MINISTÈRE
CHARGÉ
DES TRANSPORTS

Liberté
Égalité
Fraternité

dgac direction générale de l'Aviation civile

BONNES PRATIQUES POUR LA RÉDACTION D'EXIGENCES DE SÉCURITÉ

GUIDE TECHNIQUE

MINISTÈRE
CHARGÉ
DES TRANSPORTS
Liberté
Égalité
Fraternité

dgac direction générale de l'Aviation civile

BONNES PRATIQUES POUR LA RÉDACTION D'EXIGENCES DE SÉCURITÉ

GUIDE TECHNIQUE

service technique de l'Aviation civile

Département Environnement, Sécurité des Systèmes et des Opérations, Planification

1ère édition du 01/02/2021

RÉDACTEURS

Laurent **PLATEAUX**
Chef de programme (DGAC/DSAC)

Laurence **MORIN**
Chef de programme (DGAC/STAC)

COMITÉ DE RELECTURE

André **BARKAT**
Chef de division « Navigation Aérienne » (DGAC/STAC/Département Environnement, Sécurité des Systèmes et des Opérations, Planification)

Victor **BOULANGER**
Chef de projet, Division Sécurité et Capacités aéroportuaires (DGAC/STAC/Département Environnement, Sécurité des Systèmes et des Opérations, Planification)

Romain **BUFFRY**
Chef de division « Equipements » (DGAC/STAC/Département Sûreté, Equipements)

Guillaume **ROGER**
Conseiller scientifique et international – Affaires internationales (DGAC/STAC/Direction)

Résumé

Le nouveau règlement d'implémentation (UE) n°2017/373, dit IR ATM/ANS, a intégré dans ses exigences réglementaires et ses moyens de conformité tous les principes développés dans les standards d'ingénierie système actuels. En effet, ces standards ayant pour objectif d'assurer la maîtrise de la complexité des systèmes, les approches proposées sont parfaitement adaptées à des systèmes socio-techniques complexes comme le sont les systèmes fonctionnels du contrôle aérien.

En reprenant tous les codes de l'ingénierie système, l'IR ATM/ANS met la qualité des exigences au centre de l'argumentation des évaluations de sécurité ou de support à la sécurité.

Le présent guide a vocation à accompagner cette évolution en compilant quelques bonnes pratiques en termes de rédaction des exigences. Ces bonnes pratiques ont pour but de faciliter la gestion des exigences, d'assurer leur bonne prise en compte dans le cadre de la conception d'un nouveau système ou dans le cadre d'un changement ainsi que de veiller à la vérification efficace et complète de celles-ci.

Les différents types d'exigences sont abordés ainsi que leurs propriétés et les méthodes de vérification adaptées. Quelques règles de rédaction sont également proposées.

Ce guide s'adresse à toute personne ou organisation ayant vocation à développer des évaluations de sécurité ou de support à la sécurité ainsi qu'à toute personne impliquée dans la spécification de systèmes socio-techniques.

Mots-clés

Navigation aérienne, études de sécurité, exigences, rédaction, bonnes pratiques, guide

SOMMAIRE

TABLES DES FIGURES

1. PRÉAMBULE

1.1. RAPPEL DU CONTEXTE

Ce guide de bonnes pratiques pour la rédaction des exigences est rédigé dans un but d'accompagnement des prestataires de la navigation aérienne ou de tout autre acteur de l'aviation. Il ne se substitue en aucun cas aux règlements en vigueur s'il venait en contradiction avec ces derniers. Il s'adresse à toute personne contribuant au processus de rédaction des exigences, que ce soit pour la rédaction ou pour la vérification de celles-ci. Bien que les principes de la rédaction des exigences développés dans ce guide sont aisément abordables, les éléments de contexte ainsi que les justifications de certaines pratiques nécessitent d'avoir une sensibilisation aux principes élémentaires de l'ingénierie système. Cela concerne notamment la notion de niveau d'ingénierie, le rôle des différents processus ainsi que les différentes données d'ingénierie type produites lors d'un processus d'ingénierie système complet.

Dans le cadre des évaluations de sécurité, ou des évaluations de support à la sécurité, le prestataire doit définir des exigences dont la satisfaction permet de garantir le respect des critères ou objectifs de sécurité et ainsi assurer un niveau de sécurité ou de service en adéquation avec le besoin réglementaire et opérationnel.

Ces exigences, dites « de sécurité » ou « de support à la sécurité », jouent, en effet, un rôle primordial dans la démonstration réglementaire de maîtrise des changements, telle que requise dans le cadre du règlement UE 2017/373. La qualité des exigences est un point clé vis-à-vis de la construction de l'argumentaire et nécessite l'application de bonnes pratiques. La mise en œuvre de ces dernières permet d'assurer à la fois une bonne compréhension des objectifs recherchés ainsi qu'une démonstration adaptée.

1.2. OBJET DU DOCUMENT

Ce guide propose quelques règles de bonnes pratiques pour la rédaction des exigences de sécurité et de support à la sécurité nécessaires à la maîtrise des changements des systèmes fonctionnels des prestataires de services de la Navigation aérienne.

Même si ce guide a pour vocation d'encadrer la rédaction d'exigences de sécurité définies dans le cadre des évaluations réglementaires, il est applicable à tous types d'exigences pour tous types de projets.

1.3. CONTENU DU GUIDE

Les chapitres 1 et 2 cadrent le document et définissent les principaux termes utilisés.

Le chapitre 3 présente une partie théorique sur la définition de la notion d'exigence et les grands principes de rédaction des exigences.

Dans le chapitre 4, quelques exemples d'exigences sont fournis avec ce qu'il faut faire et ne pas faire.

Ce guide sera amené à être enrichi au fil du temps en termes d'exemples ou de nouvelles pratiques.

Pour plus de précision sur les processus de définition, de rédaction ou de gestion des

1.4. DOCUMENTS ET STANDARDS DE RÉFÉRENCE

exigences ainsi que sur les principes généraux de l'ingénierie système, chacun pourra se référer aux standards listés au §1.4.

[Ref 1] ISO/IEC/IEEE 15288:2015 - Ingénierie des systèmes et du logiciel - Processus du cycle de vie du système.

[Ref 2] ISO/IEC/IEEE 29148:2018 - Ingénierie des systèmes et du logiciel — Processus du cycle de vie - Ingénierie des exigences.

[Ref 3] ED-153 - Guidelines for ANS Software Safety Assurance - Issued in August 2009.

[Ref 4] ED-109A - Software Integrity Assurance Considerations for Communication and Navigation and Surveillance and Air Traffic Management (CNS/ATM) Systems - Issued in January 2012

[Ref 5] SEBoK Editorial Board. 2020. The Guide to the Systems Engineering Body of Knowledge (SEBoK), v. 2.2, R.J. Cloutier (Editor in Chief). Hoboken, NJ: The Trustees of the Stevens Institute of Technology. www.sebokwiki.org. BKCASE is managed and maintained by the Stevens Institute of Technology Systems Engineering Research Center, the International Council on Systems Engineering, and the Institute of Electrical and Electronics Engineers Computer Society.

[Ref 6] specief.org - Société pour la Promotion Et la Certification de l'Ingénierie des Exigences en langue Française - http://www.specief.org

2. DÉFINITIONS

BESOINS

Les besoins représentent ce dont un utilisateur ou une organisation doit disposer pour accomplir une mission donnée. Ils sont exprimés du point de vue de l'utilisateur final et généralement en langage naturel. Leur rédaction peut suivre plus ou moins de formalisme comme par exemple celui des User Stories fréquemment utilisé dans le cadre des méthodes agiles.

CONCEPT OPÉRATIONNEL (D'UN SYSTÈME)

« Énoncé des hypothèses et des intentions d'une organisation à l'égard d'une opération ou d'une série d'opérations d'un système ou d'un ensemble de systèmes coopérant », SEBoK [Ref 5].

Nota : à noter que la [Ref 5] fait une distinction entre le « concept opérationnel » qui est spécifique à un système ou un ensemble de systèmes donnés et le « concept des opérations » qui a une portée plus large, au niveau de l'organisation. Le terme CONOPS est plutôt associé à la seconde notion cependant dans le cadre du développement d'un système donné on s'autorisera souvent à appeler CONOPs le concept opérationnel du système considéré. En effet, à l'échelle d'un système donné, le concept des opérations et le concept opérationnel sont équivalents. La notion de CONOPS dans ce guide, et notamment dans les abréviations, fera référence au concept opérationnel d'un système et non au concept des opérations d'une organisation.

CONCEPT DES OPÉRATIONS

« Énoncé, dans les grandes lignes, des hypothèses et des intentions d'une organisation à l'égard d'une opération ou d'une série d'opérations », SEBoK [Ref 5].

EXIGENCE

« Expression qui définit la propriété ou la contrainte d'un système, produit ou processus, qui est non ambiguë, claire, unique, cohérente, complète, vérifiable et jugée nécessaire pour répondre à un besoin opérationnel », SEBoK [Ref 5].

« Caractéristique observable de l'extérieur d'une entité donnée», Alan Davis [« 201 principles of software development »].

INGÉNIERIE SYSTÈME

L'ingénierie des systèmes ou ingénierie système est une approche scientifique structurée et inter-disciplinaire, dont le but est de formaliser et d'appréhender la conception, la validation et la vérification de systèmes complexes. Elle a pour objectif de maîtriser et de contrôler la conception de systèmes dont la complexité ne permet pas une approche simple.

SPÉCIFICATION

La spécification d'un système contient l'ensemble des exigences fonctionnelles et non-fonctionnelles et des contraintes de conception nécessaires à la description précise et complète d'un système. Elle constitue le référentiel technique pour la conception, la vérification et la validation du système.

SYSTÈME

Ensemble de composants structurés de sorte à accomplir un ou plusieurs objectifs.

Le terme « système », dans ce guide, est pris au sens large et peut faire référence à un système technique, un logiciel, un système socio-technique, un système de systèmes, un système de management, un système de processus, etc.

VALIDATION

Toutes les activités nécessaires à la démonstration que les exigences spécifiées sont correctes et complètes par rapport au besoin opérationnel. La validation répond à la question : « Avons-nous spécifié et concevons-nous le bon produit ? ».

VÉRIFICATION

Toutes les activités nécessaires à la démonstration que les exigences spécifiées sont satisfaites par le système tel que réalisé. La vérification répond à la question : « Le système se comporte-t-il tel que spécifié ? ».

3. CONSIDÉRATIONS THÉORIQUES

3.1. NOTION D'EXIGENCE : DÉFINITION ET OBJECTIF

3.1.1. DÉFINITION

Une des définitions les plus communément admise dans le monde de l'ingénierie système est que, pour un système donné, une exigence est l'expression d'une **propriété que ce système doit satisfaire pour être conforme au besoin** pour lequel il est conçu.

DÉFINITION

Une exigence est l'expression d'une **propriété** qu'un système doit satisfaire pour être **conforme au besoin** pour lequel il est conçu.

Le terme « propriété » peut faire référence à différentes caractéristiques du système comme une fonctionnalité, une caractéristique non-fonctionnelle, une contrainte, une performance, une interface, etc. Nous verrons plus loin comment sont exprimées les exigences portant sur ces différentes caractéristiques.

Cette définition est généralement complétée par une propriété forte qui est qu'une exigence est une **caractéristique observable de l'extérieur** du système concerné.

PROPRIÉTÉ D'UNE EXIGENCE

Une exigence est une caractéristique **observable de l'extérieur** du système concerné.

Nota bene : à titre d'information, dans la littérature anglaise, le terme « exigence » sera traduit par « requirement ».

3.1.2. OBJECTIF

Une exigence a la difficile mission d'exprimer ce qui est attendu d'un système. Elle doit exprimer le « quoi », « que doit faire le système ? », par opposition au « comment » qui relève de la conception du système. Une exigence qui exprimerait le « comment » n'est généralement pas conforme à la propriété qui est qu'une exigence doit être « observable de l'extérieur », cependant comme ce sera vu plus loin §3.2.3, il est parfois nécessaire de contraindre la conception.

Les statistiques sur les échecs et retards dans la conduite des projets (Standish Group par exemple) mettent en avant que plus de **50% de ces échecs sont dus à des défauts d'exigences :** ambiguïté, incomplétude, inexactes, oubliées, implicites, obsolètes, etc.

Les exigences jouent un rôle primordial dans le processus de conception d'un système. En effet cette exigence sera :

- Pour le commanditaire du système, le support pour évaluer la bonne compréhension de son besoin ;
- Pour le développeur/concepteur, le support pour la réalisation à proprement parler ;
- Pour les vérificateurs et valideurs, le support pour évaluer la pertinence et la complétude des activités de vérification et validation.

Compte tenu de la multitude d'acteurs amenés à manipuler une exigence et du rôle central de l'exigence dans l'activité d'ingénierie système, il convient d'y attacher une grande importance.

L'atteinte de l'objectif repose en très grande partie sur la qualité de la rédaction des exigences.

3.1.3. LA NOTION DE SPÉCIFICATION

Associée aux exigences, il est courant de rencontrer la notion de « **spécification** ».

Une « spécification » consiste en un recueil d'exigences et fait souvent office de référentiel contractuel ou référentiel produit pour définir l'ensemble des propriétés d'un système. Alors qu'une exigence doit être précise et complète tout en ayant un caractère unitaire, l'enjeu d'une spécification est d'être exhaustive, structurée et cohérente. En effet, elle devra contenir toutes les typologies d'exigences nécessaires à la caractérisation complète et cohérente du système.

DÉFINITION

Une spécification est un **recueil d'exigences** qui spécifie l'ensemble des propriétés d'un système.

Au-delà du fait de compiler les exigences, la spécification porte généralement une part de l'explication du contexte et une part de la justification et de la compréhension des exigences. Ces éléments apportent du liant avec les choix de conception ainsi qu'avec les exigences ou les besoins du niveau supérieur. Ils comportent également des éléments essentiels à la validation et à la vérification des exigences. Par ailleurs, il est commun que la spécification contienne un chapitre dédié à la traçabilité entre les exigences ou besoin du niveau supérieur et celles qu'elle contient.

Nota bene : à titre d'information, dans la littérature anglaise, le terme « spécification » est traduit par « specification » et il est souvent question de « requirements specification » qui signifie « spécification des exigences ». Ce document constitue généralement le recueil d'exigences en entrée d'une réalisation ou d'un contrat.

3.1.4. LA NOTION DE BESOIN

Il est également souvent fait allusion à la notion de « **besoins** » en ingénierie avec un lien particulier entre « besoins » et « exigences ». Bien qu'il n'y ait pas ici non plus de définition absolue du terme besoin, il est communément admis qu'il s'agit d'exprimer le « Pour qui ? » et le « Pour quoi faire ? », et se situe habituellement au niveau de l'utilisation opérationnelle du système. Le ou les besoins sont à mettre en relation avec les missions opérationnelles attendues et sont généralement exprimés du point de vue de l'utilisateur final et de façon plus naturelle et moins formalisée que les exigences. On les trouvera néanmoins dans certains contextes sous la forme d'exigences tout à fait formalisées. A l'instar des exigences qui sont compilées dans une spécification, les besoins sont eux rassemblés au sein d'un **concept opérationnel** ou **expression des besoins**.

La spécification du ou des systèmes doit permettre de couvrir tous les besoins identifiés dans le concept opérationnel.

Nota bene : à titre d'information, dans la littérature anglaise, le terme « besoin » est traduit par « system needs » et le « concept opérationnel », « Operational Concepts ». Il existe aussi le « concept des opérations », traduit par « Concept of Operations » qui constitue une vision haut niveau des opérations d'une organisation. À l'échelle d'un système, le concept opérationnel et le concept des opérations sont équivalents.

3.2. CARACTÉRISATION ET TYPOLOGIE DES EXIGENCES

Comme vu précédemment une exigence est une propriété d'un produit ou système qui permet de définir ce qui est attendu vis-à-vis d'un besoin particulier. Il est assez commun que pour un besoin donné, plusieurs exigences soient nécessaires ainsi que plusieurs types d'exigences.

Trois grandes familles d'exigences sont habituellement définies :

- Les exigences fonctionnelles ;
- Les exigences non-fonctionnelles ;
- Les contraintes : contraintes de conception et contraintes dites « autres ».

Nous allons voir qu'au sein de ces 3 familles, il peut y avoir plusieurs autres sous-familles. Cependant, il est essentiel de comprendre, d'abord, quel est l'objectif de typer les exigences.

Comme vu plus haut, une exigence sert à la fois au concepteur/développeur du système ainsi qu'aux équipes en charge de sa vérification ou de sa validation. Or, ces processus nécessitent la mise en œuvre d'activités variées et c'est précisément pour cette raison qu'il est utile de typer les exigences : selon le type, les activités à mener seront différentes et la formulation de l'exigence sera adaptée aux activités à mettre en œuvre.

Typer une exigence permet **d'anticiper** les activités de **conception** et de **vérification/validation** à mener.

Figure 1 : Typologie des exigences

3.2.1. LES EXIGENCES FONCTIONNELLES

Les **exigences fonctionnelles** sont les exigences les plus couramment rencontrées pour ce qui concerne notamment les systèmes techniques. Elles ont pour but de spécifier le comportement attendu du système et répondre à la question « que doit faire mon système ». Nous le verrons plus loin dans les règles de rédaction, une exigence fonctionnelle est souvent formulée sous la forme « Le système X doit [faire quelque chose] [avec telle performance] ». De nouveau, il faut s'attacher au « quoi » et non au « comment ».

Les exigences fonctionnelles doivent être formulées de sorte à ne décrire autant que possible qu'un seul comportement. Ce comportement doit être décrit aux bornes du système de sorte à ce que l'exigence respecte la propriété d'être observable de l'extérieur du système. Bien qu'elles doivent être unitaires, elles doivent également être complètes, c'est à dire qu'elles doivent définir complètement, au sein même de l'exigence :

- Le système devant implémenter l'exigence ;
- Le contexte de validité de l'exigence, les prérequis ou préconditions (« dans le mode X », « lorsque telle condition est satisfaite », « dans tel contexte technique ou opérationnel », etc.) ;
- Le comportement attendu (la sollicitation nécessaire, l'action ou le résultat de l'action, etc.) ;
- Les interfaces précises qui entrent en jeu (sollicitation humaine, interface réseau, bouton, souris, interrupteur, relais, protocole, etc.) ;
- Éventuellement les post-conditions (changement de mode, indisponibilité temporaire, etc.) ;
- Les performances attendues de l'action (temps d'exécution, débit, consommation électrique, temps de réponse, gigue, latence, hauteur, puissance, distance, etc.) et/ou du résultat de l'action (précision, tolérance, durée, force, température, etc.).

Les **exigences de robustesse**, bien que souvent considérées comme à part, ne sont rien d'autres que des exigences fonctionnelles qui ont vocation à définir le comportement attendu en cas de situations inattendues ou inadaptées (erreur de syntaxe, valeur non prévue, débit trop important, absence de donnée, appui intempestif sur un bouton, détection d'action). La grande difficulté dans la formulation d'une exigence de robustesse est de bien encadrer les conditions dans lesquelles le comportement est attendu. Autant il est aisé de définir les données et conditions pour lesquelles nous attendons un comportement précis, autant il peut être extrêmement fastidieux de définir toutes les conditions non prévues ou non attendues pour lesquelles une action de robustesse sera nécessaire. Il faudra veiller dans tous les cas à éviter la formulation négative et bien spécifier le comportement attendu (et non celui interdit !) en cas de situations anormales. Plusieurs exemples sont proposés dans ce guide au paragraphe 4.2.

Les exigences de **robustesse** sont avant tout des **exigences fonctionnelles** et doivent être spécifiées pour être validées et vérifiées.

Toutes les exigences fonctionnelles se traduisent par une **fonctionnalité** devant être réalisée/accomplie par un ou plusieurs composants du système : une fonction d'un composant logiciel, un comportement mécanique, électronique, une action humaine, une activité quelconque.

Les **exigences fonctionnelles** se traduisent toujours par une ou plusieurs **fonctions** devant être implémentées dans un ou plusieurs composants du système.

En termes d'activité de validation et de vérification, dans la très grande majorité des cas, les exigences fonctionnelles seront :

- Validées, a priori, par simulation, analyse, équivalence ou prototypage ou, a posteriori, par test, démonstration, évaluation opérationnelle, ou autre, et ;
- **Vérifiées** par tests.

La spécification des exigences fonctionnelles nécessite de connaître précisément les données d'entrée manipulées, les données de sorties, les types de sollicitations ou d'interaction possibles et leurs caractéristiques. Ces informations sont généralement capitalisées au sein d'**exigences d'interfaces**.

Les exigences d'interface doivent spécifier de façon **exhaustive** toutes les **interfaces intervenant entre le système lui-même et le monde extérieur**. Elles sont des données d'entrée indispensables à la rédaction des exigences fonctionnelles.

Celles-ci permettent de définir toutes les interfaces intervenant dans les échanges entre le **système considéré et le monde extérieur**. Tout comme une exigence fonctionnelle doit pouvoir être observée de l'extérieur, une exigence d'interface ne doit concerner que des interfaces servant aux interactions avec l'extérieur du système. Attention, les interfaces entre les composants du système, donc internes au système, relèveront de la spécification des composants de niveau inférieur.

Les enjeux de la description des interfaces sont l'exhaustivité et la précision. Si les défauts d'exigences sont la cause de plus de 50% des échecs projets, les problèmes d'interfaces, bien que souvent plus faciles à résoudre et ne conduisant que rarement à des abandons de projets, sont responsables pour une grande part des re-conception et reprises projets. S'agissant d'exigences qu'il n'est effectivement possible de vérifier qu'en contexte simulé voire opérationnel, donc très tard, le coût de reprise de ces anomalies est bien souvent très élevé. S'il peut s'avérer, pour celui qui spécifie le système, extrêmement laborieux d'identifier et de lister les interfaces de façon exhaustive, il faut avoir conscience que tout manque ou toute imprécision fera défaut dans la réalisation car personne n'est mieux placé que le prescripteur du besoin pour caractériser l'environnement d'exploitation du système cible.

Dans la mesure du possible, ces exigences d'interface devront être les plus précises possible afin d'identifier, d'une part, les espaces nominaux de fonctionnement et, d'autre part, les modes dégradés ou appelant à assurer de la robustesse.

Concernant la vérification des exigences d'interfaces, elle se fait à 2 niveaux. Dans un premier temps, il s'agit de relecture en s'assurant que les exigences d'interface sont toutes couvertes par les exigences fonctionnelles et correctement référencées. En particulier, toutes les interfaces de sortie doivent faire l'objet d'au moins une exigence fonctionnelle.

Dans un second temps, il s'agit de vérifier la bonne implémentation de celles-ci. Cependant, cela se fait de façon indirecte à travers la vérification des exigences fonctionnelles qui s'appuient sur les interfaces. Il n'y a donc pas de test d'interface à proprement parler, en revanche les exigences fonctionnelles doivent être testées de sorte à exercer efficacement la totalité de l'interface.

3.2.2. LES EXIGENCES NON FONCTIONNELLES

Les **exigences non-fonctionnelles** regroupent toutes les exigences qui n'ont pas vocation à exprimer un comportement. Elles traduisent habituellement une **caractéristique générale** du système (la liste suivante n'est pas exhaustive) :

- Facteur de forme (taille, volume, poids) ;
- Ergonomie ;
- Performances générales par rapport aux ressources disponibles (consommation électrique, utilisation CPU/Mémoire, nombre d'opérateurs, débits maximum, etc.) ;
- Environnement opérationnel et technique (température, humidité, vibrations, bruit, sismicité, etc.) ;
- Performances intrinsèques du système (sûreté de fonctionnement [fiabilité, disponibilité, maintenabilité, testabilité], sûreté, sécurité, …).

La formulation de ces exigences est bien différente de celle des exigences fonctionnelles puisqu'il n'y a pas d'action spécifique attendue mais plutôt une propriété générale du système. Ces exigences devront néanmoins être précises et complètes, tout comme les exigences fonctionnelles et devront préciser :

- Le système concerné par l'exigence ;
- Le contexte de validité de l'exigence ;
- La caractéristique non-fonctionnelle abordée dans l'exigence ;
- L'objectif visé ;
- La tolérance ou la précision avec laquelle doit être atteint l'objectif (un objectif absolu est souvent irréaliste).

Les **exigences non-fonctionnelles** influencent dans la plupart des cas **l'architecture** du système et non directement les fonctionnalités des composants du système.

Contrairement aux exigences fonctionnelles qui sont directement déclinées en fonctions sur les composants du système jusqu'à être finalement implémentées, les exigences non-fonctionnelles influencent généralement les **choix de conception ainsi que l'architecture**. C'est particulièrement bien illustré par une exigence de type disponibilité qui ne correspondra à aucune fonction particulière au niveau du système considéré mais qui contraindra vraisemblablement l'architecture pour obtenir le niveau de disponibilité attendu, en faisant appel, par exemple, à de la redondance au sein des composants du système. Ces exigences non-fonctionnelles peuvent néanmoins nécessiter la mise en œuvre de composants complémentaires, pour les besoins de l'architecture et donc générer (et non décliner !) des exigences fonctionnelles sur les composants du système. Par exemple, dans le cas de la redondance, il est probable que cela induira des exigences sur le basculement, sur la reprise des fonctions, sur la synchronisation des états, etc.

En termes de validation et de vérification, les activités mises en œuvre sont généralement différentes des activités menées pour les exigences fonctionnelles et nécessitent la plupart du temps des moyens différents, des durées plus longues, des analyses et synthèses de plusieurs tests. Certaines caractéristiques, difficilement appréciables dans des durées de test acceptables, seront vérifiées par analyse et confirmées lors d'une utilisation prolongée ou nécessiteront des moyens permettant d'accélérer les observations. C'est le cas, par exemple, pour des performances comme la fiabilité qui peuvent être vérifiées a priori par analyse ou simulation et confirmées au fur et à mesure de l'utilisation. De la même manière, une exigence spécifiant l'interdiction de toute cause de panne unique, devra être vérifiée par analyse de l'architecture.

Un point sur les **exigences de performance**. Le lecteur attentif aura noté qu'il est question de performance à la fois pour les exigences fonctionnelles et pour les exigences non-fonctionnelles. Il est important de dissocier, d'une part, les performances associées à une **fonction donnée** et, d'autre part, les performances **générales d'un système**. Les premières doivent être spécifiées au sein des exigences fonctionnelles concernées afin que ce soit clairement identifié pour l'implémentation de l'exigence ainsi que pour sa vérification qui devra mesurer cette performance.

La performance d'une fonction doit être spécifiée dans l'**exigence traitant de cette fonction**, tandis qu'une **performance globale** doit faire l'objet d'une **exigence de performance spécifique**. Le cadre de leur vérification sera éminemment différent.

On spécifiera ainsi au sein d'une même exigence : « Lorsque l'opérateur appuie sur le bouton X, le système Y doit afficher la position de l'objet Z en moins de 1 seconde avec une précision de 5 mètres ». Une fonctionnalité réalisée avec des performances dégradées ne pourra pas être jugée satisfaite.

Le second type d'exigences de performance sont des attributs du système global et peuvent être formulées de façon plus générale, sans concerner une fonctionnalité en particulier. Généralement des tests spécifiques (charge, endurance, environnement, etc.) devront être mis en œuvre, au-delà d'un simple test fonctionnel. Il sera donc possible de spécifier « le système X doit avoir une disponibilité opérationnelle de 360 jours sur 365 » ou « l'occupation mémoire du système X doit toujours être inférieure à 80% de sa capacité totale ».

Dans le cadre des exigences de performances, il est également possible de trouver des exigences de performance humaine portant sur l'efficacité attendue des tâches réalisées, des temps de réaction ou autre.

3.2.3. LES CONTRAINTES

Parmi les contraintes, il est possible de faire une distinction entre 2 catégories. La première porte sur des contraintes dites de conception qui ont une influence délibérée et immédiate sur les choix de conception du système. La seconde porte sur des contraintes qui relèvent plutôt de l'environnement de développement ou de production du système et n'exerce pas une contrainte immédiate sur l'architecture. Elle peut contraindre la conception du système mais de façon indirecte.

Quelle que soit la catégorie qui nous intéresse, l'aspect fondamental des contraintes est qu'elles ne doivent en aucun cas venir en contradiction avec les exigences fonctionnelles ou non-fonctionnelles. Si cela se produisait, cela signifierait que nous aurions in fine des problèmes de faisabilité en termes fonctionnels ou de performances. La difficulté de cet aspect est que l'incompatibilité peut n'être détectable que très tardivement dans le cycle de développement. Aussi, il est important de ne pas abuser des contraintes quelles qu'elles soient et de préférer une approche favorisant la spécification du juste besoin de façon précise et complète.

3.2.3.1. CONTRAINTES DE CONCEPTION

Les **contraintes de conception** sont des exigences particulières dans le sens où elles contraignent la conception d'un système non pas par des besoins fonctionnels mais par des choix de conception imposés. Ces contraintes de conception peuvent être issues de contraintes règlementaires, de normes et standards applicables à un domaine donné ou de retours d'expérience sur des systèmes similaires. Ces contraintes consistent à contraindre la solution.

Les **contraintes de conception contraignent l'architecture** du système considéré, dans son organisation ou dans les choix de ses composants. Il est nécessaire de veiller à la **compatibilité des contraintes de conception avec les exigences fonctionnelles** qui portent le besoin opérationnel.

À titre d'exemple, une contrainte de conception pourrait être : « Le système TEST doit s'appuyer sur le système d'exploitation Windows 7 update x. ». Cette exigence contraint de fait l'architecture du produit vis-à-vis du système d'exploitation et contraint, en plus, indirectement, les performances, les fonctionnalités et la plateforme matérielle. Il faudra dans ce cas être vigilant vis-à-vis d'exigences fonctionnelles nécessitant du temps réel dur, incompatible avec une technologie telle que Windows, d'exigences de performance en termes de consommation électrique incompatibles avec les plateformes matérielles supportées par Windows 7, l'utilisation d'applications incompatibles avec celui-ci, etc. Malgré les risques ou contraintes que cela apporte, une telle contrainte de conception peut être justifiée par un besoin d'harmonisation du parc informatique en vue de son administration, par des questions de formation ou de compétence des personnels ou des questions de coût. Il faudra donc veiller à sa cohérence par rapport aux besoins fonctionnels du produit.

Les contraintes de conception ont un impact immédiat sur l'architecture du système : soit en termes d'organisation des composants, soit en termes de choix de composants eux-mêmes. Si l'usage de contraintes de conception peut être tout à fait justifié, il sera préférable de spécifier le besoin réel imposant ce choix. Cela permet :

- D'ouvrir les possibilités en termes de solutions techniques,
- D'éviter les risques d'incohérence entre une exigence fonctionnelle ou de performance et une contrainte de conception,
- De mieux capter le besoin réel dans la perspective d'une évolution ultérieure
- De ne pas avoir à revoir la spécification en cas de gestion d'obsolescence.

La vérification des exigences de conception se fait généralement par analyse ou inspection. En effet, comme elles ne spécifient que des contraintes sur l'architecture ou des choix de conception, elles ne correspondent pas à un comportement spécifique. Il sera cependant nécessaire de s'assurer lors des tests de vérification que les contraintes de conception n'empêchent pas la satisfaction d'une exigence fonctionnelle.

Les contraintes de conception peuvent être diverses et concerner des problématiques d'architecture, de conception, de conformité réglementaire, d'application de standards de conception, de réutilisation de l'existant, etc. Elles sont dans la grande majorité des cas liées à l'environnement opérationnel et technique du système.

3.2.3.2. CONTRAINTES « AUTRES »

La seconde catégorie de contraintes, dites contraintes « autres », ne relève pas d'un besoin opérationnel ou technique du système à proprement parler mais plutôt du contexte ou de l'environnement de développement. Ces contraintes portent généralement sur la phase du cycle de vie du système qui correspond au développement ou à la production du système et non à son utilisation.

Parmi ces contraintes, on peut citer par exemple un coût de développement maximum, des contraintes de délais, des processus spécifiques à appliquer, des contraintes réglementaires, un environnement de production, etc.

La vérification de ces contraintes ne se fera pas non plus par test et consistera plutôt en de l'inspection ou de la revue de plans de développement, d'éléments de planification projet ou de démonstration de conformité réglementaire.

Comme celles de conception, ces contraintes « autres » ne devraient pas avoir d'influence sur les exigences fonctionnelles du système à concevoir. Si tel était le cas, cela constituerait une incohérence et pourrait conduire à un problème de faisabilité. Par exemple, développer un système d'une grande complexité mettant en œuvre des fonctionnalités extrêmement innovantes avec des contraintes de coût et de délai tellement fortes qu'elles en deviennent incompatibles avec les objectifs fonctionnels recherchés. On pourrait également envisager une contrainte portant sur un environnement de test incompatible avec les niveaux de performances recherchés et à mesurer.

Si les contraintes « de conception » sont généralement formalisées dans les spécifications systèmes compte tenu de leur influence immédiate dans la conception du système, les contraintes « autres » figurent plutôt dans les documents de cadrage projet.

3.3. LA SÉCURITÉ : UNE TYPOLOGIE ET UN ATTRIBUT

Nous avons vu dans les paragraphes précédents différentes typologies d'exigences qui permettent de décrire un système donné de façon la plus exhaustive et précise possible. Toutes ces exigences trouvent leur place dans la conception d'un système, que ce soit dans l'architecture (exigences non-fonctionnelles) ou dans les fonctionnalités des composants (exigences fonctionnelles).

Dans le domaine du contrôle aérien et, plus généralement, dans les domaines dits critiques ou sécuritaires, il y a un type d'exigences qui revêt un rôle tout particulier. Ce sont les **« exigences de sécurité »**.

Parmi les exigences de sécurité il faut bien distinguer les exigences non-fonctionnelles de sécurité, qui fixent les objectifs de sécurité à atteindre, et les exigences ayant un attribut sécurité, visant à atteindre les objectifs de sécurité.

Il est important de distinguer deux types d'exigence de sécurité :

- Le premier est effectivement une typologie d'exigence : il s'agit d'un type d'exigences qui appartient aux exigences non-fonctionnelles et qui ont généralement pour but de fixer les objectifs ou critères de sécurité du système : taux de défaillance par heure opérationnelle, disponibilité opérationnelle, sévérité maximale d'une défaillance, etc. Ces critères sont souvent déclinés directement de matrices d'acceptabilité du risque ou de méthodes de type HAZOP (*Hazard and Operability analysis*), ALARP (*As Low As Reasonably Practical*), FHA *(Functional Hazard Analysis*) ;
- Pour ce qui concerne le second type, il s'agit plutôt d'un attribut transverse aux typologies vues précédemment. En effet, nous rencontrerons à la fois des exigences fonctionnelles de sécurité ainsi que des exigences non-fonctionnelles de sécurité. Cet attribut relève d'analyses de sécurité identifiant certaines propriétés du système nécessaires pour assurer un niveau de sécurité donné. Cet attribut facilite le suivi des exigences contribuant à atteindre les objectifs ou critères de sécurité mais ne constitue pas une typologie à proprement parler. Par exemple, on retrouvera dans ces exigences, des exigences fonctionnelles portant sur les fonctions de sécurité (« En cas de détection d'une défaillance matérielle, le système X devra envoyer un message [FAIL_MAT] au système de supervision en moins 5 secondes », « En cas de non-réception de message pendant plus de 5 sec, le système X affichera un bandeau rouge [Perte de Connexion] tel que défini dans la définition d'IHM réf [YYY]. ») ainsi que des exigences non-fonctionnelles (« La disponibilité opérationnelle du système X doit être supérieure 363 jours sur 365 avec des durées d'indisponibilités totales inférieures à 4h », « Le système X ne doit pas comporter de cause de panne unique », « Le système X doit disposer de 2 chaines fonctionnelles redondées. », etc.).

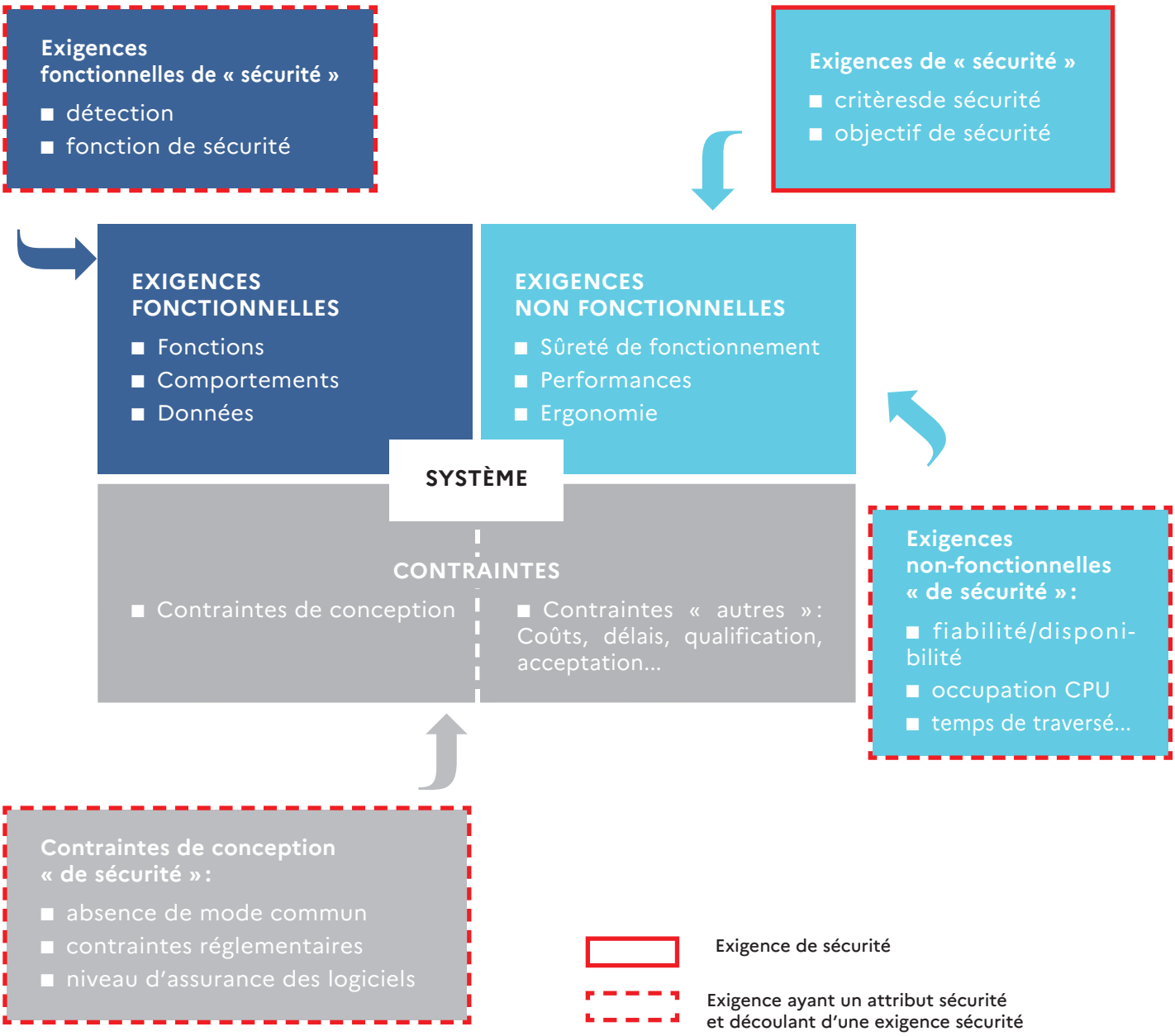

Figure 2 : Typologie et attribut de sécurité

3. CONSIDÉRATIONS THÉORIQUES

Dans le cadre des exigences de sécurité, il est également aussi commun de rencontrer des exigences se rapportant plutôt aux processus, à des contraintes « autres » d'ordres contractuels ou d'ingénierie. Si ces exigences ont bel et bien de l'importance quant à la satisfaction des objectifs et critères de sécurité, elles n'ont généralement pas leur place dans une spécification système tant leur vérification relève d'activités bien plus globales que celles engagées pour la vérification d'un système. Elles doivent donc être formalisées dans les plans de développement, dans la partie processus et gestion de projet d'un contrat, etc. Un cas typique d'exigence de sécurité de ce type est « Le logiciel du système X doit être développé avec un niveau d'assurance développement ED109A/AL4 ou ED153/SWAL3 ». Cette exigence n'apporte aucune information sur ce que doit faire le logiciel mais contraint les processus de développement pour réaliser les fonctionnalités spécifiées par les exigences fonctionnelles et non-fonctionnelles. Seule la mise en œuvre de processus de développement et d'une organisation adaptée permettra d'assurer la tenue de cette exigence. La conformité de cette exigence de sécurité devra toujours être associée à un référentiel fonctionnel donné pour le système et le logiciel.

Le règlement UE n°2017/373 met en évidence l'importance des exigences de sécurité qui permettent, si satisfaites, de démontrer la tenue des critères de sécurité d'un changement pour un prestataire dit ATS. Les exigences de sécurité évoquées dans le règlement rassemblent à la fois des exigences de sécurité à proprement parler (objectifs/critères de sécurité) et des exigences fonctionnelles/non-fonctionnelles/contraintes de sécurité, les secondes découlant généralement des premières. C'est bien la **démonstration de la satisfaction de l'ensemble de ces exigences qui portera la validité de l'argumentaire**.

Ce règlement UE n°2017/373 définit également la notion d'« Exigences de support à la sécurité » pour les services certifiés des prestataires dits non-ATS. Cette notion de « support à la sécurité » se rapproche de l'attribut de sécurité et ne constitue pas en soi une typologie d'exigence. Ces exigences peuvent être fonctionnelles, non-fonctionnelles ou de contrainte. Elles sont qualifiées de « support à la sécurité » pour mettre en évidence leur importance dans le rendu du service non-ATS final compte tenu de leur contribution directe sur le niveau de sécurité du service ATS l'utilisant.

3.4. NIVEAU DES EXIGENCES ET TRAÇABILITÉ

On parle de niveau des exigences pour situer le niveau d'ingénierie auquel on se situe. Lorsque l'on exprime une exigence, il faut impérativement identifier le niveau du système auquel on se situe. C'est pourquoi nous verrons qu'une exigence doit être formulée sous une forme du type « le système X doit… ». Cette notion de niveau est indispensable à maîtriser si l'on veut respecter la propriété qui dit qu'une exigence doit être observable aux bornes du système.

Il n'est pas possible, dans ce guide, de développer plus avant la notion de niveau des exigences ainsi que le rôle que joue le processus de conception dans cette notion, cependant cet aspect est fondamental pour réaliser des formulations correctes d'exigences et pour comprendre comment assurer un flux d'exigences corrects d'un niveau à l'autre. Il sera utile pour cela de se référer à des ouvrages ou des standards (voir §1.4) abordant en détail le sujet de l'ingénierie système.

Pour ce qui concerne ce guide, il faut avoir en tête qu'une des activités importantes en matière de vérification des exigences consiste à assurer la complétude de la couverture des exigences d'un niveau par les exigences de niveau inférieur. Cela assure, de proche en proche, le fait que le besoin opérationnel est bien implémenté ou réalisé et disponible. Cela permet également d'identifier d'éventuelles limitations lorsque certaines exigences ne sont pas satisfaites ou lorsque les performances ne sont pas au rendez-vous. Ces considérations ont d'autant plus de sens lorsque les exigences concernées sont des exigences de sécurité.

Associée au niveau d'exigence, nous retrouvons systématiquement la notion de traçabilité. Cette traçabilité, qui lie entre elles les exigences amont et aval, a précisément pour but de faciliter l'évaluation de la couverture des exigences d'un niveau à l'autre et nous verrons que ce besoin de traçabilité nécessite de favoriser d'une part l'unicité du comportement décrit dans une exigence et d'autre part l'identification unique des exigences. Grâce à ces propriétés, la traçabilité permettra notamment :

- Pour la traçabilité aval :
  - De vérifier la couverture des exigences du dessus par les exigences du dessous ;
  - De réaliser les analyses d'impact lors d'un changement des exigences amont.
- Pour la traçabilité amont :
  - De vérifier/justifier la nécessité d'une exigence de bas niveau (fonction non désirée) ;
  - De réaliser une analyse d'impact sur le système lors du changement dans un composant (notamment COTS).

3.5. EXIGENCES, TESTS ET TRAÇABILITÉ

Comme vu précédemment, les exigences constituent le référentiel applicable pour définir les activités de vérification et de validation. À chaque niveau d'exigence, il est possible d'associer un niveau de vérification et/ou de validation, du composant le plus unitaire vers le système intégré complet. La pertinence de la vérification et de la validation est d'autant plus grande qu'il est possible d'établir la complétude des tests vis-à-vis des différents comportements attendus du système et ce dans toutes les configurations envisageables.

La traçabilité entre les exigences d'une part et les **activités de vérification et de validation** d'autre part apporte le **support indispensable à la démonstration de la complétude** des activités de vérification et validation.

La démonstration de cette complétude s'appuiera notamment sur la traçabilité entre les exigences et les activités de vérification et validation. Ce pourra être classiquement un lien de traçabilité entre une exigence et un ou plusieurs tests dans le cadre d'une activité de vérification mais aussi un lien de traçabilité entre une exigence de conception et une activité d'analyse démontrant la satisfaction de l'exigence ou encore un lien de traçabilité entre une exigence et un compte-rendu de démonstration dans le cadre d'une activité de validation d'exigence.

Ce besoin de traçabilité implique notamment l'unicité des exigences ainsi qu'une identification claire.

3.6. SYNTHÈSE

L'ensemble des typologies d'exigences vues plus tôt et le flux d'exigences entre les différentes activités d'ingénierie sont synthétisés dans la Figure 3. La projection sur un cycle en V permet d'identifier clairement les différentes relations de traçabilité ainsi que les différents flux. Cependant, cette représentation ne préjuge pas de la dynamique du développement ou du cycle de vie appliqué ni de la forme que peuvent prendre les exigences, les liens de traçabilité ou les transferts de données.

Cette représentation est tout à fait valable pour un cycle de développement de type modèle en cascade, en spirale ou de type Scrum.

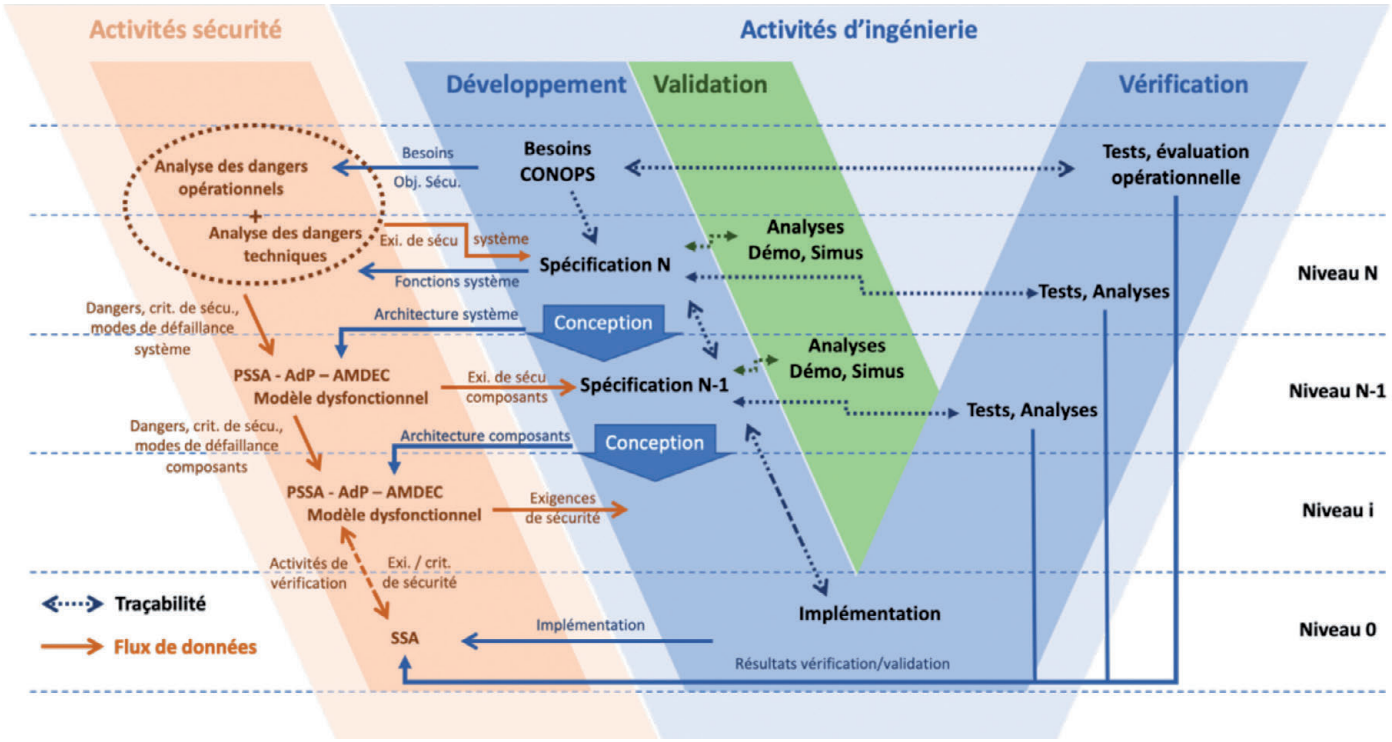

Figure 3 : synthèse des flux d'exigence et traçabilités

Glossaire de la figure

- AdP : Arbre de Panne
- AMDEC : Analyse des Modes de Défaillance, de leurs Effets et de leur Criticité
- CONOPS : Concept Opérationnel.
- PSSA: Preliminary System Safety Assessment
- SSA: System Safety Assessment

4. BONNES PRATIQUES POUR RÉDIGER DES EXIGENCES

4.1. LES BONNES PROPRIÉTÉS D'UNE EXIGENCE

Comme vu plus haut, les exigences sont d'abord un outil de coopération qui permet d'échanger sur l'attendu du système entre le prescripteur du besoin, le réalisateur et le vérificateur. En effet, les exigences sont écrites pour :

- Établir de façon non ambigüe et en termes techniques ce qui doit être réalisé ;
- Fournir la référence pour :
  - Réaliser la conception du produit ;
  - Mener les activités de validation et de vérification, dont l'établissement des tests.

Pour cela, les exigences doivent suivre des règles de rédaction afin de limiter les risques d'ambigüité et s'assurer qu'elles sont comprises par toutes les parties prenantes.

4.1.1. LES 11 CARACTÉRISTIQUES D'UNE BONNE EXIGENCE

Une exigence doit être…	Ce qui veut dire…	Justification
Identifiée	L'exigence doit disposer d'un identifiant unique.	Cette identification permet de mettre en place de la traçabilité entre les exigences, entre les exigences et les tests et également lors des analyses d'impact et évolutions.
Utile, Nécessaire	L'exigence doit refléter le besoin exprimé, et seulement le besoin exprimé (rejoint la traçabilité voir ci dessous).	Des fonctionnalités additionnelles et optionnelles ajoutent des risques d'incohérence ou de modes de défaillance supplémentaires. En sécurité, il faudra veiller en particulier à limiter les fonctionnalités superflues.
Concise, non ambigüe	L'exigence établit le besoin de façon simple, courte et claire. Elle doit être facilement lisible et compréhensible par l'ensemble des parties prenantes. On veillera à limiter les considérations implicites. Elle ne contient pas d'explication, de raisonnement, de justification. Ces ajouts peuvent être rajoutés en commentaire ou peuvent faire l'objet de notes annexes. Un même mot doit avoir une seule signification.	L'ambiguïté peut venir d'un vocabulaire métier spécifique, de mots vagues ou d'informations implicites. En cas d'une sémantique métier très présente, d'acronymes ou de termes techniques un glossaire et une liste des définitions devra être établie. L'ambiguïté des exigences pourra être améliorée par de la relecture par des acteurs différents.
Simple, Unique	L'exigence ne spécifie qu'un seul comportement. Ce comportement doit néanmoins être défini de façon complète.	L'unicité de l'exigence permet de faciliter la traçabilité entre un besoin (ou plusieurs besoins) et l'exigence ainsi qu'entre cette exigence et les exigences de niveau inférieur.
Indépendante de l'implémentation	L'exigence indique ce qui doit être fait, mais pas comment cela doit être fait. Elle ne doit pas décrire la façon de réaliser ou d'implémenter ce besoin sauf s'il existe une contrainte venant du client ou de l'environnement, dans ce cas, elle doit être classée dans les « contraintes ».	Une exigence spécifiant comment réaliser une fonction en particulier risque de créer des incohérences avec d'autres exigences fonctionnelles et de mettre à mal la faisabilité du produit.

4. BONNES PRATIQUES POUR RÉDIGER DES EXIGENCES

Une exigence doit être…	Ce qui veut dire…	Justification
Complète, auto-suffisante	L'exigence doit se suffire à elle-même et contenir l'ensemble des informations nécessaires à son implémentation et à sa vérification. Elle rappelle le contexte si nécessaire (pas d'implicite)	L'exigence indique le système concerné, les interfaces, les conditions d'exécution, le comportement attendu, les prérequis, etc.
Non redondante	Il n'y a pas recouvrement avec une autre exigence	Des exigences redondantes peuvent amener à des incohérences à terme en cas d'évolution. Si une redondance est détectée, il faut reformuler les 2 exigences concernées ou éliminer une des exigences.
Cohérente, compatible avec les autres exigences	Une exigence ne doit pas contredire une autre exigence. Toutes les exigences doivent être compatibles les unes avec les autres.	Il s'agit d'une caractéristique portant sur une spécification complète ou sur un ensemble d'exigences.
Vérifiable	L'exigence est suffisamment bien décrite pour identifier les critères de succès des vérifications et pour définir des moyens efficaces qui permettent de vérifier l'exigence. Les différents moyens existants sont : l'inspection, l'analyse, la démonstration et les tests (IADT).	L'aspect vérifiable des exigences doit être pensé dès l'écriture de l'exigence. Il est souhaitable d'impliquer, dans la relecture des exigences, des personnes en charge de la réalisation des tests.
Atteignable, faisable	Une exigence doit être réaliste. Il existe un moyen technique satisfaisant permettant de remplir cette exigence dans les limites du budget et délais impartis.	
Traçable	Toute exigence doit pouvoir être rattachée à un besoin ou une exigence de plus haut niveau afin d'être capable de tracer son origine.	Cette traçabilité implique aussi le respect de l'unicité et de l'identification unique d'une exigence.

4.1.2. LE MUST : MESURABLE/UNIQUE/SIMPLE/TRAÇABLE

Parmi les moyens mnémotechniques pour retenir les caractéristiques essentielles d'une exigence, MUST est un des plus simples :

- **Mesurable** : l'exigence contient des critères quantitatifs et/ou qualitatifs permettant d'apprécier le niveau à atteindre. Il existe une méthode de vérification du système vis-à-vis de l'exigence (inspection, analyse, démonstration, test etc.)
  - Exemple :
    - « Le système d'affichage des pistes radar doit avoir de bonnes performances » => non mesurable
    - « La piste radar doit apparaître en moins de 500 ms dans 95% du temps » => mesurable.
- **Unique** : Pas de redondance dans les exigences : il faut un identifiant unique, une seule information, strictement nécessaire et précise.
- **Simple** : Pas d'information inutile ou redondante, formulation claire et précise, compréhensible par toutes les parties prenantes, un seul comportement défini.
- **Traçable** : la traçabilité permet d'identifier l'origine de l'exigence, et de retrouver facilement sa justification. Elle permet également de tracer vers les tests.

4.1.3. SMART : SIMPLE/MESURABLE/ATTEIGNABLE/ RÉALISABLE/TRAÇABLE

SMART est un second moyen mnémotechnique pour estimer la qualité d'une exigence :

- **Spécifique/Simple** : Pas d'information inutile ou redondante, formulation claire et précise, compréhensible par toutes les parties prenantes, un seul comportement défini.
- **Mesurable :** l'exigence contient des critères quantitatifs et/ou qualitatif permettant d'apprécier le niveau à atteindre. Il existe une méthode de vérification du système vis-à-vis de l'exigence (inspection, analyse, démonstration, test etc.)
  - Exemple :
    - « Le système d'affichage des pistes radar doit avoir de bonnes performances » => non mesurable
    - « La piste radar doit apparaître en moins de 500 ms dans 95% du temps » => mesurable.
- **Atteignable** : l'exigence est réalisable techniquement ;
- **Réalisable :** l'exigence est réalisable dans les contraintes données (coût, ressources, temps, etc.)
- **Traçable** : la traçabilité permet d'identifier l'origine de l'exigence, et de retrouver facilement sa justification. Permet également de tracer vers les tests.

4. BONNES PRATIQUES POUR RÉDIGER DES EXIGENCES

4.2. EN PRATIQUE

4.2.1. DÉFAUTS CONSTATÉS ET COMMENT LES CORRIGER

- **Éviter les adjectifs vagues** :

« Tolérant aux fautes », « fidèle », « adaptable », « rapide », « lent », « ergonomique », « convivial », « suffisant », « sécurisé », « ad hoc », « robuste », « pertinent », « différent », « bon », « excellent », « efficace »

- **À proscrire** :

« etc… », « et/ou », « un/une/des » (à remplacer par « le/la/les »), « plusieurs ».

- **Utiliser des verbes d'action** :

« fournit », « affiche », « calcule », éviter les verbes plus vagues comme « gérer », « supporter », « maximiser », « minimiser », « optimiser », « améliorer », « accommoder »

- **Utiliser des verbes au présent** :

« doit » et non « devrait », « pourrait »

- **Eviter les termes ambigus** :

« état de l'art », « presque toujours », « à peu près », « proche de », « assez », « souvent », « facilement », « approximativement », « peu de », « beaucoup de », « assez de », « approprié », « efficace », « si possible », « quand nécessaire », « si nécessaire », « mais pas limité à », « autant que faire se peut »,

- **Préciser les termes trop généraux** :

« gérer », « le système », « l'équipement », « la fonction », « les entrées », « le but », etc…

- **Proscrire la forme négative dans les exigences** :

• Il est souvent tentant de formuler une exigence sous forme négative lorsqu'il a été identifié un comportement ou une propriété qui semble néfaste ou nuisible au système : « le système ne doit pas faire ça » ou « la performance XXX du système ne doit pas être inférieure/supérieure à telle valeur ». Cependant, cette formulation négative a plusieurs inconvénients :

- Les comportements qu'un système ne doit pas avoir sont en réalité probablement infinis. Même si certains ont des impacts sécurité ou opérationnels plus graves que d'autres, il n'est pas possible de caractériser un système par ce qu'il ne doit pas faire. Cela ferait appel à trop d'implicite. Caractériser complètement et précisément ce qu'il doit faire dans les cas nominaux et de robustesse permet d'anticiper tous les fonctionnements et dysfonctionnements y compris nuisibles.

- Une exigence négative ne peut pas être vérifiée de façon objective et complète. Autant, une approche par classe d'équivalence permet de vérifier le comportement spécifié dans une exigence positive autant la vérification qu'il n'existe pas une situation pour laquelle le système se comportera de telle façon nécessite de parcourir l'ensemble des situations opérationnelles potentielles. Ce qui n'est pas réalisable de façon générale.

• La formulation négative relève souvent du besoin et exprime donc plutôt une justification des exigences. Il faudra alors définir de façon explicite les exigences permettant de répondre à ce besoin. Par exemple :

- « L'implémentation de cette nouvelle fonctionnalité ne doit pas entrainer de régression » relève du besoin et se traduit par « Le développement de cette nouvelle fonctionnalité sera réalisé en accord avec les processus de développement XXX permettant d'assurer la non-régression du système » ;

- « La consommation CPU ne doit pas dépasser 80% » se traduira par « la consommation CPU du système doit rester 90% du temps en-dessous de 80%. » et « Lorsque la consommation CPU du système dépasse 80%, le système doit envoyer une alerte [ALERTE_CPU_80] à la supervision et doit arrêter les processus non prioritaires ».

4.2.2. EXEMPLES

■ Exigences système

À ne pas faire	Problème	Remplacer par (à titre d'exemple seulement, certaines choses ont été rajoutées pour compléter)
Pour minimiser la place disque nécessaire pour les archivages, on limitera dans la mesure du possible les composants présents sur le média socle système du serveur à ceux nécessaires pour un serveur sans interface graphique afin que l'image d'installation tienne sur un CD (utilisation d'image iso « minimal »).	Cette exigence relève plutôt d'une contrainte de conception. Termes trop vagues « minimiser », « dans la mesure du possible ». Dans la mesure où il faut que ça tienne sur un CD, on connaît la taille maximale. Le fait que le socle système retenu ne contienne pas l'interface graphique est soit une solution soit une autre contrainte. Quid si la taille du socle avec interface graphique est inférieure à la taille d'un CD ? « Pour minimiser la place disque nécessaire pour les archivages » : c'est la raison de l'exigence, ne pas le mettre dans l'exigence elle-même mais dans la justification ou dans le besoin générant l'exigence.	**Contrainte_1** : l'image d'installation du logiciel X doit tenir sur un unique CD de 700Mo. **Contrainte_2** : l'image d'installation du socle système du logiciel X doit contenir tous les éléments nécessaires à l'installation d'un serveur sans interface graphique. **Contrainte_3** : tous les composants non nécessaires au fonctionnement d'un serveur sans interface graphique pour le logiciel X doivent être supprimés de l'image d'installation.
Le choix du format de la source horaire locale doit être paramétrable.	Il s'agit à la fois d'une exigence fonctionnelle et d'une contrainte de conception qui requiert qu'un certain nombre de choses soit paramétrable. A préciser : au démarrage ? via un fichier de configuration ? à chaud ? Y a-t-il un format par défaut ? Il manque également la référence au dossier de définition d'interface. La possibilité de paramétrage doit être définie : quels formats disponibles, définition de l'interface.	**Exi_Fonc_1** : Le système doit permettre de paramétrer le format de la source horaire locale tel que défini dans le document d'interface réf XXX. **Contrainte_1** : Le choix du format de la source horaire fait partie des paramètres hors ligne définis dans l'ICD [1] réf YYY et doit être pris en compte au démarrage du logiciel Ou **Exi_IHM_1** : l'IHM du logiciel, telle que définie dans la spécification d'IHM réf ZZZ, doit permettre de modifier le format de la source horaire locale qui doit être pris en compte sans redémarrage en moins de 30 sec.

[1] ICD : Interface Control Document

4. BONNES PRATIQUES POUR RÉDIGER DES EXIGENCES

À ne pas faire	Problème	Remplacer par (à titre d'exemple seulement, certaines choses ont été rajoutées pour compléter)
Pour pouvoir investiguer sur des problèmes éventuels, on devra pouvoir activer par paramétrage un enregistrement des trames reçues.	Ceci n'est pas une exigence et contient 2 considérations : La possibilité d'enregistrer des trames et le fait que ce soit un paramétrage. L'objectif de l'investigation est le besoin et non l'exigence à proprement parler.	**Exi_fonct_Enreg_1** : Le système X doit permettre d'activer et désactiver l'enregistrement de toutes les trames IP selon le format défini dans le document réf YYY. **Exi_perf_Enreg_1** : L'augmentation de la charge processeur due à l'activation de l'enregistrement des trames reçues doit être inférieure à 10%. **Exi_perf_Enreg_2** : L'augmentation de l'occupation mémoire due à l'activation de l'enregistrement des trames reçues doit être inférieure à 10%. **Contrainte_Enreg_1** : Le système X doit disposer d'un fichier de paramètres hors ligne tel que défini dans l'ICD réf ZZZ permettant l'activation et la désactivation de l'enregistrement des trames et qui doit être pris en compte au démarrage du logiciel.
Le serveur doit disposer des outils/méthodes nécessaires pour établir un suivi de performance de synchronisation horaire (offset courant, distance de synchronisation, …)	L'ensemble des performances devant faire l'objet d'un suivi doivent être listées précisément, il ne faut pas utiliser de points de suspension dans l'exigence. Il faut choisir entre mettre une contrainte de conception sur le fait que le système contient des outils spécifiques (il faudra les lister et relier ces contraintes à des besoins particuliers) ou mettre une exigence fonctionnelle qui porte sur des capacités du système. Il est préférable d'avoir recours à la seconde solution.	**Exi_fonct_Suivi_Perf_1** : le logiciel X doit permettre de présenter l'offset courant à la milliseconde près. **Exi_IHM_Suivi_Perf_1** : l'IHM du logiciel X, telle que définie dans la spécification d'IHM réf ZZZ, doit permettre de visualiser l'offset courant à la milliseconde près sur un historique de 30 min par pas de 10 sec. **Exi_fonct_Suivi_Perf_2 :** le logiciel X doit permettre de présenter la distance de synchronisation courante à la minute près. **Exi_IHM_Suivi_Perf_2** : l'IHM du logiciel X, telle que définie dans la spécification d'IHM réf ZZZ, doit permettre de visualiser la distance de synchronisation courante à la minute près avec des délais de rafraîchissement inférieurs à la minute.

À ne pas faire	Problème	Remplacer par (à titre d'exemple seulement, certaines choses ont été rajoutées pour compléter)
Le partitionnement effectué par l'installation du serveur doit : • laisser de la place disponible sur le disque ; • séparer (au minimum) la partition /var de la partition système (la saturation des logs ne doit pas empêcher le fonctionnement du serveur).	Termes trop vagues « place disponible », « au minimum », emploi de la forme négative. Le fait de « laisser de la place disponible sur le disque » n'est pas compréhensible. Place disponible pour qui ? pour quoi ? autre partition ? Il s'agit ici à la fois d'une exigence de conception (séparation /var et système) et d'une expression de besoin (laisser de la place disponible et empêcher saturation).	**Exi_Install_1** : l'installation du système doit garantir que la taille des données liées à l'enregistrement des logs est limitée à x Go. Remarque sur cette exigence : celle-ci pourrait s'exprimer sous la forme d'une durée d'enregistrement de logs. Cette exigence peut aussi s'exprimer sous la forme d'une contrainte : **Contrainte_Install_1** : la taille de la partition /var devra être au minimum de x Go et la taille de la partition système au moins de 3 fois la taille du système installé. **Exi_Fonc_Logs_1** : si la taille des données d'enregistrement de logs dépasse 80% des x Go alloués aux logs, le système doit envoyer une alerte à la supervision selon l'ICD XXX et purger les données les plus anciennes.
Le composant permet de configurer un délai par rapport à l'heure reçue de la source horaire.	Quel délai ? quelle source horaire ? Exigence non testable car on ne sait pas ce qu'on doit observer.	**Exi_Fonct_Offset_1** : Le logiciel X doit permettre de diffuser l'heure reçue par la source horaire avec un [Offset_Source_horaire] positif ou négatif tel que défini dans l'ICD paramétrage XXX. **Contrainte_Offset_1** : Le système X doit disposer d'un fichier de paramètres hors ligne tel que défini dans l'ICD réf YYY permettant le paramétrage de l'[Offset_Source_Horaire] et qui doit être pris en compte au démarrage du logiciel. **Dans l'ICD XXX** : [Offset_Source_horaire] : entier, unité : seconde, range [-60 ;60], valeur par défaut : 0

4. BONNES PRATIQUES POUR RÉDIGER DES EXIGENCES

À ne pas faire	Problème	Remplacer par (à titre d'exemple seulement, certaines choses ont été rajoutées pour compléter)
La trame DATEL est convertie en une structure d'heure utilisée pour la diffusion NTP [2]	Exigence non codable, non testable. Que doit-on observer ?	**Exi_Fonct_DATEL_1**: le serveur X doit diffuser l'[Heure_NTP] reçue par les [Trames_DATEL] aux composants NTP de strate inférieure, avec une période inférieure à 10 minutes. La diffusion de l'heure doit se faire selon le format défini dans l'ICD réf XXX et moins de 500ms après réception de la trame DATEL effectivement diffusée. **Interface_Struct_NTP_1**: l'[Heure_NTP] doit respecter le format XXXXXXX défini dans le RFCYYY.
L'état de la chaine horaire (GO/NOGO, mot d'état) et la cohérence de l'heure reçue sont vérifiés.	Exigence incomplète: l'état est vérifié, et alors ? Que doit-on observer ?	L'exigence devrait plutôt prendre la forme suivante: **Exi_Fonct_Etat_1**: À la réception d'une trame d'heure DATEL, si le champ « ETAT » de la chaine horaire (spécifié dans l'ICD XXX) est différent des états prévus, le système X doit envoyer une alarme à la supervision et rejeter la trame. **Exi_Fonct_Cohérence_1**: À la réception d'une trame d'heure DATEL, si l'heure reçue n'est pas cohérente avec l'heure courante, le système X doit envoyer une alarme à la supervision et rejeter la trame. La vérification de cohérence se fera en appliquant l'algorithme suivant: XXX.

[2] NTP: Network Time Protocol

Exigences de sécurité

À ne pas faire	Problème	Remplacer par
(Non régression) L'ajout du paramétrage de basculement Normal/secours par le composant X pour le site 1 ne doit pas perturber le fonctionnement nominal du composant sur les sites autres que le site 1.	Forme négative Non testable (préciser ce que signifie « ne doit pas perturber » et « fonctionnement nominal » ?) La non-régression DOIT être garantie par l'application des processus d'ingénierie et des processus de sécurité (QMS et SMS) et ne peut être envisagée sous la forme d'une exigence, même de sécurité.	La non-régression relève de l'application des processus d'ingénierie internes et ne devrait pas faire l'objet d'une exigence. Si jamais cela devait être spécifié (pour un contrat par exemple), cela devrait prendre la forme suivante : **Exi_Secu_Processus** : le développement du système X doit suivre le processus de développement de l'entreprise XXX afin d'assurer la non-régression du système après modification.
(Non régression) Les évolutions de la version V1.2 du composant n'entrainent pas d'augmentation significative de la charge CPU du composant.	Il ne s'agit pas de non-régression à proprement parler mais uniquement d'une exigence de performance. La forme négative n'est pas recommandée. Il est préférable de spécifier des exigences de façon absolue dans la mesure où une augmentation même faible d'une charge déjà saturée n'est pas acceptable. De plus, l'exigence n'est pas testable : Il faut préciser ce que signifie une « augmentation significative » ?	**Exi_Perf_CPU_1**: La charge CPU du logiciel X doit rester inférieure à 50% de charge CPU en moyenne sur les scénarios de charge nominaux tels que définis dans le document réf XXX et à 90% de charge sur des périodes de 60 secondes lors de scénarios chargés tels que définis dans le document Ref XXX.
Valider la cohérence des cartes affichées lors d'un basculement Normal/secours durant la phase d'évaluation.	Trop vague. Activité humaine qui nécessite d'expliciter l'activité.	**Exi_Proc_Secu_Evalop_1**: Durant la phase d'évaluation, lors d'un basculement Normal/secours, le contrôleur doit vérifier la cohérence des cartes affichées en appliquant la procédure XXX. Ou **Exi_Proc_Secu_Evalop_1bis**: Durant la phase d'évaluation, lors d'un basculement Normal/secours, le contrôleur doit vérifier que les cartes affichées sont identiques en tout point (alternative: sur les points suivants à expliciter).

4. BONNES PRATIQUES POUR RÉDIGER DES EXIGENCES

À ne pas faire	Problème	Remplacer par
Vérifier que la procédure de basculement est prise en compte dans le Manex.	Il ne s'agit pas d'une exigence à proprement parler et cela ne spécifie pas un comportement du système ou de ses composants. Il s'agit d'une activité de vérification.	**Exi_Proc_Secu_Basculement_Operateur :** En cas d'alerte des dysfonctionnements suivants du système principal (lister TOUS les cas de dysfonctionnements concernés), l'opérateur doit réaliser une action de basculement sur le système secours en moins de 2 min. **Exi_Proc_Secu_Basculement_Manex :** Le Manex des opérateurs doit contenir la procédure de basculement du système principal vers le système secours **Exi_Proc_Secu_Basculement_Formation :** La formation des opérateurs doit permettre aux opérateurs de réaliser une opération de basculement du système principal vers le système secours en moins de 2 min dans 95% des cas. Toutes ces exigences plutôt centrées sur l'opérationnel doivent trouver leur contrepartie sur le système technique : **Exi_Secu_Basculement_Système** : Lorsque l'opérateur lance une commande de basculement et que le système en opération est le système principal, le système opérationnel doit devenir le système secours en moins d'1 min. **Exi_Secu_Alarme_Déf_Syst_Princ** : Si le système principal a l'un des défauts suivants (mettre la liste exhaustive des défauts visés), le système de supervision doit émettre une alarme à l'attention de l'opérateur et afficher une alarme sur l'IHM de supervision en moins de 30 sec.

- **Exemple d'exigence de sécurité :**

Exi_Objectif_Sécurité : Le taux d'occurrence des défaillances du système conduisant à des pertes de séparation supérieures à 50% doit être inférieur à 10-6 pannes/heure opérationnelle

- **Exemples d'exigences fonctionnelles et non-fonctionnelles taguées [sécurité] :**

Exi_supervision : [Sécurité] En cas de panne matérielle du système, la supervision doit remonter une alarme (telle que définie dans la spécification d'IHM_Superv réf XXX) aux opérateurs de supervision en moins de 15sec.

Exi_Performance_Affichage : [Sécurité] tous les objets présents à l'écran doivent être affichés avec une précision relative de 1% et une précision absolue toujours inférieure à 0.5Nm.

Conception : STAC/Division documentation et diffusion des connaissances

Couverture : © Marie-Ange FROISSART DGAC /STAC
© Richard METZGER DGAC /STAC

Crédit photos : © Fotolia page 7
© Marie-Ange FROISSART DGAC /STAC page 9
© Richard METZGER DGAC /STAC pages 20, 24

Février 2021

Direction générale de l'Aviation civile
service technique de l'Aviation civile
CS 30012 - 31 avenue du Maréchal Leclerc
94 385 Bonneuil-sur-Marne cedex FRANCE
Téléphone : 01 49 56 80 00
www.stac.aviation-civile.gouv.fr
www.ecologie.gouv.fr