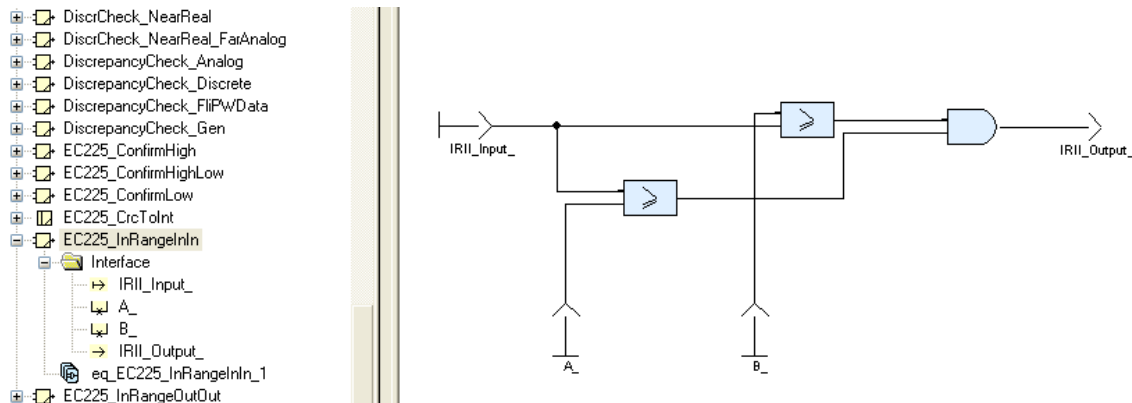


Figure 11 - SCADÉ (1)

...ou un peu plus compliquées, comme sur la figure 12, où l'on doit vérifier selon différents critères si un paramètre moteur devient critique et qu'une alarme doit être levée.

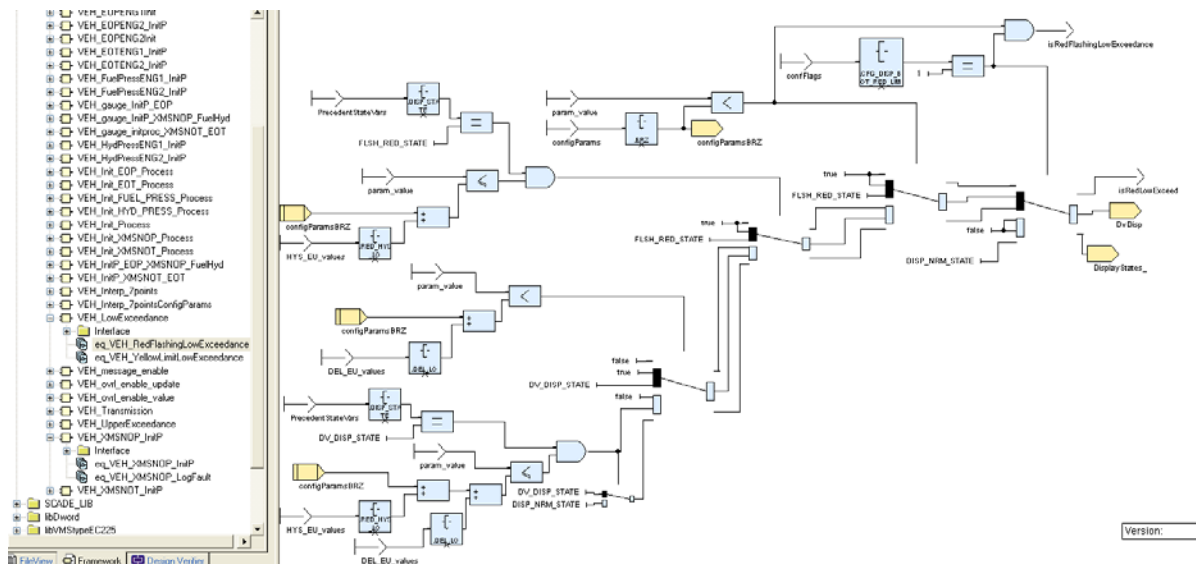


Figure 12 - SCADÉ (2)

Problèmes et solutions

Simulation

Comme énoncé précédemment, le principal problème dont l'équipe AMC+ se plaint se situe dans cette phase de développement. En regardant la figure 12, il est facile de comprendre leur détresse. Sans moyen de simulation pour exécuter ce modèle, il est très difficile de détecter toutes les erreurs potentielles.

L'idéal serait de pouvoir tester sa planche SCADE juste après sa réalisation, alors qu'actuellement seule une relecture peut découvrir des problèmes.

Je me suis mis en relation avec le support informatique MARCEL Impact pour essayer de trouver une solution. Malheureusement, du fait de l'éclatement sur plusieurs serveurs des différentes librairies utilisées sur les projets Eurocopter, il est techniquement très difficile de pouvoir utiliser l'outil de simulation livré dans SCADE.

Ce problème est donc toujours d'actualité. En attendant que le support trouve une solution de leur côté, j'ai demandé que la phase de relecture d'une planche SCADE soit effectuée très rapidement après son développement. De cette manière, si un œil nouveau découvre un problème, le développeur d'origine ne perdra pas trop de temps à se remémorer l'exigence développée.

Traçabilité

Le problème du manque de traçabilité des exigences est un double problème.

Pour les développements à venir, cela a été résolu avec mes fiches de prise en main des outils, de création de modèles SCADE et le processus de cycle du PR, où j'indique clairement qu'il faut indiquer les exigences couvertes lors d'un développement, d'un test ou d'une SWCN, et comment le faire. Ainsi, tout le monde est clairement au courant.

Pour les développements déjà effectués (repris de l'AMC+), il faut trouver où les exigences orphelines sont couvertes et testées. Malheureusement, à part avec des relectures effectuées par les développeurs, je n'ai pas trouvé d'alternatives.

Commentaires

Au niveau des commentaires manquants, là aussi, ma documentation sur la création ou la modification d'une planche SCADE explique comment les renseigner.

Bien que sur une planche simpliste, un nom explicite puisse suffire, il ne s'agit que d'une faible minorité.

A la création d'une nouvelle planche, ou lors d'une modification d'une planche déjà existante, le développeur doit commenter les interfaces, entrées et sorties, aussi bien sur le type que sur la plage de validité. Un texte décrivant la fonction doit aussi être ajouté (ou complété) pour qu'un testeur extérieur sache quoi et comment tester.

Règles de codage

Le point positif est qu'il existe un document Eurocopter sur les règles de codage, aussi bien SCADE que code manuel, pour le développement embarqué. Le point négatif est que seul le chef de projet est au courant qu'un tel document existe...

Pour éviter cet écueil à l'avenir, j'ai ajouté dans la documentation pour les nouveaux arrivants sur le projet, en plus du parcours administratif, l'obligation de prise de connaissance des normes de codage SCADE avant tout développement ou relecture.

Bien évidemment, j'ai diffusé à toute l'équipe déjà présente les deux documents pour qu'ils se mettent en conformité dans leurs modèles.

Fiche de relecture fonctionnelle SCADE (Annexe 1)

J'ai créé notre fiche de relecture fonctionnelle SCADE en me basant sur le tableau A-4 Contrôle du processus de conception logiciel de la norme DO178B (Figure 4). La fiche est en anglais, mais j'ai transmis les explications des différents points en français à chaque membre du projet, pour que tout le monde soit en phase. Pour rappel, les exigences de haut niveau correspondent à la SRS fournie par les architectes, et les exigences de bas niveau correspondent aux modèles SCADE que nous développons.

Point n°1: Are the SCADE nodes traced to model higher level requirement?

Ce point correspond à l'objectif 6 du tableau. L'objectif est de s'assurer que l'exigence de haut niveau été développée sous la forme d'exigences de bas niveau. Il faut vérifier que dans les propriétés des planches, l'exigence est bien identifiée.

Point n°2: Are the SCADE nodes comply to model higher level requirement?

Ce point correspond aux objectifs 1 et 8 du tableau. L'objectif est de s'assurer que les exigences de bas niveau du logiciel satisfont l'exigence de haut niveau. Il faut vérifier que l'ensemble des planches répertoriées couvrent bien l'ensemble de l'exigence.

Point n°3: Are the SCADE nodes accurate, consistent and implement accurate algorithm?

Ce point correspond aux objectifs 2, 7, 9 et 13 du tableau. Au niveau précision et cohérence il faut s'assurer que chaque exigence de bas niveau est claire et sans ambiguïté et que les exigences ne sont pas mutuellement conflictuelles.

Pour les aspects relatifs aux algorithmes, il faut s'assurer de la validité et du comportement des algorithmes proposés. Il faut vérifier les bibliothèques utilisées, la cohérence des entrées et des sorties par rapport à l'exigence.

Point n°4: Are the SCADE Nodes compatible with target computer?

Ce point correspond aux objectifs 3 et 10 du tableau. L'objectif est de s'assurer qu'il n'existe pas de conflits entre les exigences de bas niveau et les caractéristiques matérielles et logicielles de la machine cible, en particulier en ce qui concerne l'utilisation de ressources, les temps de réponse du système, et les dispositifs d'entrée et de sortie.

Point n°5: Are the SCADE Nodes verifiable?

Ce dernier point correspond aux objectifs 4 et 11 du tableau. L'objectif est de s'assurer que toutes les exigences de bas niveau sont vérifiables. Il faut donc avoir des commentaires assez détaillés pour effectuer des tests. La lecture des entrées, des sorties et des commentaires des traitements intermédiaires doit pouvoir suffire.

Fiche de relecture formelle SCADE (Annexe 3)

En plus de la relecture fonctionnelle des exigences, j'ai mis en place une fiche de relecture formelle qui reprend les normes de codage SCADE à respecter au niveau d'Eurocopter.

Ce document est essentiel puisqu'il énonce les bonnes pratiques en matière, entre autre, de forme, de nommage ou de complexité des planches SCADE. Cela correspond aux objectifs 5 et 12 du tableau A-4.

Lors d'une relecture fonctionnelle, une fiche de relecture formelle doit être ouverte pour chaque modèle listé en entête de la fiche, si la fiche n'existe pas déjà.

De ce fait, à la fin du projet, il y aura autant de fiches fonctionnelles que d'exigences, et autant de fiches formelles que de modèles SCADE.

Evidemment, la personne en charge d'une relecture ne doit pas être la personne qui a développé la planche.

Fiche de relecture de code manuel (Annexe 4)

Dans cette phase de développement, j'ai mis en place une dernière fiche de relecture. Comme énoncé précédemment, il peut être nécessaire d'utiliser du code manuel. Et dans le respect de la norme DO-178B, il faut vérifier ce code. Les points suivants se basent sur le tableau A-5 Contrôle du processus de codage (Figure 5).

Point n°1: Is the manual code compliant to low level requirement?

Ce point correspond à l'objectif 1 du tableau. Il s'agit de vérifier que dans l'entête, les exigences prévues pour être implémentées au niveau de cet objet sont bien identifiées et qu'elles sont bien réellement implémentées.

Point n°2: Is the manual code compliant with software architecture?

Ce point correspond à l'objectif 2 du tableau. L'objectif est de vérifier que les flux et les structures sont bien conformes aux définitions dans SCADE. Puisque les nœuds importés (code manuel) sont appelés via SCADE, il faut s'assurer que les interfaces soient cohérentes entre l'appelant, et l'appelé.

Point n°3: Is the manual code verifiable?

Ce troisième point correspond à l'objectif 3 du tableau. Il faut vérifier que la description des fonctionnalités est suffisamment détaillée pour pouvoir créer les tests nécessaires.

Point n°4: Is the manual code conforms to standards?

Ce point correspond à l'objectif 4 du tableau. L'objectif est de vérifier que les règles de codage sont respectées, ou que les écarts sont justifiés.

Point n°5: Is the manual code traceable to low-level requirements?

Ce point correspond à l'objectif 5 du tableau. Il faut s'assurer que les exigences de bas niveau du logiciel ont été développées dans le Code Source. Avec SCADE et la génération automatique de code, ce point devient inapplicable. En effet, l'outil certifie que l'exigence de bas niveau, la planche SCADE est générée sous forme de code C. Dans mon projet, ce n'est donc pas à vérifier.

Point n°6: *Is the manual code accurate and consistent, and the output of software integration process complete and correct?*

Ce dernier point correspond aux objectifs 6 et 7 du tableau. L'objectif est d'établir l'exactitude et la cohérence du Code Source, et notamment pour l'utilisation des piles, les débordements et la résolution de l'arithmétique en virgule fixe, les conflits de ressources, les temps d'exécution les plus défavorables, la gestion des interruptions, les variables ou constantes inutilisées et l'altération des données due à des conflits de tâches ou d'interruptions.

Il faut vérifier les adresses incorrectes, les chevauchements de mémoire et les composants logiciels manquants.

Comme pour le point n°5, tous les problèmes listés ici sont gérés par le générateur de SCAD, et il n'y a donc pas besoin de les vérifier.

Les points n°5 et n°6 sont donc non applicables. Je les ai quand même laissés dans la fiche de relecture pour montrer qu'ils ne sont pas oubliés et que j'ai bien pris en compte tous les éléments du tableau de la norme.

Tests unitaires

Une fois la relecture terminée et le module validé, il faut le tester. Le testeur ne doit pas être le développeur, mais il peut être déjà le relecteur. L'outil pour les plans de tests unitaires (PTU) d'Eurocopter est RT/RT.

IBM Rational Test RealTime (RT/RT)

La mise en œuvre d'un processus de test pratique, efficace et professionnel est devenue essentielle en raison du risque accru qui accompagne la complexité logicielle. Le temps et le coût consacré au test doivent être mesurés et gérés précisément. Très souvent, le manque de prévision de tests cause des dépassements de planning et de budget sans garantie de qualité.

RT/RT fournit une gamme complète de réponses à ces défis en permettant l'automatisation de système et des processus logiciels de test. C'est un ensemble d'outil de tests complets pour des systèmes embarqués, en temps réel.

De ce projet, l'outil RT/RT est certainement le plus difficile à prendre en main, sûrement parce qu'il utilise un langage propriétaire pour la rédaction des PTU. Je pense donc que de tous les documents que j'ai rédigé, celui qui décrit le processus à respecter pour créer un plan de test unitaire, avec la prise en main du logiciel RT/RT, est certainement le plus utile pour l'équipe et les nouveaux arrivants.

Exemples

Lorsque le PTU est lancé et terminé, RT/RT génère un rapport de test détaillé ainsi que la couverture structurale du code testé.

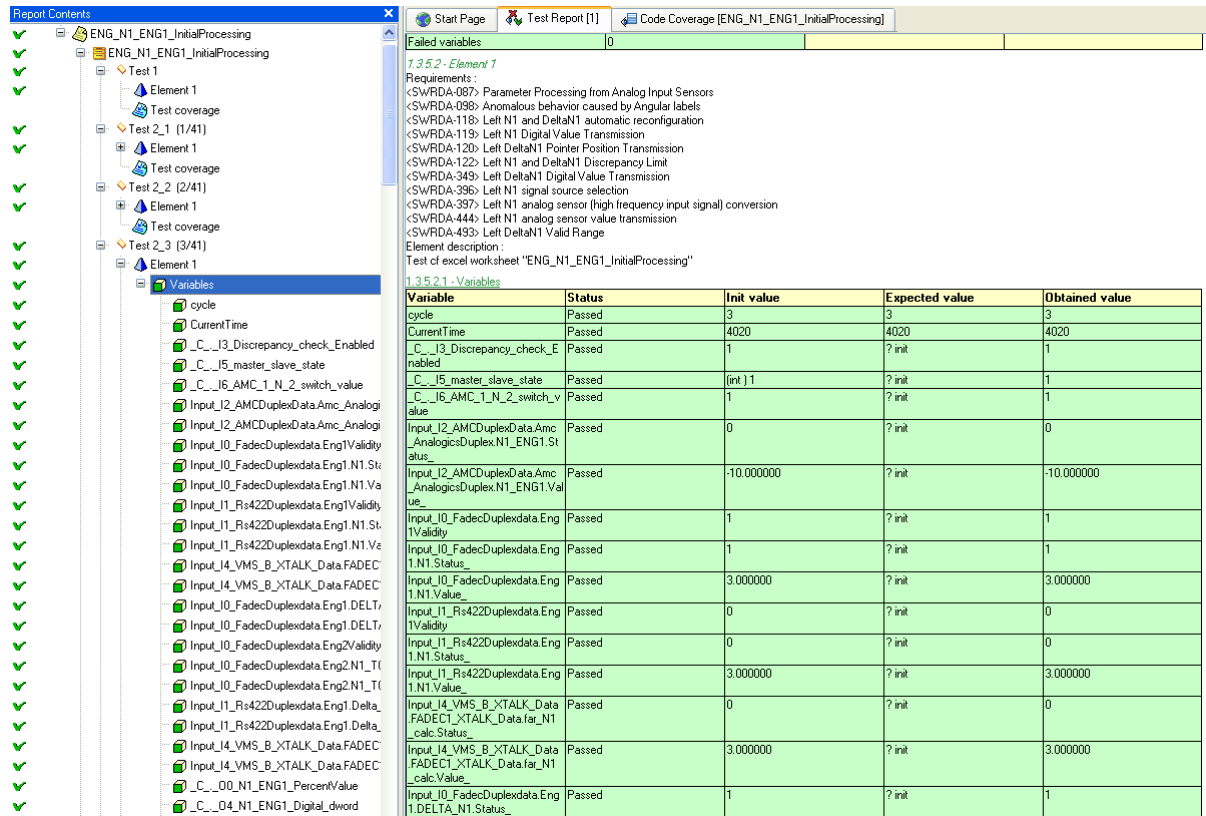


Figure 13 – Rapport de test

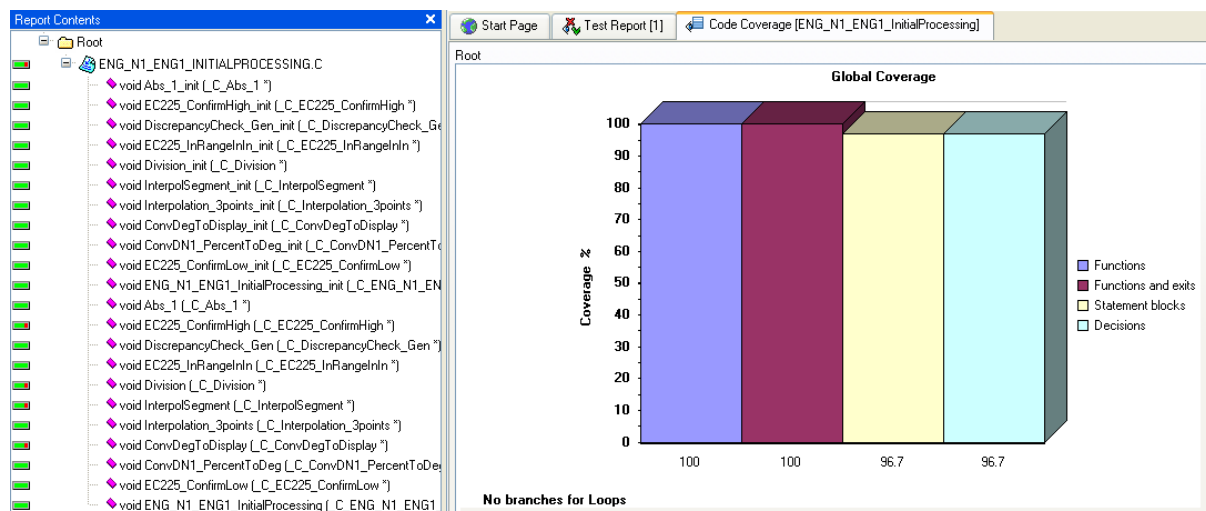


Figure 14- Taux de couverture structurale

Fiche de relecture Plan de Tests Unitaires (Annexe 2)

Comme pour un modèle SCADE ou un code manuel, il est nécessaire de relire un PTU. Le relecteur du PTU ne doit évidemment pas être son créateur, mais il peut être le développeur de la planche SCADE que le PTU doit tester. De ce fait, pour respecter la norme DO-178B dans mon périmètre, deux personnes peuvent suffire.

J'ai donc créé une fiche de relecture en me basant sur le tableau A-7 Contrôle du processus de vérification de la norme DO178B (Figure 5).

Point n°1: Are source file name and functional description of the PTU included in the header?

Ce point correspond à l'objectif 1 du tableau. Il faut vérifier la présence et la cohérence de l'en-tête du PTU.

Point n°2: Is each functional aspect tested?

Ce second point correspond à l'objectif 1 du tableau. L'objectif est de vérifier que les jeux de test ont été développés avec précision sous la forme de procédures de test et de résultats attendus et couvre entièrement l'exigence (couverture fonctionnelle). Il faut vérifier également que la description fonctionnelle est en commentaire et explicite.

Point n°3: Is there a test for every input parameter different input values (boundaries test, nominal test, invalid domain test)?

Ce troisième point correspond à l'objectif 1 du tableau. Il faut s'assurer que les domaines de valeurs possibles de la variable sont couverts : débordement de plage de valeurs, en erreur, correcte, écarts...

Point n°4: If the algorithm contains singularity, is there a test with input data for the singularity?

Ce point correspond également à l'objectif 1 du tableau et termine de le compléter. Il faut vérifier que les éventuels cas particuliers sont bien pris en compte dans les tests.

Point n°5: Are tests results corrects (and discrepancies explained)?

Ce point correspond à l'objectif 2 du tableau. L'objectif est de s'assurer que les résultats de tests sont corrects et que les éventuels écarts entre résultats attendus et obtenus sont expliqués. Le rapport de test RT/RT fait référence.

Point n°6: Is the coverage of high and low level requirements demonstrated?

Ce point correspond aux objectifs 3 et 4 du tableau. Cette analyse a pour objectif de déterminer dans quelle mesure l'ensemble de tous les PTU identifiés couvre l'exigence.

Point n°7: Is the coverage of software structure analysis achieved?

Ce dernier point correspond aux objectifs 6, 7 et 8 du tableau. Cette analyse a pour objectif de vérifier la couverture structurelle du logiciel par les tests, en utilisant l'outil RT/RT. Si la couverture RT/RT n'est pas à 100%, il doit y avoir obligatoirement une justification.

Tests d'intégration logiciel

Lorsqu'une des versions intermédiaires est terminée (codée, relue, testée et validée) elle est installée sur un AMC réel au banc d'intégration logiciel, quand la disponibilité le permet, pour des tests d'intégration logiciel, et des tests fonctionnels.

Méthode d'intégration de l'AMC+

Actuellement, les tests d'intégration logiciel sont réalisés sur le logiciel d'Eurocopter ARTIST.

ARTIST (Avionic Research, Test, Integration & Simulation Tools) est un outil logiciel et matériel qui permet aux utilisateurs, à travers une interface unique de tester les éléments d'un système avionique grâce à plusieurs fonctionnalités :

- Espionnage et acquisition des signaux émis par les éléments en test.
- Stimulation des éléments en tests.
- Simulation d'éléments manquants.
- Enregistrement de tout ou partie des informations échangées pendant le test.
- Dépouillement d'essais en vol ou piste.
- Rejeu d'un vol ou d'une partie d'un vol.

L'équipe AMC+ utilise les « planches de bord » ARTIST pour réaliser manuellement leurs procédures de tests sur banc et rédige ensuite le rapport de test. L'idée est bonne mais la durée de chaque procédure est longue, contrairement au temps qui nous est imparti sur le banc. De plus, on faisant des tests à la main, nous ne sommes pas à l'abri d'une erreur humaine.

Exemple ARTIST

Les planches de bord ARTIST sont paramétrables, et chaque entrée ou sortie du calculateur peut être modifiée ou visualisée.

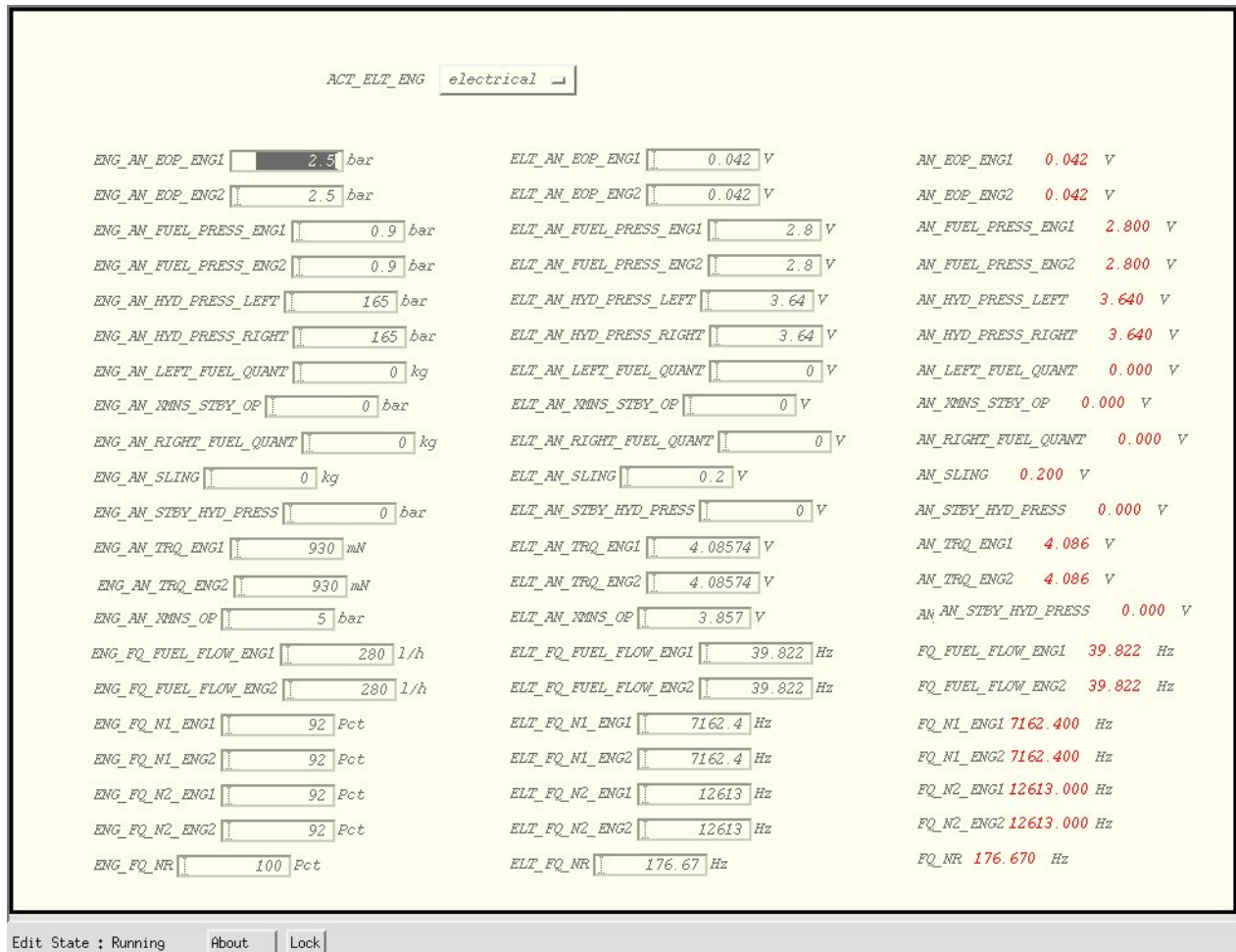


Figure 15 - Planche de bord ARTIST

Nouvelle approche

Lors de mon passage sur les bancs d'intégration du Tigre, j'ai pu constater que leur équipe avait des outils performants et génériques pour réaliser ces tests. Notamment des fichiers Excel paramétrables qui génèrent des scripts pour réaliser des tests automatiques. Je me suis donc servi de cette base pour l'adapter aux spécificités du MK1+. Cela nous fait gagner énormément de temps sur cette étape finale, par rapport à l'AMC+.

Vue d'ensemble

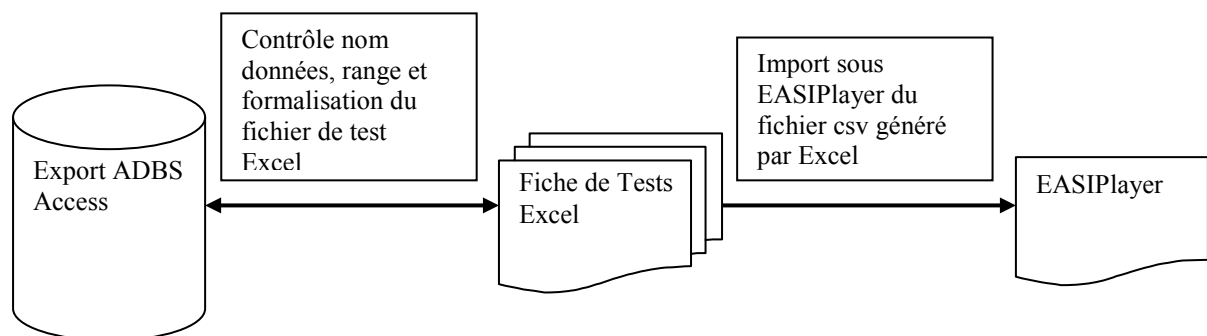


Figure 16 - Processus de tests automatiques

La base de données ADBS est celle dont se sert ARTIST pour connaître le nom et le typage des données. En alliant un export Access de cette base, et une macro Excel développée par une personne d'Eurocopter, le banc Tigre a réalisé un fichier de tests contrôlé et paramétrable. Ils génèrent ensuite un fichier csv en vue d'alimenter EASIPlayer.

EASIPlayer (Eurocopter Avionic System Integration Player) est un plug-in d'ARTIST, développé également par Eurocopter. Il permet l'automatisation des tests grâce à des scripts en entrée (au format CSV ou XML) et génère ensuite un rapport de tests.

Il y a de multiples avantages à cette nouvelle méthode. D'abord, on peut préparer tous ses jeux de tests en amont, sans avoir besoin du banc d'intégration. Ensuite, l'automatisation des tests est un gain de temps évident et la potentielle erreur humaine est retirée.

Description fiche de test Excel

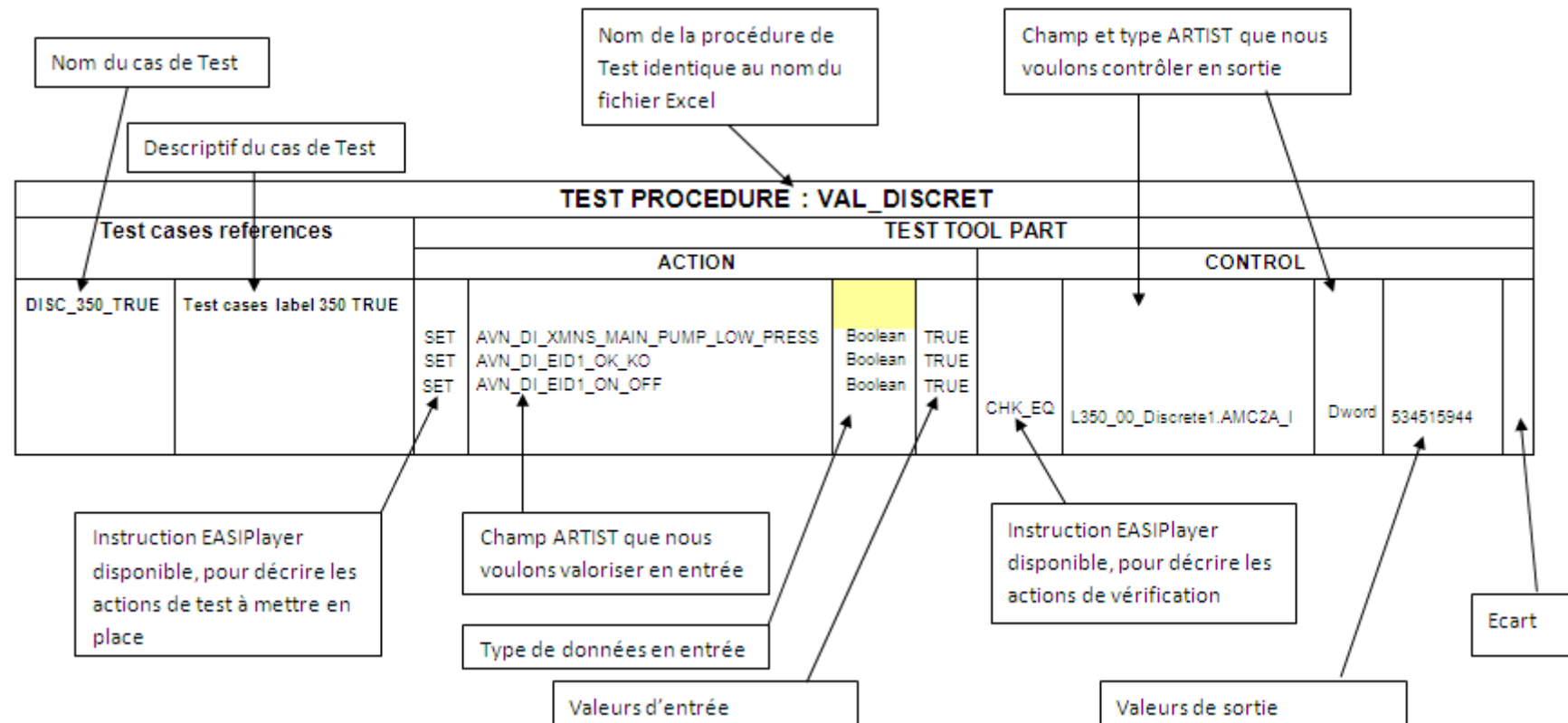


Figure 17 - Fiche de test automatique

Exemples

La figure suivante représente le test de la variable Torque du moteur 1 : Avec une tension en entrée de 4.318V, nous devons avoir une valeur calculée en milli Newton du torque de 997.2 mN +/- 0.25. La valeur en pourcentage du torque devra être de 32.14% +/- 0.02.

| | | ACTION | | | | CONTROL | | | | |
|-----------|---------------------------|--------|-----------------|------|-------|---------|---------------------|------|-------|------|
| IN_ANALOG | DC12_04 - ENG_AN_TRQ_ENG1 | SET | ELT_AN_TRQ_ENG1 | real | 4,318 | CHK_EQ | L335_01_Trq1_Sensor | real | 997,2 | 0,25 |
| | | | | | | CHK_EQ | L336_00_Trq1_Dv | real | 32,14 | 0,02 |

Figure 9 - Ligne d'une fiche de tests automatiques Excel

La figure suivante représente le rapport du test réalisé automatiquement par EASiPlayer. Cela montre que la valeur calculée est correcte, mais qu'il y a un problème sur la fonction qui la transforme en pourcentage.

| | | | | |
|---|--|--------------|--------|------------------------------|
| ✗ | DC12_04 - ENG_AN_TRQ_ENG1 | | FAILED | |
| ✓ | SET MEMORY.ELT_AN_TRQ_ENG1 to 4.318 | 08:53:33.922 | PASSED | |
| ✓ | CHK_EQ OPER_AMC1A_OPER.AMC1A_IL335_00_Trq1_Sensor.TRQ1_SENSOR = 997.2 +/- 0.25 | 08:53:34.269 | PASSED | Have 'TRQ1_SENSOR', '997.25' |
| ✗ | CHK_EQ OPER_AMC1A_OPER.AMC1A_IL336_00_Trq1_DvTRQ1 = 32.14 +/- 0.02 | 08:53:36.618 | FAILED | Have 'TRQ1', '0' |

Figure 10 - Rapport de tests automatiques EASiPlayer

Disponibilité du banc

Pour le problème de disponibilité du banc, outre le fait que la mise en place de l'automatisation des tests nous enlève beaucoup de pression, j'ai trouvé une solution qui est toujours en développement à l'heure actuelle. Plutôt que de dépendre du banc d'intégration pour les essais finaux, j'avais appris au cours de mon ancienne mission que l'on pouvait préparer une machine Unix pour simuler le calculateur et faire fonctionner notre partition à l'intérieur : la simulation sur hôte. C'est un de nos stagiaires qui est actuellement à la réalisation de ce mini projet. Il devrait bientôt voir le jour et nous aider grandement.

Gestion des problèmes

Fort de mon expérience en intégration système, j'ai eu l'occasion de rédiger beaucoup de 'Problem Report' (PR). De ce fait, j'ai pu créer un processus standard que toute l'équipe doit respecter à la création d'un PR. J'ai aussi rédigé un processus pour les 'Software Change Note' (SWCN) à respecter lors de la résolution d'un PR. Enfin, j'ai rédigé le processus 'Cycle de vie d'un PR' qui montre toutes les étapes à respecter, et comment utiliser les outils, de la création du PR à sa fermeture.

Problem Report

Origine des PR

Les problèmes peuvent être découverts lors des phases de relectures ou de tests (plan de tests unitaires, intégration logiciel, intégration système et essais en vols).

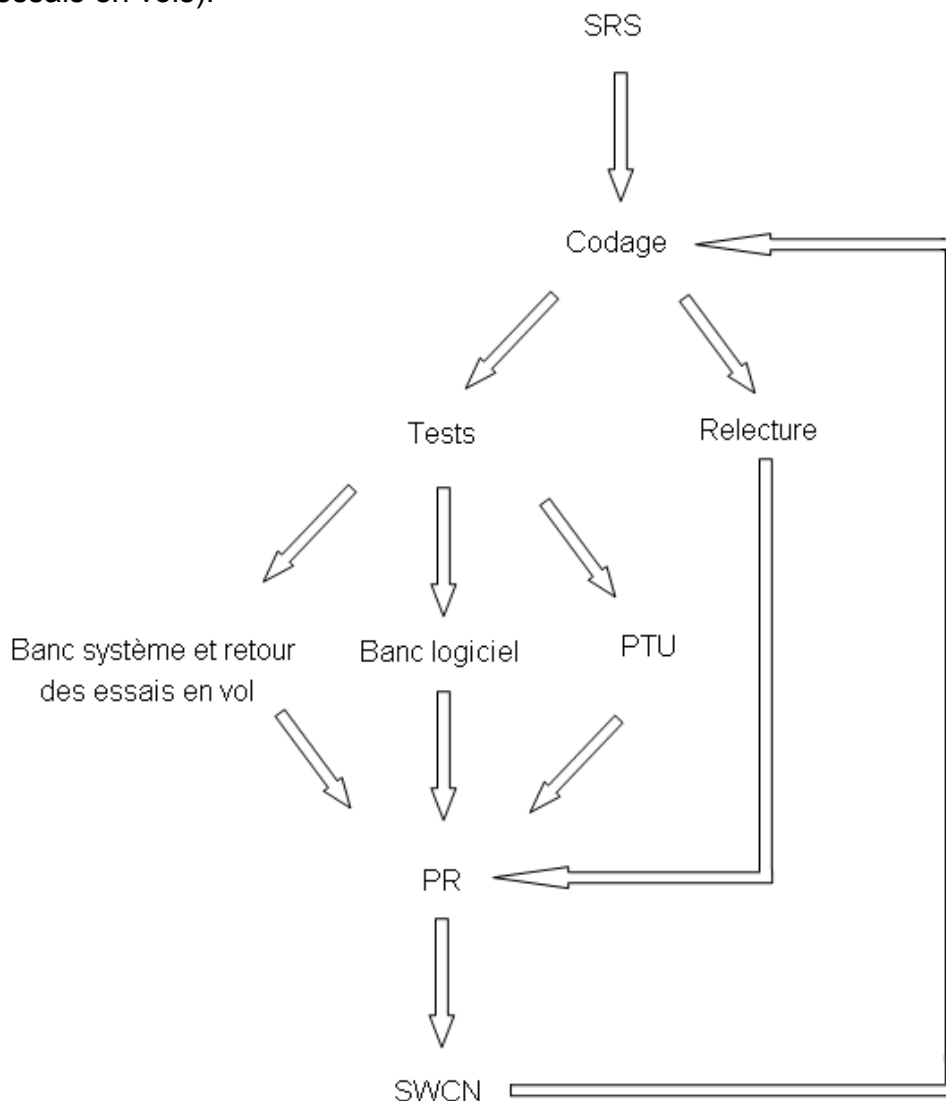


Figure 20 - Origine d'un PR

Caractéristiques d'un PR

Pour générer un PR, il faut se placer dans le projet voulu (1) puis dans la rubrique Report Issue (2).

Les champs sont les suivants :

- Category (3) : permet d'indiquer où se trouve la modification à apporter. Cette modification peut se situer sur les documents de la SRS et de l'IRS ou sur le logiciel (SW).
- Severity (4) : indique la gravité du PR, selon la classification DAL de la norme D0-178B.
- Product version (5) : indique la version du logiciel sur laquelle le problème a été détecté.
- Summary (6) : Titre du problème.
 - o Dans le cas où le PR découle d'un PTU ou que l'exigence est identifiée, celui-ci doit respecter la forme suivante : <SWRDA-XXX> + résumé du problème.
 - o Dans tous les autres cas, seul le résumé du problème doit figurer.
- Description (7) : description du problème rencontré. Elle doit décrire ce qui est attendu et ce qui est constaté. De plus, tout élément susceptible d'aider à la compréhension du problème doit y être référencé.
 - o Exemple : S'il s'agit d'un nœud SCADE, la référence du nœud et l'équation impactée doivent y figurer.
- Steps To Reproduce (8) : permet de décrire la procédure nécessaire à la reproduction de l'erreur détectée.
 - o Pour un PTU, la référence du PTU doit y figurer.
 - o Pour un problème relatif aux tests banc, le mode opératoire doit être référencé.
- Detected on (9) : Renseigne comment l'erreur a été découverte (voir chapitre Origine des PR)
- Report Stay (10) : réouvre un nouveau formulaire de saisie de PR.

The image shows a web application interface for reporting an issue. At the top, there is a navigation bar with links: [Main](#), [My View](#), [View Issues](#), [Report Issue](#) (highlighted with a red circle and arrow 2), [Change Log](#), [Roadmap](#), [Docs](#), [My Account](#), and [Logout](#). To the right of the navigation bar, there is a 'Project:' dropdown menu set to 'PR_VMS_MK1e' (arrow 1) and a 'Switch' button. Below the navigation bar, there is a 'Recently Visited' section with a list of issue IDs: 0000899, 0000940, 0000981, 0000967, 0000972. The main form is titled 'Enter Report Details' and contains several fields: 'Category' (dropdown menu, arrow 4), 'Severity' (dropdown menu, arrow 5), 'Product Version' (dropdown menu, arrow 3), 'Summary' (text input field, arrow 6), 'Description' (large text area, arrow 7), 'Steps To Reproduce' (large text area, arrow 8), 'Detected on' (dropdown menu with options: System Bench, Flight, Ground, PF, PTU, arrow 9), and 'Report Stay' (checkbox, arrow 10). A 'Submit Report' button is located at the bottom right of the form. A red asterisk and the word 'required' are visible at the bottom left of the form.

Figure 11 - Fiche d'un PR

Cycle de vie d'un PR

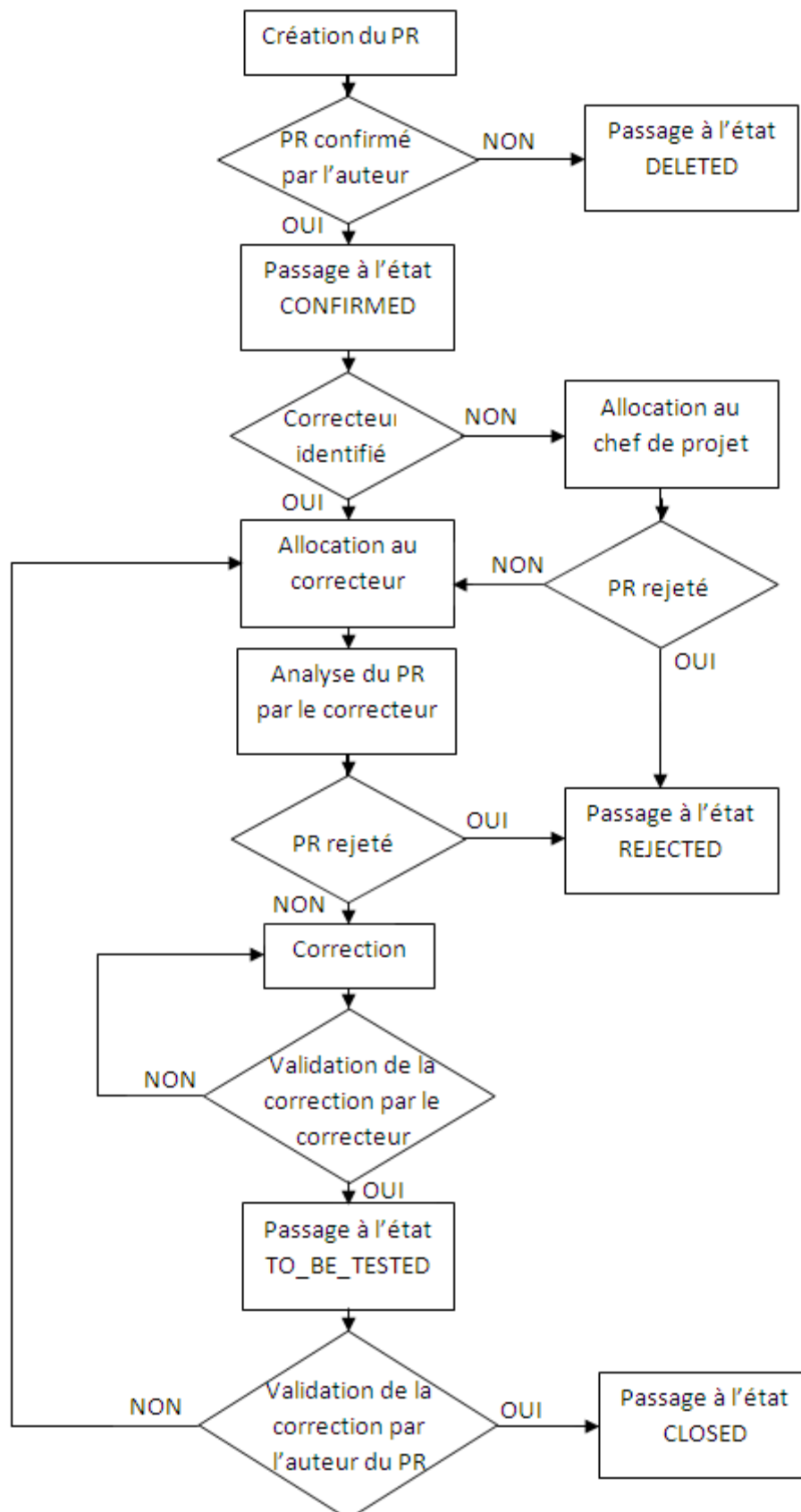


Figure 12 - Cycle de vie d'un PR

Il existe plusieurs états pour un PR : NEW, CONFIRMED, REJECTED, ALLOCATED, TO_BE_TESTED, CLOSED, DELETED

Lorsque le PR est créé, il doit être alloué au correcteur ou à défaut au chef de projet pour que celui-ci l'analyse et détermine par quel membre de l'équipe la correction devra être effectuée.

Le correcteur doit alors définir si le PR a lieu d'être ou non. Dans ce dernier cas, le PR doit être placé en état REJECTED.

Dans la phase ALLOCATED, les objets SWCN doivent être créés et liés au PR.

Lorsque le PR est corrigé et la correction validée par le correcteur (rejeu de cas de test sur banc logiciel ou PTU, relecture), la SWCN doit passer à l'état CLOSED et le PR passe à l'état TO_BE_TESTED.

Dans la phase TO_BE_TESTED, l'auteur du PR doit vérifier que le problème a été résolu en rejouant son test. Si le problème a été corrigé, le PR passe à l'état CLOSED. Dans le cas contraire, le PR est renvoyé au correcteur et repasse à l'état ALLOCATED. Si une erreur différente intervient, un nouveau PR doit être créé et lié au PR initial.

Software Change Note

Caractéristiques d'une SWCN

Pour générer une SWCN, il faut se placer dans le projet voulu (1) puis dans la rubrique Report Issue (2).

- Category (3) : Renseigner de la même manière que le PR source.
- Priority (4) : Laisser à 'normal'.
- Target version (5) : Indique la version du logiciel sur laquelle le problème va être corrigé (version courante de développement).
- Summary (6) : Reprendre le même titre que le PR source
- Description (7) : La description doit comporter tous les éléments qui peuvent permettre identifier clairement les corrections effectuées :
 - Le modèle SCADÉ
 - Le nœud
 - L'équation
 - La description de la modification

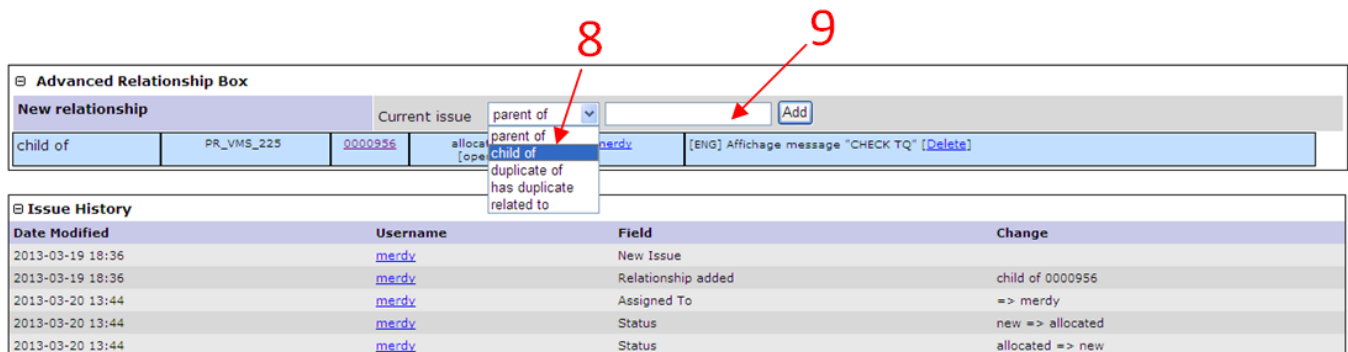
The screenshot shows the 'Report Issue' form in the VMS system. The form is titled 'Enter Report Details' and contains several fields with red arrows pointing to them:

- 1: Project dropdown menu (set to 'Sw/CN_VMS_225')
- 2: 'Report Issue' link in the navigation bar
- 3: Category dropdown menu (set to '(select)')
- 4: Priority dropdown menu (set to 'normal')
- 5: Target Version dropdown menu
- 6: Summary text input field
- 7: Description text input field

At the bottom of the form, there is a 'Report Stay' section with a checkbox 'check to report more issues' and a 'Submit Report' button. A red asterisk indicates required fields.

Figure 13 - Fiche d'une SWCN

Il faut ensuite créer le lien child of (8) et renseigner le numéro du PR à l'origine de la SWCN(9) dans Advanced Relationship Box. Ce lien permet de naviguer facilement entre le PR la ou les SWCN associée(s).



Advanced Relationship Box

New relationship

Current issue: parent of [dropdown] [Add]

child of PR_VMS_225 0000956 allocated [dropdown] merdy [ENG] Affichage message "CHECK TQ" [Delete]

Issue History

| Date Modified | Username | Field | Change |
|------------------|----------|--------------------|------------------|
| 2013-03-19 18:36 | merdy | New Issue | |
| 2013-03-19 18:36 | merdy | Relationship added | child of 0000956 |
| 2013-03-20 13:44 | merdy | Assigned To | => merdy |
| 2013-03-20 13:44 | merdy | Status | new => allocated |
| 2013-03-20 13:44 | merdy | Status | allocated => new |

Figure 14 - Relation PR / SWCN

Cycle de vie d'une SWCN

Lorsque la SWCN est créée, elle apparaît en état NEW dans la rubrique 'Status'. Par défaut, elle n'est assignée à personne, il faut se l'allouer à soi-même.

Lorsque la correction est terminée, la SWCN doit passer à l'état IMPLEMENTED.

Si la correction effectuée répond au PR et qu'elle est validée par le correcteur lors de son test, la SWCN passe dans l'état CLOSED. Dans le cas contraire, elle reste à IMPLEMENTED.

Lorsque la SWCN est passée à l'état CLOSED, le correcteur doit mettre le PR source à TO_BE_TESTED.

Rôle de chaque entité

| Actions | Auteur du PR | Correcteur | Chef de projet |
|---|--------------|------------|----------------|
| Création du PR | X | | |
| Allocation du PR | X | | X |
| Création SWCN | | X | |
| Correction | | X | |
| Clôture SWCN | | X | |
| Informar que la correction est terminée et valide | | X | |
| Validation de la correction | X | X | |
| Clôture du PR | X | | |

Cas particulier

Puisque je travaille sur le projet MK1+ qui est le « petit frère » du projet AMC+, si un problème est détecté sur le premier, il y a de fortes chances qu'il soit applicable au second, et vice versa. De ce fait, pour ne pas qu'il y ait d'oubli, un problème vu sur un projet doit également être pris en compte dans l'autre. Ainsi, un duplicata du PR doit être créé. L'analyse déterminera ensuite si il est ou non applicable.

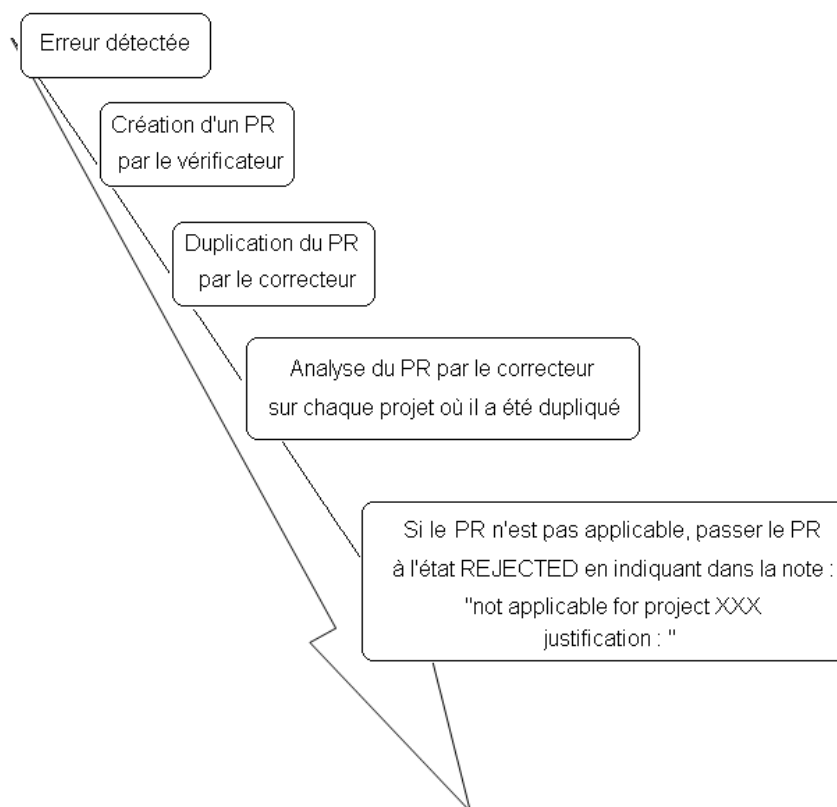


Figure 15 - Report d'un PR sur d'autres projets

Livraison

Une fois la version finale validée sur le banc d'intégration logiciel, elle sera livrée au banc d'intégration système en vue de la certification. Cela signera la fin de notre projet MK1+. Il sera ensuite installé dans l'hélicoptère Cougar pour des essais au sol, puis des essais en vol. Il est évident que d'autres problèmes seront découverts lors de chacune de ces phases. Nous passerons alors à de la maintenance applicative.

Si j'écris au futur, alors que la livraison aurait déjà du avoir lieu, puisque nous étions dans les temps, c'est que malheureusement, les priorités ont été redéfinies par le client. Toutes les ressources sont maintenant déployées sur le projet AMC+, pour enfin le terminer.

Conclusion sur le projet

Le projet MK1+ est le premier projet dans lequel j'ai une vraie responsabilité. Ma hiérarchie a compté sur moi pour démarrer le projet sur des bases solides et ne pas prendre du retard comme le projet AMC+ a su en accumuler.

J'ai beaucoup appris, aussi bien au niveau qualité que gestion de projet, analyse, développement, et tests. J'ai su être force de proposition, capitaliser mon expérience personnelle en récupérant les idées intéressantes et les outils performants, pour en faire profiter l'équipe.

Même si je regrette à ce jour que la certification du MK1+ ait été repoussée, je pense avoir rempli ma mission puisqu'au moment où nous avons changé de priorité, à la version 0.08, nous étions dans les temps et le client pleinement satisfait lors des recettes.

Dans tous les cas, mon travail effectué sur le MK1+ est loin d'être inutile. Ayant prouvées leur efficacité, mes méthodes sont reprises sur l'AMC+ pour le remettre sur la bonne voie, et seront utilisées également pour le nouveau projet AMC+ V2, dont Steria s'est vu confié la réalisation par Eurocopter.

Références bibliographiques

Software aspects of certification (DO-178B/ED-12B) par Frederic POTHON

Considérations sur le logiciel en vue de la certification des systèmes et équipements de bord par l'EUROCAE

EASI Player concepts par Gilles DUPEYROU (Eurocopter)

Rational Test RealTime Online Documentation par IBM

Introduction à SCADE par François TOUCHARD (Polytech Marseille)

Formation ARTIST/UNIX par Jean-Luc TAITON (Eurocopter)

SCADE Usage Standard par M. BOSSY (Eurocopter)

Software Code Standards par Olivier MOREL (Eurocopter)

Annexe 1 : Fiche de relecture fonctionnelle SCADE

SCADE Checklist

| | | | | |
|------------------|--|-------------|-----------------------|-----------------------|
| Project : | SCADE Verification Check List for <SWRDA-XXX> requirement | Date | Inspector Name | Page 1 / 1 |
|------------------|--|-------------|-----------------------|-----------------------|

| | |
|--|--|
| STD_SCA Reference | |
| SRS Reference/issue | |
| Requirement number | |
| Impact Version | |
| Reviewed SCADE models (delete or insert lines if any) | |
| | |
| | |
| | |
| | |
| | |

| N° | Points to Check | Yes | No | N/A | DO178B objectives |
|----|--|-----|----|-----|---------------------------------------|
| 1 | Are the SCADE nodes traced to model higher level requirement? | | | | Table A-4 objective #6 |
| 2 | Are the SCADE nodes comply to model higher level requirement? | | | | Table A-4 objectives: #1, #8 |
| 3 | Are the SCADE nodes accurate, consistent and implement accurate algorithm? | | | | Table A-4 objectives: #2, #9, #7, #13 |
| 4 | Are the SCADE Nodes compatible with target computer? | | | | Table A-4 objectives: #3, #10 |
| 5 | Are the SCADE Nodes verifiable? | | | | Table A-4 objectives: #4, #11 |
| | Comment/justification (if NO or N/A) | | | | |

| Remarks / Comments | Answer | Check | Who | Date |
|--------------------|--------|-------|-----|------|
| Point X: | | | | |

Annexe 2 : Fiche de relecture Plan de Tests Unitaires

| | | | | |
|------------------|--|-------------|-----------------------|-----------------------|
| Project : | Tests Verification Check List for <SWRDA-XXX> requirement | Date | Inspector Name | Page 1 / 1 |
|------------------|--|-------------|-----------------------|-----------------------|

| | |
|--|--|
| SRS Reference/issue | |
| Requirement number | |
| Impact Version | |
| Reviewed component (PTU) Name (delete or insert lines if any) | |
| | |
| | |
| | |
| | |
| | |
| | |

| N° | Points to Check | Yes | No | N/A | DO178B objective |
|--------------------------------------|--|-----|----|-----|---------------------------------|
| 1 | Are source file name and functional description of the PTU included in the header? | | | | Table A-7 objective #1 |
| 2 | Is each functional aspect tested? | | | | Table A-7 objective #1 |
| 3 | Is there a test for every input parameter with the following input values <ul style="list-style-type: none"> boundaries test (min/max ;False/True for Boolean) nominal test invalid domain test | | | | Table A-7 objective #1 |
| 4 | If the algorithm contains singularity, is there a test with input data for the singularity? | | | | Table A-7 objective #1 |
| 5 | Are tests results corrects (and discrepancies explained)? | | | | Table A-7 objective #2 |
| 6 | Is the coverage of high and low level requirements demonstrated? | | | | Table A-7 objectives #3, #4 |
| 7 | Is the coverage of software structure analysis achieved? | | | | Table A-7 objectives #3, #4, #8 |
| Comment/justification (if No or N/A) | | | | | |

| Remarks / Comments | Answer | Check | Who | Date |
|--------------------|--------|-------|-----|------|
| Point X: | | | | |

Annexe 3 : Fiche de relecture formelle SCADE

| | | | | |
|------------------|--|-------------|-----------------------|-----------------------|
| Project : | SCADE File Verification Check List for NODE NODE NAME | Date | Inspector Name | Page 1 / 7 |
|------------------|--|-------------|-----------------------|-----------------------|

| | |
|--------------------------|-----------------------------|
| STD_SCA Reference | 332 A 89 8292 / A004 |
| Impact Version | |

| Point to Check | Yes | No | N/A | DO178B objectives |
|---|------------|-----------|------------|--|
| Does the SCADE nodes satisfy the SCADE coding rules (ref. 332 A 89 8292) | | | | Table A-4 objectives: #5, #12 |
| Comment/justification (if NO or N/A) | | | | |

SCADE CODING RULES CHECKLIST

| RULE | DEFINITION | LEVEL | Compliance |
|--|--|------------------|-------------------|
| Scade structure | | | |
| Rule STR/1: many models | SCADE specification has to be split in many models. Each model is a set of nodes used to build a major algorithm like FLI or having the same properties like failures management | optional | |
| Rule STR/2: external application interfaces | External application interfaces have to be defined in two dedicated models; one for inputs (Inputs) another for outputs (Outputs). Each structure exchanged between partitions has to be defined in a dedicated SCADE node. This node belongs to the sender partition specification and can be modified only by the responsible of this partition. The receiver partition specification includes this node in the Inputs model but no modification is allowed. | mandatory | |
| Rule STR/3: manual and SCADE interface | If the partition is split in two entity kinds (one developed by SCADE the other developed in manual code) then the interface between these two entities has to be defined in a dedicated model. | mandatory | |
| rule STR/4: SCADE and SRS structure | SCADE node or equation structure has to be near as far as possible to the upper requirement document structure (generally SRS document). | mandatory | |

| RULE | DEFINITION | LEVEL | Compliance |
|--|--|-----------|------------|
| Rule STR/5: additional node | If the same function is used more than 3 times (not necessary in the same node) then it has to be defined in a dedicated node and not in lined in the using node. | mandatory | |
| Rule STR/6: unnecessary node | Avoid increasing abstraction level by adding node if the treatment is used less than 2 times. | mandatory | |
| Rule STR/7: A4 format | Equation size is limited to A4 format. If the node is split in many equations the distribution has to follow functional aspect and to build understanding sub function. | mandatory | |
| Rule STR/8: symbol occurrence | SCADE symbol occurrence facilities has to be used, then the an occurrence file must be defined in SCADE model ("occurrence number file" parameter in "SCADE" option defined in "properties" of the model) The file name has to be <model>.occ and the base has to be 10. | mandatory | |
| Rule STR/9: links cross | Cross between many operator links has to be avoided as far as possible. | mandatory | |
| Rule STR/10: superimposed link | Partial overlap between many operator links has to be strictly avoided. | mandatory | |
| Rule STR/11: saofd content | Each saofd file has to contain only one kind of node (type, constant, or equation). | mandatory | |
| Rule STR/12: equation input/output place | The equation input place has to be at left of the equation sheet and the outputs at the right. | optional | |
| Rule STR/13: textual node | The textual node must not be used in a SCADE project. | mandatory | |
| Rule STR/14: unused node | Nodes that are not used in a SCADE Model must be removed from the model, and the associated saofd files must be deleted. | mandatory | |

| RULE | DEFINITION | LEVEL | Compliance |
|----------------------------------|--|-----------|------------|
| Rule LIB/1: library definition1 | The nodes defining elementary treatments without application specificities have to be grouped in dedicated model called library. | mandatory | |
| Rule LIB/2: library definition2 | Library model has to be split in many models according common attribute. A big library with all nodes has to be avoided. | optional | |
| Rule LIB/3: Esterel library | The SCADE library basically delivered by Esterel must not be used and must be suppressed at model creation. | mandatory | |
| Rule LIB/4: Library limitation | Library model has to be limited to the nodes which are or will be really used by the application. | optional | |
| Rule LIB /5: graphic symbol | Each node has to have a graphical symbol used by SCADE to display the node call. | mandatory | |
| Rule LIB/6: hidden input | If an input is generally defined by a constant, then the input has to be defined as a hidden input. | optional | |
| Rule LIB/7: elementary input | Each inputs of a library node has to be an elementary entity and not a structure with many elements. | optional | |
| Rule LIB/8: basic operator using | The node has to be generally developed with basic SCADE operator and not with other node. | optional | |
| | | | |
| Rule NAM/1: file naming | File and node have to have the same name. SCADE editor modifies automatically filename if option "propagate entity name changes to file" in "General" option in menu "Tools" is selected. | mandatory | |
| Rule NAM/2: entity naming | Inputs, outputs, internal data, nodes, constants, type and structure fields have to have name in mixed format with an upper character at the beginning of each word (example EngineOilPressure). | optional | |

| RULE | DEFINITION | LEVEL | Compliance |
|--|---|-----------|------------|
| Rule NAM/3: generic type naming | Node having inputs or outputs with SCADE generic type (identify with *) has to have a name ended by “_” character. | mandatory | |
| Rule NAM/4: name continuity1 | The data name used to put/get value from/to a structure field has to have the same name as the field except when the field name is ended by “_” character. | optional | |
| Rule NAM/5: name continuity2 | The data name used to put/get value to input/output node has to have the same name as the field except when the input/output name is ended by “_” character. | optional | |
| | | | |
| Rule CFG/1: type definition | Predefined SCADE type, real, bool, int and char have to be respectively defined by type float, signed char, unsigned int and unsigned char. Integer are always positive number. | mandatory | |
| Rule CFG/2: dedicated symbol for each predefined input/output type | Predefined SCADE type, real, bool, int and char have to have dedicated graphical symbols to identify graphically the type of the external or internal node input or output. | optional | |
| | | | |
| Rule LAN/1: Pre using | Fby operator has to be used instead of Pre operator with Init operator. | mandatory | |
| Rule LAN/2: structured constant | Structured constants have to be defined in constant node | mandatory | |
| Rule LAN/3: Fby and conduct | Fby operator using inside node used with Conduct has to be justified or reinitialized at each Conduct wakeup condition. The rule is applicable too if the conduct is on the nodes calling indirectly the node with the Fby operator (| optional | |
| Rule LAN/4: hidden input usage | When node with a hidden input is called and the corresponding input is a constant, then the constant has to be put inside the symbol. | optional | |

| RULE | DEFINITION | LEVEL | Compliance |
|--|---|-----------|------------|
| Rule LAN/5: State management | If version SCADE 5.1 is selected, basic state machines operator have to be used to define application state transition. If SCADE version is upper Safe State Machines have be used. | optional | |
| Rule LAN/6: State and conduct | State machines using inside node used with Conduct has to be justified. | mandatory | |
| Rule LAN/7: multi switches | A set of switches with the same condition has to be merged in one switch | optional | |
| Rule LAN/8: library operator using | Library nodes have to be used instead to inline the corresponding SCADE description except in library model. | mandatory | |
| Rule LAN/9: node output merge | A set of data sent through two node call levels have to be merged inside a SCADE structure. | mandatory | |
| Rule LAN/10: data structure level | Reduce the data structure hierarchical level as far as possible. | optional | |
| Rule LAN/11: data structure field | Data structure with more than 30 fields has to be avoided. | optional | |
| Rule LAN/12: node input structure | When a node uses only a few set of fields from a structure, use the fields as input (extracted by the caller) instead of the complete structure. The more the node is being used (lot of occurrences) and the bigger the structure is, the more this rule has to be taken into account. | optional | |
| Rule LAN/13: structure field access | Structure fields have to be acceded only by elementary naming projection. | mandatory | |
| Rule LAN/14: structure building location | Data have to be grouped in a structure by the data producer and not by the node calling the producer. | mandatory | |
| Rule LAN/15: Fby location | Fby operator has to be used inside a node and not outside the node between one input and one output | mandatory | |
| Rule LAN/16: global data | Global data have not to be used. | mandatory | |
| Rule LAN/17: clock | SCADE clock has not to be used. | mandatory | |

| RULE | DEFINITION | LEVEL | Compliance |
|---|--|-----------|------------|
| Rule LAN/18: output value reusing | When node output result has to be used in another equation, the internal data used to perform this interface has to be post-fixed with " L " string. | mandatory | |
| Rule LAN/19: Division operator | Division operator has not to be used when it's not possible to demonstrate than the divisor is not null. A dedicated node with robustness behavior could be used instead. | mandatory | |
| Rule ANN/1: node description | Each node has to have annotation to describe the node. If the node matches exactly SRS requirement, then the description is limited to a reference link to the corresponding SRS requirement. If the node implements only one part of a requirement then additional information will be added to characterize this requirement part. | mandatory | |
| Rule ANN/2: node traceability | SRS traceability has to be managed by requirement reference or by naming rules. | mandatory | |
| Rule ANN/3: constant description | The defined/imported constants of the SCADE project must be describe with a Remark annotation. Whether it's possible, the annotation may be refer to a SRS. | mandatory | |
| Rule ANN/4: type description | The types of the SCADE project must be described with a Remark annotation. | mandatory | |
| Rule OPT/1: Predefined operator weight (CPU load) | Operator weight has to be taken into account to optimize SCADE node and reduce CPU load. | optional | |
| Rule OPT/2: Logical expression | Logical operator number has to be reduced in logical expression. | optional | |
| Rule OPT /3: Local variable | Local variable used only one time has to be removed. | optional | |
| Rule OPT /4: Projection | Data projection has to be made near the use. | optional | |
| Rule OPT /5: Double computation | Double computation has to be removed. | optional | |
| Rule OPT /6: Division by a constant | Division by a constant has to be replaced by a multiplication by the inverse. | optional | |

Annexe 4 : Fiche de relecture de code manuel

Manual code Checklist

| | | | | |
|------------------|---|-------------|-----------------------|-----------------------|
| Project : | Manual code object reference (file name and issue) : | Date | Inspector Name | Page 1 / 9 |
|------------------|---|-------------|-----------------------|-----------------------|

| | |
|--|--------------------------|
| STD-COD Reference | 332 A 89 8293 |
| SRS Reference/issue | 332A89.8295 / B001 draft |
| Impact version | |
| Verified requirements (delete or insert lines if any) | |
| | |
| | |
| | |
| | |
| | |

| N° | Points to Check | Yes | No | N/A | DO178B objectives |
|---|--|-----|----|-----|------------------------------|
| 1 | Is the Manual code compliant to low-level level requirements? | | | | Table A-5 objective: #1 |
| 2 | Is the Manual code compliant with software architecture? | | | | Table A-5 objective: #2 |
| 3 | Is the Manual code verifiable? | | | | Table A-5 objective: #3 |
| 4 | Is the Manual code conforms to standards.? | | | | Table A-5 objective: #4 |
| 5 | Is the Manual code traceable to low-level requirements.? | | | X | Table A-5 objective: #5 |
| 6 | Is the Manual code accurate and Consistent, and the output of software integration process complete and correct? | | | X | Table A-5 objectives: #6, #7 |
| Comment/justification (if NO or N/A) N°5 : This Table A-5 objective #5 can't be verified at code object file level. It will be demonstrated by verification of traceability matrix . N°6 : Certified by the production line | | | | | |

| Remarks / Comments | Answer | Check | Who | Date |
|--------------------|--------|-------|-----|------|
| Point X: | | | | |

MANUAL CODE RULES CHECKLIST

| RULE | DEFINITION | LEVEL | Compliance |
|-------------------------|--|-----------|------------|
| Header | | | |
| [DC-1] | a common header description shall be used for any new modules. | mandatory | |
| [DC-2] | each function/procedure shall be described through a standard header, describing purpose, interfaces, input/output parameters, status return if any. | mandatory | |
| [DC-3] | comment and instruction should be defined in different lines, except for definitions and declarations. | optional | |
| Code Description | | | |
| [DG-1] | each line of code should contain 1 instruction only (including declaration), 1 structure field only or 1 enumerated value only. | optional | |
| [DG-2] | brackets should be used for the value returned by a function | optional | |
| [DG-3] | parameters of function should be ordered as this: (input parameter, ..., input/output parameters, ..., output parameters, ...). | optional | |
| [DG-4] | brackets of a bloc should be in a specific line without instruction (except for comments). | optional | |
| [DG-5] | <p>a comment should be added after the closing bracket of a bloc.</p> <p><i>Note:</i> this rules eases lisibility of code vs comments in the same page.</p> <p><i>Example:</i> if (Mode_Init)</p> <p> {</p> <p> .</p> <p> .</p> <p> }</p> <p> }/* endif (Mode_Init)*/</p> | optional | |
| [DG-6] | <p>preprocessor directives (#define, #undef, #if, #ifdef, #else, #endif, ...) should be joined to "#".</p> <p><i>Note:</i> C syntax may require to declare "#" at the beginning of the line.</p> <p><i>Example:</i> appropriate coding is:</p> <p> #ifdef VARIABLE</p> <p> .</p> <p> not appropriate coding is:</p> <p> # ifdef VARIABLE</p> <p> .</p> | optional | |
| [DG-7] | preprocessor conditional directives "#if", "#ifdef" or "#ifndef" + "#endif" should be defined in the same bloc to | optional | |

Réalisation de la partition embarquée
Vehicle Management System (VMS)
de l'hélicoptère COUGAR dans le cadre de sa certification

| RULE | DEFINITION | LEVEL | Compliance |
|----------------|---|-----------|------------|
| | <p>enforce lisibility of directives. <i>Example:</i> <code>#ifdef TYPE_C</code> <code>#define C_MIN 0</code> <code>#else</code> <code>#define C_MIN 1</code> <code>#endif TYPE_C</code> <i>Note:</i> some preprocessors do not accept this rule.</p> | | |
| [DG-8] | <p>a blank character should precede the first open bracket of a set of brackets. <i>Note:</i> the rule is not applicable to macro with parameter, if open bracket shall be joined to the macro's name. <i>Examples:</i> <code>return= function (parameter);</code> <code>if ((Val_1 > C_1) && (Val_2 < C_2))</code></p> | optional | |
| [DG-9] | <p>following operators should be joined to the variable: ++, --, *, &, ~, cast. <i>Examples:</i> <code>int *PtrInt;</code> <code>PtrInt = &MyInt;</code> <code>Data1 = ~MyInt;</code> <code>MyInt++;</code> <code>MyFloat = (float)MyInt;</code></p> | optional | |
| [DG-10] | <p>operators to access structures and unions (., ->) should be joined to the variable or structure field. <i>Example:</i> <code>MyData = MyStructure.Field1;</code></p> | optional | |
| [DG-11] | <p>" []" should be joined to the index or variable. <i>Example:</i> <code>Value = Tab1[IndX][IndY] +</code> <code>Tab2[IndX][IndY];</code></p> | optional | |
| Naming | | | |
| [NG-1] | identifier names should be explicit of their purpose. | optional | |
| [NG-2] | <p>except specific cases, identifiers should mix characters typology (uppercase for 1st letter of each word, lowercase for other), underscore between words. <i>Example:</i> <code>Set_Low_Temp</code></p> | optional | |
| [NG-3] | <p>There should not exist two identifier names identical except for the case. <i>Example:</i> <code>VMS_Process</code> and <code>Vms_process</code></p> | optional | |
| [NG-4] | <p>All defines shall be in uppercase. <i>Example:</i> <code>#define MAX(100)</code></p> | optional | |
| General | | | |
| [SG-1] | coding shall be compliant with C-ANSI standard. | mandatory | |
| [SG-2] | Type and constant definitions shall be as maintainable | mandatory | |

| RULE | DEFINITION | LEVEL | Compliance |
|-------------|---|-----------|------------|
| | and localized as possible. Particularly, types and constants shall not be re-defined, but defined in a centralized file shared between all the consumers. | | |
| Types | | | |
| [ST-1] | <p>enumerated values shall be handled with variable using identical enumerated type.</p> <p>Example: if enumerated type is defined:</p> <pre>typedef enum { VAL_1=0, VAL_2=1 }TYPE_Enum;</pre> <p>Declaration shall be: TYPE_Enum Variable = VAL_1;</p> <p>Declaration shall not be: Int Variable =VAL_1;</p> | mandatory | |
| [ST-2] | <p>types and type structures, enumerated should be defined through "typedef" instruction.</p> <p>Example: typedef struct { int Field1; int Field2; }TYPE_Structure;</p> <p>TYPE_Structure MyStructure;</p> <p>do not code as this:</p> <pre>struct TYPE_Structure { int Field1; int Field2; }TYPE_Structure; struct TYPE_Structure MyStructure;</pre> | optional | |
| Data | | | |
| [SD-1] | <p>type definition and variable definition shall be separated.</p> <p><i>Status Mandatory</i> Note: this rule is justified to enforce analysis for type definition and to limit volume of types</p> | mandatory | |
| [SD-2] | <p>each variable shall be declared and initialized independently of other.</p> <p><i>Status Mandatory</i></p> <p>Example:</p> <pre>TYPE_MyType Data_1=0; TYPE_MyType Data_2=0;</pre> <p>do not code as this:</p> | mandatory | |

Réalisation de la partition embarquée
Vehicle Management System (VMS)
de l'hélicoptère COUGAR dans le cadre de sa certification

| RULE | DEFINITION | LEVEL | Compliance |
|-------------------|---|-----------|------------|
| | TYPE_MyType Data_1 = 0, Data_2 = 0; | | |
| [SD-3] | variables shall be declared "local" if not used outside the module. Local variables shall be defined at the beginning of the function/procedure rather than in the several coding blocs | mandatory | |
| [SD-4] | dynamic memory allocation shall be forbidden | mandatory | |
| [SD-5] | the directive "register" shall be forbidden. | mandatory | |
| [SD-6] | size of table should be symbolic constants (const variable). | optional | |
| [SD-7] | numeric values shall be handled through constants. <i>Note:</i> objectives are to avoid duplication of numeric values in order to ease maintenance. Possible solutions are: variables (using "const" instruction) defined in global or local specification files symbolic constants (using "#define") defined in specification files definition can be local to a module or function/procedure if data is internal to the module or function/procedure | mandatory | |
| Macros | | | |
| [SM-1] | for function macro, parenthesis shall be used for parameters and substitution expression <i>Example:</i> #define M_SUM(a,b) ((a) + (b)) | mandatory | |
| [SM-2] | a function macro shall not change its parameters. <i>Example:</i> following macro is forbidden: #define M_INCREASE(a) ((a)++) | mandatory | |
| [SM-3] | a function macro shall not implement implicit access to a structure field. <i>Example:</i> following macro is forbidden : #define M_LENGTH(text) ((text).Length) | mandatory | |
| [SM-4] | C keywords shall not be redefined. <i>Example:</i> following macro is forbidden : #define float double | mandatory | |
| Expression | | | |
| [SE-1] | parenthesis shall be used for deterministic order of operation evaluation. | mandatory | |

| RULE | DEFINITION | LEVEL | Compliance |
|--------|---|-----------|------------|
| [SE-2] | type conversion shall be defined. | mandatory | |
| [SE-3] | <p>multiple affectation shall be forbidden. <i>Example:</i> appropriate coding is: Site=C_PI; Gis=C_PI;</p> <p>not appropriate coding is: Site = Gis = C_PI;</p> | mandatory | |
| [SE-4] | <p>implicit affectation shall be forbidden either in condition or through increment/decrement operation (++ , --). <i>Example:</i> appropriate coding is:</p> <pre> if (Counter < C_MAX_COUNTER) { Counter++; <instructions>; } </pre> <p>not appropriate coding is: if (Counter++ <= C_MAX_COUNTER)</p> <pre> { <instructions>; } </pre> | mandatory | |
| [SE-5] | <p>affectation shall be forbidden for function/procedure/macro call. <i>Example:</i> appropriate coding is:</p> <pre> i=0; toto(i); </pre> <p>not appropriate coding is: toto(i=0);</p> | mandatory | |
| [SE-6] | <p>index shall be tested before any table access. <i>Note:</i> to access a table through a loop index, it is useless to perform the test of the loop index.</p> | mandatory | |
| [SE-7] | <p>absolute comparison (==, !=) shall be forbidden for float types. <i>Note:</i> this comparison raises 2 problems: - validity of algorithm - Validity of representation for portable code Only comparison to particular values can be done (ex: 0.0).</p> | mandatory | |

| RULE | DEFINITION | LEVEL | Compliance |
|-----------------|---|-----------|------------|
| [SE-8] | <p>an expression shall not modify a variable used in the expression</p> <p><i>Example:</i> following expression is forbidden: var_b = (2 + var_a++);</p> | mandatory | |
| Function | | | |
| [SF-1] | <p>function type shall be defined.</p> <p><i>Example:</i> appropriate coding is: extern int MyFunction (int v_Argument); extern void MyFunction (int v_Argument);</p> <p>not appropriate coding is: extern MyFunction (int v_Argument);</p> | mandatory | |
| [SF-2] | <p>recursive function call shall be forbidden.</p> <p><i>Note:</i> indirect recursive call shall be verified, means iteration in the call graph (function 1 calls function 2, function 2 calls function 1).</p> | mandatory | |
| [SF-3] | optional arguments in a function prototype shall be forbidden | mandatory | |
| [SF-4] | "Return" statement shall be implemented for any functions to return a status or value, except for "void" functions | mandatory | |
| [SF-5] | <p>a function shall implement 1 "return" only.</p> <p><i>Note:</i> to simplify the algorithm for error cases, specific "return" may be added in this case.</p> | mandatory | |
| [SF-6] | <p>a function shall not return a pointer to local variable.</p> <p><i>Example:</i> next coding is forbidden :</p> <pre> int *Max (int v_I1, v_I2) { int MaxLocal; if (v_I1 > v_I2) { MaxLocal=v_I1; } else { MaxLocal=v_I2; } return(&MaxLocal); } </pre> | mandatory | |

Réalisation de la partition embarquée
Vehicle Management System (VMS)
de l'hélicoptère COUGAR dans le cadre de sa certification

| RULE | DEFINITION | LEVEL | Compliance |
|--------------------------|--|-----------|------------|
| | <i>Note:</i> result is not deterministic. This error is generally not detected neither by the compiler nor by the syntax analyzer (lint). | | |
| [SF-7] | for output or input/output parameter (passing by reference), the same variable shall not be allocated to 2 formal parameters. <i>Example:</i> if F_1 prototype is: Int F_1 (int v_E1, int v_E2, int *r_S1, int *r_S2); Such coding is forbidden : Return = F_1 (data1, data2, &data3, &data3); | mandatory | |
| [SF-8] | update of output/input-output parameter should be grouped and done before the return instruction (possibly, intermediate local variables used for calculation). <i>Note:</i> this rule improves code lisibility, understanding and robustness (side effects, refer also to [SF-7]). | optional | |
| [SF-9] | for functions returning a status or value, this return value shall be analyzed by calling function/procedure (the function caller must use function result). | mandatory | |
| Control structure | | | |
| [SC-1] | the instruction "goto" shall be restricted for jump to an error treatment bloc, within the function. <i>Note:</i> the use of "goto" can simplify the program structure for error cases. Use for any other contexts may decrease the lisibility of the algorithm. | mandatory | |
| [SC-2] | the instruction "break" shall be forbidden, except for bloc "switch". | mandatory | |
| [SC-3] | the instruction "continue" shall be forbidden. <i>Note:</i> the instruction may be authorized for a loop interruption to manage error case. | mandatory | |
| [SC-4] | each case bloc "switch" shall be finished by a "break". | mandatory | |
| [SC-5] | any bloc "switch" shall use the instruction "default". | mandatory | |
| [SC-6] | any bloc of instructions shall be enclosed through brackets even if not required by C syntax. <i>Example:</i> appropriate coding: if (Value > Treshold) { Treshold_Max = TRUE; } not appropriate coding: | mandatory | |

Réalisation de la partition embarquée
Vehicle Management System (VMS)
de l'hélicoptère COUGAR dans le cadre de sa certification

| RULE | DEFINITION | LEVEL | Compliance |
|---------|---|-----------|------------|
| | if (Value > Treshold) Treshold_Max = TRUE; | | |
| [SC-7] | modification of a loop index in the loop shall be forbidden. | mandatory | |
| [SC-8] | no assumption shall be done on a loop index value after the loop instruction (determinism). <i>Note:</i> use an intermediate variable to be incremented in the loop, to memorize loop index value. | mandatory | |
| [SC-9] | number of nested levels shall be limited to 5. <i>Note:</i> the 1 st bloc to be measured is the 1 st instruction bloc of the function.. | mandatory | |
| [SC-10] | the operator " ? : " shall be forbidden. <i>Note:</i> an instruction using this operator cannot be instrumented by some structural test tool(s). | mandatory | |

Résumé

En mars 2010, ma société s'est vu confier le projet AMC+ : Le calculateur embarqué du super puma qui gère les données moteur et véhicule doit être remplacé. L'AMC fabriqué par General Electric Company laisse sa place à l'AMC fabriqué par Elbit Systems. Le matériel devant donc changer, Eurocopter décide de modifier aussi la partition VMS installée (codée en C) pour la ré-implémenter au nouveau standard en vigueur : SCADE.

L'hélicoptère Cougar, équivalent militaire du Super Puma, doit être équipé d'un calculateur équivalent à l'AMC+. Ce calculateur porte le nom de MK1+ et le projet démarre en parallèle du projet AMC+.

Le but final au projet est la certification de l'hélicoptère par les autorités compétentes en juillet 2013. La norme DO-178B/ED-12B, qui fixe les conditions de sécurité applicables aux logiciels critiques de l'aviation doit donc être respectée tout au long du processus.

L'enjeu du projet est de réaliser le MK1+ en tenant compte du retour d'expérience du projet AMC+, ceci afin de respecter le planning.

Mots clés

AMC, SCADE, MK1, NORME DO-178B, PROCESSUS, PLANIFICATION, QUALITE, DEVELOPPEMENT, TESTS, GESTION DE CONFIGURATION, CALCULATEUR EMBARQUE.

Abstract

In March 2010, my company obtained the AMC+ project: Embedded computer of Super Puma which manages engine and vehicle data must be replaced. The AMC manufactured by General Electric Company is replaced by the AMC manufactured by Elbit Systems. In the same time, Eurocopter also decided to change the VMS partition installed (coded in C) to re-implement it in the new standard in place : SCADE.

The Cougar helicopter, military equivalent of the Super Puma, must be equipped with a similar board computer than AMC+. This computer is named MK1+ and this project starts in parallel of AMC+.

The final objective of the project is the helicopter certification by the competent authorities in July 2013. DO-178B/ED-12B standard which sets the security requirements applicable to critical aviation software must be applied throughout the process.

The aim of the project is to achieve the MK1+ taking into account the feedback from the AMC + project, in order to meet the schedule.