



Université du Québec
École de technologie supérieure

LIVEUV

Présenté à

DAVID GUILMAINE

LOG410 : Analyse de Besoins et Spécification Logiciel

Software Requirements Specification

David Lauzon
LAUD01028300

Anton Zakharov
ZAKA12038406

8 décembre 2010

Table des matières

1	Introduction	6
1.1	Objectif	6
1.2	Portée	6
1.3	Références	7
1.4	Hypothèses et Dépendances	7
2	Survol du Modèle des Cas d'Utilisation	8
2.1	Diagramme des cas d'utilisation	8
2.2	CU01 – Ajouter un widget	8
2.3	CU02 – Connecter un widget	8
2.4	CU03 - Supprimer un widget	9
2.5	CU05 - Détacher un widget	9
2.6	CU06 - Sauvegarder une perspective	9
2.7	CU07 - Supprimer une perspective	9
2.8	CU08 – Charger une perspective	9
2.9	CU09 – Déplacer un Widget	9
2.10	CU10 - Créer une alerte	9
2.11	CU11 - Supprimer une alerte	9
2.12	CU12 - Attacher un widget	10
3	Les Acteurs	10
3.1	Utilisateur	10
3.2	libJAUS	10
4	Les Exigences	10
4.1	Les exigences fonctionnelles	10
	REQ001 <i>Barre d'outils de widgets</i>	10
	REQ002 <i>Conteneur de widgets</i>	10
	REQ003 <i>États de widget</i>	10
	REQ004 <i>Widgets spécifiques</i>	10
	REQ005 <i>Widgets génériques</i>	10
	REQ006 <i>Connexion automatique à JAUS</i>	11
	REQ007 <i>Arbre JAUS</i>	11
	REQ008 <i>Importation d'un widget</i>	11
	REQ009 <i>Sections de modification des widgets</i>	11
	REQ010 <i>Importer une perspective</i>	11
	REQ011 <i>Console d'erreurs</i>	11
	REQ012 <i>Theme couleur</i> (PROJETÉ DANS LE FUTUR)	11
4.2	Les Exigences Non Fonctionnelles	11
4.2.1	Functionality > Accuracy	11
	NFR003 L'utilisateur change la valeur connectée d'un widget	11
	NFR004 L'utilisateur change la valeur responsable du bouton d'arrêt du sous-marin	11
4.2.2	Reliability > Maturity	11

NFR008	L'utilisateur utilise l'application et l'application <i>crash</i>	11
NFR009	Le programmeur veut tester une fonctionnalité critique (faire une liste avec le client)	12
NFR015	Le programmeur veut tester les cas d'utilisations suivants : (faire une liste avec le client)	12
4.2.3	Reliability > Fault tolerance	12
NFR016	L'utilisateur utilise le système. Un widget crash (arrête de fonctionner dû à une erreur dans le code ou autre)	12
NFR017	L'utilisateur utilise le système normalement et le système <i>crash</i>	12
4.2.4	Reliability > Recoverability	12
NFR002	L'utilisateur redémarre l'application après un <i>crash</i> et charge sa perspective	12
4.2.5	Usability > Learnability	12
NFR001	L'utilisateur reproduit, à l'aide du système, une perspective représentée dans une image	12
4.2.6	Usability > Operability	12
NFR011	L'utilisateur veut ajouter un nouveau widget	12
NFR007	L'utilisateur veut lier la valeur d'un widget à libJAUS	12
NFR012	L'utilisateur veut utiliser la perspective sauvegardée par un autre usagé sur une autre machine. Le fichier de sauvegarde est dans l'ordinateur de l'utilisateur.	13
NFR010	L'utilisateur L'utilisateur change la donnée d'une valeur liée à JAUS du widget .	13
NFR018	L'utilisateur veut installer LiveUV	13
4.2.7	Efficiency > Resource behavior	13
NFR014	L'utilisateur veut installer LiveUV	13
4.2.8	Maintainability > Changeability	13
NFR005	Le programmeur code un nouveau widget avec 2 données à liables et 2 données modifiables non-liables	13
NFR006	Le programmeur code une ligne texte affichable et non-modifiable par l'utilisateur d'un widget	13
NFR013	Le programmeur remplace libJAUS par libNouveau avec la même interface publique	13
NFR019	Le programmeur veut importer un nouveau widget dans le système . . .	13
5	Documentation en direct pour l'utilisateur et exigences du système d'aide	14
6	Contraintes de conception	14
CR001	<i>Support multi-plateforme</i>	14
CR002	<i>Conception modulaire</i>	14
CR003	<i>Compatibilité avec plusieurs véhicules autonomes</i>	14
CR004	<i>Utilisation de la librairie de communication libJAUS</i>	14
CR005	<i>Langage de programmation JAVA</i>	14
7	Composants achetés	14

8 Interfaces	15
8.1 Interfaces Utilisateur	15
8.2 Interfaces Matérielles	15
8.3 Interfaces Logicielles	15
8.4 Interfaces de Communications	16
9 Exigences de Licenses	16
10 Remarques légales, de droits d'auteur, et diverses	16
11 Normes et Standards Applicables	16
11.1 Glossaire	16
11.2 Quint-2	17
12 Bibliographie	17
A Spécifications des Cas d'Utilisation	18
B Matrice de Tracabilité	18
C États du système et des widgets	18
D Légendes des diagrammes UML	20
E Diagrammes UML	22

Liste des tableaux

1 Historique des Révisions	5
2 Glossaire	16
3 Matrice de Tracabilité	18
4 Environments (mode d'opération) du système	18
5 États de connexion de l'application	19
6 États de connexion de l'application	19
7 États d'un widget	19
8 États d'une valeur d'un widget	19
9 États de la position du Widget	19

Table des figures

1 Diagramme de la portée du projet	6
2 Diagramme des cas d'utilisation	8
3 Fenêtre Principale de LiveUV	15
4 Légende des diagrammes de cas d'utilisation (référence [3])	20
5 Légende des diagrammes de séquence (référence [3])	20
6 Légende des diagrammes d'activité (référence [3])	21

7	Légende des modèles du domaine (référence [3])	21
8	Légende des diagrammes d'activité	22
9	Diagramme d'Activité AD01 - Référence : CU09 Déplacer un widget	23
10	Diagramme d'État STATED01 - Référence : REQ003 États de widget	24

Historiques des Révisions

TABLE 1: Historique des Révisions

Date	Version	Description	Auteur
1/dec/2010	v0.9	Polissage du rapport. Correction des erreurs de tout genre.	Anton Zakharov,David Lauzon
30/nov/2010	v0.8	Corrections suite au rapport de prototype	Anton Zakharov
29/nov/2010	v0.7.1	Ajout des sections 1, 5, 7 et annexes	David
21/nov/2010	v0.7	Integration des modification suite a la revue du 2010_11_17. Intégration des requis additionels suite à la discussion avec le client: voir Req a implementer.txt	Anton Zakharov
15/nov/2010	v0.6	Revisions des uses case jusqu a CU07	David Lauzon
14/nov/2010	v0.5	section 8	Anton Zakharov
13/nov/2010	v0.4	section 6	David Lauzon
12/nov/2010	v0.3	section 4.1	Anton Zakharov
11/nov/2010	v0.2	sections 2 et 3	David Lauzon
18/oct/2010	v0.1	Gabarit initial	David

1 Introduction

1.1 Objectif

Le but de ce document est de rassembler, d'analyser, et de définir les exigences du LiveUV. Le LiveUV sera la prochaine interface graphique de l'outil télémétrie du club étudiant SONIA, un club étudiant de l'ÉTS (École de technologie supérieure) oeuvrant dans la conception d'un sous-marin intelligent et autonome. Ce document détermine les exigences fonctionnelles et non fonctionnelles nécessaires pour réaliser le produit spécifié par le client dans le document de Vision.

1.2 Portée

Le système de la télémétrie est un système qui est déjà en place et qui doit être refait. Sa position, dans l'ensemble de l'architecture système de SONIA demeure la même, c'est-à-dire branché directement avec le AUV Decision Center (actuellement AUV5). Cependant, comprenons que dans le mandat du présent projet, une couche intermédiaire serait fournie, qui s'occuperait de la communication à travers le protocole JAUS avec AUV5, out toute autre module au besoin. Ce module s'appelle libJAUS.

Dans le schéma ci-dessous (¹), on peut voir que la portée du projet se limite au module LiveUV.

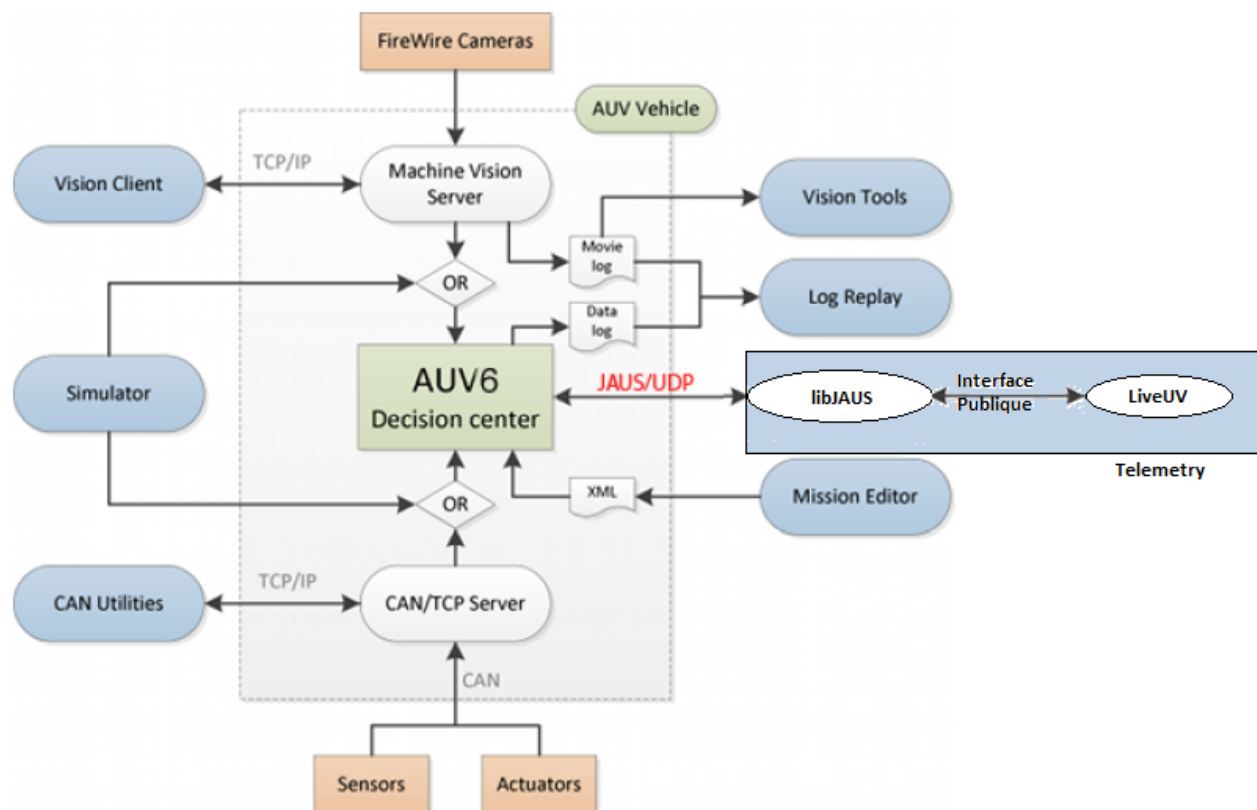


FIGURE 1 – Diagramme de la portée du projet

1. Référence du Schéma : Vision Reference A10.pdf, Section Portée

Dans cette optique, LiveUV pourra être compatible avec tout autre véhicule autonome ayant un module libJAUS, respectant les interfaces publiques définies dans le libJAUS du club SONIA. L'aéronef de Dronolab et la tondeuse de Capra, deux autres clubs étudiants de l'ÉTS, pourront créer leurs propres libJAUS et utiliser LiveUV directement.

1.3 Références

Le format du document a été inspiré du document : Polymtl - LOG3900 - Example de SRS.pdf

Le document de vision : Vision Reference A10.pdf

Le document des cas d'utilisation : Use_Cases_EQ12.pdf

Le rapport de prototype : Rapport Prototype EQ12.docx

1.4 Hypothèses et Dépendances

Il est assumé dans ce document que libJAUS est opérationnel. De plus libJAUS doit avoir une interface publique pour communiquer avec le système.

2 Survol du Modèle des Cas d'Utilisation

2.1 Diagramme des cas d'utilisation

Le diagramme des cas d'utilisation décrit la relation entre les cas d'utilisation, les acteurs et la portée du système.

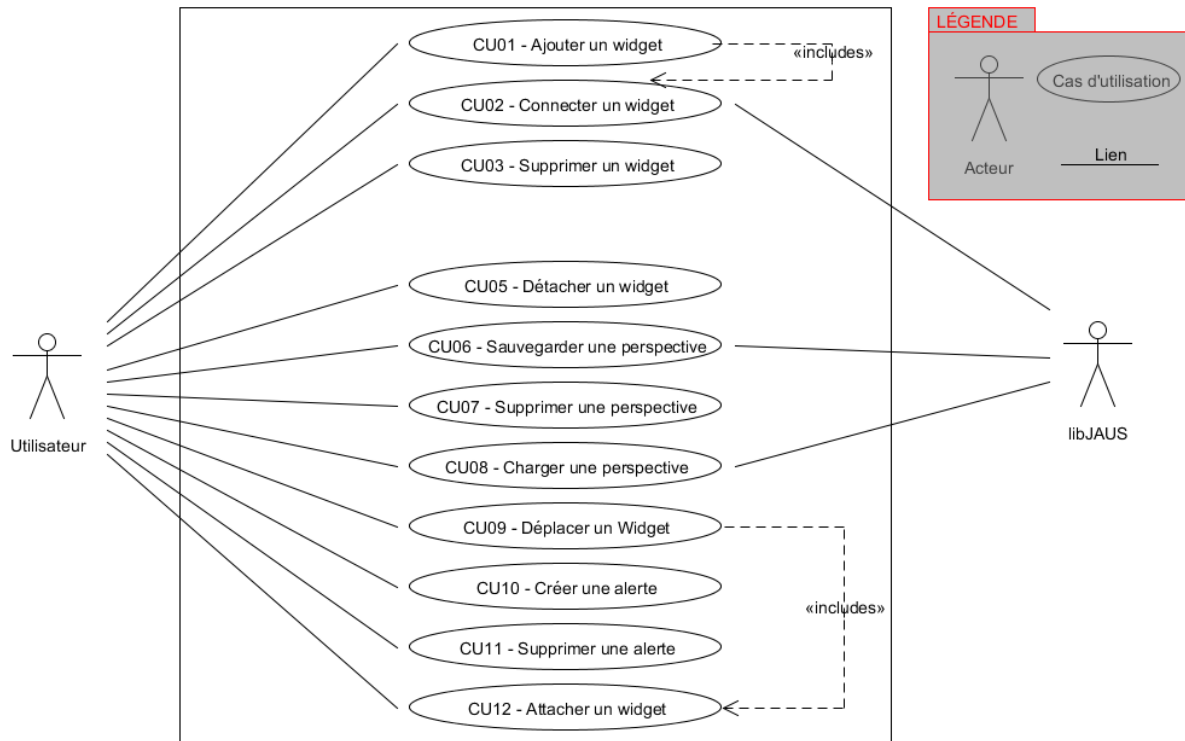


FIGURE 2 – Diagramme des cas d'utilisation

Cette section présente une description pour chaque cas d'utilisation et leurs acteurs. Les cas complets se trouvent dans l'annexe A.

Certains cas d'utilisation font référence à des états du système et des widgets définis à l'annexe C.

2.2 CU01 – Ajouter un widget

L'utilisateur se sert du système pour ajouter un widget sur la perspective. Les widgets disponibles se trouvent dans une barre d'outils.

Acteur : Utilisateur

2.3 CU02 – Connecter un widget

L'utilisateur utilise les outils de notre système pour connecter un widget à des données disponibles dans JAUS.

Acteur : Utilisateur, libJAUS

2.4 CU03 - Supprimer un widget

L'utilisateur supprime le widget de la perspective en utilisant LiveUI.

Acteur : Utilisateur

2.5 CU05 - Détacher un widget

L'utilisateur utilise le système pour détacher le widget du conteneur principal, et le mettre dans une petite fenêtre séparée.

Acteur : Utilisateur

2.6 CU06 - Sauvegarder une perspective

L'utilisateur utilise LiveUI pour sauvegarder les informations de connections et la position de tous les widgets de la perspective dans un fichier.

Acteur : Utilisateur, libJAUS

2.7 CU07 - Supprimer une perspective

L'utilisateur utilise l'outil de recherche de fichiers du système pour trouver une sauvegarde de perspective et la supprimer.

Acteur : Utilisateur

2.8 CU08 – Charger une perspective

L'utilisateur utilise l'outil de recherche de LiveUI pour trouver un fichier de perspective et le charger. La perspective restitue tous les widgets ainsi que leurs états de connexion (si possible) lors de la création de la sauvegarde.

Acteur : Utilisateur, libJAUS

2.9 CU09 – Déplacer un Widget

L'utilisateur utilise le système pour changer la position du widget sur l'écran.

Acteur : Utilisateur

2.10 CU10 - Créer une alerte

L'utilisateur utilise les outils du système pour créer une alerte et la connecter à une valeur/des valeurs critiques d'un widget.

Acteur : Utilisateur

2.11 CU11 - Supprimer une alerte

L'utilisateur utilise LiveUI pour rechercher une alerte parmi les alertes existantes et la supprimer.

Acteur : Utilisateur

2.12 CU12 - Attacher un widget

L'utilisateur utilise le système pour ré-attacher au conteneur principal, un widget ayant été détaché. Voir cas d'utilisation dans la sous-section : 2.5.

Acteur : Utilisateur

3 Les Acteurs

3.1 Utilisateur

Personne utilisant le système pour accomplir des actions ou obtenir des données du sous-marin

3.2 libJAUS

Système gérant la communication des données depuis le sous-marin vers la télémétrie (le système qui fait objet de ce SRS).

4 Les Exigences

Certaines exigences font référence à des états du système et des widgets définis à l'annexe C.

4.1 Les exigences fonctionnelles

Voici une liste des exigences, en termes de fonctionnalités, du logiciel. Ceux-ci décrivent des fonctionnalités non-incluses dans les cas d'utilisation.

REQ001 *Barre d'outils de widgets*

Le système doit avoir une boîte contenant des widgets représentant des contrôles. Voir cas d'utilisation dans la sous-section 2.2. La boîte est séparée du conteneur de widgets; elle peut être ouverte (visible) et fermée (cachée) par l'utilisateur.

REQ002 *Conteneur de widgets*

Le système doit avoir une boîte contenant les widgets et permettant de gérer la taille / position / attachement / détachement de ceux-ci. Lorsque le widget est détaché, il n'est plus dans le conteneur et peut être déplacé librement comme une autre fenêtre graphique. Il doit y avoir un seul conteneur.

REQ003 *États de widget*

Le système doit avoir des widgets qui doivent supporter les états suivants : voir l'annexe C (États du système et des widgets) pour la description de ces états et le diagramme d'état STATED01 (Figure 10).

REQ004 *Widgets spécifiques*

Le système doit avoir des widgets spécifiques à un véhicule autonome. Par exemple : un widget qui montre la position du sous-marin, un widget qui montre la profondeur du sous marin et un widget qui montre l'état de la batterie.

REQ005 *Widgets génériques*

Le système doit avoir des widgets ré-utilisables d'un club étudiant à l'autre. Par exemple : un widget représentant un tableau.

REQ006 *Connection automatique à JAUS*

Le système doit se connecter automatiquement au véhicule autonome via libJAUS une fois que l'application LiveUI est démarrée. Le système utilise les données dans le fichier de configuration.

REQ007 *Arbre JAUS*

Le système doit avoir une fenêtre (nommé arbre JAUS) présentant la structure de toutes les composantes disponibles via libJAUS dont il est possible de voir/modifier une valeur. Les données seront structurées sous la forme d'un arbre.

REQ008 *Importation d'un widget*

Le système doit permettre l'importation de nouveaux widgets déjà fait en moins de 1 heure par un programmeur expert.

REQ009 *Sections de modification des widgets*

Le système doit avoir des widgets qui possèdent au moins une des valeurs suivantes : 1) Modifiable non liable 2) Liable

REQ010 *Importer une perspective*

Le système doit permettre l'importation d'un fichier de perspective sauvegardée par une autre instance du système sur une autre machine.

REQ011 *Console d'erreurs*

Le système doit afficher les erreurs d'exécution dans une console.

REQ012 *Theme couleur* (PROJETÉ DANS LE FUTUR)

Le système offre des thème couleurs de l'interface.

4.2 Les Exigences Non Fonctionnelles

Les exigences qui ne sont pas liées aux fonctionnalités du logiciel, mais plus à sa qualité. Ces exigences sont regroupées en fonction du modèle de qualité Quint-2 (expliquée à la section 11.2).

4.2.1 Functionality > Accuracy

NFR003 *L'utilisateur change la valeur connectée d'un widget* (1.2.1. Failure ratio)

Le système est en opération journalière. Le système affiche la valeur retournée par le libJAUS. La valeur converge vers le changement de l'utilisateur 98% du temps.

NFR004 *L'utilisateur change la valeur responsable du bouton d'arrêt du sous-marin* (1.2.1. Failure ratio)

Le système est en opération journalière. Le système affiche la valeur du bouton on retournée par le libJAUS. Cette valeur est la même que rentrée par l'utilisateur 99.99% du temps.

4.2.2 Reliability > Maturity

NFR008 *L'utilisateur utilise l'application et l'application *crash** (2.1.1. Mean time between failures (MTBF))

Le système est en opération journalière. L'utilisateur redémarre l'application et l'application fonctionne normalement. Il y a au plus 1 crash en 2 heures.

NFR009 Le programmeur veut tester une fonctionnalité critique (faire une liste avec le client) (2.1.6. Test density)

Le système est en maintenance. Un test unitaire est disponible pour la fonctionnalité 100% du temps.

NFR015 Le programmeur veut tester les cas d'utilisations suivants: (faire une liste avec le client) (2.1.6. Test density)

Le système est en maintenance. Les cas d'utilisations sont testés automatiquement par un programme et les résultats sont envoyées au programmeurs. 100% des cas sont testés.

4.2.3 Reliability > Fault tolerance

NFR016 L'utilisateur utilise le système. Un widget crash (arrête de fonctionner dû à une erreur dans le code ou autre) (2.2.2. Vulnerability)

Le système est en opération journalière. La défaillance du widget n'affecte pas la stabilité du système 99.99% du temps.

NFR017 L'utilisateur utilise le système normalement et le système *crash* (2.2.2. Vulnerability)

Le système est en opération journalière. Les systèmes externes à l'application (ex: système autonome, internet, os) fonctionnent normalement même si l'application crash. Les systèmes externes doivent fonctionner 99.99% du temps.

4.2.4 Reliability > Recoverability

NFR002 L'utilisateur redémarre l'application après un *crash* et charge sa perspective (2.3.3. Mean recovery time)

Le système est en opération journalière. Le système affiche la perspective avec les valeurs réelles du libJAUS en moins de 1 minute après que l'utilisateur clique pour redémarrer le système.

4.2.5 Usability > Learnability

NFR001 L'utilisateur reproduit, à l'aide du système, une perspective représentée dans une image (3.2.1. Average learning time)

Le système est en opération journalière. Le système permet à l'utilisateur de créer une nouvelle perspective avec les valeurs présentes/changées des widgets sur l'image de référence en moins de 20 min (moins de 10 minutes pour un utilisateur expérimenté en télémétrie).

4.2.6 Usability > Operability

NFR011 L'utilisateur veut ajouter un nouveau widget (3.3.18. Number of keystrokes)

Le système est en opération journalière. Le widget est ajouté à la perspective principale en moins de 6 clics de souris.

NFR007 L'utilisateur veut lier la valeur d'un widget à libJAUS (3.3.21. Ultimate operation time)

Le système est en opération journalière. Sa prends moins de 2 mins pour un novice, et moins de 1 min pour un expert pour connecter une valeur.

NFR012 L'utilisateur veut utiliser la perspective sauvegardée par un autre usagé sur une autre machine. Le fichier de sauvegarde est dans l'ordinateur de l'utilisateur. (3.3.21. Ultimate operation time)

Le système est en opération journalière. La perspective peut être chargée dans l'application normalement en moins de 5 minutes.

NFR010 L'utilisateur L'utilisateur change la donnée d'une valeur liée à JAUS du widget (3.3.26. Response time for user)

Le système est en opération journalière. Le système communique la nouvelle donnée que libJAUS doit mettre à jour, et le système affiche une confirmation (ou un message d'erreur) que le changement a été fait sur le véhicule autonome en moins de 1 seconde.

NFR018 L'utilisateur veut installer LiveUV (3.3.4. Set-up installation time)

Le système est en installation. Le système s'installe correctement en moins de 3 min par un usager expert en ordinateurs.

4.2.7 Efficiency > Resource behavior

NFR014 L'utilisateur veut installer LiveUV (4.2.2. Internal memory occupancy, 4.2.4. Processor occupancy)

Le système est en installation. Le système s'installe correctement sur une machine qui est au minimum un dual core avec 1GB de ram.

4.2.8 Maintainability > Changeability

NFR005 Le programmeur code un nouveau widget avec 2 données à liables et 2 données modifiables non-liables (5.2.1. Modification effort per unit volume)

Le système est en maintenance. Le nouveau widget peut prendre tous les états de widgets et réagit correctement. La fonctionnalité est codée en moins de 4 heures.

NFR006 Le programmeur code une ligne texte affichable et non-modifiable par l'utilisateur d'un widget (5.2.1. Modification effort per unit volume)

Le système est en maintenance. Le widget affiche dans l'application la nouvelle valeur. La fonctionnalité est codée en moins de 1 heure.

NFR013 Le programmeur remplace libJAUS par libNouveau avec la même interface publique (5.2.1. Modification effort per unit volume)

Le système est en maintenance. Le système compile. L'utilisateur peut charger une perspective et tous les widgets peuvent lier leurs valeurs à libNouveau. Cette compatibilité est codée en moins de 1 mois par 1 personne.

NFR019 Le programmeur veut importer un nouveau widget dans le système (5.2.1. Modification effort per unit volume)

Le système est en maintenance. Le programmeur code le script pour l'intégration du nouveau widget. Ce widget est totalement opérationnel pour l'utilisateur. L'importation est effectuée en moins de 2 heures.

5 Documentation en direct pour l'utilisateur et exigences du système d'aide

Pas de documentation dans le logiciel en tant que tel. mais un wiki sera disponible pour supporter l'utilisation du logiciel (plus de détails dans le document de vision).

6 Contraintes de conception

Voici la liste de contraintes que le système doit respecter :

CR001 *Support multi-plateforme*

Le système doit supporter les systèmes d'exploitation suivants : MAC OS X, Linux et Microsoft Windows.

CR002 *Conception modulaire*

Le système doit être conçu en utilisant la programmation orientée objet, de façon à ce que les responsabilités soit bien réparties en classes et / ou modules, dans le but que le système soit facilement maintenable.

CR003 *Compatibilité avec plusieurs véhicules autonomes*

Le système doit être compatible avec d'autres véhicules autonomes implémentant la librairie libJAUS (par exemple des autres clubs étudiants de l'ÉTS). S'ils ont besoin de widgets spécifiques, il suffira à leur équipe de programmation de les intégrer dans le système.

CR004 *Utilisation de la librairie de communication libJAUS*

Le système doit obligatoirement communiquer au véhicule autonome avec les interfaces publiques de libJAUS (il ne doit pas communiquer en utilisant directement le protocole JAUS).

CR005 *Langage de programmation JAVA*

Le système doit être écrit dans le langage de programmation JAVA, en utilisant la convention de nommage des noms de Java par Sun (Sun Java Naming Convention).

7 Composants achetés

Aucune composante supplémentaire ne devrait être ajoutée.

8 Interfaces

8.1 Interfaces Utilisateur

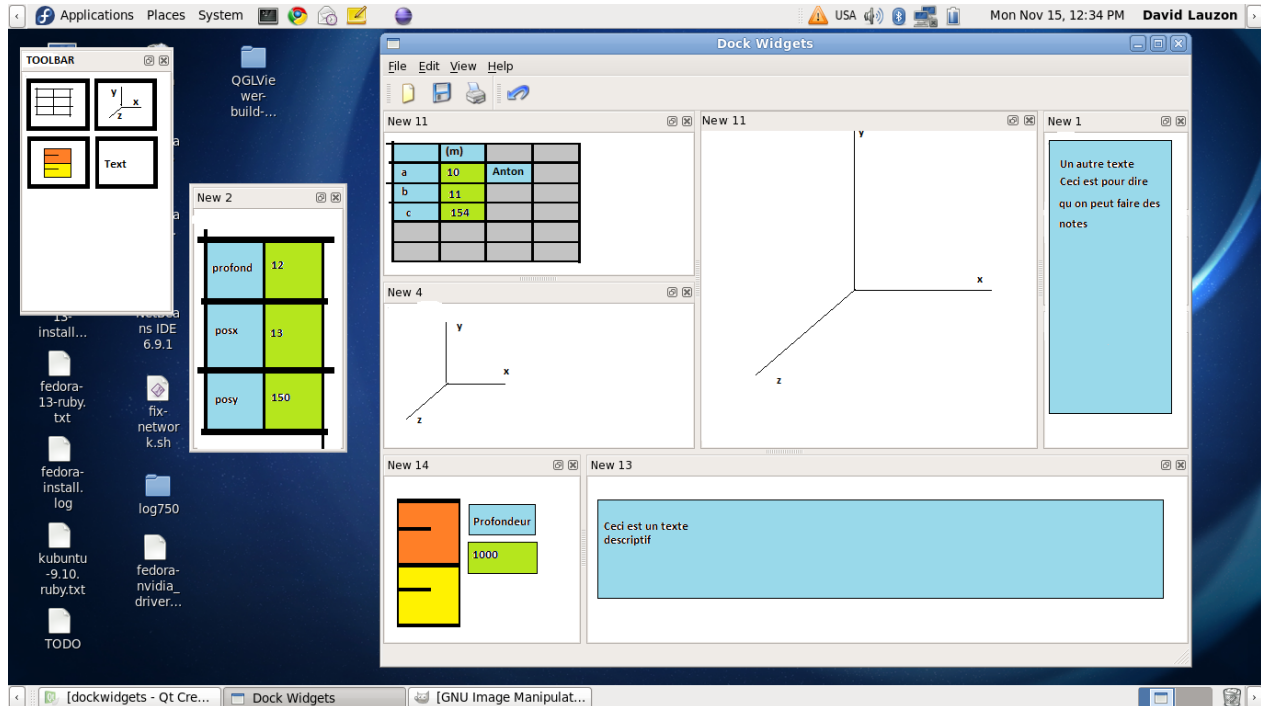


FIGURE 3 – Fenêtre Principale de LiveUV

La figure ci-dessus (Figure 3) est un exemple de screenshot du système en état de marche. Les widgets sont délimités par des zones avec un petit x en haut à droite de chaque section. Le x sert à effacer le widget. Il y a 2 sections qui ne sont pas dans le conteneur principal : la barre d'outils des widgets en haut à gauche de l'écran et le widget détaché au centre de l'écran.

À l'intérieur des widgets :

- Les sections en bleu pale sont des section ou l'utilisateur peut rentrer du texte, mais qui ne sont pas reliés à libJAUS.
- Les section en vert pale sont des valeurs qui sont reliés au valeurs de libJAUS. Modification de ces valeurs entraîne une modification des valeurs de libJAUS.

8.2 Interfaces Matérielles

Aucune.

8.3 Interfaces Logicielles

IL01 : libJAUS : Librairie JAUS en Java développé par SONIA fournissant une couche d'abstraction modulaire au dessus de JAUS. La librairie récupère les données des différentes composantes depuis le véhicule autonome (Ex : vitesse, profondeur, etc.) à travers JAUS.

IL02 : libFS : Librairie fournissant une couche d'abstraction au système de fichier et permettant au système de sauvegarder les perspectives, les paramètres de connections libJAUS (adresse IP du sous-marin, etc.) sur le disque dur peu importe le système de fichier. Cela est requis car différents systèmes de fichiers existent pour chaque plateforme.

8.4 Interfaces de Communications

IC01 : libJAUS : Librairie décrit plus haut. Il est sous-entendu que la librairie communique avec JAUS à travers un réseau local par TCP/IP. Toutefois, le système peut communiquer directement avec libJAUS vu que celle-ci gère les communications réseaux.

9 Exigences de Licenses

Cette section ne s'applique pas dans le cadre de ce projet.

10 Remarques légales, de droits d'auteur, et diverses

Cette section ne s'applique pas dans le cadre de ce projet.

11 Normes et Standards Applicables

11.1 Glossaire

TABLE 2: Glossaire

Terme	Définition
Capra	Club étudiant de l'ÉTS oeuvrant dans la conception d'une tondeuse intelligente et autonome.
Dronolab	Club étudiant de l'ÉTS oeuvrant dans la conception d'un aéronef intelligent et autonome.
ÉTS	École de technologie supérieure (Université d'ingénierie située à Montréal faisant partie du réseau de l'Université du Québec)
JAUS	Joint Architecture for Unmanned Systems. Dans le contexte, cette spécification décrit une architecture, mais également un protocole de communication entre la libJAUS et AUV6.
libJAUS	Système gérant la communication des données depuis le sous-marin vers LiveUV.
Perspective	Disposition des widgets selon des critères prédéfinis ou personnalisés
Protocole	Langage de communication utilisé entre deux composants du système.
SONIA	Système d'opération nautique intelligent et autonome. Club étudiant de l'ÉTS oeuvrant dans la conception d'un sous-marin intelligent et autonome.

TABLE 2: (suite)

UDP	User Datagram Protocol. Un protocole de transport permettant la communication entre deux composants connectés en réseau.
Télémétrie	Logiciel client permettant de récupérer ou d'affecter les valeurs des différents capteurs ou périphériques d'un véhicule autonome.
Widget	Outil graphique facilitant la lecture ou l'affectation de données télé-métriques d'un périphérique du véhicule autonome.
Système	Lorsqu'on parle du système, il s'agit du produit que l'on développe, c'est-à-dire l'interface de la télémétrie.

11.2 Quint-2

Les exigences non-fonctionnelles (section 4.2) ont été classifiées selon le modèle de qualité du standard Quint-2 (Extended ISO 9126-1 Model of Software Quality). Ce modèle constituait le modèle le plus complet lors de la rédaction de ce document, car il classe les exigences en cinq grandes classes : Functionality, Reliability, Efficiency, Maintainability, et Portability.

Pour plus d'information, se référer à : <http://www.serc.nl/quint-book/>

12 Bibliographie

- [1] *Page d'accueil*. 5 Oct. 2010. Club Étudiant SONIA. Visité en Octobre 2010.
<http://sonia.etsmtl.ca/fr>
- [2] *Télémétrie (informatique)*. 10 fév. 2010. Wikipédia. Visité en Octobre 2010.
[http://fr.wikipedia.org/wiki/T%C3%A9l%C3%A9m%C3%A9trie_\(informatique\)](http://fr.wikipedia.org/wiki/T%C3%A9l%C3%A9m%C3%A9trie_(informatique))
- [3] *Underwater Systems Corps - SRS*. 26 mars 2009. LOG410 - Analyse de besoins et spécifications. Visité en Novembre 2010.
<https://cours.etsmtl.ca/log410/private/Labo%203/Exemple-SRS-Editeur-etats.pdf>

Annexes

A Spécifications des Cas d'Utilisation

Les cas d'utilisation sont référés dans le document "Use_Cases_EQ12.pdf".

B Matrice de Tracabilité

TABLE 3: Matrice de Tracabilité

ID	Caractéristique	Exigence	Exigences non fonctionnelles	Cas d'utilisations
CA01	Ajouter un widget	REQ001, REQ002, REQ003	NFR011	CU01, CU02
CA03	Supprimer un widget	REQ002, REQ003		CU03
CA04	Importer un widget	REQ008	NFR019	
CA05	Détacher un widget	REQ002, REQ003		CU05
CA06	Sauvegarder une perspective	REQ003		CU06
CA07	Supprimer une perspective			CU07
CA08	Créer une alerte	REQ003		CU10
CA09	Supprimer une alerte			CU11

C États du système et des widgets

TABLE 4 – Environments (mode d'opération) du système

Opération journalière	Le système opère normalement.
Maintenance	Des ajouts ou correction sont apportés au système.
Installation	Le système est en train de s'installer et peut ne pas avoir encore installés tous les pré-requis à son bon fonctionnement.

TABLE 5 – États de connexion de l'application

Connecté	L'application est connecté par JAUS à un véhicule autonome.
Déconnecté	Pas de connexion à JAUS.

TABLE 6 – États de connexion de l'application

Connecté	L'application est connecté par JAUS à un véhicule autonome.
Déconnecté	Pas de connexion à JAUS.

TABLE 7 – États d'un widget

Opérationnel	Toutes les valeurs nécessaires au bon fonctionnement du widget ont été liées à une valeur correspondante valide de l'arbre JAUS.
Opérationnel Incomplet	Le widget est opérationnel (voir description ci-haut) mais il reste des valeurs optionnelles qui n'ont pas été liées.
Non-Opérationnel	Une ou plusieurs valeurs du widget doivent être liées à des valeurs de l'arbre JAUS pour que le widget soit opérationnel.
En Liaison	L'utilisateur est en train de lier les valeurs du widget à des valeurs de JAUS.

TABLE 8 – États d'une valeur d'un widget

Liée	La valeur d'entrée du widget a été liée (mapping) à une valeur de l'arbre JAUS.
Non-Disponible	La valeur d'entrée du widget a été liée, mais la valeur JAUS n'est plus disponible (ex : composante défaillante ou retirée, etc.).
Non-Liée	La valeur n'a pas été encore assignée ou laissée volontairement vide.
Modifiable non-liable	Une valeur qu'il est possible de modifier mais qui n'est pas liée à libJAUS. Ex : Les étiquettes dans un tableau.

TABLE 9 – États de la position du Widget

Détaché	Détaché du conteneur principal (cad. dans sa fenêtre à lui).
Attaché	Attaché au conteneur principal.
Immobilisé	Le widget ne peut être déplacé ou effacé ou détaché. Les bords du widget sont minimes.
Dans le toolbar	le widget est présent dans le toolbar. À noter que le widget se trouve dans cet état SEULEMENT si c'est indiqué explicitement dans le texte.

D Légendes des diagrammes UML

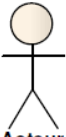
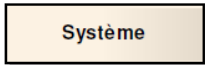
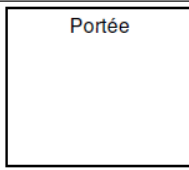
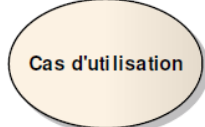
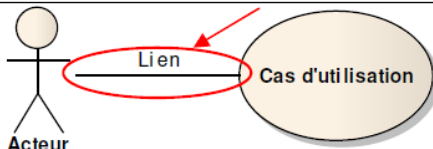
Légende	
 Acteur	Acteur humain du système
 Système	Acteur système (sous-système) du système
 Portée	Ce qui est inclus dans le système, c'est-à-dire, la limite du système et/ou la portée du système.
 Cas d'utilisation	Cas d'utilisation qui permet à un acteur d'effectuer une tâche à l'aide du système.
 Acteur Lien Cas d'utilisation	Signifie que l'acteur utilise le cas d'utilisation ou que le cas d'utilisation utilise le système externe, si le lien est pointillé.

FIGURE 4 – Légende des diagrammes de cas d'utilisation (référence [3])

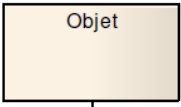
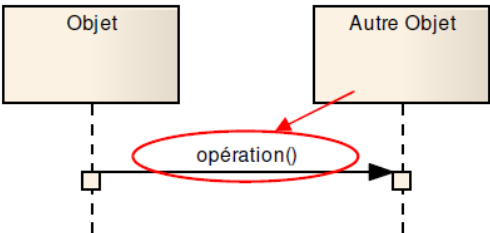
Légende	
 Objet	Objet du système qui exécute des opérations.
 Objet Autre Objet opération()	L'opération exécutée sur l' « Autre objet ».

FIGURE 5 – Légende des diagrammes de séquence (référence [3])

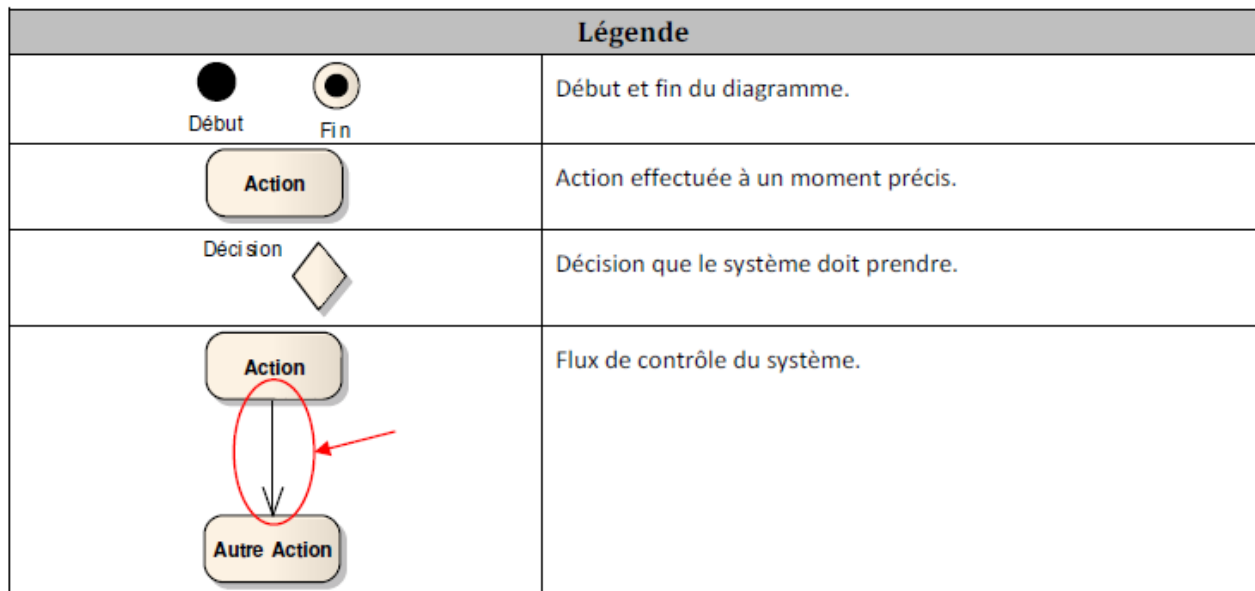


FIGURE 6 – Légende des diagrammes d'activité (référence [3])

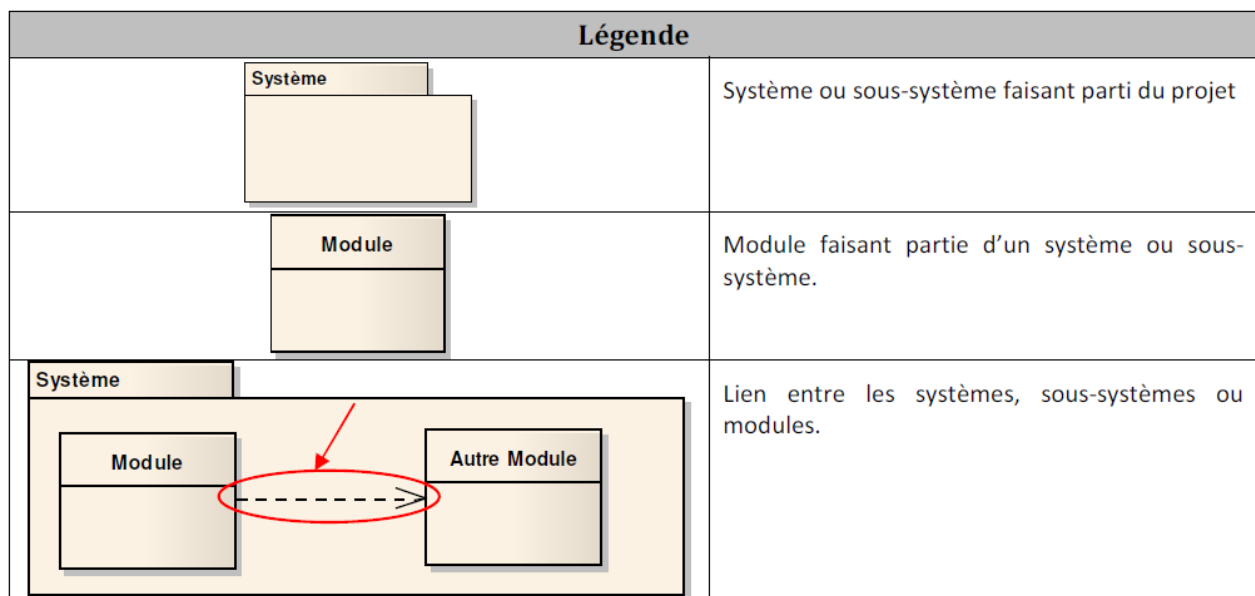


FIGURE 7 – Légende des modèles du domaine (référence [3])


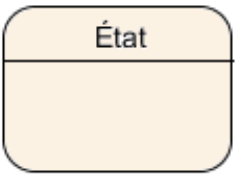

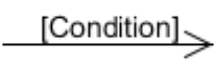
Légende	
	Début et fin de du diagramme
	L'état d'un objet à un moment précis
	Décision quand il y a plusieurs changements d'état possible pour le même flux de changement
	Représente le flux de changement d'états du système La condition représente la condition de changement d'état

FIGURE 8 – Légende des diagrammes d'activité

E Diagrammes UML

Voir les légendes suivantes :

- Légende des diagrammes d'activité (figure 6)
- Légende des diagrammes d'activité (figure 8)

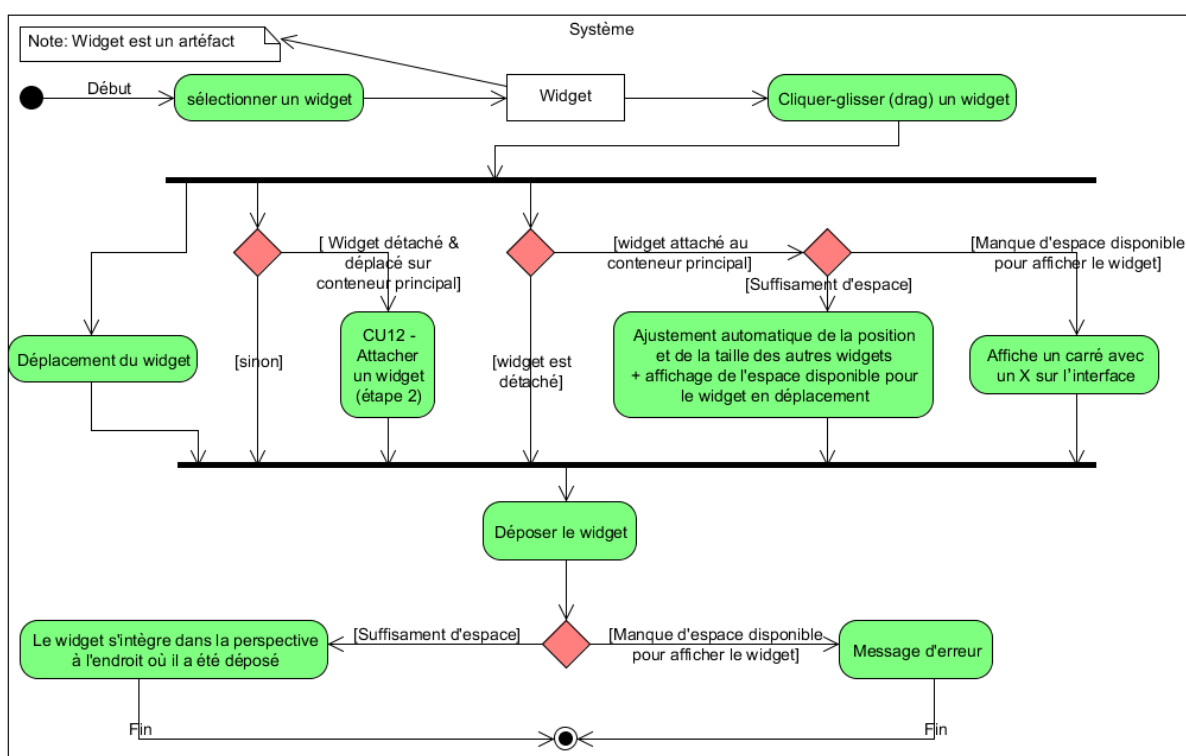


FIGURE 9 – Diagramme d'Activité AD01 - Référence : CU09 Déplacer un widget

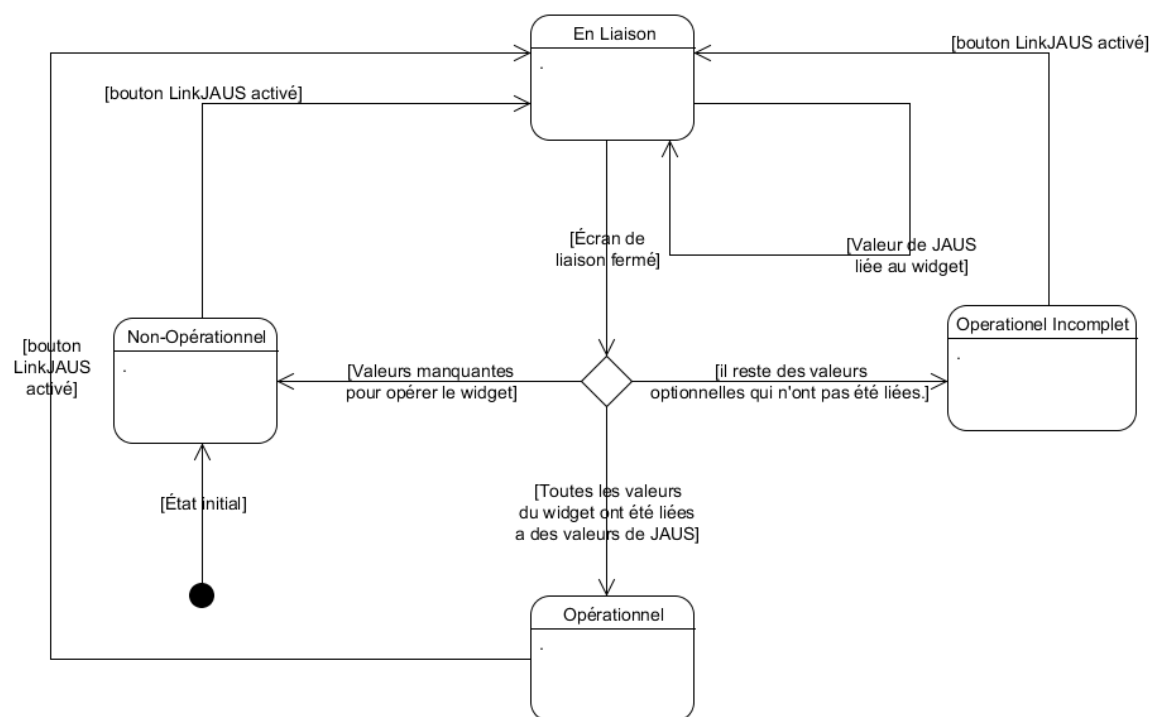


FIGURE 10 – Diagramme d’État STATED01 - Référence : REQ003 États de widget