TraceX

Equipe de projet L2S1

Rapport de Projet TraceX

Automatisation de la Traçabilité Documentaire

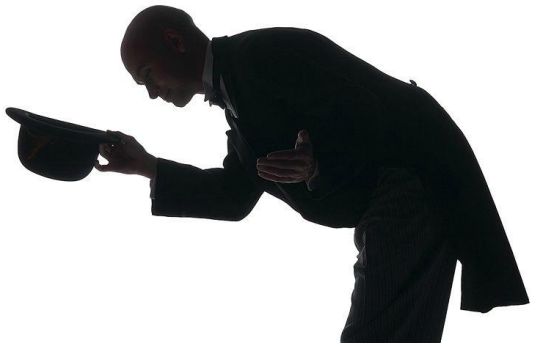
Etablissement	Université de Paris Cité
Date du document :	17/02/2025
Version :	A.02
Auteur(s) :	ALLAHOUM Abdelmalek Said, KIM Léa, HUANG Maxime, ZHENG Jacques
Type de diffusion :	Document électronique (.pdf)
Confidentialité :	Réservé aux étudiants UFR Maths-Info de l'université Paris Cité

Projet TraceX – Equipe de projet L2S1
17/05/2025

Remerciements

Nous tenons à exprimer notre profonde gratitude à M. Jeremy Meynadier, notre encadrant, pour sa bienveillance, son accompagnement rigoureux et la qualité de ses conseils tout au long de ce projet. Sa disponibilité, sa pédagogie et sa capacité à nous guider avec justesse ont grandement contribué à la réussite de ce travail.

Nous remercions également l'ensemble de nos enseignants, dont l'expertise et l'engagement ont nourri notre réflexion et renforcé notre compréhension des enjeux techniques et méthodologiques abordés dans le cadre de ce projet.



Un grand merci à nos camarades, avec qui les échanges, les retours constructifs et le partage d'expériences ont été particulièrement enrichissants et stimulants.

Nous souhaitons enfin remercier l'équipe pédagogique et administrative de notre établissement pour avoir mis à disposition un environnement de travail propice à la réalisation de ce projet, tant sur le plan matériel qu'organisationnel.

Résumé

Dans l'industrie aéronautique ainsi que dans d'autres industries sujettes à des exigences strictes en matière de sécurité et de traçabilité, la traçabilité des exigences entre différents documents est cruciale pour assurer la conformité, la qualité et la sécurité des systèmes développés. Toutefois, la gestion manuelle de ces liens de traçabilité demeure une tâche complexe, sujette à erreurs et très chronophage. Ce projet a été conçu pour automatiser cette traçabilité documentaire inter-niveaux entre les documents SSS (System Segment Specification), SRS (Software Requirements Specification) et SDD (System Design Description), tout en offrant une interface utilisateur intuitive et des outils d'analyse visuelle.

L'objectif principal était de développer une application logicielle capable d'importer des documents dans divers formats (.docx, .xlsx, .csv), d'extraire les exigences contenues selon des styles prédéfinis, de calculer le taux de traçabilité entre les documents, d'afficher dynamiquement les relations sous forme de graphes, et de produire un rapport récapitulatif listant les documents ainsi que leurs exigences. Le développement a été réalisé en C++ avec le framework Qt pour l'interface graphique, la bibliothèque pugixml pour le parsing des fichiers XML.

La réalisation du projet s'est appuyée sur une planification précise, une répartition des tâches claire au sein de l'équipe, et une méthodologie de développement incrémentale avec intégration de tests unitaires et d'intégration. Les fonctionnalités principales ont été validées avec succès : importation multi-format, hiérarchisation des documents, extraction et analyse des exigences, visualisation interactive et application des filtres, ainsi que la génération d'un rapport récapitulatif.

Ce projet apporte une réponse concrète, efficace et adaptée à un besoin industriel réel, en proposant des solutions alignées avec les exigences du terrain, notamment en termes de fiabilité, de traçabilité et d'optimisation des processus. Il illustre la capacité à transformer une problématique métier en un outil fonctionnel répondant aux standards et aux contraintes des environnements industriels, tout en permettant aux étudiants de se confronter à des défis techniques, collaboratifs et organisationnels propres aux projets de développement logiciel en environnement critique.

Table des matières

1. Introduction

1.1. Contexte

L'évolution des systèmes d'information dans les domaines critiques, tels que l'aéronautique, s'accompagne d'une complexité croissante des processus de validation documentaire. Chaque exigence définie dans les documents de spécification doit pouvoir être suivie tout au long du cycle de développement logiciel et système. Cette traçabilité inter-documentaire est une obligation tant réglementaire que contractuelle, garantissant que les exigences initiales sont bien prises en compte dans les phases de conception et d'implémentation. Pourtant, dans de nombreux cas, cette gestion reste manuelle, fastidieuse, sujette à erreurs et difficilement scalable.

Dans ce contexte, le projet que nous avons mené propose une solution innovante d'automatisation de la traçabilité documentaire, adaptée à un environnement sensible, que ce soit dans l'aérospatial ou dans d'autres domaines où la rigueur et la sécurité sont primordiales. L'outil logiciel développé permet de prendre en charge des documents hétérogènes (à savoir : SSS, SRS et SDD), d'en extraire automatiquement les exigences selon des styles définis, de visualiser les liens sous forme de graphe interactif, d'appliquer des filtres et de produire des rapports de synthèse.

1.2. Problématique

Comment automatiser et fiabiliser la gestion de la traçabilité documentaire inter-niveaux (SSS → SRS → SDD) ?

1.3. Objectif du projet

Concevoir une application permettant d'importer, analyser, filtrer, visualiser et générer des rapports de traçabilité documentaire afin de réduire les erreurs humaines, d'accroître la productivité et de renforcer la qualité du suivi documentaire.

1.4. Portée

Le logiciel doit être multiplateforme (Windows, Linux, MacOS) et compatible avec les formats de documents les plus répandus dans l'industrie (.docx, .xlsx, .csv).

1.5. Méthodologie

Une approche de développement agile et itérative a été adoptée, combinant une analyse préalable des besoins, une conception modulaire orientée objet, un développement en C++ avec le framework Qt en utilisant l'IDE Qt Creator, et une intégration de tests automatisés (QtTest) pour valider chaque composant.

1.6. Organisation du rapport

Le rapport est structuré en neuf parties, allant de l'introduction à la bibliographie, en passant par l'analyse des besoins, la planification, la réalisation, les tests, les résultats et les perspectives. ...

2. Analyse des besoins / Cahier des charges

2.1. Destinataires du projet

Ce projet s'adresse principalement à des acteurs confrontés à la nécessité de garantir la traçabilité des exigences tout au long du cycle de développement des systèmes. Il peut s'agir de :

- Ingénieurs systèmes ;
- Responsables qualité ;
- Chefs de projets techniques ;
- Ou encore de développeurs en environnement normé.

L'outil est conçu pour répondre à des besoins métiers concrets dans un environnement critique, où la conformité, la traçabilité et l'efficacité documentaire sont non seulement stratégiques mais aussi réglementairement exigées.

2.2. Enjeux et contraintes

La gestion manuelle de la traçabilité inter-documentaire (entre SSS, SRS et SDD) est source d'erreurs, de retards et de surcharge de travail. Les principales contraintes identifiées sont :

- **Techniques** : compatibilité avec les formats DOCX, XLSX, CSV ; fluidité de l'interface utilisateur ; performance de l'analyse sur des fichiers volumineux.
- **Fonctionnelles** : affichage graphique clair ; filtrage intelligent des exigences.
- **Matérielles** : fonctionnement sur des postes standards, sans besoin de ressources matérielles spécifiques.
- **Organisationnelles** : délais courts (12 semaines), disponibilité limitée de certains membres, communication entre les rôles (développeur, chef de projet, encadrant).

2.3. Objectifs fonctionnels

L'application doit permettre :

- L'importation et la gestion intuitive de fichiers multiples ;
- L'extraction automatique des exigences à partir de styles typographiques définis ;
- L'établissement des relations logiques entre les documents (SSS ↔ SRS ↔ SDD) ;
- Le calcul du taux de traçabilité entre chaque niveau ;

- La visualisation dynamique sous forme de graphe interactif ;
- Le filtrage par statut de développement, radical, cible, ou client pour affiner l'analyse ;
- La génération automatique de rapports (CSV) identifiant les exigences non tracées.

2.4. Livrables attendus

- Une application logicielle multiplateforme avec une interface graphique développée sous Qt.
- Des modules fonctionnels autonomes : importation, extraction, filtrage, visualisation, génération de rapports.
- Un manuel d'utilisation à destination de l'utilisateur final.
- Un cahier de tests / recette fonctionnelle.
- Le code source documenté et les fichiers de configuration associés.

2.5. Spécifications techniques

- Langages : C++ (principal).
- Frameworks & bibliothèques : Qt (GUI), Pugixml (parsing XML), QtTest (tests unitaires), Figma (maquettes).
- Systèmes cibles : Windows, Linux, MacOS.
- Contraintes de performance : analyse en temps réduit (<1s pour des documents de taille moyenne), consommation mémoire optimisée.

2.6. Schémas et représentation des fonctionnalités

Un diagramme fonctionnel a été établi, représentant :

- Le flux de données entre modules :
  - (Importation des fichiers ;
  - Extraction des données pertinentes ;
  - Analyse et calcul du taux de traçabilité ;
  - Visualisation des liens ;
  - Génération d'un rapport récapitulatif.
- Les interactions utilisateur (glisser-déposer, filtres dynamiques, navigation intuitive) ;
- La structure hiérarchique des documents importés (SSS en parent, SRS en enfant, SDD en petit-fils).

Ce cahier des charges a servi de base contractuelle entre les membres du groupe, l'encadrant et les attentes pédagogiques. Il a guidé l'ensemble du développement et a permis d'assurer une correspondance stricte entre les spécifications initiales et les fonctionnalités livrées.

3. Planification

3.1. Organisation de l'équipe

L'équipe projet était constituée de quatre membres, chacun ayant un rôle spécifique aligné avec ses compétences et les besoins fonctionnels du projet :

Membre	Rôle principal	Tâches clés réalisées
ALLAHOUM Abdelmalek Said	Développeur backend	Extraction des exigences, Extraction des styles, Extraction des critères de filtrage, application des filtres, parsing XML, traitement des fichiers, logique métier, Affichage du graphe
KIM Léa	Développeur backend	Génération de rapports, calcul du taux de traçabilité
HUANG Maxime	Développeur frontend	Interface graphique sous Qt, ergonomie, navigation, gestion des événements, importation et structuration des fichiers
ZHENG Jacques	Développeur frontend	Affichage graphique des statistiques, implémentation de la page help, Importation des fichiers, Affichage des exigences dans la page du graphe

Chaque membre a également contribué à la rédaction des documents techniques (cahier des charges, conception générale et détailléé, manuels utilisateurs) et à la phase de tests.

3.2. Répartition des tâches

Les tâches ont été réparties en suivant une logique modulaire, afin de permettre un développement parallèle et une intégration progressive. Les modules principaux étaient :

- Importation des documents ;
- Extraction des styles et des exigences ;
- Filtrage et hiérarchisation ;
- Calcul de la traçabilité ;
- Visualisation interactive ;
- Génération de rapports.

3.3. Étapes du projet

Le projet s'est déroulé sur 12 semaines, selon les jalons suivants :

Semaine	Tâche
Semaines 1 à 2	Analyse des besoins, recherche documentaire, cadrage du périmètre
Semaine 3	Rédaction du cahier des charges, validation par l'encadrant
Semaine 4	Conception de l'architecture logicielle et modélisation des classes
Semaines 5 à 6	Développement du module d'analyse documentaire (lecture et extraction)
Semaines 6 à 7	Développement du module de calcul de statistiques
Semaines 8 à 10	Intégration de l'interface graphique avec Qt, interactions utilisateur
Semaine 10 à 11	Intégration des modules, test fonctionnel, validation et corrections
Semaine 12	Finalisation, production du livrable, rédaction du rapport de projet

3.4. Diagramme de Gantt (représentation simplifiée)

Lien du diagramme de Gantt :

https://www.notion.so/lea729/1d154784fe948099ad1df6d8d348005f?v=1d354784fe9480a7bd13000c4a5b2313

3.5. Suivi et gestion

Le projet a été suivi de manière hebdomadaire via :

- Des réunions de coordination (15 à 30 minutes) ;
- L'utilisation d'un gestionnaire de versions (SVN) pour le code ;
- Un partage de fichiers sur un espace collaboratif en ligne (docs.google.com) ;
- Une feuille de route partagée avec échéances et priorités.

Les décisions techniques majeures ont été soumises à validation de l'encadrant.

4. Réalisation / Développement

4.1. Vue d'ensemble

La réalisation du projet s'est faite de manière incrémentale, en s'appuyant sur une conception modulaire orientée objet. Chaque fonctionnalité a été développée, testée et intégrée au fur et à mesure, avec une coordination étroite entre les membres de l'équipe.

Le projet s'est articulé autour de cinq grandes briques fonctionnelles :

- L'importation et la structuration des fichiers ;

- L'analyse documentaire et l'extraction de données pertinentes ;
- Le calcul du taux de traçabilité ;
- La visualisation interactive des liens ;
- La génération de rapports et visualisation des statistiques.

4.2. Technologies utilisées

Élément	Technologie choisie	Rôle / Justification
Langage principal	C++	Performance, contrôle mémoire, compatibilité Qt, appel système
Interface graphique	Qt (Qt Creator)	Développement multiplateforme, ergonomie
Parsing XML	pugixml	Extraction rapide et fiable de styles / exigences / critères de filtrage
Visualisation graphique	Qt	Génération dynamique du graphe
Prototypage UI	Figma	Maquettes fonctionnelles
Système de versionnage	SVN	Suivi des modifications, travail collaboratif
Tests automatisés	QtTest	Fiabilité du code, tests unitaires modulaires

4.3. Déroulement par modules

a) Importation et structuration des fichiers

Un module d'importation a été développé pour gérer les fichiers au format .docx, .xlsx et .csv, avec une interface de type glisser-déposer. Une table liste les fichiers chargés, et un arbre hiérarchique permet de les classer manuellement (SSS → SRS → SDD).

Des fonctions spécifiques ont été implémentées pour lire le contenu XML des fichiers Word et Excel qui sont en réalité des dossiers compressés (ZIP), ainsi que pour traiter les fichiers CSV en lecture ligne par ligne, suivie d'une phase de nettoyage des données.

b) Extraction des styles et exigences

L'extraction des exigences repose sur la détection de styles créés et utilisés par l'utilisateur (Titre1, Titre2, Exigence, Traçabilité, etc.) à l'intérieur des fichiers au format .docx. Un module permet à l'utilisateur de sélectionner le style associé aux exigences. Le parsing XML est effectué via **pugixml**, avec des classes spécialisées pour les documents SSS, SRS et SDD en parcourant les balises des fichiers XML, en particulier le fichier ''document.xml'' où se trouve

le contenu du fichier .docx et le fichier ‘’styles.xml’’ où on retrouve les styles utilisés dans le fichier, pour en extraire les données significatives.

Pour les documents au format XLSX, des fonctions dédiées traitent respectivement les fichiers sharedStrings.xml et ainsi que les balises contenant le contenue des cellules.

Pour les documents au format CSV, le traitement était similaire à celui d’un fichier .txt en lisant le document ligne par ligne.

c) Filtrage et hiérarchisation

Une fois les exigences extraites, l’utilisateur peut appliquer des filtres selon :

- Le statut (développé / non développé) ;
- La cible matérielle (ex. matériel embarqué) ;
- Les clients (nécessaire ou non pour tel ou tel client) ;
- Les radicaux extraits.

La hiérarchisation est gérée via un arbre dynamique dans l’interface, reflétant les relations parent / enfant entre fichiers.

d) Calcul de la traçabilité

Un algorithme de comparaison croisée a été développé pour :

- Détecter les exigences tracées entre SSS → SRS et SRS → SDD ;
- Calculer le taux de traçabilité pour chaque lien ;
- Générer un rapport récapitulatif listant les documents ainsi que leurs exigences de manière à pouvoir identifier les exigences tracées et non tracées.

La traçabilité est exprimée en pourcentage :

Taux de traçabilité = (Nombre d’exigences tracées/Nombre total d’exigences) ×100

e) Visualisation interactive

La représentation graphique des liens de traçabilité a été réalisée avec Qt, intégrée dans une fenêtre Qt (QGraphicsView). Le graphe affiche :

- Les nœuds représentant les documents ;
- Les arêtes représentant les liens d’exigences ;
- Le taux de traçabilité entre chaque paire ;
- Les exigences de chaque document ;

- Le nom de chaque document.

Des boutons de zoom (+ / –), des options de sélection et de survol dynamique améliorent l'expérience utilisateur.

f) Génération de rapports

Une fonctionnalité permet d'exporter un rapport au format **CSV** contenant :

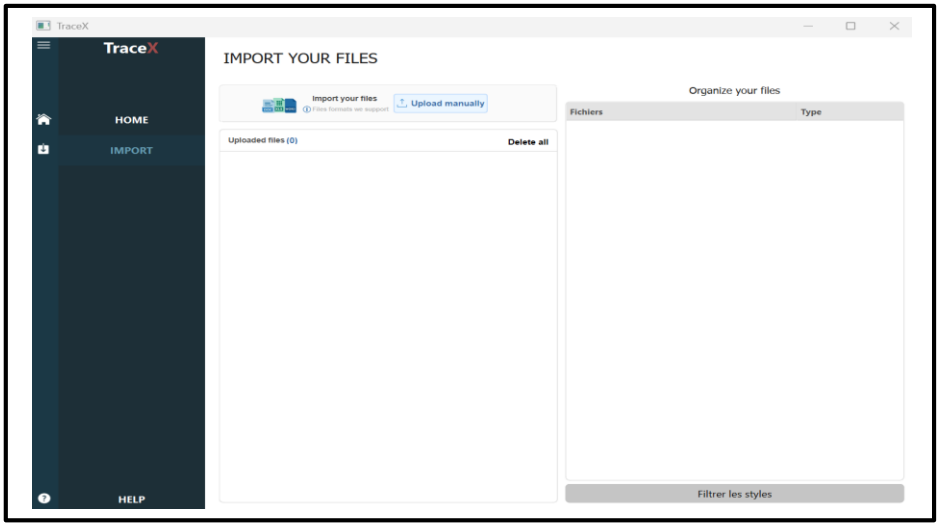
- Les exigences des documents ;
- Les documents.

4.4. Difficultés rencontrées

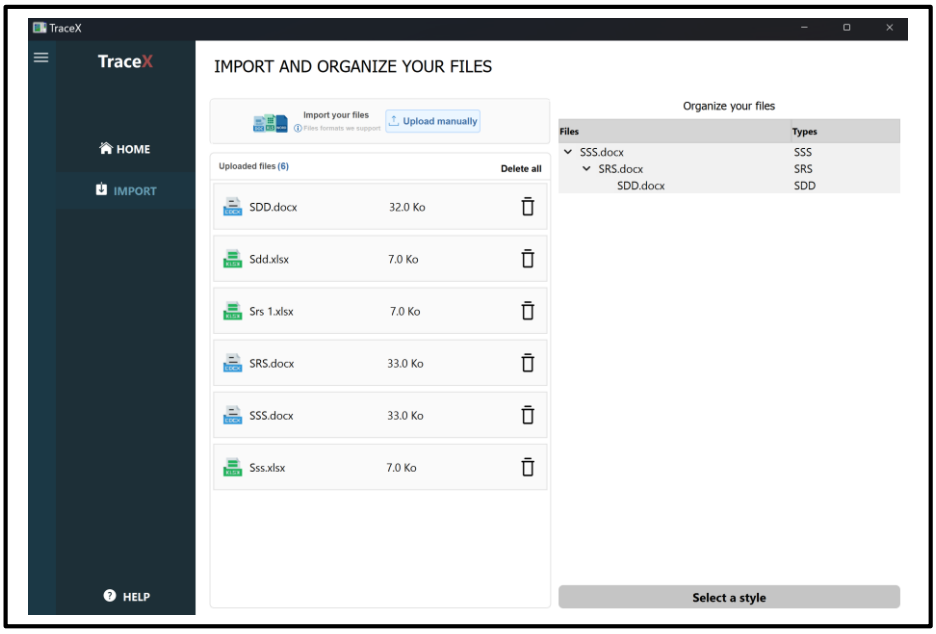
Problème rencontré	Solution apportée
Extraction XML de fichiers .docx complexes	Analyse ciblée des balises pertinentes (document.xml, styles.xml)
Identification correcte des styles	Sélecteur visuel avec cases à cocher pour validation utilisateur
Gestion du glisser-déposer hiérarchique (arbre Qt)	Classes personnalisées QListWidget_custom, QTreeWidget_custom
Performance sur fichiers lourds (>1000 exigences)	Traitement optimisé, parsing ligne par ligne
Intégration des modules dans une interface cohérente	Centralisation des événements dans la classe MainWindow

4.5. Illustrations et captures d'écran

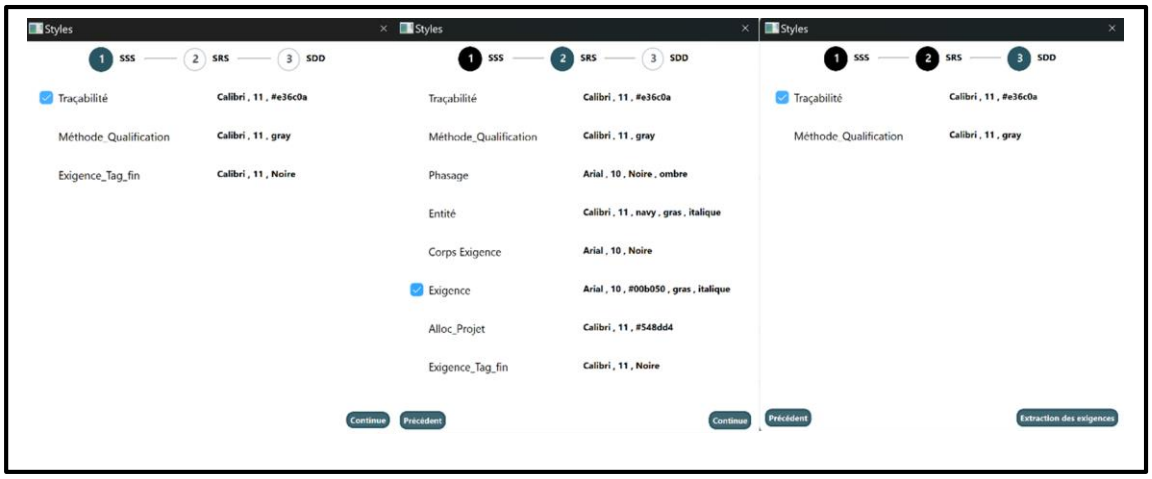
- Interface d'importation avec tableau de fichiers



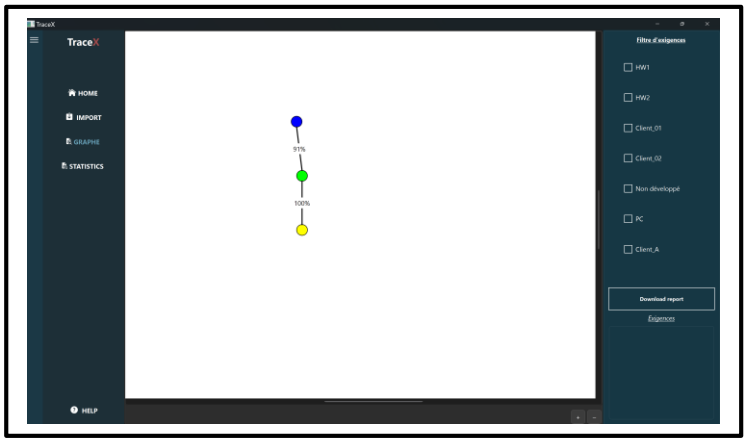
- Arbre hiérarchique SSS → SRS → SDD



- Fenêtre de sélection des styles



- Graphe de traçabilité interactif



- Histogramme des taux de traçabilité

- Rapport CSV exporté

4.6. Installateurs

Le logiciel développé par le groupe est conçu pour être facilement déployé sur les principaux systèmes d'exploitation, à savoir macOS, Linux (Ubuntu) et Windows. Afin de simplifier au maximum le processus d'installation pour l'utilisateur, des installateurs spécifiques ont été créés pour chaque plateforme. Ces installateurs prennent en charge automatiquement l'installation des dépendances nécessaires, telles que les bibliothèques, packages et Frameworks requis au bon fonctionnement de l'application, sans intervention manuelle.

Cependant, sur macOS, bien que l'installateur fonctionne correctement sur la machine où il a été généré, nous avons constaté une limitation liée à la politique de sécurité d'Apple. En effet, macOS impose une validation (notarisation) des applications par ses services avant de les exécuter librement sur d'autres machines. Cette procédure, qui implique une inscription au programme développeur Apple et un processus payant de vérification, ne nous a pas été accessible dans le cadre de ce projet. Par conséquent, nous n'avons pas pu tester ni garantir le bon fonctionnement de l'installateur sur des machines macOS autres que celle utilisée pour sa création. En revanche, sur les systèmes Windows et Linux-Ubuntu, les installateurs sont pleinement opérationnels. La seule différence observée concerne un léger écart de performance au lancement de l'application, selon le système.

5. Tests et validation

5.1. Objectifs des tests

L'objectif principal de la phase de test était de garantir la conformité fonctionnelle et technique du logiciel par rapport aux exigences définies dans le cahier des charges. Les tests visaient à :

- Vérifier le bon fonctionnement de chaque module indépendamment (tests unitaires) ;
- Valider l'interaction entre les modules (tests d'intégration) ;
- Évaluer la performance et la robustesse de l'application ;
- Contrôler la conformité avec les cas d'usage réels à travers des scénarios utilisateurs.

5.2. Méthodologie de test

Une approche orientée tests a été adoptée pour les composants critiques du projet (comme l'extraction, la traçabilité et les calculs), tandis que la validation fonctionnelle de l'interface graphique a été réalisée manuellement. Les tests unitaires en C++ ont été automatisés à l'aide de la bibliothèque QtTest.

Type de test	Outils utilisés	Modules concernés
Tests unitaires	QtTest	Extraction, parsing, calcul de traçabilité
Tests d'intégration	tests manuels	Interaction entre importation, analyse, visualisation
Tests fonctionnels	Scénarios utilisateurs	Parcours complet de l'application
Tests UI	Vérification manuelle	Navigation, affichage, filtres
Tests de performance	Mesure temps d'exécution	Parsing et génération de graphes pour fichiers volumineux

5.3. Scénarios de test fonctionnel

Des cas d'usage concrets ont été définis pour simuler les actions d'un utilisateur final dans des conditions réalistes :

- **Scénario 1** : Importer un fichier SSS et un fichier SRS, vérifier leur affichage dans l'arbre hiérarchique.
- **Scénario 2** : Sélectionner un style d'exigence pour chaque document, lancer l'analyse, et observer la génération du graphe.
- **Scénario 3** : Appliquer un filtre par cible ou statut (développé/non développé) et valider l'effet immédiat sur le graphe.

- **Scénario 4** : Exporter le rapport des exigences non tracées au format CSV, vérifier l'exactitude des résultats.
- **Scénario 5** : Supprimer un fichier via l'interface, observer la mise à jour automatique du tableau et de l'arborescence.

5.4. Résultats des tests

Test effectué	Résultat	Observations
Importation de fichiers multiples	**Réussi**	Chargement instantané, fichiers listés dynamiquement
Extraction des styles et exigences	**Réussi**	Détection correcte des styles, affichage clair pour sélection utilisateur
Hiérarchisation par glisser-déposer	**Réussi**	Interface intuitive, hiérarchie respectée
Calcul du taux de traçabilité	**Réussi**	Valeurs exactes validées sur jeux de données simulés
Visualisation du graphe	**Réussi**	Génération fluide, arêtes claires, taux affichés
Filtrage dynamique	**Réussi**	Rafraîchissement automatique du graphe selon les filtres cochés
Exportation CSV du rapport	**Réussi**	Contenu fidèle aux exigences non tracées détectées
Test de performance sur fichier lourd	**Acceptable**	Temps de traitement légèrement supérieur à 1s sur fichiers > 100Mo

5.5. Analyse des écarts

Seul un léger ralentissement a été constaté lors du traitement de fichiers très volumineux (> 100Mo), principalement dû au parsing XML en profondeur. Des optimisations par lecture segmentée sont en cours de réflexion pour améliorer encore les performances.

5.6. Couverture des exigences

Fonctionnalité	Statut
Importation multi-format	**Implémentée**
Hiérarchisation des documents	**Implémentée**
Extraction automatisée des exigences	**Implémentée**
Sélection personnalisée des styles	**Implémentée**

Fonctionnalité	Statut
Calcul du taux de traçabilité	**Implémentée**
Visualisation graphique	**Implémentée**
Filtres dynamiques (statut, radical, cible, client)	**Implémentés**
Export des exigences non tracées (CSV)	**Implémenté**

6. Résultats / Bilan

6.1. Fonctionnalités pleinement opérationnelles

Au terme du développement, toutes les fonctionnalités clés prévues dans le cahier des charges ont été mises en œuvre avec succès et validées par des tests fonctionnels. En particulier :

- **Importation multi-format** : l'application prend en charge les fichiers .docx, .xlsx, .csv, avec une interface fluide et intuitive.
- **Extraction automatisée des exigences** : via le parsing XML des documents Word/Excel, selon des styles configurables.
- **Hiérarchisation des documents** : système dynamique de glisser-déposer avec classification parent/enfant.
- **Calcul du taux de traçabilité** : comparaisons précises entre exigences, avec taux exprimé en pourcentage.
- **Visualisation graphique** : graphe interactif généré avec taux affichés, navigation fluide, zoom, filtres.
- **Filtres avancés** : filtrage des exigences par statut (développé/non développé), radical (à partir d'une expression régulière), cible et client.
- **Export de rapports** : génération de fichiers CSV contenant les exigences non tracées et leurs statistiques.

6.2. Ce qui pourrait être amélioré

- **Support du format PDF** : Le support du format .pdf, initialement envisagé comme une amélioration facultative en fin de développement si le temps le permettait, a finalement été écarté en raison de sa complexité d'intégration dans les délais impartis. Il pourra être ajouté dans une version ultérieure avec des bibliothèques adaptées (par ex. Poppler, PDFium).

- **Optimisation des performances sur gros volumes** : bien que fonctionnelle, l'analyse de fichiers très volumineux (> 100 Mo) pourrait être accélérée par l'implémentation de techniques de traitement différé ou asynchrone.
- **Affichage graphique depuis un rapport** : il peut être utile à l'utilisateur de reconstituer le graphe à partir des données d'un rapport.

6.3. Comparaison avec les objectifs initiaux

Objectif défini	Réalisé ?	Remarques
Automatiser l'extraction des exigences	**Oui**	Extraction fiable et paramétrable
Représenter visuellement les liens	**Oui**	Graphe interactif complet
Calculer le taux de traçabilité	**Oui**	Calcul validé via tests simulés
Filtrer les exigences	**Oui**	Filtrage multi-critères implémenté
Générer un rapport exportable	**Oui**	Fichiers CSV produits et testés
Support du format PDF	**Non**	Option abandonnée temporairement
Traitement performant sur très gros fichiers	**Partiel**	Acceptable mais perfectible au-delà de 100Mo

6.4. Indicateurs qualitatifs

- **Taux de réussite fonctionnelle** : 100% des cas d'usage testés ont été validés sans anomalie.
- **Satisfaction de l'encadrant** : les démonstrations hebdomadaires ont permis d'ajuster rapidement les livrables.
- **Robustesse du logiciel** : aucune erreur critique bloquante détectée lors des tests d'intégration.

7. Conclusion

Ce projet d'automatisation de la traçabilité documentaire dans des environnements sensibles tel que l'aéronautique a permis de répondre à une problématique réelle : la complexité, la lenteur et la fragilité des méthodes manuelles de suivi des exigences dans les documents techniques. En seulement douze semaines, notre équipe a pu concevoir, développer et valider une application fonctionnelle, intuitive et efficace, capable de transformer un processus traditionnellement fastidieux en une opération fluide, structurée et visuellement claire.

La réalisation de ce projet nous a permis d'acquérir des compétences concrètes sur toute la chaîne du développement logiciel :

- Analyse des besoins, formalisation et rédaction d'un cahier des charges ;

- Conception logicielle orientée objet, architecture modulaire ;
- Programmation en C++ avec Qt pour l'interface utilisateur ;
- Manipulation avancée de fichiers XML et formats bureautiques ;
- Tests unitaires et scénarios de validation utilisateur ;
- Collaboration de groupe et gestion de versions (SVN).

Nous avons également pris conscience de l'importance de la traçabilité dans les environnements critiques comme l'aéronautique, où la conformité réglementaire, la qualité documentaire et la sécurité sont indissociables. L'outil développé constitue un prototype fonctionnel susceptible d'être étendu et adapté à d'autres secteurs exigeants (médical, ferroviaire, défense).

7.1. Améliorations possibles

Plusieurs pistes d'amélioration ont été identifiées pour une future version du projet :

- Intégration du format PDF pour une couverture documentaire complète ;
- Optimisation du traitement de fichiers volumineux avec des techniques de lecture différée ou multi-threading ;
- Renforcement de la sécurité documentaire (ex. : contrôle d'accès, chiffrement des données sensibles) ;
- Ajout d'une base de données locale pour gérer des projets plus complexes avec historisation.

7.2. Perspectives de poursuite

L'outil pourra être intégré dans un workflow de gestion qualité ou dans un système de gestion des exigences (Requirements Management System). Il pourra aussi être proposé comme base pour un projet de fin d'études, un stage en entreprise, ou une industrialisation plus poussée dans le cadre d'un partenariat avec un acteur du secteur.

8. Annexes

- Captures d'écran de l'interface
- Diagramme de Gantt

9. Bibliographie / Sources

- Qt Documentation
- Pugixml Manual
- OpenClassroom : Programmation C++
- LinkedIn Learning : Qt avancé
- Tutoriels divers (YouTube, StackOverflow)
- ChatGPT : aide ponctuelle pour choix technologiques