Vers une certification continue des logiciels critiques en aéronautique

Vincent Louis, Claude Baron

Vers une certification continue des logiciels critiques en aéronautique

Vincent LOUIS DGA Techniques aéronautiques
47 rue Saint-Jean 31131 Balma France
vincent1.louis@intradef.gouv.fr

Claude BARON INSA LAAS-CNRS
7 avenue du colonnel Roche 31031 Toulouse France
cbaron@laas.fr

Résumé— De nombreux secteurs utilisent des systèmes et des logiciels critiques, dont les défaillances pourraient entraîner des pertes de vies humaines. Cet article s'intéresse au développement et à la certification des logiciels critiques, avec une illustration dans le domaine de l'aéronautique. Il souligne les difficultés rencontrées en entreprise et l'état des pratiques pour mettre en perspective quelques pistes d'améliorations possibles de celles-ci. Il démontre notamment l'intérêt d'intégrer plus étroitement et de façon continue les contraintes de la certification au processus de développement des logiciels. Notre proposition est de nous inspirer des valeurs agiles pour répondre le plus tôt possible aux exigences de certification applicables au développement des logiciels. C'est ce que nous appelons la certification continue.

Mots clefs—Agilité, Ingénierie du Logiciel, Processus de Développement, Assurance Qualité, Certification, Aéronautique, Systèmes Embarqués, Standards, DO-178 C, DO254, IEC 61508

I. INTRODUCTION

En aéronautique mais plus largement dans les transports, la production d'énergie, le nucléaire, la finance, la santé ou les applications militaires, de nombreux systèmes sont qualifiés de 'critiques'. La criticité[1] d'un système est définie en regard des conséquences qu'une défaillance de ce système pourrait entrainer des pertes de vie humaines ou une destruction de matériel. Quand une fonction d'un système est jugée critique, une autorité, comme l'European Aviation Safety Agency (EASA) pour les aéronefs civils, la Direction Générale de l'Armement (DGA) pour les aéronefs d'Etat militaires et gouvernementaux, ou l'Autorité de Sûreté Nucléaire (ASN) pour le domaine nucléaire, exige généralement que le processus de développement du système ait suivi une méthodologie dont les démonstrations sont reconnues comme acceptables. Cette autorité, ou organisme de certification, réalise ou fait réaliser un ou plusieurs audits afin de s'assurer de la conformité du processus de développement industriel avec les objectifs prescrits dans les référentiels normatifs reconnus dans le domaine concerné. En aéronautique, seuls l'aéronef, une hélice ou un moteur sont officiellement certifiés. Cependant, avec l'intensification de l'utilisation de logiciel dans les systèmes embarqués critiques (pour la supervision, le contrôle, la commande, la sécurité…), il est indispensable d'appliquer des objectifs spécifiques au développement de systèmes dont les fonctions sont essentiellement supportées par les logiciels ; on parlera par abus de langage de 'certification logicielle'.

L'efficacité de la démarche de certification est reconnue et elle est incontournable dans les domaines critiques réglementés. Ces évaluations indépendantes, demandées par l'autorité de certification, sont primordiales pour garantir à l'environnement du système et à ses futurs utilisateurs un niveau de confiance conforme à celui attendu pour le contexte d'emploi. Dans notre société actuelle, le rôle de l'autorité est de se porter garant devant les tiers qu'un système critique peut être mis en service avec un niveau de risque acceptable pour éviter de mettre en danger la vie d'autrui. C'est un tiers de confiance qui délivre des certificats.

Des normes ont été écrites pour définir des activités techniques et des processus avec pour objectifs de détecter des erreurs avant qu'elles ne soient plus rattrapables. Cependant, on constate que la mise en œuvre des référentiels normatifs par les industriels est souvent considérée comme très lourde et très coûteuse [Hilderman 2007] [Hilderman 2009]. La certification est souvent vue comme une contrainte, induisant des activités complémentaires jugées superflues et engendrant des surcouts, comme la rédaction de document de spécification du logiciel, la conduite de revues (d'exigences, de procédures et de résultats de test), et la gestion en configuration de l'ensemble des éléments d'ingénierie. En pratique, ces activités sont très peu souvent conduites par les industriels en phase de prototypage. Elles sont souvent conduites quelques temps avant les audits de certification. Certes, la certification a un coût, puisqu'il est vrai qu'elle entraîne des activités complémentaires à un processus de développement standard, mais « faire de la certification » au dernier moment, après une phase de prototypage, présente un risque encore plus fort d'augmentation des coûts dus aux activités de rétro-ingénierie.

Ces activités de certification mériteraient d'être conduites au fil du développement pour en minimiser l'impact financier.

[1] La criticité (C) des fonctions assurées par un système est établie en regard des événements redoutés qui portent sur ces fonctions. Elle est calculée en prenant en compte la fréquence d'occurrence (F) des évènements redoutés, leur gravité (G) et leur détectabilité (D) : C = F * G * D.

En outre, être capable d'apporter de manière continue des garanties de conformité rendrait le processus non seulement moins couteux mais aussi plus performant. Il serait donc intéressant de mieux intégrer les activités requises par la certification dans le processus de développement pour qu'elles soient réalisées plus efficacement, et avec un effort moindre.

Face à cet état des pratiques, le congrès américain incite l'autorité américaine de certification la Federal Aviation Administration à alléger les contraintes afin de réduire les coûts. Ceci pourrait cependant se faire au détriment de la sécurité. Une alternative à cette option serait de plutôt chercher à faciliter la certification, par exemple en l'intégrant de façon plus naturelle aux processus de développement des systèmes, et en l'outillant.

Cet article traite de la problématique de la certification, en soulignant son utilité sur le plan sociétal. Il offre l'intérêt et l'originalité de donner le point de vue d'un membre d'un organisme de certification. Il adresse ensuite plus spécifiquement le développement de logiciels critiques, avec une illustration dans le domaine de l'aéronautique. Il souligne les difficultés rencontrées en entreprise et l'état des pratiques pour mettre en perspective quelques pistes d'améliorations possibles de celles-ci. Il démontre notamment l'intérêt d'intégrer plus étroitement les contraintes de certification au processus de développement des systèmes de façon continue.

Notre proposition est d'aller au-delà et de s'inspirer des valeurs agiles (produire régulièrement des livrables opérationnels, accepter des changements de besoins, valoriser les interactions entre les individus) pour être capable d'apporter plus tôt dans le développement et de façon continue, des garanties de conformité aux normes. C'est ce que l'article appelle « Certification Continue ».

La section II positionne le contexte de ce travail, introduit la problématique générale de la certification et propose un état des pratiques industrielles dans différents domaines d'activité (automobile, santé, nucléaire…). La section III rappelle les pratiques actuelles en matière d'ingénierie logicielle et explique en quoi les objectifs de certification contraignent le processus de développement logiciel en aéronautique (objectifs, référentiels, contraintes induites, etc..). Cette section analyse également comment la certification est vécue par les industriels du secteur. La section IV développe quelques pistes permettant de mieux intégrer les exigences de certification au processus de développement logiciel. La section V démontre qu'il est possible d'introduire de l'agilité dans le processus de développement de logiciels critiques soumis à certification. L'article conclut sur la nécessité et l'intérêt de faire évoluer les pratiques en les rendant plus agiles et en s'appuyant sur des approches basées modèles ; elle étend ces conclusions pour la certification de logiciels critiques à des domaines autres que l'aéronautique ou même pour des systèmes non logiciels.

II. CONTEXTE ET ETAT DES PRATIQUES

Notre société prévoit, qu'en cas d'incident, la responsabilité soit répartie entre les différentes parties prenantes : le concepteur du système, son utilisateur et/ou l'autorité qui en a autorisé la mise en service en garantissant un risque acceptable[2] en regard de la mise en danger de la vie d'autrui [JORF 2013].

Après un rappel sur le concept de safety (Section A), la section II montre l'importance sociétale de la certification, introduit le rôle des autorités et explique de façon générique comment celles-ci s'assurent de la mise en conformité des systèmes industriels vis-à-vis de la réglementation (Section B). Elle précise ensuite les missions, les rôles et la responsabilité de la Direction Générale de l'Armement dans ce contexte (Section C). Puis elle propose un panorama des processus de certification dans différents domaines métiers (Section D) et détaille la réglementation en aéronautique (Section E). Elle analyse ensuite les postures et pratiques actuelles des industriels dans un contexte de certification (Section F). Elle introduit pour finir la problématique considérée ainsi que les objectifs de ces travaux de recherche (Section G).

A. Introduction à la safety

La *safety*[3] fait appel à la théorie des systèmes et l'ingénierie des systèmes pour prévenir les accidents prévisibles et réduire au minimum les conséquences d'accidents d'imprévus. La safety considère les pertes en vie humaines ou les blessures, la destruction de biens, la perte d'une mission et des dommages environnementaux [Leveson 2003]. Il s'agit d'une approche planifiée, disciplinée et systématique pour identifier, analyser, évaluer, éliminer et contrôler les dangers tout au long du cycle de vie d'un système afin de prévenir ou de réduire les accidents.

Les normes relatives à la safety prescrivent un ensemble d'étapes, de documents livrables et de critères de sortie axés sur la planification, l'analyse et la conception, la mise en œuvre, la vérification et la validation, la gestion de configuration et l'assurance qualité pour le développement d'un système critique [Rempel 2014]. En outre, elles formulent généralement des attentes concernant la création et l'utilisation de la traçabilité dans le projet.

Les activités liées à la safety commencent dès les premières étapes de l'élaboration du concept dans un projet et se poursuivent tout au long de la conception, de la production, des tests, de l'utilisation opérationnelle et du retrait de service. L'un des aspects intéressant de la safety est de mettre l'accent sur l'identification et la classification précoces des dangers afin que des mesures correctives puissent être prises pour éliminer ou minimiser ces dangers avant que les décisions de conception finales ne soient prises. Certains principes généraux la distinguent des autres approches de la safety et de la gestion des risques. Parmi eux :

- construire la safety en parallèle du système et ne pas la traiter à la fin de la conception du système ;
- privilégier des approches qualitatives plutôt que quantitatives ;

[2] On parle de 'risque acceptable' car il n'est pas possible d'atteindre un niveau de risque zéro

[3] Nous préférons conserver le terme anglais de safety pour éviter les confusions possibles entre sécurité (security) et sûreté (safety)

- rien n'est absolument sûr; la safety est rarement le but premier dans la conception de systèmes, des conflits surviennent et des compromis sont nécessaires.

À l'aide de ces principes généraux, les analyses safety tentent de gérer les dangers au moyen de procédures d'analyse, de conception et de gestion.

Les principales activités comprennent [Leveson 2003]:

- des analyses descendantes des dangers relatifs au système, dès la conception initiale, pour éliminer ou contrôler les risques, et ensuite durant toute la durée de vie du système, pour évaluer les modifications du système ou de l'environnement,
- la documentation, le suivi des dangers, et leur résolution (avec traçabilité),
- la conception pour éliminer ou contrôler les risques et minimiser les dommages,
- la maintenance des systèmes et documents d'information de sécurité, la création de rapports et d'information.

Les industries utilisent différentes approches pour assurer un haut niveau de safety ; cependant, les techniques d'analyse reposent uniquement sur les compétences et l'expertise de l'ingénieur safety. On peut diviser les approches en deux familles : les méthodes qualitatives et les méthodes quantitatives. Les deux approches partagent l'objectif de trouver des dépendances (relations de cause à effet) entre un danger au niveau du système et des défaillances de composants. Les approches qualitatives se concentrent sur la question "Que doit-il se passer pour qu'il n'y ait pas de risque systémique ?", tandis que les méthodes quantitatives visent à fournir des estimations sur les probabilités, les taux et/ou la gravité des conséquences des défaillances. Les approches traditionnelles les plus courantes en matière de safety sont l'analyse des modes de défaillance et de leurs effets [Leveson 1995] ainsi que l'analyse des arbres de défaillances [Wessiani 2018]. Ces techniques ne constituent que des moyens de trouver des problèmes et d'élaborer des plans pour faire face aux défaillances, comme dans l'évaluation probabiliste des risques. Elles sont mises au défi par l'introduction de nouvelles technologies et par la complexité croissante des systèmes que nous cherchons à construire.

Nous avons besoin d'une compréhension meilleure et moins subjective des causes des accidents et des moyens de les prévenir à l'avenir. Les modèles les plus efficaces doivent permettre d'approfondir l'analyse pour aller au-delà de l'attribution des responsabilités et aider les ingénieurs à en apprendre le plus possible sur tous les facteurs impliqués, y compris ceux liés aux structures sociales et organisationnelles [Leveson 2004].

Au cours de la dernière décennie, les approches basées sur des modèles sont devenues prédominantes. Contrairement aux méthodes traditionnelles, elles permettent aux ingénieurs système et aux ingénieurs safety d'améliorer leurs interactions par le partage de modèles du système étendus aux comportements dysfonctionnels. La synchronisation de ces modèles a pour objectif d'établir une cohérence d'ensemble pour la conception de systèmes critiques complexes [IRT 2018]. L'utilisation de modèles et l'automatisation de certaines parties de l'analyse safety permettent à la fois de réduire le coût et d'améliorer la qualité des analyses safety.

B. Certification de systèmes critiques : pourquoi, comment ?

D'un point de vue sociétal, il est nécessaire d'avoir un avis contradictoire quant à la mise en service d'un système critique pouvant conduire à la perte de vie humaine. Lorsqu'un accident conduit à une enquête judiciaire ou à un dédommagement par une assurance, la responsabilité est souvent portée par l'opérateur du système : accident de la route, crash d'un aéronef, et avec l'augmentation des automatismes (véhicule autonome, drones) le nombre d'accidents va diminuer mais la responsabilité va peu à peu se déporter de l'opérateur vers le concepteur du système ou du logiciel.

Dans le domaine automobile, le groupe d'assurance AXA a produit un document pour clarifier la responsabilité selon plusieurs scénarios lors d'un accident impliquant un véhicule autonome [Salmon 2018]. La Convention de Vienne, qui définit les règles internationales pour la circulation routière, stipule que le conducteur doit toujours rester maître de son véhicule mais autorise depuis 2016 les systèmes automatisés "à condition qu'ils puissent être contrôlés voire désactivés par le conducteur". Plusieurs véhicules autonomes circulent donc déjà sur les routes, mais pour l'instant à titre expérimental. En France, l'ordonnance n° 2016-1057 du 3 août 2016 permet ainsi la circulation "à des fins expérimentales d'un véhicule à délégation partielle ou totale de conduite", sur délivrance d'une autorisation du ministre des Transports [JORF 2016]. Cette tendance implique la définition d'une réglementation qui devra être respectée par les concepteurs et évaluée de façon indépendante. Des travaux sont en cours pour réglementer la conception de véhicule autonome par différentes autorités nationales [DGT 2018] [Macron 2018].

Dans le domaine aéronautique, le processus de certification consiste à justifier les exigences réglementaires préconisées pour chacun des types d'aéronef (Certification Spécification : 25 - Large Aircraft, 29 - Large Rotorcraft) auprès d'une autorité de certification : l'EASA pour les aéronefs civils, la DGA pour les aéronefs d'état miliaires et gouvernementaux ou de son délégataire. La certification s'applique aux aéronefs, aux moteurs ou aux hélices. Les autorités de certification considèrent le logiciel comme faisant partie du système ou de l'équipement embarqué installé sur le produit certifié. Autrement dit, les autorités de certification ne certifient pas le logiciel en tant que produit unique et autonome. Les systèmes et équipements, y compris les logiciels embarqués, doivent être approuvés pour pouvoir être acceptés dans le cadre d'une certification. L'approbation par les autorités de certification dépend de la réussite de la démonstration ou de l'examen des produits du cycle de vie du logiciel, on parle de certification logicielle.

Le rôle d'une entité indépendante d'évaluation se justifie pleinement. Par contre, il est inconcevable d'imaginer une totale maîtrise technique du système complexe expertisé par un évaluateur externe ou d'une implication d'une autorité aussi avancée que les concepteurs du système ou du logiciel. Le concepteur du système reste responsable des choix techniques retenus même si l'autorité de certification peut parfois être remise en cause en cas de manquement dans l'exercice de ses

attributions[4]. Le processus de certification permet de rendre efficace cette évaluation à travers l'échantillonnage d'un sous-ensemble des données d'ingénierie. Par extrapolation, il permet de rendre compte d'une maîtrise globale du processus de développement et de vérification.

C. Mission de la Direction Générale de l'Armement en tant qu'Autorité Technique

Au-delà de ses missions principales, équiper les armées de façon souveraine, préparer le futur des systèmes de défense, promouvoir la coopération européenne et soutenir les exportations, l'une des missions de la DGA est d'assurer, par délégation ministérielle, le rôle d'Autorité Technique pour l'ensemble des aéronefs d'états, dont elle vérifie la conformité des systèmes aux exigences réglementaires. Cette responsabilité est assurée par des personnels DGA indépendants, du fait des divers intérêts en jeu ; ils ont pour mission de garantir la sécurité des biens et des personnes. Pour le domaine aéronautique civil, c'est l'EASA qui exerce le rôle d'autorité depuis sa création en 2002. Pour le domaine aéronautique militaire, la DGA a depuis 2006, la responsabilité de la certification de type [5] et du suivi de navigabilité[6] des aéronefs d'état (avions et drones du ministère des armées, de la gendarmerie, des douanes et présidentiels). En effet, le décret de navigabilité 2006-1551 du 7 décembre 2006 [Regulation 2016] abrogé par le décret 2013-367 du 29 avril 2013 [JORF 2013], a transposé la règlementation civile aux aéronefs d'état. Depuis cette date, l'ensemble de la flotte des aéronefs gouvernementaux est donc soumis aux exigences réglementaires de certification. Pour mener ces activités, la DGA s'appuie principalement sur les compétences de l'EASA en matière de certification de fonctions civiles et réalise elle-même le complément de certification pour les aéronefs militaires de base civile et pour les aéronefs sans équivalents civils. La DGA est également ponctuellement mandatée par l'EASA pour conduire des activités de certification dans le domaine civil.

La DGA expertise donc des systèmes issus de domaines différents : spatial, aéronautique, terrestre, naval, médical, missile, drone. Chacun des domaines ayant défini son propre référentiel normatif en matière de développement logiciel, la DGA est régulièrement confrontée à des niveaux de maturité extrêmement variables. Certains industriels historiques sont habitués depuis des décennies à appliquer des processus rigoureux pour développer leurs logiciels critiques, quand d'autres viennent tout juste de mettre en place une première méthode structurée de travail. La DGA voit aussi se positionner sur des marchés de défense de nouveaux entrants (start-up) qui n'ont jamais eu à démontrer le niveau de fiabilité de leur développement informatique.

D. Panorama des normes et processus de certification

Comme indiqué en introduction de nombreux domaines disposent d'une autorité de contrôle pour certifier les logiciels critiques. L'autorité doit valider les bases de certification qui décrivent les méthodes retenues par les industriels pour atteindre les objectifs réglementaires. Son rôle sera aussi de définir, à travers un document appelé Certification Review Item, les adaptations nécessaires pour traiter les nouvelles problématiques non couvertes par les règlements existants. Seule la bonne application d'un processus d'ingénierie permet de s'assurer que le produit répondra aux objectifs de sûreté. Des audits permettent de vérifier le contenu technique produit par les processus mis en place. Lors de l'évaluation, une part non négligeable de l'avis de l'auditeur repose sur la confiance qu'a réussi à donner le candidat à la certification (*applicant*). Même si l'audit ne peut pas être exhaustif car il porte juste sur un échantillon des données d'ingénierie, cette méthode aléatoire est jugée satisfaisante. Lors de ce type d'exercice, il est nécessaire pour l'applicant de réussir à présenter sa capacité à concevoir son logiciel, à gérer les problèmes et dimensionner les ressources pour répondre à tous les objectifs réglementaires. Ces objectifs seront déterminés par le niveau de criticité du logiciel établi sur la base d'une analyse de niveau système qui identifie la contribution des logiciels aux scénarios redoutés (cf. Section E). Une équipe qui applique des processus rigoureux n'a aucune difficulté à exposer les preuves requises pour la certification. Dans le cas contraire, la démonstration peut s'avérer difficile et l'auditeur trouve facilement des écarts pouvant être rédhibitoires pour la certification.

Ces processus à appliquer sont décrits dans de nombreuses normes[7] adaptées à chaque domaine. A titre d'exemple, l'IEC 61508 [IEC 61508 2010] répond aux attentes du domaine électronique industriel, la norme DO-178C /ED12C [RTCA DO-178C 2012] s'applique au développement des logiciels aéronautiques, l'ISO 26262 [ISO 26262 2018] traite les logiciels du domaine automobile, l'IEC 62304 [IEC 62304 2012] est propre au domaine médical ou encore les normes ECSS EQ-60 [ECSS EQ-60 2013] et EQ-80 [EQ-80 2009] décrivent les pratiques attendues pour le développement de logiciels spatiaux. Toutes ces normes décrivent les étapes à suivre pour démontrer qu'un logiciel implémente correctement les fonctions attendues.

---

[4] Pour exemple, l'administrateur de la FAA (Federal Aviation Authority) a dû justifier devant le Congrés Américain des activités conduites par son administration dans le cadre de la certification du système MCAS du Boeing 737 Max impliqué lors deux crashes : vol Lion Air n°610 (189 morts) le 29 octobre 2018, et vol Ethiopian Airlines n°303 (157 morts) le 10 mars 2019.

[5] Un certificat de type est propre à une catégorie particulière d'aéronefs. Il confirme que l'avion est construit selon une conception approuvée conformément à la fabrication "type" et que la conformité aux exigences de navigabilité est assurée.

[6] La navigabilité est définie comme l'aptitude d'un aéronef (quel que soit le type : avion, hélicoptère, planeur, drone, etc.) à effectuer sa mission dans des conditions acceptables de sécurité vis-à-vis des passagers et de l'équipage, des populations survolées et des autres usagers de l'espace aérien.
En Europe, la navigabilité, est régie par des règles appelées PART (Partie du décret). Ces règles définies par l'Agence Européenne de la Sécurité Aérienne (EASA) visent à garantir « un niveau élevé et uniforme de sécurité aérienne » en Europe. Dans le reste du monde chaque pays a sa propre réglementation à l'exemple des FAR (Federal Aviation Regulations) édités par la FAA aux États-Unis.
La navigabilité s'organise autour de trois fonctions : le suivi, le contrôle et le maintien (ou suivi) de la navigabilité. Le suivi de navigabilité concerne les agréments d'organismes indépendants (ou du transporteur aérien lui-même) pour la maintenance et le suivi administratif des aéronefs. Ils ont notamment pour tâche d'établir des programmes d'entretien d'aéronefs et de commander ou de coordonner les travaux de modification, de réparation ou d'entretien nécessaires. Ils réalisent également l'examen technique périodique des aéronefs et gèrent les dossiers techniques des appareils.

[7] On appelle norme dans l'article, des guides de bonnes pratiques ou de recommandations à caractère normatif.

Les grandes lignes en sont complètement similaires (même si de légères différences existent).

Dans l'industrie, les systèmes critiques sont souvent développés conformément à l'IEC 61508 [IEC 61508 2010]. Il s'agit d'une méta-norme qui doit être particularisée suivant les contraintes et les objectifs de chaque domaine. Cette norme générique a été déclinée dans de nombreux métiers tels que l'ISO 26262 [ISO 26262 2018] pour le domaine automobile, l'EN50128 [EN50128 2011] pour le domaine ferroviaire ou l'IEC 61511 [IEC 61511 2016] pour l'industrie pétrochimique. Ensuite, cette déclinaison fait l'objet d'évaluations indépendantes par des organismes reconnus tels que TUV (Technischer Überwachungs-Verein) ou Bureau Veritas qui vérifient la conformité des développements aux objectifs de la norme. L'IEC 61508 est donc intéressante par son caractère générique mais elle présente quelques inconvénients. Elle est bien écrite pour concevoir une fonction de sécurité ajoutée à un système existant mais traite mal la conception d'un nouveau système complet. Par ailleurs, elle mélange les aspects quantitatifs et qualitatifs pour attribuer une criticité au logiciel ce qui peut parfois surprendre car si un défaut est présent dans un logiciel, la probabilité de défaillance en cas d'activation sera forcément de 1 du fait du comportement déterministe des logiciels. Les auteurs de la norme comptent la faire évoluer pour clarifier la manière d'allouer la criticité de chaque logiciel en s'inspirant du concept de coupes minimales fonctionnelles [ARP4754A 2011].

Dans le domaine nucléaire, l'Autorité de Sûreté Nucléaire s'appuie sur l'IRSN (Institut de RadioProtection et de Sûreté Nucléaire) pour conduire les évaluations des logiciels critiques selon l'IEC-61513 [IEC-61513 2013] et l'IEC 60880 [IEC 60880 2013].

Dans le domaine ferroviaire, l'organisme Certifier a pour mission d'évaluer la conformité des développements logiciels critiques en regard des normes EN 50126 – EN 50128 – EN 50129.

Dans le domaine médical, l'ANSM (l'Agence Nationale de Sécurité du Médicament) assure l'évaluation des dispositifs médicaux électroniques selon l'IEC-62304 [IEC-62304 2012]. Elle émet simplement des recommandations auprès des industriels du domaine. La conformité à ces recommandations n'est pas systématiquement évaluée par audit. La responsabilité est principalement portée par le fabricant. L'agence sera mise en cause uniquement si elle reste inactive à la suite d'un incident. L'étude conduite par SERMA Ingénierie et présentée dans le rapport [SERMA 2016] répertorie plusieurs axes d'amélioration de la norme suite à l'analyse de plusieurs incidents. Le rapport préconise d'introduire un nouveau chapitre dans la norme IEC-62304 pour décrire un processus d'évaluation pour la certification et un autre chapitre pour préciser les attentes en matière de conception détaillée et technique de test (couverture structurelle). La norme actuelle n'est pas assez précise sur ces points et mérite d'être améliorée compte tenu des enjeux à venir en matière de développements logiciels critiques médicaux (hôpital numérique, télémédecine et télé-chirurgie, objets connectés, nouvelles technologies : nano-implants, multi-cœur, virtualisation, …). A l'heure actuelle, l'ANSM ne conduit pas d'audit pour évaluer la conception des logiciels. Seuls les fabricants sont responsables en cas d'incident grave. L'ANSM engage sa responsabilité uniquement si les mesures appropriées ne sont pas prises suite à un incident.

Dans le domaine spatial, il n'y pas d'audit formel prévu par les référentiels du domaine (standards ECSS de l'European Cooperation for Space Standardization) pour évaluer le développement des logiciels. Par contre, la conformité du développement aux ECSS est assurée par un groupe d'experts indépendants qui conduit plusieurs revues selon les stades d'avancement du projet.

Dans le domaine aéronautique, le processus de certification est incontournable. Ce processus est considéré comme le plus efficace pour qu'une autorité de certification puisse obtenir un niveau de confiance suffisant pour la mise en service d'un système critique. Chaque produit aéronautique (avion, hélice et moteur) doit avoir suivi un processus de certification. Dans ce domaine, la FAA[8] ou l'EASA assurent le rôle d'autorité. Les référentiels existants (DO-178C et ARP4754A) sont réputés robustes pour évaluer la sécurité des systèmes informatiques embarqués.

Dans le domaine automobile, l'industrie applique naturellement le référentiel ISO 26262 pour développer les systèmes critiques. L'UTAC CERAM (Union Technique de l'Automobile, du motocycle et du Cycle- Centre d'Essais et de Recherche Appliqué à la Mobilité) est le seul organisme qui a la délégation de l'état français pour homologuer les véhicules soumis à la réglementation. Il effectue les audits de sûreté de fonctionnement qui sont exigés par l'annexe sur les systèmes complexes de certaines réglementations par exemple le règlement UNECE R13 [UNECE R13 2012] pour la fonction Freinage et le règlement UNECE R79 [UNECE R79 2017] pour la fonction direction. Cet organisme envisage d'étendre son périmètre d'activité et ses ressources à l'évaluation de la safety et de la cyber-sécurité des véhicules autonomes connectés. Des travaux sont en cours et prévoient l'arrivée prochaine en 2019-2020 d'une norme et d'une règlementation automobile relative à la cyber-sécurité définissant des objectifs à respecter et une démarche d'audit sans doute assez similaires et synergiques avec ceux du domaine safety. Ainsi la DGEC (Direction Générale de l'Energie et du Climat), l'UTAC et l'ANSSI (Agence Nationale de la Sécurité des Systèmes d'Information) s'organisent sur ce sujet.

De nombreux acteurs industriels ont compris les enjeux stratégiques quant à la certification des systèmes et logiciels critiques et sont en train de recruter massivement des compétences issues des domaines déjà réglementés (comme l'aéronautique). Plusieurs centaines d'offres d'emploi avec des compétences en safety multi-domaines sont actuellement proposées chez Renault, PSA, EasyMile, Waymo (Autonomous Vehicle Google Startup), Apple, Uber, Tesla et Amazon. Afin d'illustrer l'intérêt de ces acteurs pour la maîtrise de cette

[8] Federal Aviation Administration. Autorité américaine qui établit les règlements pour l'aviation civile appelés FAR (Federal Aviation Regulations)

discipline, la Fig. 1 présente la liste des postes actuellement ouverts chez Uber avec le mot clé « DO-178 ».

Fig. 1. Liste des postes ouverts au centre technologique de la société Uber obtenus avec la recherche par mot clé “DO-178” (Février 2019)

E. Réglementation et exigences en aéronautique

Cette section a pour objet la présentation du cadre réglementaire aéronautique qui a conduit à la définition des différentes normes reconnues applicables par les industriels du domaine. Nous exposerons les principes décrits dans la norme DO-178C/ED-12C et les réflexions actuelles sur leurs évolutions.

Cadre réglementaire

Dans le domaine aéronautique, des travaux initiés en 1910 pour établir des règles communes internationales dans les domaines aéronautique et spatial ont conduit à l’écriture de la convention de Chicago entrée en vigueur le 4 avril 1947 [Convention 1947]. Cette convention couvre divers domaines relatifs aux opérations et utilisations des aéroports, aux espaces survolés, à la conception, à la fabrication, à la maintenance des aéronefs, à leur navigabilité et est à l’origine de la création de l’Organisation de l’Aviation Civile Internationale (OACI[9]), agence spécialisée des nations unies ayant pour mission de coordonner et réguler le transport aérien international. Elle rédige des règlements en réponse à la nécessité sociétale de garantir un haut niveau de sécurité du trafic aérien. Ces règlements sont repris par la Commission Européenne qui a créé l’EASA pour définir les exigences propres à chaque type de flotte d’aéronef.

En effet, les articles de la convention de Chicago doivent être déclinés par la législation des états signataires. La Commission Européenne a établi à travers le règlement N°1592/2002 [EASA-1592 2002] suivi du N°216/2008 (Hard Law[10]) [EASA-216 2008] la transcription des règles communes applicables à l’aviation civile, et description des missions de l’Agence Européenne de la Sécurité Aérienne. Cette dernière a pour mission principale de promouvoir le plus haut niveau possible de sécurité pour l’aviation civile. Elle s’appuie sur des règlements de certification (parmi lesquelles les Certification Specifications for large aeroplanes [EASA CS-25 2007], large rotorcraft [EASA CS-29 2012], des Acceptable Means of Compliance (AMC), et des standards auxquels doivent se conformer les entreprises qui souhaitent certifier leur aéronef.

Après avoir défini des exigences réglementaires et des moyens de conformité, les acteurs du domaine (industriels et autorité) ont convenu de guides pour répondre aux exigences et développer les systèmes et logiciels aéronautiques conformément à la réglementation.

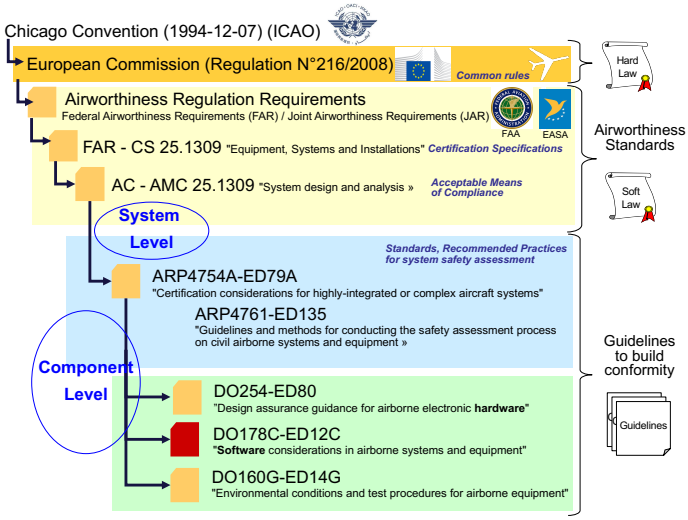

Fig. 2. Cadre règlementaire aéronautique

Dans cet article, nous nous intéresserons plus particulièrement à la certification des aéronefs et de leurs équipements. La Fig. 2 illustre le cadre réglementaire du processus de certification des équipements informatiques (hardware et software) dans le domaine aéronautique.

La CS-25 (amendement 21 du 27 mars 2018, Soft Law[11]) [EASA CS-25 2018] comprend plusieurs sous-parties séparées en deux volets (Book 1: Certification Specifications et Book 2: Acceptable Means of Compliance).

Dans la section SUBPART F – EQUIPEMENT du Book 1, les exigences CS 25.1301 et CS 25.1309 concernent particulièrement la certification des logiciels. La première exige que ‘chaque équipement installé doit être d'un type et d'une conception appropriés à sa fonction prévue’. Le logiciel doit faire ce pour quoi il a été prévu, pas plus, pas moins. La seconde est orientée safety : ‘les systèmes et composants doivent être conçus de manière à ce que tout événement redouté[12] soit

[9] En anglais, International Civil Aviation Organization (ICAO)
[10] Signifie que les lois sont contraignantes
[11] Réfère à des instruments « quasi-juridiques » tels que les résolutions, les déclarations et les lignes directrices élaborées par les gouvernements et les organisations privées. Bien qu'elle ne possède aucun pouvoir juridique et n'exerce aucun mécanisme coercitif, la « loi souple » est néanmoins souvent largement adoptée et généralement suivie [Druzin, 2016].
[12] Dans cet article, nous traduisons le terme ‘failure condition’ par ‘événement redouté’; on trouve parfois d’autres traductions, comme condition de panne, ou scénario de panne.

extrêmement improbable et ne résulte pas d'une défaillance unique'.

Dans le Book 2, des AMC sont proposés pour répondre à chaque exigence. Par exemple, pour l'exigence CS 25.1309 (la 1309e exigence règlementaire…), il est décrit la justification du taux de défaillance reconnu acceptable[13] pour chaque évènement redouté catastrophique (en substance, prendre l'avion ne doit pas être plus risqué que de marcher dans la rue).

L'exigence CS 25.1309 référence les standards applicables (AMC 20-115 pour la norme DO-178C) ou des documents industriels appelés « Aerospace Recommended Practice » (ARP) comme l'ARP 4754A [ARP4754A 2011] et l'ARP 4761 [ARP4761 1996] qui décrivent les méthodes classiques pour conduire les analyses safety.

Les systèmes et les composants, vus seuls ou interconnectés doivent être conçus de telle sorte que l'apparition de défaillances catastrophiques limitant la sécurité du vol ou de l'atterrissage soit « extrêmement improbable » et ne résulte pas d'une panne simple : c'est le concept appelé « fail safe design concept »[14]. Le système doit être conçu pour assurer les fonctions attendues.

La TABLE I. propose une classification des « failure conditions » (ou évènements redoutés) en fonction de la sévérité de leurs effets, sur une échelle de 5 niveaux allant de 'No Safety Effect' à 'Catastrophic'. Ainsi si l'événement redouté entraîne des effets de type « Fatalities or Incapacitation » sur l'équipage ou « Multiple fatalities » sur les occupants ou « Normally with hull loss » sur l'aéronef alors il sera considéré comme « catastrophique ».

TABLE I. RELATION ENTRE LA SEVERITE DES EFFETS ET LA CLASSIFICATION DES EVENEMENTS REDOUTES (BOOK 2 CS25.1309 SECTION 8 §B)

Severity of the Effects						
	Effect on Aeroplane	No effect on operational capabilities or safety	Slight reduction in functional capabilities or safety margins	Significant reduction in functional capabilities or safety margins	Large reduction in functional capabilities or safety margins	Normally with hull loss
	Effect on Occupants excluding Flight Crew	Inconvenience	Physical discomfort	Physical distress, possibly including injuries	Serious or fatal injury to a small number of passengers or cabin crew	Multiple fatalities
	Effect on Flight Crew	No effect on flight crew	Slight increase in workload	Physical discomfort or a significant increase in workload	Physical distress or excessive workload impairs ability to perform tasks	Fatalities or incapacitation
Classification of Failure Conditions		No Safety Effect	Minor	Major	Hazardous	Catastrophic

La TABLE II. établit une relation entre la sévérité des effets d'une « failure condition » (cf. classification précédente) et sa probabilité d'occurrence. Ainsi si l'événement redouté est considéré comme « catastrophique » alors sa probabilité d'occurrence (probabilité quantitative acceptable) devra être inférieure à 10^{-9} par heure de vol et sa probabilité qualitative acceptable devrait être « Extremely improbable ». Ramené au niveau avion qui doit supporter 100 failure conditions catastrophiques et pour lequel 10% de crash sont dûs à des défaillances techniques, il est jugé économiquement et socialement acceptable de perdre un avion tous les millions d'heures de vol (probabilité < 10^{-6}).

TABLE II. RELATION ENTRE LA CLASSIFICATION DES EVENEMENTS REDOUTES ET LA PROBABILITE (BOOK 2 CS25.1309 SECTION 8 §B)

Classification of Failure Conditions	No Safety Effect	Minor	Major	Hazardous	Catastrophic
Allowable Qualitative Probability	No Probability Requirement	<-Probable->	<--Remote-->	Extremely <----------> Remote	Extremely Improbable
Allowable Quantitative Probability: Average Probability per Flight Hour on the Order of:	No Probability Requirement	<----------> $<10^{-3}$ Note 1	<----------> $<10^{-5}$	<----------> $<10^{-7}$	$<10^{-9}$
Note 1: A numerical probability range is provided here as a reference. The applicant is not required to perform a quantitative analysis, nor substantiate by such an analysis, that this numerical criteria has been met for Minor Failure Conditions. Current transport category aeroplane products are regarded as meeting this standard simply by using current commonly-accepted industry practice.					

En ce qui concerne les logiciels, l'enjeu est de minimiser le risque d'introduction d'erreurs lors du développement. En cas d'activation d'une erreur latente, le comportement déterministe des logiciels entraine systématiquement une défaillance. En conséquence, l'option retenue est de contraindre le processus d'ingénierie du logiciel. Ces contraintes se traduisent par des objectifs à respecter décrits dans la norme DO-178C.

La norme DO-178C/ED-12C

Parmi les référentiels utilisés en aéronautique, on trouve notamment la norme DO-178C (Software Considerations in Airborne Systems and Equipment Certification) ou ED-12C développée en commun et éditées respectivement par l'EUROCAE côté européen (ED-12C) et l'organisme RTCA Inc côté américain (DO-178C) [RTCA DO-178C 2012]. Cette norme fixe les conditions de sécurité applicables aux logiciels critiques de l'avionique dans l'aviation commerciale et l'aviation générale. Elle s'appuie sur quatre grands principes :

- Premier principe : Un logiciel est si complexe qu'il est pratiquement impossible de garantir qu'il est exempt d'erreurs. Par conséquent, si le produit final lui-même ne peut être garanti, la manière dont il est produit doit être aussi fiable que possible.
- Deuxième principe : Même si le processus de développement est fiable, des erreurs peuvent survenir. De multiples activités de vérification doivent être effectuées à chaque étape afin d'éliminer toutes les erreurs résiduelles potentielles.
- Troisième principe : La DO-178C est un document axé sur la maîtrise de trois processus pour atteindre des objectifs techniques : le processus de développement, le processus de vérification et le processus de gestion de configuration. Aucune méthode ou technique n'est spécifiée. Seuls les

[13] Les données historiques indiquent que la probabilité d'un accident grave attribuable à des causes liées à l'utilisation opérationnelle d'un aéronef était d'environ de 1 accident par million d'heures de vol. De plus, seulement environ 10% du total peut être attribué à des conditions de défaillance causées par les systèmes de l'avion. Enfin une centaine de conditions de défaillance potentiellement « Catastrophiques » sont identifiées au niveau d'un avion. Il en résulte un objectif de probabilité de défaillance inférieur à 10^{-9} par heure de vol pour chaque condition de défaillance. (10^{-6} * 10% * 1/100).

[14] Book 2 CS25.1309 section 6 §b)

objectifs le sont. La manière d'atteindre les objectifs est une décision industrielle.

- Quatrième principe : Il est supposé que la plupart des considérations de safety ont été faites au niveau du système et que l'assurance du développement logiciel doit garantir qu'elles ont été correctement mises en œuvre.

Les spécifications des logiciels et la façon dont ils sont produits jouent ainsi un rôle majeur dans la sécurité. La norme DO-178C impose que certaines activités soient conduites avec indépendance entre celui qui réalise l'activité et celui qui la vérifie. Ainsi l'applicant devra fournir les preuves de cette indépendance à travers l'enregistrement des personnes qui ont conduit les activités.

Les analyses safety permettent d'allouer un niveau de criticité pour chaque logiciel en regard de la gravité des événements redoutés auxquels ils contribuent. Ce niveau de criticité est appelé Development Assurance Level (DAL) et permet de connaître, en utilisant la norme DO-178C, les activités d'ingénierie logicielle à conduire pour certifier le logiciel. Plus largement, le DAL peut être alloué à une fonction, on parlera alors de FDAL (Functional Development Assurance Level) mais aussi aux *items*, un item étant défini comme « un élément matériel ou logiciel ayant des interfaces délimitées et bien définies » [ARP4754A 2011], on parlera alors de IDAL (Item Development Assurance Level).

Les logiciels sont classés selon 5 niveaux de criticité, déterminant ainsi un niveau d'assurance de développement ou DAL (de E à A). Plus ce niveau est proche du niveau A, plus le nombre d'objectifs par la DO-178C est élevé (cf. TABLE III.). Ainsi, si l'événement redouté est considéré comme catastrophique alors le niveau d'assurance de développement du logiciel contributeur sera DAL A, et la DO-178C imposera 71 objectifs à tenir. Des activités de certification doivent être réalisées pour atteindre ces objectifs. Ces activités doivent être implémentées dans des processus qui permettent d'atteindre les objectifs.

Par exemple, l'un des objectifs du processus de développement est de développer les exigences logicielles de haut niveau « High Level Requirements are developped » (DO-178C). Ce processus définit pour cela les neuf activités à conduire (§5.1.2 et §5.5) pour produire les données des sorties attendues (la spécification du logiciel, la traçabilité). La TABLE III. expose le nombre d'objectifs à tenir selon la DO-178C en regard du niveau d'assurance de développement.

TABLE III. LES OBJECTIFS DE LA NORME DO-178C VERSUS LE NIVEAU D'ASSURANCE DE DEVELOPPEMENT

Failure Condition Severity	No Safety Effect	Minor	Major	Hazardous	Catastrophic
Development Assurance Level	E	D	C	B	A
Number of Objectives DO-178C	0	26	62	69	71

Relations entre l'avionneur et l'autorité

La norme DO-178C explique aussi comment l'applicant doit se mettre en relation avec l'autorité pour valider les choix d'implémentation des fonctionnalités (Table A-10 « Certification Liaison Process »).

La DO-178 prévoit un Plan de Certification pour les Aspects Logiciels (Plan for Software Aspects of Certification - PSAC). Il doit être fourni et approuvé dès le début d'un développement logiciel. À la fin de chaque cycle d'approbation du logiciel, l'applicant soumet à l'Autorité un « Software Accomplishment Summary » (SAS) qui présente les résultats du développement ainsi que les éventuels écarts et leurs justifications vis-à-vis du PSAC.

Suivant la nature du produit logiciel et sa criticité, l'implication de l'Autorité peut varier. Le niveau d'implication est établi par l'EASA suivant les critères issus du document intitulé *Draft Criteria for the determination of the EASA level of involvement in product certification*. Pour chaque Level of Involvment (LOI), le nombre d'évaluations, appelées *Stage Of Involvement* (SOI) est défini. Pour un nouveau produit logiciel, quatre SOI sont nécessaires (planification, développement, vérification et conformité).

L'évaluation effectuée lors des premières réunions de certification déterminera le niveau d'implication des experts logiciels de l'EASA dans les activités de certification. Cinq grands critères peuvent influencer les résultats de cette évaluation :

(1) Le niveau de DAL/IDAL du logiciel, le niveau de rigueur des tâches d'assurance de développement effectuées sur le logiciel ;

(2) La complexité du développement logiciel ;

(3) L'expérience de l'équipe de développement et/ou du candidat en matière d'approbation de logiciels ;

(4) L'historique de service du logiciel ;

(5) La nécessité d'une nouvelle politique de l'EASA en raison de toute nouveauté telles que de nouvelles technologies, de nouvelles méthodes de conception, des outils inhabituels, etc.

La TABLE IV. expose une matrice qui permet d'identifier le niveau de risque du projet (Unidentified non-compliance). Un projet nouveau et complexe réalisé par une entreprise avec un niveau de performance moyen aura un niveau de risque « high ».

TABLE IV. NIVEAU DE RISQUE DU PROJET [LOI 2017]

Step1: Likelihood of an unidentified non-compliance			
CDI / Organisation Performance	no novel or complex aspects	no novel, but complex aspects; novel, but no complex aspects	novel and complex aspects
High	Very Low	Low	Medium
Medium	Low	Medium	High
Low or unknown	High	High	High

CDI : Compliance Demonstration Items

Une fois la probabilité du risque définie, la sévérité du logiciel est utilisée pour définir la classe d'intervention (cf. **Erreur ! Source du renvoi introuvable.**Par exemple, un logiciel développé DAL D pour un risque « Medium » sera « Class 1 ».

TABLE V. IDENTIFICATION DE LA CLASSE DE RISQUE

Severity \ Likelihood	Very Low	Low	Medium	High
Non-critical - DAL D	Class 1	Class 1	Class 1	Class 2
Non-critical - DAL C	Class 1	Class 1	Class 2	Class 3
Critical - DAL B	Class 1	Class 2	Class 3	Class 3
Critical - DAL A	Class 1	Class 2	Class 3	Class 4

Enfin, la TABLE VI. précise le niveau d'implication (nombre d'audits sur site, revue documentaire, …) de l'EASA en fonction de la classe d'intervention. Par exemple, pour un logiciel « Class 1 » aucun audit ne sera conduit par l'autorité.

TABLE VI. NIVEAU D'IMPLICATION DE L'EASA [LOI 2017]

EASA LOI	Desktop and on-site audits					Document review		
	Number of on-site audits	Planning audit (Desktop)	Development audit (on-site)	Verification audit (on-site)	Final audit (Desktop)	PSAC/PHAC and related SW/AEH plans	SAS/HAS SCI/HCI	Applicant SW/AEH review report
Class 1	None	None	None	None	None	None	None	None
Class 2	None	Yes	None	None	Yes	Yes	Yes	None
Class 3	1	Yes	Yes		Yes	Yes	Yes	Upon Request
Class 4	As necessary	Yes	Yes	Yes	Yes	Yes	Yes	Upon Request

Décrire les objectifs sans imposer les moyens

En synthèse de cette section, nous retenons que les normes en aéronautique n'imposent pas les moyens mais décrivent les objectifs qui doivent être atteints à travers la mise en œuvre de processus.

Cependant, une fois que l'autorité a établi les règles à respecter, elle propose souvent des moyens de conformité acceptables en regard de chaque exigence règlementaire. Il s'agit en fait de techniques reconnues qui permettent de répondre aux objectifs de sûreté. Par exemple les objectifs de la DO-178C peuvent être couverts de différentes manières. Les méthodes formelles sont une alternative pour répondre aux objectifs sans faire de tests. La méthode des graphes de Markov [Chan 2012] est également un moyen de conformité acceptable pour démontrer que les objectifs de fiabilité d'un système critique sont atteints.

La tendance actuelle de l'industrie aéronautique est d'essayer de relâcher certaines contraintes en définissant uniquement des méta-objectifs car les objectifs décrits dans les différents tomes (DO-178C, DO-330-Tool Qualification [RTCA DO-330 2011], DO-331-Model Based Development [RTCA DO-331 2011], DO-332-Object Oriented Programmation [RTCA DO-332 2011], DO-333-Formal Methods [RTCA DO-333 2011] sont trop nombreux. Le projet de recherche RESSAC[15] a permis de définir 3 propriétés essentielles (Overarching properties) : l'intention, l'exactitude et l'acceptabilité.

Même si le constat sur l'augmentation du nombre d'objectifs à atteindre est partagé, nous avons pu constater dans le domaine militaire que des directives moins précises ne permettaient pas d'obtenir le même niveau de confiance. Certes, les industriels ayant beaucoup d'expérience auront toutes les compétences pour répondre convenablement à cette nouvelle démarche. Mais nous pensons que toute une frange de l'industrie n'est pas armée pour y répondre et doit pouvoir s'appuyer sur des guides explicites pour certifier un logiciel.

F. Analyse des postures et pratiques industrielles dans un contexte de certification

Même si l'efficacité des référentiels est reconnue en aéronautique (le nombre d'accident aérien n'a jamais cessé de diminuer), la mise en œuvre des référentiels de certification par les industriels est souvent considérée comme très lourde et très coûteuse [Hilderman 2009], induisant des activités jugées superflues en phase de prototypage. Or, c'est une nécessité et une garantie vis-à-vis des tiers que chaque système puisse être utilisé avec un risque acceptable et l'absence de ces activités peut conduire à des dérives entraînant des conséquences dramatiques.

Face aux exigences de la certification, celle-ci est donc prise en compte tardivement dans le développement par les différents acteurs ; certains objectifs de ces référentiels sont même souvent atteints uniquement à la fin du développement 'parce qu'il le faut', pour montrer la conformité à la norme. Or cela présente un risque (sur la conformité) et un surcoût probable, supérieur au coût dû aux activités de certification. En effet, comme le montre la Fig. 3, la certification a un coût, puisqu'elle entraine des activités complémentaires à un processus de développement standard ; mais le coût des activités purement liées à la certification est seulement d'environ 3% par projet.

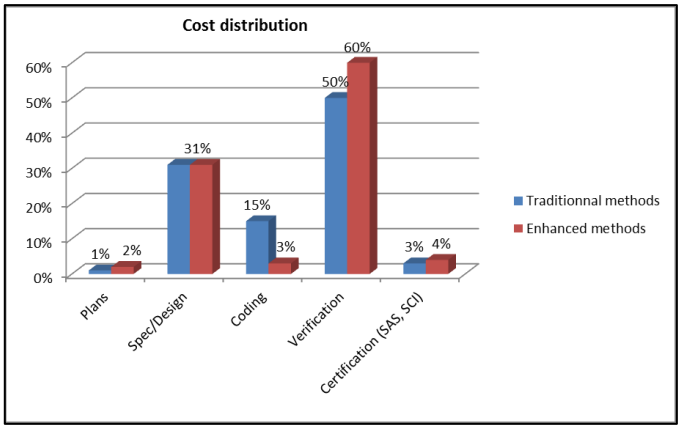

Fig. 3. Distribution des coûts de développement des logiciels - Document interne DGA Techniques aéronautiques

[15] RESSAC (Re-Engineering and Streamlining Standards for Avionics Certification) est un projet de recherche piloté par l'IRT Saint-Exupery, Toulouse, 2015 (https://sahara.pf.irt-saintexupery.com/RESSAC/)

S'intéresser à la question de la certification tardivement présente un risque encore plus fort d'augmentation des coûts. Par exemple, un objectif de couverture structurelle[16] partiellement atteint va demander des activités de retro-ingénierie ou des compléments d'analyse couteux.

En outre, de nombreux acteurs industriels ont une mauvaise estimation des coûts induits pour passer d'un niveau d'assurance à l'autre. Le DAL A est souvent vu comme un graal coûtant excessivement cher à produire. En réalité, l'écart le plus important de coût et de calendrier se situe entre le niveau D et le niveau supérieur (30 % pour passer de D à C, 50 % pour passer de D à B et 57 % pour passer de D à A). L'expérience de la DGA et l'étude HighRely [Hilderman 2009] montrent que la plus importante marche financière se situe entre le DAL D et DAL C.
L'écart de coût et de calendrier pour l'application de la norme DO-178B selon HighRely [Hilderman 2009] sont indiqués en Table VII.

TABLE VII. EVOLUTION DU COUT VERSUS NIVEAU D'ASSURANCE DE DEVELOPPEMENT

Level E	Level D	Level C	Level B	Level A
Baseline	E + 5%	D + 30%	C + 15%	B + 5%

La conception détaillée et les tests qui sont requis pour les logiciels DAL C imposent des activités supplémentaires qui ont pour finalité de garantir l'absence de comportements fonctionnels non voulus. Elles sont à l'origine du surcoût de développement en comparaison avec un logiciel DAL D. La Fig. 4 fournit des extrapolations de métriques.

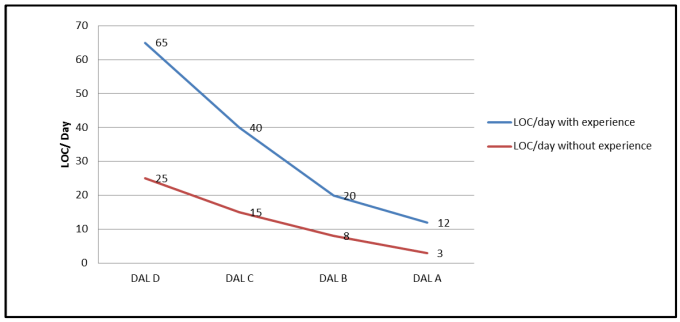

Fig. 4. Nombre de lignes de code produites par jour versus niveau d'assurance de développement [Hilderman 2009]

Les objectifs requis pour atteindre un DAL A adressent des classes d'erreur spécifiques à la sûreté de fonctionnement (traçabilité code source / code objet, couverture structurelle MC/DC[17] (Modified Condition / Decision Coverage). Ils n'ont pas d'incidence majeure sur le coût du développement. Il serait certainement plus performant (au niveau des garanties de conformité), de démontrer la conformité de manière continue, en intégrant mieux la certification dans le processus de développement pour qu'elle soit mieux appliquée et à moindre coût.

G. Vers une certification continue

Notre proposition, développée dans cet article, est de rendre plus agiles les activités de certification, en distinguant ce que recouvre aujourd'hui l'agilité dans le développement de logiciels, les concepts compatibles avec le processus de certification, ceux qui le sont moins et ceux qui sont particulièrement intéressants afin de montrer comment il serait possible de procéder. L'idée est de s'appuyer sur une proposition méthodologique intégrant la certification tout au long du processus développement tout en restant compatible à la norme DO-178C.

Les objectifs de cet article sont ainsi de montrer que l'agilité est compatible avec la certification et la norme DO-178C. Plusieurs méthodes supportent cette approche agile, Scrum [Canals 2019] [Scrum 2018], eXtrem Programming [Beck 2004], DevOps [Goudeau 2016] [Verona 2016]. Notre étude considèrera les valeurs et principes fondamentaux de l'agilité indépendamment de la méthode. Grâce à certains concepts issus de l'agilité, nous montrerons qu'une meilleure intégration de la certification tout au long du processus (appelée 'certification continue') permet de maîtriser les coûts, le processus de développement et l'intégration de la certification. Avoir des indicateurs concrets sur la certification, permet d'avoir une assurance sur le niveau de confiance que l'on peut avoir sur les développements. Délivrer un logiciel « certifiable » ou « pré-certifié » ('certification ready') plus fréquemment à un client final assure un besoin bien couvert et donne une indication concrète sur l'avancement de la certification. L'effet tunnel est réduit car l'état d'avancement des documents requis pour la certification est connu en permanence. L'idée fondamentale est de rapprocher les intérêts, en apparence antagonistes, des différents acteurs (équipe de développement, opérationnels, équipe qualité/certification) dans un objectif de fonctionnement plus fluide, davantage performant, collaboratif, pour assurer au final collectivement les garanties sociétales recherchées.

III. CERTIFICATION ET DEVELOPPEMENT LOGICIEL POUR LES SYSTEMES AERONAUTIQUES CRITIQUES

Cette section rappelle tout d'abord quelles contraintes les objectifs de certification induisent sur un processus de développement logiciel classique en aéronautique (section A), et souligne l'importance du processus de vérification (section B). Elle introduit ensuite l'assurance développement et l'assurance qualité, dans ce contexte de certification aéronautique (Section C). Elle termine par un retour d'expérience et une analyse sur les

[16] La couverture de code est une mesure utilisée pour décrire le taux de code source exécuté d'un programme par une série de tests. Elle montre les parties du code source testées ou non testées. Elle s'exprime par le pourcentage du code exécuté par rapport au code total.
[17] Il existe plusieurs niveaux de couverture : Fonction (Function Coverage), Instruction (Statement Coverage), Condition ou point de test - formule logique insécable de type booléen (Condition Coverage), Decision - formule logique composée de conditions et de connecteurs logiques (Decision coverage). En plus des critères de couverture de condition et de décision, la couverture des conditions / décisions modifiées (MC/DC) impose que chaque condition de la décision soit évaluée (au moins une fois) dans un cas où elle a une influence sur la valeur finale de la décision.

pratiques des industriels confrontés à des activités de certification (section D).

A. Un processus d'ingénierie logicielle contraint par les exigences de certification

L'ingénierie logicielle est une discipline qui s'intéresse aux méthodes systématiques qui permettent de développer des logiciels correspondant aux besoins du client, fiables, maintenables et performants. La **Erreur ! Source du renvoi introuvable.** présente les étapes classiques de développement d'un logiciel : les exigences des utilisateurs sont transformées en exigences sur le logiciel, qui servent à leur tour de référence pour élaborer une architecture logicielle, avant de passer à la conception détaillée puis au codage. On procède alors aux tests de chaque fonction prise de manière unitaire, puis aux tests d'intégration, aux tests logiciels et tests d'acceptation par le client. Chaque plan de test est préparé dans la partie descendante du cycle (à gauche sur la figure).

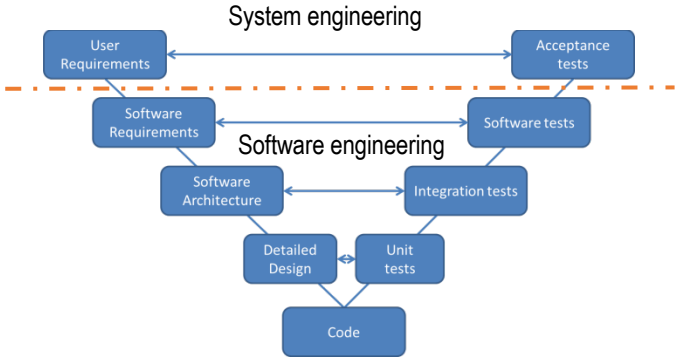

Fig. 5. Cycle de vie logiciel classique

Comme introduit en section II.C, la certification des aéronefs, via la norme DO-178C, n'impose pas de cycle de vie mais définit des processus séparés qui seront combinés pour décrire le cycle de vie d'un projet donné :

- Le processus de planification : plans de développement, de vérification, de gestion de configuration…
- Le processus de développement : spécification, conception, codage et intégration,
- Les processus intégraux : vérification, gestion de configuration, assurance qualité et coordination pour la certification.

Pour chaque processus sont définis : des objectifs d'assurance (ex : définir l'architecture et les éléments permettant de coder), des moyens de les satisfaire, des données d'entrées (ex : Spécifications, Plan de développement, Règles de conception), des activités (ex : définir l'architecture, les exigences dérivées, etc.), des produits (ex : description de conception), des critères de transition.

La norme DO-178C précise aussi les objectifs nécessaires à respecter pour obtenir la certification. A titre d'exemple, voici un certain nombre d'objectifs que l'on rencontre dans la norme :

- Les fonctions du logiciel devront être systématiquement spécifiées à travers une spécification générale (SRS : Software Requirement Specification) écrite à partir des exigences système
- Une architecture et une conception détaillée seront requises pour les logiciels les plus critiques (LLR : Low Level Requirement). Ainsi, la norme DO-178C est très claire sur le niveau de granularité élémentaire (Low Level Requirement) attendu pour développer un logiciel sécuritaire dont la défaillance peut avoir des conséquences Major, Hazardous ou Catastrophic. Le processus d'ingénierie peut prévoir un ou plusieurs niveaux de High Level Requirement, raffinés jusqu'à une exigence détaillée qui doit pouvoir être directement implémentée en code source sans aucune information supplémentaire [RTCA DO-178C 2012].
- Chaque élément de spécification ou de conception doit être développé, précis, cohérent, traçable, vérifiable. Ceci implique notamment d'effectuer des retours vers les processus amont en cas de découvertes de problèmes pouvant les impacter.
- Le code source sera développé à partir de ces éléments avant de servir pour générer le code objet exécutable.
- Pour les logiciels critiques, toutes les exigences (HLR et LLR) devront être testées. Les tests doivent être basés sur les exigences pour couvrir le comportement nominal et les cas de robustesse, et non sur le code (Requirement Based Testing) [Bender 2009].
- La couverture structurelle du code source, obtenue par l'exécution de ces tests basés sur les exigences, doit être mesurée. Les critères de couverture structurelle sont modulés en fonction de la criticité du logiciel (Statement Coverage, Decision coverage, Modified Condition/Decision Coverage)
- Le code source lui-même doit être développé conformément à des standards de codage.
- Une traçabilité doit être établie entre chaque donnée.
- Les données d'ingénierie doivent être gérées en configuration. Dans certains cas, une indépendance est requise entre la production d'une donnée et sa vérification.
- Enfin les activités d'Assurance Qualité doivent être conduites et enregistrées.

Notons que dans le domaine de l'aéronautique et de l'espace, selon les pays, les normes traitent la certification avec un angle de vue spécifique. Sur certains points, elles font des propositions particulièrement intéressantes.

Ainsi, pour le domaine aéronautique, le niveau de criticité (DAL) des modules logiciels se base sur l'ordre des coupes minimales fonctionnelles (Functional Failure Set) des événements redoutés auxquels contribuent les logiciels. Ce concept novateur a été présenté par [Rouahi 2007] et standardisée dans l'ARP4754A [ARP4754A 2011].

La norme américaine Mil-Std-882 E [DOD SYSTEM SAFETY 2012] met en corrélation la gravité des événements redoutés avec les conséquences financières (ainsi, une perte financière de plus de 10M€ sera considérée comme « Catastrophique »).

La norme DO-278 (Air Traffic Management) [RTCA DO-278 2011] propose un niveau intermédiaire (SWAL 4) entre le DAL D et le DAL C qui réduit l'écart en termes de coût de développement entre ces deux niveaux.

Enfin les normes Spatiales ECSS ou les recommandations du CNES sont assez prescriptives sur les mesures quantitatives pouvant être réalisées sur le code source. En fonction du niveau de criticité, la complexité cyclomatique [Gill 1991] qui représente le nombre de décisions obtenues en parcourant le graphe de contrôle des algorithmes, la profondeur d'imbrication des structures, et le nombre de paramètres seront limités avec des bornes de plus en plus contraignantes. La TABLE VIII. précise les contraintes à respecter en fonction de la catégorie du logiciel. Par exemple pour un logiciel cible de catégorie C la complexité cyclomatique des algorithmes ne devra pas dépasser la valeur de 15.

Les mesures sur le code source peuvent être complétées par le concept de « dette technique [18]» [Osetskyi 2018] qui permet de mesurer le coût induit par la correction des anomalies en regard des standards de codage [NT DGATA 2016]. Une stratégie de mise sous contrôle de la dette technique mesurant le ratio de dette technique [Letouzey 2012] sur le nouveau code facilite la résorption progressive des anomalies antérieures.

TABLE VIII. METRIQUES QUALITE LOGICIELLES DANS LE DOMAINE SPATIAL

Quality Charateristics	Metric	Target Category A	Target Category B	Target Category C	Target Category D
Reliability Evidence	Structural code coverage	Coverage => 100% Modified Condition / Decision Coverage =>	Decision Coverage => 100%	Coverage => 100% for on board software Statement Coverage => 100% for ground	
Reliability Evidence	Requirement coverage	100%	100%	100%	100%
Maintainability Modularity	Cyclomatic complexity	<10	<12	<15	<=20
Maintainability Modularity	Nesting level	<5	<5	<5	<7
Maintainability Modularity	Number of statements (per functions)	<100	<100	<100	<200
Maintainability Stability	Requirement stability	2%	5%	10%	15%

B. Critical software development & verification process

La vérification constitue le chapitre le plus important de la norme DO-178C, en volume - 13 pages de description (contre 5 en moyenne pour les autres chapitres) - et en charge de travail induite - pour l'A380 on compte 4 lignes de test pour 1 ligne de code embarqué. C'est un processus transverse ; il s'applique à tous les autres processus. Il recommande une combinaison de revues (inspection d'un produit par une personne indépendante – analyse qualitative), d'analyses (examen détaillé d'un produit, éventuellement effectué par un outil – analyse quantitative) et de tests (exécution d'un logiciel pour comparer les résultats obtenus avec les résultats attendus – tests fonctionnels, analyses de couverture fonctionnelle et structurelle) pour détecter et rendre compte des erreurs introduites au cours du développement.

Notons que la norme DO-178C ne fait pas de différence entre les activités de « validation » et de « vérification ». Les deux activités sont nommées indifféremment « vérification ». La norme DO-254 est plus claire sur cet aspect. Cette dernière parle de « validation » pour l'activité qui consiste à s'assurer que l'exigence considérée est conforme et complète avec celle du niveau supérieur (Construisons-nous le bon produit ?). Quant à la vérification, elle consiste à s'assurer que le résultat obtenu à travers l'exécution de l'implémentation est conforme aux attendus (Construisons-nous correctement le produit ?). La figure Fig. 6 illustre visuellement la différence entre les activités de « validation » et de « vérification ».

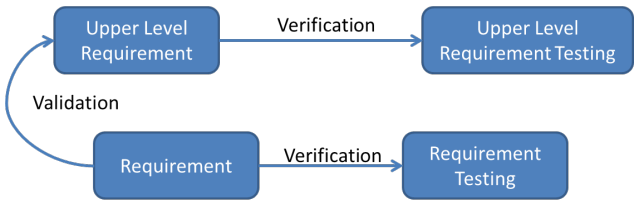

Fig. 6. Différence entre Validation et Vérification (DO-254) [RTCA DO-254]

De façon générale, et indépendamment du domaine d'application, les normes d'ingénierie logicielle décrivent toutes un processus de développement et de vérification. Comme on le voit sur la **Erreur ! Source du renvoi introuvable.**, d'une norme à l'autre les activités peuvent être légèrement différentes. Les objectifs sont également plus ou moins contraignants mais les grands principes sont semblables.

TABLE IX. COMPARAISON DE DIFFERENTES NORMES D'INGENIERIE LOGICIELLE

DO-178C/ED-12C Aeronautical: DAL		DAL E	DAL D	DAL C	DAL B	DAL A
DO-278/ED-109 Air Traffic Management: SWAL	SWAL 6	SWAL 5	SWAL 4	SWAL 3	SWAL 2	SWAL 1
IEC 61508 Electronic Industry: SIL			SIL 1	SIL 2	SIL 3	SIL 4
EN 50128 Railway: SSIL		SSIL 0	SSIL 1	SSIL 2	SSIL 3	SSIL 4
IEC 61513 - IEC 60880 Nuclear: Category				Cat C	Cat B	Cat A
IEC 62304 Medical: Class		Cl A	Cl B	Cl C		
ISO 26262 Automotive: ASIL		ASIL A	ASIL B	ASIL C	ASIL D	
IEC 61511 Instrumented System: SIL			SIL 1	SIL 2	SIL 3	SIL 4
ISO 13849: PI - IEC 62061: SIL Control System (Machinery)		PL a	PL b SIL 1	PL c SIL 2	PL d SIL 3	PL e SIL 4
ECSS Space: Criticity		Critic D	Critic C	Critic B	Critic A	
Mil-STD-882E Military: SwCI	SWCI5	SWCI 4	SWCI 3	SWCI 2	SWCI 1	
Technical Note DGATA P1301261003001-1P-C Military: Level			Level 1	Level 2	Level 3	Not Allowed

Les contraintes associées à chaque niveau d'ingénierie auront un impact sur l'organisation des services de l'entreprise. Au niveau du projet, les contraintes seront déclinées pour conduire les revues requises avec ou sans indépendance. Les activités techniques telles que l'instrumentation pour la mesure

[18]La dette technique est un concept qui reflète l'effort de travail à fournir aujourd'hui pour corriger des anomalies dues au non-respect des règles de conception qui induiront systématiquement des coûts supplémentaires dans le futur. On parle de dette car ces surcoûts sont assimilés à des intérêts.

de couverture structurelle du code, la vérification de la précision des algorithmes, le calcul du pire cas d'exécution, l'élaboration de tests de robustesse et des classes d'équivalence et la production documentaire devront aussi être réalisées avec un niveau de rigueur de plus en plus contraignant.

Le critère d'indépendance permet d'illustrer l'impact pour les organisations pour développer un logiciel DAL A ou DAL B. Deux types d'indépendance sont considérés :

- Indépendance des revues ou analyses, dans ce cas la personne qui conduit la revue doit être différente de la personne qui a produit la donnée ;
- Indépendance entre les personnes qui réalisent les activités, par exemple entre l'activité de codage (§5.3) et l'activité de sélection des cas de tests basés sur les exigences (§6.4.2).

Ce critère nécessite une répartition des responsabilités pour garantir que les activités ou revues ont bien été conduites de façon indépendante. La taille minimale d'une équipe pour développer un logiciel DAL A répondant au critère d'indépendance est de 3 personnes : le développeur et/ou relecteur d'une donnée produite par le vérificateur, le vérificateur et/ou relecteur d'une donnée produite par le développeur, et le responsable d'assurance qualité.

C. Assurance de développement, assurance qualité et certification

L'assurance de développement est la principale méthode reconnue pour garantir la fiabilité des logiciels des systèmes critiques. Le comportement déterministe d'un logiciel induit une défaillance systématique si une erreur de développement du logiciel est activée en phase opérationnelle. Les contraintes imposées sur les processus de développement et vérification permettent ainsi d'augmenter le niveau de confiance concernant l'absence d'erreur de développement introduite dans le logiciel en vue d'une autorisation de mise en service par une autorité de certification.

L'activité de certification d'un logiciel est souvent interprétée comme une simple activité d'assurance qualité logicielle. Pourtant un bon nombre de points à traiter adressent des problématiques techniques propres aux activités de sûreté de fonctionnement. Là où le responsable qualité limitera son intervention à vérifier simplement l'existence du document requis pour la certification, l'auditeur devra s'assurer que le contenu des documents correspond à ce qui a été implémenté et testé. Est-ce que la fonctionnalité logicielle a bien été implémentée conformément à la spécification système ? Est-ce que le test associé permet de garantir le bon fonctionnement de la fonction, est-il basé sur l'exigence et non sur le code ? Est-ce que ce test couvre les classes d'équivalence des ensembles de valeurs numériques ? Est-ce que le séquenceur permet d'exécuter des tâches conformément aux contraintes temps réel et déterministes requis par le système ? Lorsque les logiciels assurent les fonctions les plus critiques, la personne en charge de la certification du logiciel doit veiller à ce que le choix du jeu de tests permette de traiter les différentes classes d'équivalence des données numériques, les points singuliers, les valeurs limites, les transitions de mode et la combinatoire des entrées logiques [RTCA DO-178C 2012].

D. La certification aéronautique telle qu'elle est vécue et pratiquée par les industriels dans l'aéronautique

Les industriels du domaine aéronautique sont bien organisés pour conduire les activités de certification aéronautique. Pour chaque projet un CVE (Compliance Verification Engineer) est nommé pour établir l'interface avec l'autorité. Il est reconnu par l'EASA pour ses compétences et est responsable de la navigabilité du produit de manière indépendante vis-à-vis du management programme. Son rôle est de préparer les équipes aux différentes évaluations, il sert de médiateur en cas de conflit. D'autres sociétés s'octroient les services d'experts de la certification logicielle pour préparer le travail.

Peu d'acteurs du domaine aéronautique utilisent les méthodes agiles. Ils préfèrent utiliser une méthode éprouvée et déjà validée par l'autorité de certification. Beaucoup ont peur de devoir convaincre l'autorité pour la mise en place d'une nouvelle méthode. D'autres ne veulent pas faire évoluer leur atelier d'ingénierie par crainte de devoir démontrer que les projets existants ne sont pas impactés par l'évolution. Cette approche est légitime mais elle prive les équipes de développement de beaucoup d'évolutions technologiques qui facilitent le travail de production et la conduite d'audits de certification.

Même si leur efficacité est reconnue et qu'ils sont incontournables dans les domaines critiques comme en aéronautique, la mise en œuvre de standards de certification par les industriels est souvent considérée comme trop lourde et trop coûteuse [Hilderman 2009].

Ainsi, la mise en œuvre de ces standards est parfois négligée. Par exemple, dans le domaine militaire, rares sont les industriels qui appliquent des processus rigoureux et auditables pour développer des logiciels critiques. Un argument souvent entendu est que le système est fait pour faire la guerre donc les problèmes de safety sont secondaires. Mais ces arguments sont de moins en moins recevables par l'Autorité Technique DGA, en particulier pour les phases d'entraînement en métropole : il serait vraiment dommageable de voir un drone s'écraser dans une zone non sécurisée ou de voir un missile dévier de sa trajectoire et toucher une cible non prévue.

A l'opposé, on note quelques initiatives vertueuses, comme celle de Nexter, l'un des acteurs historiques du domaine de la défense, qui a adopté une démarche générale pour la conception des logiciels critiques ; cette entreprise a adapté ses pratiques à la norme IEC61508 et l'applique dorénavant sur tous les nouveaux développements. Avec l'appui de la société SERMA, elle a également mis en place des moyens pour évaluer la conformité des développements de ses sous-traitants au référentiel IEC61508. Cette initiative est très positive et sera gage d'une maîtrise de la sécurité pour les futures évolutions des systèmes de défense.

De plus, une mauvaise interprétation des référentiels est souvent constatée. Par exemple, la norme DO-178C ne prescrit pas de cycle de développement mais souvent les industriels l'interprètent ainsi en s'imposant un cycle en V. En vérité, elle n'interdit pas l'utilisation des principes agiles.

Comme le montre la Fig. 7, les contraintes et obligations de certification semblent suggérer un développement linéaire des

systèmes. Les normes ARP4754 et DO-178 ont été écrites à une époque où le développement en aéronautique s'appuyait sur un cycle en V ou en cascade. Il serait utile d'introduire les nouvelles méthodes de développement logiciel.

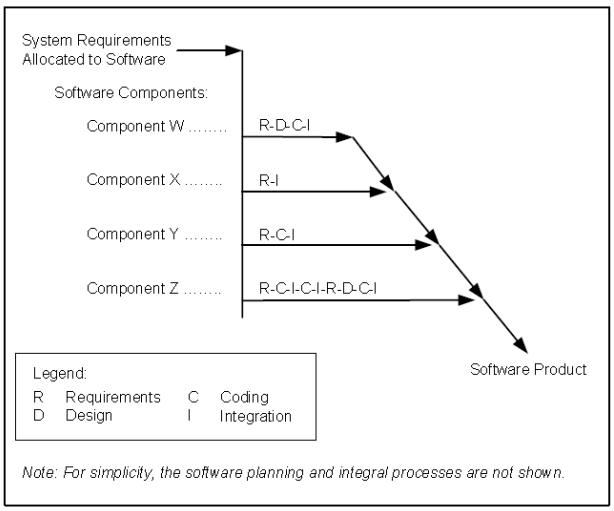

Fig. 7. Cycle de développement illustrés dans [RTCA DO-178C 2012], Figure 3-1

Enfin, les règlements relatifs à la certification sont souvent invoqués à la fin du développement ou sont réalisées des activités supplémentaires, qui viennent 'en verrue' sur le processus de développement, terminé, juste pour montrer la conformité à la norme. La démonstration de la conformité n'est pas effectuée de manière continue, alors que l'intégrer au processus de développement aurait plusieurs avantages, notamment d'assurer de meilleures garanties sur la safety, et d'engendrer de moindres coûts.

Dans les sections qui précèdent, nous avons exposé l'intérêt sociétal de la certification et le mode de fonctionnement d'un processus de certification logicielle, en détaillant sa mise en œuvre dans un contexte aéronautique. Nous avons mis en perspective les différences avec d'autres domaines et introduit la nécessité d'adapter les pratiques pour augmenter l'efficacité de la certification et réduire son coût. Nous avons noté que l'industrie a très récemment et assez marginalement initié un changement de méthode pour rendre plus « agile » le développement des logiciels sécuritaires. Cette tendance nous semble toutefois prometteuse ; nous développons cette thèse dans les sections suivantes, parmi un panorama de pistes intéressantes.

IV. COMMENT AMELIORER LE PROCESSUS DE CERTIFICATION DES LOGICIELS ?

Pour améliorer le processus de développement logiciel tout en répondant aux objectifs des normes de certification, plusieurs pistes peuvent être, ou sont déjà pour certaines, envisagées, notamment l'ingénierie dirigée par les modèles, et l'intégration continue qui nous amène progressivement à glisser vers une approche plus agile.

A. L'ingénierie dirigée par les modèles

L'Ingénierie Dirigée par les Modèles (IDM, ou Model Based Systems Engineering, MBSE) est une pratique d'ingénierie des systèmes visant la description, au travers de modèles, concepts, et langages, à la fois d'un problème posé (besoin) et de sa solution. Un système peut être décrit par différents modèles[19] liés les uns aux autres, en utilisant autant de langages de modélisation différents que les aspects chronologiques ou technologiques du développement du système le nécessitent [Combernale 2008].

Le MBSE permet de partager une vision claire du produit à développer ; il améliore les échanges et la communication entre les acteurs du développement d'un système ou d'un logiciel, et par là-même la qualité des processus. Il rend opérationnels les modèles pour la génération de code et de documentation de test, la validation, la vérification, l'exécution, etc. Les modèles permettant également de simuler des comportements avant leur implémentation, certains défauts de conception peuvent ainsi être détectés très tôt dans le cycle de développement.

Les propriétés de syntaxe et sémantique non ambiguës pour exprimer le comportement attendu à travers des modèles garantissent un haut niveau d'assurance de développement ; des outils comme SCADE [SCADE 1999] ou Simulink [SIMULINK 1984] sont fréquemment utilisés pour programmer des applications critiques et contribuent à la safety des systèmes.

Le MBSE a fait ses preuves pour supporter la simulation, la vérification, le test et la génération de code. L'approche basée modèles s'étend dorénavant à d'autres domaines tels que la safety. En effet, la plupart des techniques actuelles d'analyse de la safety reposant généralement sur un modèle de système informel ; elles sont souvent incomplètes, incohérentes et comportent des erreurs [Joshi 2006]. L'approche basée modèles des analyses safety (Model Based Safety Assessment, MBSA[20]) propose d'étendre le développement basé modèle pour incorporer les activités d'analyse de la safety en plus des activités de développement traditionnelles. Elle permet d'aborder ces analyses avec davantage de rigueur ; elle favorise aussi la documentation et le partage d'information avec l'autorité de certification. Elle permet également d'effectuer des simulations.

Le MBSA fonctionne sur un modèle formel décrivant à la fois le comportement nominal du système et celui des défaillances. En utilisant une telle approche, il est possible d'obtenir un modèle précis du comportement du système, d'automatiser certaines parties du processus d'analyse de la safety et, par conséquent, de réduire les coûts et d'améliorer la qualité des analyses [Bernard 2009]. La dernière décennie a vu se développer un certain nombre de méthodes et outils MBSA [Lissagor 2011]. L'industrie les a adoptés pour certains domaines critiques (aéronautique, défense, nucléaire…). Les autorités de certification les ont acceptés ; ainsi l'annexe N de la prochaine révision de [ARP4761 1996] introduira le MBSA

[19] Un modèle est une abstraction, une simplification d'un système qui est suffisante pour comprendre le système modélisé et répondre aux questions que l'on se pose sur lui. [Combemale 2008]

[20] L'acronyme MBSA signifie aussi parfois Model Based Safety Analysis, l'assessment étant de fait le résultat des analyses.

(Model Based Safety Analysis) comme moyen acceptable pour conduire des analyses safety dans le domaine aéronautique. Le MBSA est applicable à tous les domaines (naval, terrestre, spatial, drones, …).

La DGA a développé sa propre méthodologie structurée reposant sur le MBSA pour accompagner l'évaluation de la safety par les modèles. Elle s'appuie sur le langage AltaRica, spécifique à la sûreté de fonctionnement. Il s'agit d'un langage formel de haut niveau conçu pour modéliser les systèmes et dédié aux analyses safety [Arnold 1999]. Ce langage a été développé dans le but de concevoir des modèles qui reflètent (mieux que des arbres de défaillances, des graphes de Markov et des réseaux de Petri) les architectures physiques et fonctionnelles des systèmes. La méthode repose sur l'évaluation des aspects qualitatifs des analyses de la safety des systèmes socio-techniques complexes (Analyse des modes communs, allocation de DAL, facteurs humains, dépendance multi-systèmes) [Louis 2012]. Cette méthode innovante transpose les principes prescrits par la norme DO-331 (Model Based Development) pour la génération de code à partir de modèle (type Simulink) à la production de modèles de safety pour générer des arbres de défaillance. Une librairie de plusieurs centaines de composants élémentaires réutilisables a été créée. Chaque composant est spécifié et testé avec indépendance pour garantir un haut niveau de confiance sur le comportement attendu.

La méthode préconise aussi un méta-modèle générique applicable à chaque modélisation.

- Une vue fonctionnelle qui décrit les fonctions modélisées sur lesquelles vont porter les évènements redoutés (par exemple perte de la fonction freinage).
- Une vue organique qui décrit les constituants (matériel et logiciel) du système qui permettent d'assurer les fonctions observées.
- Une vue zonale qui place chaque constituant dans les différentes zones du système sur lesquelles pourront porter des menaces spécifiques (feu, choc à l'oiseau, agression électromagnétique, éclatement moteur…).

L'aspect innovant est d'exploiter la puissance du langage AltaRica pour générer automatiquement les coupes minimales des erreurs systématiques (Functionnal Failure Set [ARP4754A 2011]) propres aux défaillances des logiciels mais aussi aux erreurs de conception des éléments matériels. Cette modélisation permet aussi de traiter, à partir du même modèle, les interactions multi-systèmes en intégrant dans l'analyse des causes de défaillances les systèmes ressources (Système électrique, système hydraulique…). Les erreurs humaines peuvent aussi être prises en compte pour s'assurer que les procédures applicables répondent bien aux enjeux de safety du système.

Cette méthode fait aujourd'hui l'objet de plusieurs axes de communication auprès de différents organismes pour promouvoir son utilisation : formation aux autorités de certification (30 experts EASA formés à ce jour), publications [Bieber 2018], formation interne DGA, formation professionnelle externe (EUROSAE), formation initiale d'ingénieur (INSA), essaimage industriel [Linty 2019]. Elle est couramment utilisée au profit de la qualification DGA des systèmes (A400M, C130, MdCN, CERES…). Des résultats probants ont été obtenus lors de son utilisation pour supporter les travaux d'expertise au profit du ministère de la justice dans le cadre de l'enquête après accident du crash du vol AF447 (Rio-Paris). Le projet MOISE [IRT 2018], auquel participe la DGA, mène des études prospectives pour mettre en cohérence les modèles issus d'une démarche d'ingénierie système reposant sur des modèles et les analyses safety système basées modèles. Son but est de définir un processus de synchronisation entre la définition du système et les analyses safety dans le but d'assurer la consistance de la description système et du modèle safety. Ces deux activités pourront se dérouler en parallèle et ainsi la conformité avec les guides de processus de développement ARP4754A et ARP4761 pourra être démontrée.

Notre vision est que la complexité des futurs systèmes critiques sera telle que les analyses de la safety classiques devront être complétées par de nouvelles méthodes évoluées.

Nous sommes convaincus que les méthodes MBSA seront à l'avenir incontournables pour formaliser les analyses de la safety des futurs systèmes socio-techniques à fortes interactions comme les véhicules autonomes.

B. Intégration continue

Les logiciels, mais aussi les équipes et l'infrastructure de déploiement augmentent en complexité. Pour développer, tester et livrer des logiciels de manière rapide et cohérente, les développeurs et les organisations ont créé des stratégies pour gérer et automatiser ces processus. Ainsi, l'utilisation des mécanismes d'intégration continue, mais plus récemment de pratiques complémentaires telles que la livraison et le déploiement continus, deviennent de plus en plus répandues, y compris dans le domaine aéronautique. L'intégration continue met l'accent sur l'intégration du travail des développeurs individuels dans un référentiel principal plusieurs fois par jour pour détecter rapidement les problèmes d'intégration et accélérer le développement collaboratif. La livraison continue consiste à réduire les frictions dans le processus de déploiement ou de publication, en automatisant les étapes requises pour déployer une version afin que le code puisse être publié en toute sécurité à tout moment. Le déploiement continu va encore plus loin en déployant automatiquement chaque fois qu'un changement de code est effectué.

Intégration continue

Intégrer recouvre l'ensemble des activités à faire une fois que le travail de développement à proprement parler est terminé, pour obtenir un produit exploitable, "prêt à l'emploi". Ainsi, vérifier la cohérence entre différents composants d'un logiciel, et corriger d'éventuelles incohérences relève de l'intégration. Faire de l'intégration continue consiste à intégrer un composant dès que possible, s'assurer de la cohérence avec les autres composants et que le résultat des modifications ne produit pas de régression dans l'application, et générer un exécutable opérationnel. C'est une étape incontournable pour automatiser toutes les tâches répétitives d'un processus de développement logiciel, permettant en cela d'exécuter et de produire certaines activités/données requises par la certification (exécution des tests, mesure du taux de couverture).

L'idée d'intégration continue apparait comme un objectif de l'organisation du projet dans [Royce 1998]. L'Intégration Continue' est ensuite popularisée avec l'eXtreme Programming, une pratique qui consiste à ce que les développeurs d'une même application réintègrent le code sur lequel ils travaillent le plus fréquemment possible, et à déclencher à chaque intégration un processus qui vérifie automatiquement le fonctionnement de l'application afin que les anomalies soient détectées à leur entrée [Pillou 2018]. [Fowler 2006] décrit ainsi les ingrédients de cette pratique : l'utilisation d'un référentiel de gestion de versions du code source, l'automatisation du processus de "build[21]", des tests unitaires et fonctionnels automatisés, l'exécution quotidienne de l'ensemble {build+test}. Elle a pour effet d'accélérer les phases de compilation, le déploiement et le test du code, engendrant de ce fait des gains de productivité.

L'intégration continue permet ainsi, par l'exécution systématique à chaque 'build' de tous les tests logiciels, de s'assurer de la qualité d'une application par des mesures de qualité du code, par une meilleure maîtrise des dépendances, par la détection au plus tôt d'erreurs d'intégration (suite à un oubli d'inclusion par exemple, ou des régressions possibles des fonctionnalités précédemment implémentées), et de s'assurer du respect des normes (de nommage, de programmation...) établies pour l'application. Elle permet aussi une meilleure réactivité face aux changements, la standardisation des sources et du cycle de vie de l'application.

L'intégration continue vise ainsi deux objectifs, réduire à presque zéro la durée et l'effort nécessaire à chaque épisode d'intégration, et pouvoir à tout moment fournir un produit exploitable. Dans la pratique, ces objectifs exigent de faire de l'intégration une procédure reproductible et dans la plus large mesure automatisée. Elle inclut l'exécution d'une batterie de tests unitaires et fonctionnels à chaque publication dans le référentiel des sources. En cas d'échec ne serait-ce que de l'un de ces tests, l'équipe cherche prioritairement à rétablir la stabilité du produit. L'exécution de cette procédure se fera de manière rapide et régulière.

Elle nécessite pour cela un certain nombre d'outils. Ceux-ci permettent de gérer le logiciel en configuration, le compiler, créer des archives, déployer une archive sur une machine de test, exécuter une suite de tests et en notifier le résultat à plusieurs destinataires, générer la documentation du code, déposer le logiciel dans un espace de livraison, créer des rapports de statistiques….

La démarche générale est détaillée par la Fig. 8.

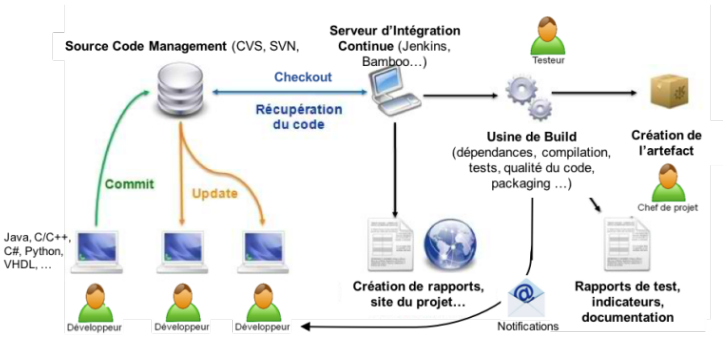

Fig. 8. Flux d'Integration Continue

L'utilisation d'un référentiel de gestion de versions du code source (CVS, SVN, Git, etc.) par les équipes de développement est la première étape qui permet d'automatiser entièrement le processus de compilation et de génération du produit. Le développeur fait un commit de code[22] sur le référentiel du gestionnaire de configuration. Les outils de gestion de configuration Git ou SVN sont fréquemment utilisés pour assurer cette tâche. Le serveur d'intégration continue (CruiseControl, Jenkins, Bamboo…) détecte le commit, fait un Checkout[23], lance les opérations de compilation et de tests. En cas d'échec, une notification est générée au chef de projet et/ou à l'équipe de développement. Le développeur concerné par l'erreur fait un update[24] du référentiel de gestion de configuration et corrige l'anomalie. Les serveurs d'intégration servent à compiler le code centralisé, ce qui permet de voir si toutes les dépendances sont satisfaites. Les tests unitaires peuvent être lancés automatiquement.

Livraison continue

La livraison continue est une extension de l'intégration continue. Selon [Fowler 2013], elle vise à construire une application de telle manière qu'elle puisse être envoyée en production en toute confiance à tout moment. Cette façon de travailler est très prisée par le mouvement DevOps dont la philosophie est : « You build it, you run it ».

Elle se concentre sur l'automatisation du processus de livraison de logiciels afin que les équipes puissent déployer facilement et en toute confiance leur code en production à tout moment. En veillant à ce que la base de code soit toujours dans un état déployable, publier le logiciel ne nécessite ni coordination complexe ni test en phase avancée [Humble 2011].

La livraison continue permet de réduire le plus possible le temps entre une idée et sa mise à disposition auprès des utilisateurs. Cette pratique est intéressante car elle automatise les étapes entre la vérification du code dans le référentiel et la décision de publier des builds fonctionnels et bien testés sur l'infrastructure de production. Les étapes qui permettent d'affirmer la qualité et l'exactitude du code sont automatisées,

[21] C'est l'ensemble d'étapes nécessaires à la compilation, à la création des livrables au lancement des tests.

[22] C'est l'opération qui permet la validation des mises à jour du code source existant sur le répertoire de travail local de la machine du développeur moyennant l'outil de gestion de configuration (tel que SVN). Le commit se fait du répertoire de travail local vers le référentiel de l'outil de gestion de configuration. Il est généralement automatique mais peut être toutefois déclenché manuellement par exécution d'une ligne de commande.

[23] C'est l'opération d'extraction d'une version d'un projet en cours de développement du référentiel du gestionnaire de configuration sur un répertoire de travail local.

[24] C'est l'opération qui permet de la mise à jour à partir du référentiel de l'outil de gestion de configuration du répertoire local.

mais la décision finale sur ce qu'il faut publier est laissée entre les mains de l'organisation pour une flexibilité maximale.

Comme pour l'intégration continue, la livraison continue est une pratique qui nécessite la mise en œuvre de processus rigoureux et outillés et d'une organisation adaptée pour être efficace.

Déploiement continu

Le déploiement continu va encore plus loin ; c'est une extension de la livraison continue qui consiste à livrer chaque changement apporté au logiciel à l'utilisateur final. Dans ce mode de fonctionnement, il n'y a pas d'intervention humaine pour décider du moment du déploiement en production, c'est un système automatique de déploiement qui déploie tous les changements, exceptés ceux qui échouent à un test. Cette pratique permet d'accélérer la boucle de feedback, et permet aussi aux développeurs de mieux se focaliser sur le développement du logiciel puisqu'il n'y a plus de " date de livraison " à anticiper.

Cependant, ce cycle de déploiement entièrement automatisé peut être une source d'anxiété pour les organisations qui s'inquiètent d'abandonner le contrôle de leur système d'automatisation sur ce qui est publié. Le compromis offert par les déploiements automatisés est parfois jugé trop dangereux pour la récompense qu'ils fournissent.

Vers la certification continue des logiciels

L'étape ultime, la certification continue, consisterait donc à conduire les activités requises par la certification au fil du développement logiciel.

La **Erreur ! Source du renvoi introuvable.** positionne les activités de production d'un logiciel et les différents processus de développement continus associés.

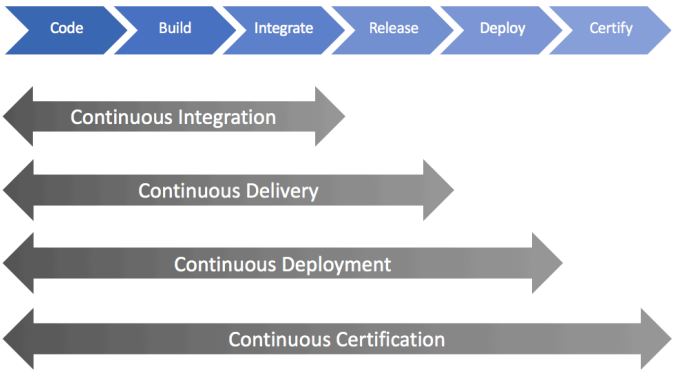

Fig. 9. Processus de développement continu

Le développement continu présente toutefois quelques inconvénients. D'abord, il s'agit de processus complexes de par la présence de nombreuses briques servant à l'automatisation des tâches. Ensuite, il doit être opérationnel en permanence. La mise en place de l'Intégration Continue représente un vrai coût qui doit être mutualisé au maximum entre plusieurs projets et équipe de développement. Les changements dans les habitudes de travail induits par la mise en place de l'Intégration Continue doivent s'accompagner d'une formation aux bonnes pratiques. Enfin, quelques obstacles doivent être surmontés pour l'appliquer dans un contexte de certification. L'absence de documentation ne sera pas acceptable, les processus devront être stables et certains objectifs requis par la norme DO-178C seront partiellement atteints (i.e. la complétude).

V. LA CERTIFICATION CONTINUE A TRAVERS L'INTRODUCTION DE L'AGILITE POUR LES LOGICIELS CRITIQUES

Cette section a pour objectif de montrer qu'il est possible, dans une dynamique de « certification continue », d'introduire de l'agilité dans le processus de développement de logiciels critiques soumis à certification. En effet, les acteurs du domaine aéronautique pensent, par exemple, que les standards de certification dans leur domaine imposent un développement linéaire (de type Waterfall ou V). Cependant si cela peut être sous-jacent dans les normes, elles n'imposent pas de cycle de vie (cf. Section III.*A*). Elles fixent des objectifs décrits dans des processus à mettre en place. Un mode de fonctionnement agile, dont les adeptes louent la performance pour le développement logiciel, n'est donc pas incompatible avec les normes aéronautiques. Il sera d'autant plus performant si la prise en compte du besoin des autorités (réponse aux objectifs de la norme DO-178C) est totalement intégrée au processus de développement.

Prenons par exemple le cas d'applications de déploiement de mises à jour logicielles sur les véhicules Tesla : celles-ci peuvent avoir une certaine forme de criticité (par rapport à l'intégrité et la confidentialité des données, la disponibilité des services, les fonctions de sécurité etc.) : des défaillances peuvent impacter fortement l'image de marque et engendrer des pertes financières et humaines. Or, ce type d'application est depuis longtemps développé en mode « agile », certainement car leur contexte leur a permis de mettre en œuvre facilement ces techniques novatrices.

Fig. 10 présente le processus de développement continu suivi par Tesla. Il reprend les principes agiles d'intégration continue et de déploiement continu. Par contre, nous pouvons observer que les analyses de safety et l'approbation des autorités américaines (National Highway Traffic Safety Administration) restent encore séquentielles.

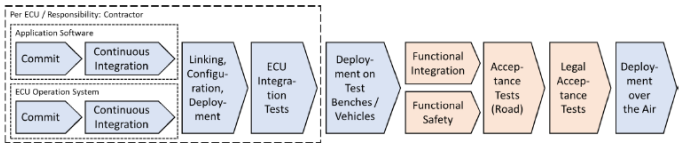

Fig. 10. Processus de développement continu suivi par Tesla [Vöst 2016]

Notre article propose de rendre le processus certification (Legal Acceptance) encore plus imbriqué dans le processus de développement. Les évaluations de la safety pourraient aussi être évaluées en continu par les autorités à travers la livraison d'un modèle MBSA qui présenterait, dès les phases préliminaires de définition d'architecture, les principes safety essentiels (Allocation de DAL, indépendance, analyse de modes communs…).

Avec son histoire, le domaine aéronautique est sûrement moins enclin à ces changements mais l'évidence de gain d'efficacité et de productivité a motivé certains acteurs pour initier l'utilisation de ces techniques pour développer des applications aéronautiques critiques. Plusieurs expériences réussies laissent penser qu'une large adoption est envisageable par l'industrie aéronautique. Cette section, en répertoriant et analysant ces expériences, a pour objectifs de montrer que les attentes de l'autorité sont compatibles avec un développement agile, d'identifier les bonnes pratiques, et d'identifier les pièges à éviter.

Cette section démontre tout d'abord les limites des approches traditionnelles, puis rappelle les concepts clefs et comment procèdent les approches agiles. Elle met ensuite en avant les intérêts d'introduire l'agilité dans un processus de développement continu de logiciels soumis à certification et en souligne les verrous. Elle termine par quelques exemples d'expériences industrielles pilotes réussies, prouvant par là-même que ces verrous peuvent être surmontés.

A. Limites des approches tradidionnelles

Selon [Standish Group 2015] seuls 32 % des projets de développement logiciel sont réalisés dans les délais et le budget initialement définis, 44 % des projets ont abouti hors délai et hors budget ou sans respecter les exigences initiales, et 24 % des projets sont arrêtés avant d'être mis en production.

Parmi les motifs d'échec arrivent en tête le manque d'implication des utilisateurs finaux (12,8%) et les changements de spécifications en cours de projet (11,8%). Quels sont les constats majeurs effectués ? Un trop fort cloisonnement des collaborateurs dans l'organisation, des remises en cause trop fréquentes des processus projet du fait que chaque équipe possède ses propres processus, un manque de communication entre les équipes, une documentation volumineuse coûteuse à générer et souvent peu adaptée. La qualité intervient souvent en dernier pour pouvoir attribuer un tampon qui valide un projet et devient un critère d'ajustement. Les besoins du client sont souvent mal pris en compte et la solution manque de valeur. Enfin au niveau de la planification on constate un manque de maîtrise des projets, un livrable non disponible à la date prévue, avec des cycles d'évolution trop longs.

Il faut donc casser les silos [Xue 2017], récupérer les informations dans les autres équipes ainsi que chez le client et avoir la possibilité d'être un peu plus libre sur la manière de fonctionner.

Pour cela, le traditionnel cycle en V n'est pas adapté. Comme on le voit sur la partie descendante de la Fig. 11, les étapes s'enchainent en cascade, selon un enchaînement linéaire qui, de façon très simplifiée, comporte 3 phases : on réfléchit à tout, on prévoit tout, on réalise tout, exactement comme ça a été prévu, et on teste et livre tout, une bonne fois pour toute. Il présente ainsi plusieurs défauts.

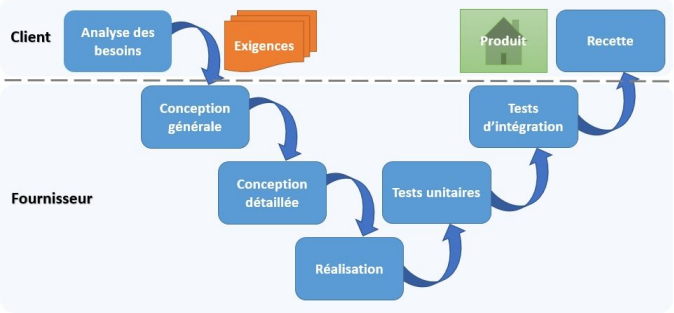

Fig. 11. Schéma simplifié d'un cycle en V [Ninni 2019]

Le client n'intervient qu'en début et fin de cycle : après avoir exprimé ses besoins, on lui présente des spécifications générales puis il attend la recette pour découvrir le résultat, ce qui donne parfois lieu à des désillusions importantes lorsqu'il reçoit le produit. Il exige une première phase d'analyse et de conception poussées si on veut anticiper au maximum tous les besoins et les problèmes possibles et s'assurer que l'on n'a pas omis le moindre détail. La phase de développement est réalisée avec des spécifications figées, engendrant de ce fait un effet tunnel[25], avec peu de vision sur ce que sera le produit au final, tout comme sur sa date de livraison effective : si le client a oublié une contrainte, ou souhaite ajouter ou modifier quelque chose, il faut en principe qu'il attende la fin du projet et la réception des produits, pour pouvoir lancer un nouveau projet qui prendra en compte ses nouvelles améliorations (on déroge souvent à cette règle, avec malgré tout, des retards et un surcoût très importants, tout changement en cours de route ayant un impact sur l'ensemble du projet). Le cycle en V autorise les retours arrière sur les étapes de même niveau (si par exemple les tests unitaires montrent des défauts, on repart en conception détaillée) ; cependant un problème d'intégration (donc découvert tardivement) pourra remettre en cause l'intégralité du projet ! Enfin, si le projet est en retard, ce sont en général les dernières tâches à réaliser (i.e. revues pour la certification, tests du produit…) qui en pâtissent ; on réduit donc le temps alloué à ces activités, ce qui a un impact négatif sur la qualité du produit.

Tout l'enjeu est d'éviter les situations qui conduisent à réaliser tardivement les activités requises par la certification sous prétexte qu'elles n'apportent pas de valeur ajoutée. Effectivement, une fois que le logiciel est livré au client et que les tests de recette sont passés, la preuve d'une revue d'exigences n'a plus beaucoup d'intérêt. Par contre, cette revue est beaucoup plus intéressante à conduire au cours du développement, au moment où la donnée est produite pour d'une part limiter l'effort et d'autre part maximiser l'impact d'une détection d'anomalie (complétude, testabilité…). Il est nécessaire de se prémunir de ce risque car la pression économique aura toujours tendance à dégrader l'intérêt de conduire ces activités de certification si elles sont réalisées trop tardivement.

Les méthodes agiles offrent alors tout leur intérêt. Elles reposent sur une organisation des projets en boucles assez

[25] On appelle 'effet tunnel' une période durant laquelle le donneur d'ordres ou l'émetteur des spécifications reste aveugle quant à la progression du projet et/ou du développement.

courtes qui se répètent plutôt qu'une longue succession linéaire d'étapes. Elles visent à livrer au client des versions intermédiaires de produits opérationnels afin qu'il puisse mesurer l'avancement du projet et valider la direction prise. Elles promeuvent la communication dans et entre les équipes, ainsi qu'avec les différentes partie-prenantes, client, autorités. L'agilité à une approche très pragmatique : l'important c'est que tout fonctionne bien et que tout le monde soit content, les autorités de certification comprises.

B. L'agilité en développement logiciel

L'agilité est une approche, correspondant à une philosophie, qui doit guider toute action et toute réflexion dans une structure organisée au service de ses clients. L'objectif premier est de produire au plus tôt un maximum de valeurs métier par incréments courts de grande qualité et industrialisés, et de réduire ainsi l'effet tunnel. Elle assure aussi une meilleure et plus rapide adaptation aux changements ; elle permet une démarche d'amélioration continue[26] et de tirer le maximum de l'intelligence collective d'une organisation. Elle vise à rendre plus naturelle une manière de fonctionner. L'agilité offre de la liberté et une certaine autonomie, tant sur le plan de l'organisation que de l'ingénierie, mais demande toutefois beaucoup de rigueur, un cadre précis et exigeant, et un suivi quotidien.

Elle s'appuie sur les valeurs promues par le manifeste agile [Manifesto 2001] :

1. la valorisation des individus et de leurs interactions, plutôt que des processus et des outils ; bien menée, l'agilité permet au fil des itérations une réelle maîtrise du projet et de la qualité du produit. Le principe consiste à fixer des objectifs petits, clairement identifiés, que l'on sait atteindre et sur lesquels un engagement sera possible. Cela permet, de respecter l'engagement posé, d'être fier et satisfait du travail accompli, puis de poursuivre.
2. un logiciel opérationnel, plus qu'une documentation exhaustive. L'agilité propose de faire des bouts d'application complets, de qualité, et que le comportement métier attendu soit vérifié en permanence ; on privilégie la fonctionnalité à la complétion des fonctionnalités. Cela ne signifie pas pour autant la fin de la documentation : au contraire le produit évolue régulièrement, il est donc nécessaire de maintenir une documentation nécessaire et suffisante et de capitaliser dessus.
3. la collaboration au sein des équipes et avec les clients (toute l'équipe est responsable d'une tâche), plutôt qu'une contractualisation rigide ; il s'agit de construire plus de valeur et la livrer au plus tôt. La vision du logiciel et du projet est portée collectivement (partage des mêmes objectifs, d'un même langage, des mêmes contraintes de budget, délai, organisation), pilotée par un représentant du client au sein de l'équipe de développement.
4. l'acceptation du changement et l'adaptation au changement, plutôt que le suivi d'une planification figée.

De ces valeurs découlent 12 principes : satisfaire le client, accepter le besoin de changement, livrer fréquemment, motiver les équipes, dialoguer face-à-face, faire simple, pour n'en citer que quelques-uns.

Ainsi, une espérance de mise sur le marché plus rapide, de productivité et qualité élevées, de coûts plus bas, d'une meilleure satisfaction des parties prenantes, d'un engagement plus fort des employés et de la satisfaction au travail, sont autant de raisons d'adopter une approche agile.

C. Scénario général d'un développement logiciel agile

On trouve plusieurs méthodes agiles, dont les plus connues sont Scrum [Scrum 2011], Safe [SAFe 2016], Lean Management [Lean 1988], Kanban [Kanban 2003]. Elles reposent sur des valeurs et des mécanismes similaires, qui sont présentés dans cette section, même si chacune présente des spécificités qui lui sont propres.

Le cycle de vie Agile est rythmé par de courtes itérations de quelques semaines. Le projet est découpé en fonctionnalités à développer (appelées User Story dans la terminologie Scrum).

Elles sont décrites en employant le vocable utilisé par le client. Elles sont souvent exprimées ainsi :

En tant que <qui>, je veux <quoi> afin de <pourquoi>

Elles représentent le besoin d'un point de vue utilisateur. A chacune sont associés une estimation de la quantité de travail nécessaire pour développer, tester, et valider cette fonctionnalité, ainsi qu'un test relativement simple qui constitue un test de validation. Les critères d'acceptation permettant d'indiquer le résultat attendu doivent être définis pour valider que la tâche a été correctement effectuée. Par exemple, pour la fonctionnalité « En tant que pilote, je ne veux pas que s'affiche le bouton sélection altitude si le mode maintenance est activé pour ne pas encombrer l'écran », le critère d'acceptation est « Si le mode maintenance est activé alors le bouton 'sélection altitude' n'est pas affiché ».

Ces fonctionnalités sont complétées par une description détaillée des choix techniques à implémenter.

La liste est dressée et hiérarchisée avec le client.

Plusieurs cérémonies cadencent l'exécution d'une itération. Un animateur (appelé Scrum master dans la méthode Scrum) est le garant du bon fonctionnement de la méthode.

Il organise, avant chaque itération, une réunion de planification, au cours de laquelle sont sélectionnées dans la liste

[26] Elle préconise de faire un point systématique à la fin de chaque projet sur ce qui s'est bien passé (les Bests) pour le reproduire et les points à améliorer (les Concerns).

les fonctionnalités les plus prioritaires pour le client. Elles seront développées, testées et livrées au client à la fin de l'itération.

Au cours de l'itération, sont organisées, chaque jour, de courtes réunions d'avancement au cours desquelles tous les membres de l'équipe indiquent les tâches effectuées la veille, les tâches prévues pour le jour et les difficultés rencontrées. Le but de cette réunion n'est pas de résoudre les problèmes mais de les identifier et les exprimer, afin de s'assurer que les objectifs de l'itération seront tenus. Suite à cette réunion, l'animateur met à jour ce qui a été fait et évalue le rythme de travail de l'équipe.

A la fin d'une itération, on fait une démonstration au client des derniers développements. C'est aussi l'occasion de faire un bilan, sur le fonctionnement de l'équipe et de trouver des points d'amélioration.

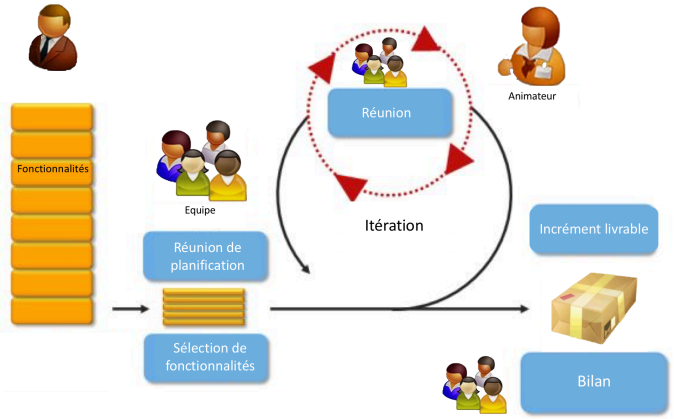

Fig. 12. Processus agile

D. *Intérêts de l'agilité dans un processus de développement de logiciels soumis à certification*

Comme nous l'avons vu précédemment, l'agilité vise à répondre au mieux au besoin du client. Dans un contexte de développement de logiciel soumis à certification, l'autorité peut être considérée comme un client suprême qu'il est nécessaire de satisfaire pour répondre aux enjeux sociétaux de sécurité aérienne.

La prise en compte de son besoin le plus tôt possible permet de réduire les risques et les coûts de développement. Elle augmente le niveau de confiance en garantissant que les activités ont été conduites au moment opportun et non juste avant la venue de l'auditeur pour lui faire plaisir. Elle facilite le déroulement de l'étape d'échantillonnage lors des audits de certification en assurant une traçabilité immédiate entre toutes les données et un enregistrement systématique des preuves de la réalisation des activités requises pour la certification (revues, résultat de test, taux de couverture structurelle, mesures des performances, ...).

Cette démarche nécessite la mise en place de processus outillés afin d'accomplir en continu l'ensemble des tâches requises pour répondre aux objectifs de la norme DO-178C ou tout autre développement logiciel critique (IEC 61508, ISO26262, …). L'enjeu est d'atteindre avant la fin de chaque itération un statut « Prêt pour la certification » pour chaque donnée ou activité produite (Exigence, Test, Revue). Cette vision novatrice est appelée ici « Certification continue ». La capacité d'avoir outillé tout le processus permet, en cas de besoin, de procéder sans difficulté à une refonte de l'application en mesurant immédiatement les activités nécessaires et suffisantes à reconduire. L'automatisation de tous les tests permettra un gain de temps non négligeable dans cette étape. Les équipes n'auront pas peur de faire évoluer leur logiciel et finalement le besoin du client et des autorités sera atteint à moindre coût.

E. *Initiatives actuelles*

Certaines initiatives ont été menées dans le domaine aéronautique pour introduire l'agilité dans le processus de développement de logiciels critiques. Ainsi, Airbus Helicopter a utilisé la méthode Scrum pour développer une nouvelle avionique (logiciels embarqués applications militaires) [Marsden 2018]. Mais la plus aboutie en la matière à l'heure actuelle concerne le développement du calculateur ADIRU par Thalès Avionics [Chenu 2013]. L'équipe en charge du développement a réussi à mettre en place un processus de « Livraison continue » avec son client. L'équipe a démontré la faisabilité de ce concept novateur, consistant à livrer très régulièrement un produit (ici, mêlant Hardware et Software) avec un périmètre fonctionnel réduit mais opérationnel. Le nombre de défauts constatés par le client a été divisé par 100 comparé à un projet similaire. Pour obtenir ces résultats, les équipes, fonctionnant de façon agile, ont compris que les objectifs de certification étaient non négociables et ont mis en place des méthodes et des outils pour automatiser le maximum d'activités (traçabilité, exécution des tests, analyse statique …).

Certes, la mise en place d'un tel processus de travail à un coût mais la valeur amenée pour le client final, la fierté de l'équipe du travail accompli et l'image renvoyée à l'autorité de certification est inestimable. La maturité affichée lors de l'évaluation d'un tel produit est gage de confiance pour autoriser sa mise en service.

Terminons en citant l'existence d'une proposition d'outillage. La société Sogilis a développé le logiciel « Autopilot » pour drone de niveau DAL A en appliquant certains principes des méthodes agiles, notamment le développement piloté par les tests [Mrabti 2018]. Ils sont allés plus loin en formalisant l'expression des cas de test à travers leur expression formelle qui facilitent l'automatisation de la vérification des fonctionnalités tout en respectant le principe primordial de tests basés sur les exigences.

F. *Verrous*

Rappelons que la norme DO-178C définit des objectifs à atteindre pendant les phases de développement à travers la mise en œuvre d'un ensemble d'activités. Ces activités sont regroupées par nature de processus. Les méthodes et les outils pour organiser et déployer les processus et les activités ne sont pas décrits dans la norme DO-178C. L'enjeu est de définir un flux de travail qui permette de planifier les activités et les processus dans des itérations Agile, en conformité avec la DO-178C [Mrabti 2018].

[Coe 2013] a examiné les valeurs agiles et les fondements de la DO-178C, et a identifié quatre sources principales de conflits possibles, comme on le voit en TABLE X.

TABLE X. VALEURS AGILES VERSUS RECOMMANDATIONS CLEFS DE LA DO-178C ADAPTE DE [COE 2013]

Valeurs agiles	**Recommandations clefs de la DO-178C**
Individus et leurs Interactions	Processus et outils
Logiciel opérationnel	Documentation exhaustive
Évolution des besoins en collaboration avec le client	Spécification rigoureuse des exigences
Adaptation au changement	Suivi d'un plan

De notre point de vue, cette étude interprète mal les principes Agiles et ceux promus par la norme DO-178C. Revenons sur chaque source potentielle de conflit identifiée une à une pour l'analyser plus en détails.

Individus et leurs interactions vs Processus et outils

Il est évident que les interactions humaines sont primordiales pour aboutir à un logiciel certifiable ; c'est encore plus vrai lorsque l'équipe doit présenter ses travaux à un auditeur. La dimension sociale est incontournable pour atteindre un niveau de confiance justifié. Une prise en considération, dès le début du projet, d'une capacité à être audité est nécessaire pour ne pas subir la situation. Une équipe fière du travail accompli, à l'aise avec son processus d'ingénierie, et qui est capable de présenter rapidement toutes les preuves requises, inspirera naturellement confiance.

Logiciel opérationnel vs Documentation exhaustive

La norme DO-178C impose seulement trois livrables documentaires, le PSAC (Plan for Software Aspects of Certification), le SAS (Software Accomplishment Summary) et SCI (Software Configuration Index). Le format des autres données d'ingénierie est laissé libre. Historiquement, l'industrie a choisi de manipuler des documents au format « Word » ou dans des bases de données type « DOORS » qui ne sont pas forcément adaptés pour répondre de façon fluide aux objectifs de la DO-178C. D'autres formats numériques, tels que HTML ou Markdown, semblent plus efficace pour assurer la traçabilité et la gestion de version des données documentaires.

Concernant la documentation, un amalgame est souvent constaté. L'agilité préconise une documentation minimale ce qui ne veux pas dire aucune documentation. Aucun projet informatique d'envergure ne peut exister sans la formalisation des informations techniques essentielles. Le processus de certification ne demande pas plus que ces informations essentielles qui permettent de garantir que le comportement implémenté dans le code source correspond à un spécification technique détaillée apte à être testée. Pour les logiciels critiques, en regard de la spécification détaillée, il ne doit pas y avoir de comportement absent ou supplémentaire au niveau du code source. Cela impose un niveau de granularité élémentaire assez faible pour qu'il n'y ait aucune interprétation possible par la personne chargée du codage. Les principes agiles d'incréments fonctionnels itératifs sont totalement adaptés à cette exigence.

Evolution des besoins vs Spécification rigoureuse

Pour un périmètre fonctionnel donné, les activités rigoureuses requises par la norme DO-178C peuvent être réalisées en mode Agile afin d'être en mesure de livrer un logiciel 'certification ready' à chaque itération. Si l'ajout d'une fonctionnalité induit un impact sur des fonctionnalités précédemment implémentées, l'itération considérée doit aussi prendre en compte les modifications requises des artefacts antérieurs (Spécifications, Tests, code source…). A défaut, elles doivent être programmées lors des itérations suivantes pour assurer le plus rapidement possible une cohérence d'ensemble. Le système de gestion de configuration doit être en mesure d'enregistrer l'historique des évolutions. En revanche, les mesures de performance globales et la complétude du comportement attendu ne pourront être établies qu'à la fin du développement du logiciel.

Adaptation au changement vs Suivi d'un plan

Enfin les plans doivent être le plus stables possible ce qui n'exclut pas qu'eux-mêmes prévoient d'être amendés dans le cadre d'un processus d'amélioration continu maitrisé.

Une difficulté nous semble toutefois encore à surmonter. Elle concerne la contractualisation de ce mode de fonctionnement où finalement le produit livré peut largement différer des fonctionnalités initiales exprimées. Dans un contrat classique, si le besoin vient à évoluer, le risque est entièrement supporté par le fournisseur. Quelques précautions peuvent être prises mais la souplesse des approches agiles sera limitée. Quelques initiatives ont vu le jour comme par exemple la forfaitisation de chaque itération avec la possibilité pour le client d'arrêter le contrat à chaque cycle. Le principe du troc d'exigences consiste à réaliser une fonctionnalité imprévue en échange du retrait d'une autre moins importante, non prioritaire et du coût équivalent. Des mécanismes plus classiques d'avenant autorisent un volume d'exigences modifiables (<10%). Enfin, certains industriels facturent à la fin de chaque itération. Cet enjeu contractuel concernant la prise en compte rapide de l'évolution du besoin au cours du développement sera de plus en plus important notamment pour traiter des problématiques de cyber-sécurité dont les cycles de menaces sont bien plus rapides que les cycles de développement actuels.

Notons toutefois pour finir que dans cet article, nous ne sommes volontairement pas rentrés dans le détail de la mise en œuvre des différentes méthodes agiles et sommes restés aux niveaux des principes fondamentaux. Il est ainsi important de préciser que les méthodes agiles présentent chacune des spécificités et que le choix de la méthode pour mener un projet devra être fonction du contexte et des contraintes. Par exemple, on ne choisira pas Scrum, ou tout au moins on l'adaptera, pour développer des logiciels soumis à certification. Scrum préconise en effet que toute l'équipe soit responsable de l'ensemble des tâches ; ce qui n'est pas directement compatible avec le principe d'indépendance des normes qui veut que l'équipe qui développe soit distincte de celle qui teste, et que les responsables des tâches soient identifiés et tracés. Dans cet

exemple, on préfèrera ainsi une méthode Lean management, dans laquelle chaque membre de l'équipe agile est responsable d'une tâche, ou alors une adaptation de la méthode Scrum pour répartir les tâches et enregistrer l'identité de la personne qui a réalisé l'activité.

VI. Conclusion et perspectives

Cet article a présenté la problématique générale de la certification du point de vue des autorités et exposé un état des pratiques industrielles dans différents domaines d'activité (automobile, santé, aéronautique, militaire…). Il a traité ensuite plus spécifiquement le développement de logiciels critiques dans le domaine de l'avionique. Il a notamment souligné les difficultés rencontrées en entreprise et mis en perspective quelques pistes d'améliorations possibles grâce à l'ingénierie dirigée par les modèles et aux méthodes agiles, ces deux pistes pouvant bien entendu être considérées en complémentarité.

Certains principes agiles, et l'interprétation qui en est faite, ne sont pas totalement compatibles avec un processus de certification. Cependant, plusieurs initiatives industrielles ont récemment expérimenté l'introduction de l'agilité dans le processus de développement. Les résultats de celles-ci ont démontré que ces nouvelles pratiques permettaient d'augmenter le niveau de confiance et de réduire les coûts de développement. En outre, l'agilité met l'accent sur les valeurs humaines dont la dimension est primordiale dans le contexte d'un développement logiciel sécuritaire, reposant sur la confiance. De telles initiatives méritent d'être encouragées, en les orientant pour satisfaire le besoin des autorités.

Le domaine aéronautique est précurseur dans l'application d'un cadre normatif strict. Il est clairement maîtrisé depuis plusieurs décennies par les concepteurs historiques. Nous voyons aussi de nouveaux acteurs se lancer dans l'aventure des drones ou des voitures autonomes en mettant en œuvre des techniques industrielles novatrices (agilité, intelligence artificielle, évaluation de la safety par les modèles). Le rythme des innovations proposées est souvent soutenu et en décalage avec la capacité des autorités à les retranscrire dans la réglementation. L'enjeu pour l'autorité est de réussir à suivre ces évolutions en les cadrant et les orientant pour que le risque acceptable reste maîtrisé.

La DGA, en tant qu'autorité de certification militaire, vise en effet à promouvoir des approches efficaces du point de vue de la certification mais pragmatiques du point de vue de la maîtrise des coûts ; elle est favorable de ce point de vue aux méthodes agiles. Comme cet article l'a illustré, elles présentent, dans un contexte de certification, un certain nombre d'avantages qui permettent d'augmenter le niveau de confiance en matière d'ingénierie logicielle et de sûreté de fonctionnement, tout en diminuant les coûts relatifs à la certification.

S'inspirant des initiatives visant à rendre la démarche plus agile et s'appuyant sur l'expertise méthodologique de la DGA, nos prochains travaux ont pour ambition de proposer une méthodologie agile basée modèles. Les domaines de la programmation des composants électroniques complexes (FPGA) et de l'évaluation des principes de sécurité par les modèles sont d'autres pistes à étudier pour étendre l'utilisation de cette approche. Les aspects juridiques méritent aussi certains éclaircissements. Enfin, les outils doivent s'améliorer pour faciliter la mise en œuvre de ces nouvelles pratiques. Des solutions comme [JIRA 2002], [TULEAP 2011] ou [Linty 2019] permettent déjà de répondre à certaines problématiques soulevées.

En complément, il semble nécessaire de proposer une approche pédagogique pour accompagner ce changement de pratiques. Une adaptation des formations initiales spécialisées vient d'être mise en place (INSA Toulouse Master en Ingénierie Système) pour prendre en compte la vision des autorités. Des modules de formation professionnelle spécifiques sont déjà proposés par différents organismes (DGA, Linty Services).

References

[1] [Hilderman 2007] Hilderman, Vance and Baghi, Tony. "Avionics Certification: A Complete Guide to DO-178 (Software), DO-254 (Hardware)". Avionics Communications. 2007.

[2] [Hilderman 2009] Hilderman, Vance. "DO-178B Costs Versus Benefits. HighRely", 2009. http://www.cems.uwe.ac.uk/~ngunton/hrwp_do_178b_cost_ benefit. pdf

[3] [JORF 2013] JORF, "Décret n° 2013-367 du 29 avril 2013 relatif aux règles d'utilisation, de navigabilité et d'immatriculation des aéronefs militaires et des aéronefs appartenant à l'Etat et utilisés par les services de douanes, de sécurité publique et de sécurité civile", 2013. https://www.legifrance.gouv.fr/eli/decret/2013/4/29/2013-367/jo/texte

[4] [Leveson 2003] Leveson, Nancy. "White Paper on Approaches to Safety Engineering", April 2003. http:// sunnyday.mit.edu/caib/concepts.pdf

[5] [Rempel 2014] Rempel, Patrick; Mäder, Patrick; Kuschke, Tobias; Cleland-Huang, Jane. "Mind the Gap: Assessing the Conformance of Software Traceability to Relevant Guidelines", International Conference on Software Engineering (ICSE), New York, USA, ACM: 943–954, 2014.

[6] [Leveson 1995] Leveson N., Safeware : System Safety and Computers. Addison-Wesley Professional, 1995.

[7] [Wessiani 2018] Wessiani N.A., Yoshio F., " Failure mode effect analysis and fault tree analysis as a combined methodology in risk management", IOP Conf. Series, April 2018.

[8] [Leveson 2004] Leveson N., "A New Accident Model for Engineering Safer Systems", Safety Science, Vol. 42, No. 4, April 2004, pp. 237-270.

[9] [Salmon 2018] Salmon, Burges. "Driverless cars: liability frameworks and safety by design : Insurance and Legal report". AXA 2018

[10] [JORF 2016] JORF n°0181 du 5 août 2016 texte n° 8, Ordonnance n° 2016-1057 du 3 août 2016 relative à l'expérimentation de véhicules à délégation de conduite sur les voies publiques. 2016. https://www.legifrance.gouv.fr/eli/ordonnance/2016/8/3/2016-1057/jo/texte

[11] [DGT 2018] Direccion General de Trafico, ministère de l'écologie et du développement durable espagnol. May 2018. http://www.automobilebarcelona.com/en/home

[12] [Macron 2018] Macron, Emmanuel, "Développement des véhicules autonomes - Orientations stratégiques pour l'action publique. Extraits du discours sur l'Intelligence Artificielle". Paris Collège de France – mars 2018. https://www.ecologique-solidaire.gouv.fr/sites/default/files/2018.05.14_rapport_vehicules_autonomes.pdf

[13] [Regulation 2016] Décret n°2006-1551 du 7 décembre 2006 relatif aux règles d'utilisation, de navigabilité et d'immatriculation des aéronefs militaires et des aéronefs appartenant à l'Etat et utilisés par les services de douanes, de sécurité publique et de sécurité civile, 2006. https://www.legifrance.gouv.fr/affichTexte.do?cidTexte=JORFTEXT000000463718&categorieLien=cid

[14] [IEC 61508 2010] International Electrotechnical Commission, "Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems", https://www.iec.ch/functionalsafety/standards/

[15] [RTCA DO-178C 2012] RTCA SC-205, EUROCAE WG-12, DO-178C/ED12C, “Software Considerations in Airborne Systems and Equipment Certification”, January 2012.

[16] [ISO 26262 2018] ISO 26262, ISO TC22/SC3/WG16, “Véhicules routiers - Sécurité fonctionnelle”, First edition 2011, retrieved 2018, https://www.iso.org/fr/search.html?q=26262

[17] [IEC 62304 2012] International Electrotechnical Commission, “Medical device software – software life cycle processes”, First edition 2006-05, Retrieved 2 June 2012.

[18] [ECSS Q-60 2013] European Cooperation for Space Standardization, “Electrical, electronic and electromechanical components”, October 2013.

[19] [EQ-80 2009] European Cooperation for Space Standardization, “Software product assurance”, March 2009.

[20] [EN 50128 2011] AFNOR, “Applications ferroviaires - Systèmes de signalisation, de télécommunication et de traitement - Logiciels pour systèmes de commande et de protection ferroviaire”, Octobre 2011.

[21] [IEC 61511 2016] International Electrotechnical Commission, “Functional safety - Safety instrumented systems for the process industry sector”, First edition 2005, Retrieved 2016.

[22] [ARP4754A 2011] Society of Automotive Engineers, Aerospace Recommended Practice “Guidelines For Development Of Civil Aircraft and Systems”, November 2011.

[23] [IEC 61513 2013] International Electrotechnical Commission, “Nuclear power plants – Instrumentation and control for systems important to safety – General requirements for systems”, First edition 2005, Retrieved 2016.

[24] [IEC 60880 2013] International Electrotechnical Commission, “Functional safety - Safety instrumented systems for the process industry sector”, First edition 2001, Retrieved 2013.

[25] [SERMA 2016] Serma Ingenierie à la demande de l’ANSM. Etude sur la sécurité des logiciels médicaux. SYS0000990_ANSM_Rapport_final_FR edition. Juillet 2016. https://ansm.sante.fr/var/ansm_site/storage/original/application/edd12a5999dc24a7fa6d6cda4e39469f.pdf

[26] [UNECE R13 2012] United Nations Economic Commission for Europe (UNECE), “Accord concernant l’adoption de prescriptions techniques uniformes applicables aux véhicules à roues, aux équipements et aux pièces susceptibles d’être montés ou utilisés sur un véhicule à roues et les conditions de reconnaissance réciproque des homologations délivrées conformément à ces prescriptions” - Révision 2, additif 12, règelement 13. December 2012. https://www.unece.org/fileadmin/DAM/trans/main/wp29/wp29regs/2013/R013r7am4f.pdf

[27] [UNECE R79 2017] United Nations Economic Commission for Europe (UNECE), Uniform provisions concerning the approval of vehicles with regard to steering equipment, UN regulation N°79, 2017.

[28] [Convention 1947] Convention on International Civil Aviation (Chicago Convention) Doc 7300. April 1947. https://www.icao.int/publications/Documents/7300_cons.pdf

[29] [EASA-1592 2002] EASA, Regulation (EC) No 1592/2002, July 2002. https://www.easa.europa.eu/document-library/regulations/regulation-ec-no-15922002

[30] [EASA-216 2008] EASA, Regulation (EC) No 216/2008, February 2008. https://www.easa.europa.eu/document-library/regulations/regulation-ec-no-2162008

[31] [EASA CS-25 2007] EASA, Certification Specifications for large aeroplanes, September 2007. https://www.easa.europa.eu/sites/default/files/dfu/CS-25_Amdt%203_19.09.07_Consolidated%20version.pdf

[32] [EASA CS-29 2012] EASA, Certification Specifications for large rotorcraft, December 2012. https://www.easa.europa.eu/sites/default/files/dfu/CS-29%20Amendment%203.pdf

[33] [EASA CS-25 2018] EASA, Certification Specifications for large aeroplanes, Amendment 21. March 2018. https://www.easa.europa.eu/sites/default/files/dfu/CS-25%20Amendment%2021.pdf

[34] [Druzin, 2016] Bryan Druzin, “Why does Soft Law Have any Power Anyway? “, 2016, https://www.cambridge.org/core/journals/asian-journal-of-international-law/article/why-does-soft-law-have-any-power-anyway/00EBCEA91F92F97E1079A80AE077BD39

[35] [ARP4761 1996] Society of Automotive Engineers, Aerospace Recommended Practice “Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment”, December 1996, https://www.sae.org/standards/content/arp4761/

[36] [Boehm 2000] Boehm B., Abts C., Winsor Brown A., Chulani S., Clark B., Horowitz E., Madachy R., Reifer D. and Steece B. *Software Cost Estimation with COCOMO II*, Englewood Cliffs, NJ:Prentice-Hall, 2000.

[37] [Vallee 2012] Vallee F., “Quantification de la fiabilité des logiciels”, 2012.

[38] [Chan 2012] Ka Ching Chan, C. T. Lenard, Terence M Mills, “An Introduction to Markov Chains”, MAV 49th Annual Conference, La Trobe University, Bundoora, Australia, December 2012.

[39] [RTCA DO-330 2011] Radio Technical Commission for Aeronautics (RTCA), “Software Tool Qualification Considerations”, December 2011. https://standards.globalspec.com/std/1461615/rtca-do-330

[40] [RTCA DO-331 2011] RTCA, “Model-Based Development and Verification Supplement to DO-178C and DO-278A”, December 2011.

[41] [RTCA DO-332 2011] RTCA, “Object-Oriented Technology and Related Techniques Supplement to DO-178C and DO-278A”, December 2011.

[42] [RTCA DO-333 2011] RTCA, “Formal Methods Supplement to DO-178C and DO-278A”, December 2011.

[43] [Hilderman 2009] Vince Hilderman, “DO-178B Costs Versus Benefits“, HighRely White Paper, 2009. http://www.highrely.com/whitepapers.php

[44] [Canals 2019] Canals Agusti H3230 "Méthode Agile Scrum", Les Techniques de l’Ingénieur, 2019. https://www.techniques-ingenieur.fr/base-documentaire/technologies-de-l-information-th9/genie-logiciel-42306210/methode-agile-scrum-h3230/

[45] [Scrum 2018] Scrum.org, “What is Scrum?”, consulted 02/12/2018. https://www.scrum.org/resources/what-is-scrum?

[46] [Beck 2004] Beck, Kent, Andres, Cynthia, “Extreme Programming Explained: Embrace Change”, Addison-Wesley, 2nd Edition (The XP Series), November 2004.

[47] [Goudeau 2016] Goudeau Stéphane, Metias Samuel,Découvrir DevOps, l’essentiel pour tous les métiers, Dunod, Mars 2016.

[48] [Verona 2016] Verona Joakim, “Practical DevOps”, Packt Publishing, February 2016.

[49] [Bender 2009] Bender RBT Inc, “Requirements Based Testing Process Overview”, 2009. http://benderrbt.com/Bender-Requirements%20Based%20Testing%20Process%20Overview.pdf

[50] [Rouahi 2007] Rouahi (J.) et Carlier (Y.) - Assurance de développement et chemins fonctionnels, séminaire Ingénierie des Systèmes Complexes à Logiciels Prépondérants organisé à DGA Techniques aéronautiques, septembre 2007.

[51] [DOD SYSTEM SAFETY 2012] Department of Defense, Standard Practice “System Safety”, 2012.

[52] [RTCA DO-278 2011] RTCA, “Software Integrity Assurance Considerations for Communication, Navigation, Surveillance and Air Traffic Management (CNS/ATM) Systems”, December 2011.

[53] [Gill 1991] Geoffrey K. Gill, Chris F. Kemerer, “Cyclomatic Complexity Density and Software Maintenance Productivity”, IEEE Transactions on Software Engineering, vol. 17, no. 12, December 1991.

[54] [Osetskyi 2018] Osetskyi Victor, “What Technical Debt Is and How to Calculate It”, Agile Zone, Opinion, July 2018. https://dzone.com/articles/what-technical-debt-it-and-how-to-calculate-it

[55] [NT DGATA 2016] DGA Techniques aéronautiques, Note Technique 16-DGATA-P1301261003001-1P-C “Référentiel d’exigences d'ingénierie des logiciels et composants électroniques complexes pour la prise en compte de la sûreté de fonctionnement“, 2016.

[56] [Letouzey 2012] J.-L. Letouzey, "The SQALE method for evaluating technical debt," in Proceedings of the Third International Workshop on Managing Technical Debt, pp. 31-36, 2012.

[57] [RTCA DO-254 2006] RTCA and EUROCAE, RTCA DO-254/EUROCAE ED-80 “Design assurance guidance for airborne electronic hardware”, 2006.

[58] [Combemale 2008] Combemale B., “Ingénierie Dirigée par les Modèles-État de l’art”, 2008. <hal-00371565>

[59] [SCADE 1999] Esterel Technologies, Safety Critical Application Development Environment (SCADE), 1999. http://www.esterel-technologies.com/products/scade-suite/

[60] [SIMULINK 1984] MathWorks, 1984, https://fr.mathworks.com/products/

[61] [Joshi 2006] Joshi A., Whalen M., Heimdahl M., “Model Based Safety Analysis Final Report”, NASA contractor report, NASA/CR-2006-213953, February 2006.

[62] [Bernard 2009] Bernard R. *Analyses de sûreté de fonctionnement multi-systèmes. Modélisation et simulation*. Université Sciences et Technologies - Bordeaux I, 2009. <tel-00441310>

[63] [Lisagor 2011] Lisagor O., Kelly T., Niu R., “Model-Based Safety Assessment Review of the Discipline and its Challenges”, International Conference on Reliability, Maintainability and Safety (ICRMS), 2011.

[64] [Leveson 2012] Leveson, N. Engineering a Safer World, MIT Press, January 2012.

[65] [Arnold 1999] Arnold, A., Point G., Griffault A., Rauzy A., “The AltaRica Formalism for Describing Concurrent Systems”, Fundamenta Informaticae, Volume 40 Issue 2-3, Pages 109-124, IOS Press Amsterdam, The Netherlands, November 1999.

[66] [Louis 2012] Louis V., “Rotorcraft HIRF qualification and accident investigations driven by model based analysis”, International Workshop on Model Based Safety Assessment, Bordeaux, France, 2012.

[67] [Bieber 2018] Bieber P., Farges J-L., Pucel X., Séjeau L-M., Seguin C., “Model-Based Safety Analysis for co-assessment of operation and system safety: application to specific operations of unmanned aircraft”, European Congress Embedded Real Time Software And Systems (ERTS), Toulouse, 2018.

[68] [Linty 2019] Linty Services, 2017. http://www.linty-services.com

[69] [IRT 2018] IRT Saint-Exupéry, MOISE ProjectMethods and tools for model-based collaborative system engineering (requirement engineering, multi-view modeling and verification, system engineering in extended enterprise), 2018. http://www.irt-saintexupery.com/expertise/embedded-systems/

[70] [Royce 1998] Royce W, “*Software project management: a unified framework*“, September 1998

[71] [Pillou 2018] Pillou J.F., “Concept de l’Intégration Continue”, issu de CommentCaMarche, 2018. https://www.commentcamarche.net/

[72] [Fowler 2006] Fowler M., Continuous Integration, https://martinfowler.com/articles/continuousIntegration.html, 2006.

[73] [Fowler 2013] Fowler M., Continuous Delivery, 2013. https://martinfowler.com/bliki/ContinuousDelivery.html

[74] [Duvall 2007] Duvall Paul, Matyas Steve, Glover Andrew. “Continuous Integration: Improving Software Quality and Reducing Risk“, 2007.

[75] [Humble 2011] Humble J, Farley D. “Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation“, 2011.

[76] [Vöst 2016] Vöst S., Wagner S., “ Towards Continuous Integration and Continuous Delivery in the Automotive Industry“, 2016.

[77] [Standish Group 2015] Standish Group International, Inc. “Report in Computer World”, 2015.

[78] [Xue 2017] Xue R., Baron C., Esteban P., “Optimizing product development in industry by alignment of the ISO/IEC 15288 Systems Engineering Standard and the PMBoK Guide”, International Journal of Product Development, vol. 22, issue 1, pp. 65-80, 2017.

[79] [Ninni 2019] Ninni L., Blog Launizo consulting, 2019. https://www.launizo.com/blog/methodes-et-outils-de-productivite-en-entreprise-1/post/les-methodes-agiles-3

[80] [Manifesto 2001] Cunningham W, Beck K., Fowler M, Thomas D, “Manifesto for Agile Software Development“, August 2001.

[81] [Scrum 2011] Sutherland J., Schwaber K., “Scrum Guide“, 2011.

[82] [SAFe 2016] Leffingwell D., “SAFe 4.5 Reference Guide: Scaled Agile Framework for Lean Enterprises “, 2018

[83] [Lean 1988] Krafcik J., “ Triumph of the Lean Production System“, 1988.

[84] [Kanban 2003] Anderson D., “Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results“, 2003.

[85] [Marsden 2018] Marsden J., Windisch A, Villermin J., Aventini C., Mayo R., Grossi J., Fabre L., ”ED-12C/DO-178C vs. Agile Manifesto – A Solution to Agile Development of Certifiable Avionics Systems”, Conférence Embedded Real Time Software And Systems (ERTS^2), Toulouse, France, Février 2018.

[86] [Chenu 2013] Chenu E., ”Integration Continue” séminaire Ingénieire des Systèmes Complexes à Logiciels Prépondérants, ISCLP, 2013.

[87] [Mrabti 2018] Mrabti A., Gautherot D., Brossard V., Moy Y., Pothon F., ”Safe and Secure Autopilot Software for Drones”, Conférence Embedded Real Time Software And Systems (ERTS^2), Toulouse, France, Février 2018.

[88] [Coe 2013] Coe David J., Kulick Jeffrey H., “A Model-Based Agile Process for DO-178C Certification”, World Congress in Computer Science, Computer Engineering, and Applied Computing, Las Vegas, USA, 2013. http://worldcomp-proceedings.com/proc/p2013/SER2918.pdf

[89] [JIRA 2002] Jira, Atlassian, 2002. www.atlassian.com/software/jira

[90] [TULEAP 2011] Tuleap, Enalean. 2011, www.tuleap.org