Réingénierie et évolution du logiciel PACIQ

par

Abderrahmane ELMOUL

RAPPORT DE PROJET PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE COMME EXIGENCE PARTIELLE À L'OBTENTION DE M. Ing. GÉNIE LOGICIEL

MONTRÉAL, LE 30 NOVEMBRE 2016

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

PRÉSENTATION DU JURY

CE RAPPORT DE PROJET A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

Professeur Alain April, directeur de projet
Département de génie logiciel & des technologies de l'information à l'École de technologie supérieure

Professeur Abdelaoued Gherbi, jury
Département de génie logiciel & des technologies de l'information à l'École de technologie supérieure

REMERCIEMENTS

L’expression de ma reconnaissance et ma gratitude s’adresse principalement à mon directeur de projet, le professeur Alain April qui m’a fait confiance et m’a encouragé tout au long de ce projet. Grâce à sa disponibilité, ses directives et ses conseils dotés d’une très bonne qualité et d’un immense réconfort, j’ai pu relever le défi et atteindre mon objectif avec une très grande motivation et dynamisme.

Mes remerciements les plus vifs vont également à mon codirecteur en entreprise monsieur Michel Lemay directeur de département de « qualité, sécurité et risque » au centre hospitalier universitaire Sainte-Justine de Montréal, ainsi que madame Isabelle Olivier coordonnatrice au même département pour leur collaboration en ce qui a trait à la compréhension de leur domaine d’affaires, afin d’accomplir mon projet dans les meilleures conditions.

Je dédie ce projet à mes parents, Rachida et Mostafa, pour tout ce qu’ils ont sacrifié pour me permettre d’atteindre ces sommités. Malgré la distance qui nous sépare, ils m’ont toujours encouragés.

Je remercie également ma femme Amal et mon fils Nizar pour leur apport moral, leur amour et leur soutien.

Je profite de cette occasion également pour exprimer ma reconnaissance et mes remerciements les plus sincères à toute personne ayant contribué à la réalisation de ce projet.

RÉINGÉNIERIE ET ÉVOLUTION DU LOGICIEL PACIQ

Abderrahmane ELMOUL

RÉSUMÉ

Dans l'industrie du logiciel, l'évaluation de la qualité est l'une des activités les plus importantes et cruciales du processus de développement logiciel, car elle permet de déterminer la capacité du produit en cours de développement à réponde aux attentes des parties prenantes.

Ce rapport de projet appliqué de maîtrise - 15 crédits, décrit de manière exhaustive la démarche d'évaluation et d'amélioration de la qualité du logiciel PACIQ *(c.-à-d. le logiciel de suivi de l'amélioration continue de la qualité pour les établissements de la santé au Québec)*. Ce dernier a été développé au profit du centre hospitalier universitaire Sainte-Justine, par des étudiants au baccalauréat et à la maîtrise à l'École de Technologie Supérieure, sous la supervision du professeur Alain April. Le but de ce logiciel est de combler les besoins de l'hôpital en termes de suivi des initiatives d'amélioration de la qualité des services de santé en fonction des activités de certifications et ses normes applicables.

Ce rapport débute par une revue de littérature concernant les méthodes d'évaluation de la qualité logicielle d'un point de vue interne et externe. La deuxième partie décrit la démarche d'évaluation de la qualité du logiciel PACIQ, réalisée selon le modèle de qualité multiperspective proposé par la norme ISO 25000.

Ensuite différentes améliorations fonctionnelles ont dues être apportées sur ce logiciel, afin d'obtenir une version plus conforme aux exigences fonctionnelles et non-fonctionnelles décrites au document de spécifications du système, ainsi qu'aux standards et bonne pratique de développement logiciel. Le dernier chapitre du rapport présente le résultat des améliorations apportées ainsi qu'une synthèse des enjeux rencontrés et des solutions mises en œuvre.

Mots-clés : Évaluation de la qualité logicielle, Amélioration de la qualité logicielle, perspective de la qualité logicielle, ISO25000, Réingénierie logicielle.

REINGENEERING AND EVOLUTION OF THE PACIQ SOFTWARE

Abderrahmane ELMOUL

ABSTRACT

The quality evaluation is one of the most important and crucial activity of the software development process, as it determines the capacity of the product under development to meet the stakeholders expectations.

This 15 credits master degree applied project report, describes the activities of quality assessment and reengineering of the PACIQ software that was developed for the Sainte-Justine University Research Hospital by a group of ETS master students, under the supervision of the professor Alain April. The objective was to meet the needs of the hospital in terms of monitoring the evaluating the quality certification activities based on applicable standards.

This report first presents a literature review of the software quality evaluation methods, from internal and external perspectives. The second part describes the activities of assessing quality and reengineering performed on PACIQ according to a multiperspective quality model proposed by ISO 25000. Then, it presents the various enhancements applied to the software source code, to obtain a new version that better meets the functional and non-functional requirements specified in the software requirements specification (SRS), the software development standards and the software engineering best practices. The final chapter of the report summarizes the impact of these enhancements and the challenges faced during this applied project at the hospital.

Keywords: Evaluation of software quality, software quality improvement perspective of software quality, ISO 25000.

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

AQL	Assurance Qualité Logicielle
CHUSJ,	Centre hospitalier de l'Université de Montréal de Sainte-Justine
CSS	Cascading Style Sheet
ÉTS	École de Technologie Supérieure
HTML	HyperText Markup Language
IDE	Integrated Development Environment, ou Environnement de Développement
IIS	Internet Information Services
ISO	International Organization for Standardization
MVC	Modèle-vue-contrôleur.
PACIQ	Logiciel de suivi de l'amélioration continue de la qualité pour les établissements de la santé au Québec
SRS	Software requirements specification
SSRS	SQL Server Reporting Service
SQL	Structured Query Language
TSQL	Transact-SQL

INTRODUCTION

A. Problématique et contexte du projet

Dans le cadre du « plan stratégique, 2010-2015 », déposé par le ministère de santé et des services sociaux afin d'orienter les actions des organisations et des professionnels fournissant des services de santé et des services sociaux à la population du Québec, un projet de recherche et de développement a été démarré, au centre hospitalier Universitaire Sainte-Justine en collaboration avec l'École de Technologie Supérieure. L'objectif de ce projet consiste à réaliser un logiciel, dénommé PACIQ, permettant l'informatisation du processus de gestion des initiatives d'amélioration de la qualité des services de santé, en fonction des normes applicables, ainsi que de soutenir l'équipe qualité concernant les activités de l'hôpital en termes d'accréditation.

Le logiciel PACIQ (c.-à-d. un premier prototype qui n'est pas encore déployé) a été développé initialement par plusieurs étudiants au baccalauréat et à la maîtrise à l'ÉTS, sous la supervision du professeur Alain April qui a joué un rôle très important, notamment en termes de gestion de projet, de contrôle de qualité et de synchronisation des travaux de réalisation entre le département de « qualité, sécurité et risques » du CHU Sainte-Justine (Maîtrise d'ouvrage) et les différents étudiants ayant travaillé sur le projet (Maîtrise d'œuvre). Toutefois, vu le nombre et l'ampleur des modifications réalisées récemment sur ce logiciel, suite à des demandes de changements et des défauts identifiés par le client durant l'étape de tests d'acceptation, le professeur Alain April a jugé nécessaire d'évaluer la qualité du prototype logiciel PACIQ, et d'y effectuer une étape de réingénierie visant à améliorer le logiciel avant de le déployer dans l'environnement de production.

Ma contribution dans ce projet de maîtrise appliquée de 15 crédits, effectuée à l'ÉTS, fût de mener d'abord une évaluation multiperspective de la qualité du logiciel PACIQ, dans le but d'identifier tous les défauts susceptibles de causer des défaillances dans l'environnement de production, et de proposer des solutions permettant de remédier chacun de ces problèmes.

B. Portée du projet

Le logiciel PACIQ est une application web à trois niveaux logiques, développée en utilisant les technologies : web ASP.NET (version 4.0 du Framework .NET), le langage de programmation C#, ainsi que le SGBDR Microsoft SQL Serveur 2008R2 pour la persistance des données.

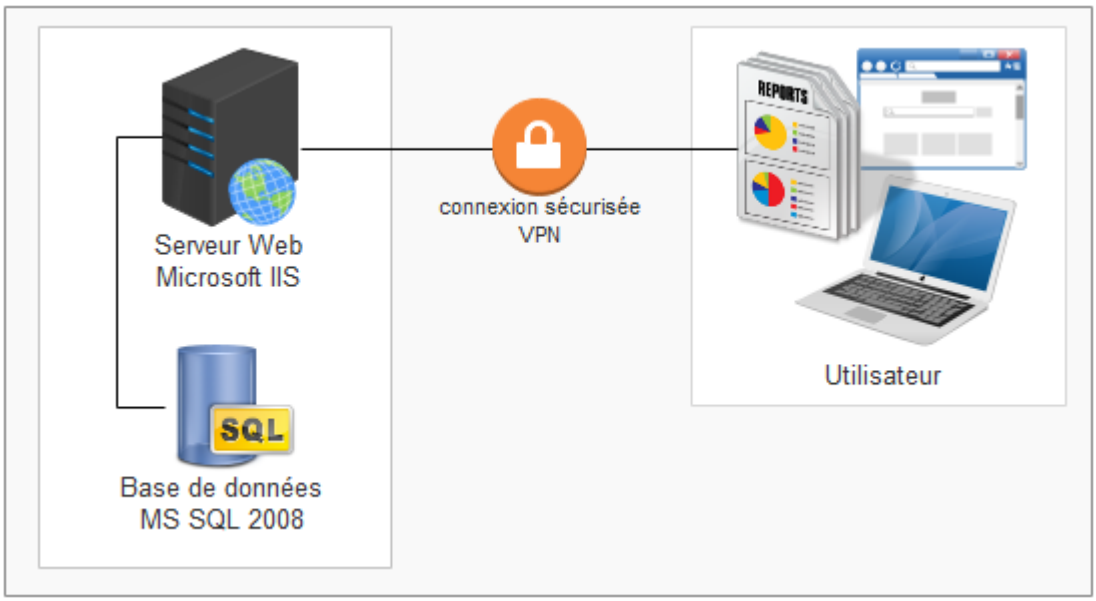


Figure 1.1 Architecture de l'application PACIQ

Les principales fonctionnalités offertes par la première version du prototype PACIQ sont:

- La gestion des plans d'amélioration;
- La gestion des activités liées aux améliorations;
- La gestion des plans d'accréditation;
- La gestion des différentes versions de cahiers de normes;
- La gestion des indicateurs de conformité et de performante des services de santé;
- La visualisation des rapports liés aux initiatives d'améliorations saisies.

Toutefois, la portée des activités d'évaluation et d'amélioration de la qualité réalisées dans le cadre de ce projet se limite à la qualité du code source du logiciel PACIQ et ses fonctionnalités offertes, par rapport aux critères suivants:

- La conformité du logiciel aux exigences fonctionnelles, non fonctionnelles, ainsi qu'aux contraintes spécifiées dans le SRS;
- La pertinence de l'architecture et de la conception du logiciel par rapport aux bonnes pratiques du génie logiciel;
- La conformité du code source avec les standards de développement C# et SQL de Microsoft;
- La pertinence du plan de gestion de la configuration du code source.

C. Conclusion

Cette section nous a permis de mettre en contexte ce projet de maîtrise appliquée de 15, tout en présentant la porter de ses activités d'évaluation et d'amélioration de la qualité du logiciel PACIQ. Le prochain chapitre présente une revue de l'état de l'art portant sur l'évaluation de la qualité logicielle.

CHAPITRE 1

REVUE DE LITTÉRATURE CONCERNANT L'ÉVALUATION DE LA QUALITÉ LOGICIELLE

1.1.Introduction

Dans le cadre compétitif et concurrentiel des organismes, les logiciels informatiques sont devenus de plus en plus une ressource stratégique omniprésente, permettant l'informatisation des processus et l'amélioration de la prise de décision stratégique des gestionnaires. Toutefois, pour garder leurs avantages compétitifs, les organismes exigent de plus en plus des logiciels de très bonne qualité, dans des délais très courts, mais avec un budget limité défiant toute compétition. Cependant le développement d'un logiciel de qualité n'est pas un objectif facile à atteindre, notamment à cause de l'ambigüité que comporte la définition de la qualité logicielle, qui est perçue différemment selon plusieurs perspectives (client, utilisateurs, spécialiste TI…etc.).

Pour cela, il est très important afin d'atteindre le niveau de qualité attendue, de mettre l'accent sur la qualité dès la première phase du cycle de développement logiciel. Cela est possible grâce à des activités d'assurance et de contrôle de qualité, qui offrent l'avantage d'évaluer et d'améliorer la qualité logicielle de façon efficace, efficiente et rationnelle, sans avoir besoin d'investir plus ou moins dans la qualité pour assurer la satisfaction des parties prenantes.

Donc, afin de bien maîtriser le concept d'évaluation de la qualité logicielle. Le présent chapitre présente une revue de l'état de l'art portant sur :

1. L'importance de l'évaluation de la qualité logicielle;
2. Les perspectives de la qualité logicielle;
3. Les critères de la qualité logicielle;
4. Les méthodes d'évaluation de la qualité logicielle.

1.2. L’évaluation de la qualité logicielle

« L’augmentation des coûts de détection et de prévention mène à une diminution des coûts liés aux défaillances (internes ou externes) et vice versa : une diminution des coûts en matière de détection et de prévention mène à une augmentation des coûts liés aux défaillances » (Alain April et Claude Y. Laporte, 2011, p. 64). Cette relation s’explique principalement par la valeur ajoutée que peuvent apporter les activités d’évaluation et de détection des erreurs, qui visent principalement à découvrir l’état actuel de la qualité du logiciel, tout en identifiant ses défauts susceptibles de causer des défaillances dans l’environnement de production. Cela permet de corriger tous les défauts durant le cycle de développement avant même que le logiciel soit livré au client.

Il est donc important de noter que l’évaluation de la qualité logicielle n’est pas une activité optionnelle qui génère des coûts supplémentaires au projet. Au contraire l'idée sous-jacente est de pouvoir réduire le coût des projets de développement logiciel, en mettant l’accent sur la qualité du logiciel tout au long de son cycle de développement. Cela permet d’atteindre le niveau de qualité attendu du premier coup, au lieu de consacrer un budget plus élevé pour couvrir tous les coûts engendrés par les défaillances causées par des défauts non détectés dans le processus de développement (par exemple, le coût du support technique, le coût de redéploiement, le coût de litige avec le client…etc.). Car il est toujours moins cher de corriger les erreurs en interne (c.-à-d. durant la phase de développement) que de le faire après la livraison au client.

En effet, la qualité dans un processus de développement ne se limite pas seulement à la qualité du produit final. Car dans certains domaines d’affaires tels que l’aérospatial et le militaire, un logiciel ne peut être considéré de bonne qualité que s’il a été développé dans le cadre d’un processus de développement normalisé qui a fait l’objet d’une évaluation et une amélioration continue (tel que, la norme ISO 29110). Toutefois, dans le présent travail nous allons nous concentrer seulement sur la qualité du produit final qui est le logiciel.

Selon la norme ISO 12207, l'évaluation se définit comme étant « la détermination systématique de la mesure dans laquelle une entité répond aux critères spécifiés ». Pour cela, il est nécessaire afin de déterminer si oui ou non un logiciel est de bonne qualité, d'évaluer sa qualité en fonction de ses critères de qualité mesurables, établis au début du projet. Ces critères de qualité sont généralement analysés par l'analyste d'affaires en termes de faisabilité et d'impacts durant la phase d'analyse et de spécification des exigences, et spécifiés explicitement dans le document de spécification des exigences (SRS) sous forme d'exigences non fonctionnelles, afin d'être implémentés et évaluer tout au long du cycle de développement du logiciel.

1.3. Les perspectives de la qualité logicielle

La qualité logicielle et une notion abstraite perçue différemment selon plusieurs perspectives. Ainsi « La mesure de la qualité se traduit souvent par la mesure d'une perception de la qualité » (Alain April et Claude Y. Laporte, 2011). Cette relation décrit la dépendance étroite entre la qualité logicielle et la perception de l'entité qui va la mesurer. Par exemple un logiciel de bonne qualité du point de vue ingénieur logiciel, n'est pas forcément un logiciel de bonne qualité du point de vue client ou utilisateur, et vice versa. Malheureusement dans l'industrie du logiciel, les perspectives de la qualité varient d'un projet à l'autre. Par conséquent, il est nécessaire d'identifier au début de chaque projet toutes les perspectives pertinentes, afin de s'intéresser à leurs critères de qualité associée, tout au long du cycle de développement du logiciel.

Afin de mieux appréhender l'importance des perspectives de la qualité, dans le processus d'évaluation de la qualité logicielle, plusieurs chercheurs (Garvin, D. A, 1988; Kitchenham, B., & Pfleeger, S, 1996) ont publié des exemples de perspectives les plus répandues, à savoir:

- **La perspective transcendantale de la qualité** : cette perspective définit la qualité comme étant un point de vue personnel, établi par un intervenant du projet suite à l'utilisation du logiciel. Malheureusement cette perspective représente un point de vue individuel et subjectif de l'intervenant, qui est difficile à satisfaire. Notamment parce

qu'il n'est pas établi au début du projet. En outre il est très difficile de satisfaire toutes les perspectives transcendantales dans un projet avec beaucoup d'intervenants, car chacun risque d'avoir sa propre perception de la qualité du logiciel.

- **La perspective de l'utilisateur** : tout comme la perspective transcendantale de la qualité. Cette perspective s'inscrit aussi dans le cadre du point personnel et reflète l'attente individuelle de chaque utilisateur du logiciel, c'est la raison pour laquelle il est important d'identifier toutes les classes d'utilisateurs au début de chaque projet, afin d'énumérer toutes leurs attentes, et d'essayer de les satisfaire durant le cycle de développement. Dans le cas d'un projet de grande envergure avec plusieurs utilisateurs, une solution serait d'identifier un utilisateur « pilote » ou « prime » pour chaque classe d'utilisateur, afin de représenter et agir au nom des utilisateurs de sa classe.

- **La perspective de la fabrication** : cette perspective vise à ce que le logiciel soit conforme aux exigences et aux contraintes spécifiées dans les spécifications. Il est donc important pour cette perspective que le SRS soit complet et à jour, en termes d'exigences et de contraintes spécifiées par les clients, les utilisateurs, ou par toute autre entité nécessaire, telle que les normes, les lois ou les standards.

- **La perspective de la qualité du produit** : cette perspective met l'accent sur le point de vue interne du logiciel, notamment l'architecture, la conception et la programmation du code source. Il est donc important afin de satisfaire les critères de cette perspective, de s'intéresser à l'aspect technique du logiciel et à la façon dont le logiciel est conçu. Pour se faire, il faut investir plus dans la phase d'architecture, et de suivre les recommandations, les standards et les meilleures pratiques du génie logiciel.

- **La perspective de la qualité comme valeur** : Cette perspective s'intéresse à la notion du coût et la valeur ajoutée de chaque composant ou fonctionnalité du logiciel. Ainsi, un logiciel de bonne qualité selon cette perspective décrit un logiciel qui inclut seulement les fonctionnalités ayant un retour sur investissement pour le client.

Il est très important de noter l'existence d'une interaction négative entre quelques critères associés aux perspectives citées précédemment. Par exemple, plus le logiciel est de bonne qualité du point de vue « **qualité du produit** », moins il est de qualité du point de vue « **qualité comme valeur** ». Car l'utilisation d'une norme tell que l'ISO 29110 qui vise à développer un logiciel de bonne qualité du point de vue génie logiciel, entraine la création de plusieurs documents (tel que, le document d'architecture et le document de procédures et cas de tests) et l'exécution de plusieurs activités (tel que, les revues, les vérifications, les validations) qui ne créent pas de la valeur pour le client, car ils sont des éléments internes au processus de développement du logiciel, et vice versa : un logiciel de bonne qualité du point de vue « **qualité comme valeur** » est un logiciel qui a été développé dans un délai très court, et avec un budget limité défiant toute.

Toutefois, malgré que les parties prenantes dans un projet de développement logiciel aient le privilège de choisir les perspectives de qualité les plus pertinentes à satisfaire, il ne faut jamais omettre la perspective interne du logiciel. Car, malheureusement dans l'industrie du logiciel, les projets qui ne s'intéressent pas à la qualité interne du produit, aboutissent toujours à la réalisation des logiciels avec beaucoup de défauts, qui se transforment en défaillances une fois le logiciel est placé dans son environnement de production.

1.4. Les critères de la qualité logicielle

Les exigences fonctionnelles dans un projet de développement logiciel décrivent de manières exhaustives toutes les fonctionnalités que le logiciel doit offrir, afin de satisfaire les besoins des clients-utilisateurs. Ces exigences sont généralement exprimées au début de chaque projet sous forme de besoins « User stories », ensuite spécifiées sous forme d'exigences fonctionnelles dans le SRS, avant d'être implémentées dans le logiciel par l'équipe de développement. Cependant, malgré que les critères de qualité n'offrent pas une fonctionnalité du point de vue utilisation, ils nécessitent aussi des efforts de développement, que le gestionnaire de projet doit prendre en considération dans sa planification de projet,

l'architecte dans son architecture, les développeurs dans leur programmation, ainsi que les testeurs dans leurs différents niveaux de tests.

Toutefois, les critères de qualité exprimés (par exemple, la maintenabilité, la fiabilité, la compatibilité, l'efficacité, la portabilité et l'efficacité du rendement et la facilité d'utilisation) sont souvent très abstraits pour être implémenté et mesuré dans le cadre d'une évaluation de la qualité. C'est la raison pour laquelle elles sont décomposées en plusieurs sous-caractéristiques plus élémentaires facilitant leurs développements et leurs évaluations. Par exemple, le critère de « maintenabilité » d'un logiciel, représenté par sa « modularité, sa réutilisabilité, sa modifiabilité ainsi que sa testabilité ». Ces sous-caractéristiques peuvent aussi être exprimées par des attributs encore plus élémentaires. Par exemple la notion de « modificabilité » peut aussi être mesurée par le biais de plusieurs attributs de qualité tels que la taille du code source, sa complexité ainsi que la densité des commentaires.

Pour faciliter cette décomposition. Les chercheurs ont proposé plusieurs modèles de qualité, que les chercheurs Alain April et Claude Y. Laporte ont citée dans leur livre (l'assurance qualité logicielle, 2011) à savoir :

- Le modèle McCall, Richards et Walter;
- Le modèle Evans et Marciniak;
- Le modèle Dromey;
- Le modèle IEEE 1061;
- Le modèle ISO 9126 et ISO 25000.

Le tableau suivant illustre un exemple de modèle de qualité (ISO 25000), qui décrit plusieurs sous-caractéristiques regroupées selon huit caractéristiques :

Caractéristique de qualité	Sous-caractéristique de qualité
Efficacité du rendement	- Comportement vis-à-vis du temps - Utilisation des ressources
Aptitude fonctionnelle	- Exhaustivité fonctionnelle - Exactitude fonctionnelle

	- Pertinence fonctionnelle
Compatibilité	- Coexistence - Interopérabilité
Facilité d'utilisation	- Pertinence - Facilité d'apprentissage - Opérabilité - Protection de l'utilisateur à faire une erreur - Interface utilisateur esthétique - Accessibilité
Fiabilité	- Maturité - Disponibilité - Tolérance aux pannes - Rétablissement
Sécurité	- Confidentialité - Intégrité - Non-réfutation - Responsabilisation - Authenticité
Maintenabilité	- Modularité - Réutilisabilité - Modifiabilité - Testabilité
Portabilité	- Adaptabilité - Installabilité - Remplaçabilité

Tableau 1.1 Le modèle de qualité proposé par la norme ISO 25000

Selon (Alain April et Claude Y. Laporte, 2011, p. 115) il est important de noter que certaines de ces exigences non fonctionnelles, peuvent avoir une interaction négative entre elles. Par exemple le développement d'un logiciel qui nécessite un niveau de sécurité très élevé requiert le développement et l'intégration de plusieurs algorithmes, mécanismes et règles de sécurité, qui peuvent nuire à l'utilisabilité du logiciel. Le tableau suivant illustre quelques exemples de ces interactions :

Attribut des exigences	Effet							
	Fonctionnalité	Efficacité	Utilisabilité	Fiabilité	Sécurité	Récupérabilité	Précision	Maintenabilité
Fonctionnalité	+	-	+	-	-	0	0	-
Efficacité	0	+/-	+	-	-	0	-	-
Utilisabilité	+	'+/-	+	+	0	+	+	0
Fiabilité	0	0	+	+	0	0	0	0
Sécurité	0	-	-	+	+	0	0	0
Récupérabilité	0	-	+	+	0	+	0	0
Précision	0	-	+	0	0	0	+	0
Maintenabilité	0	0	0	-	0	0	0	+

Tableau 1.2 Tableau d'interaction entre les caractéristiques de la qualité logicielle (Alexander Egyed, Paul Grünbacher, 2004)

1.5. Les méthodes d'évaluation de la qualité logicielle

L'évaluation de la qualité logicielle est un domaine qui a connu beaucoup de succès ces dernières années, notamment après l'apparition des notations modernes telles que l'intégration continue des logiciels, ainsi que le suivi des indicateurs de performance des praticiens du génie logiciel en fonction de la qualité et leurs codes sources. Mais malgré ce progrès, il n'y a toujours pas d'accord clair sur la façon la plus fiable et efficace dont l'évaluation de la qualité logicielle doit être menée. Ceci peut s'expliquer par les deux contraintes suivantes :

- D'abord, les erreurs sont injectées continuellement durant le processus de développement du logiciel, donc un processus d'évaluation efficace est celui qui peut supporter l'évaluation de n'importe quelle version du logiciel (c.-à-d. même les versions non terminées).
- La deuxième contrainte est liée à la particularité de chaque projet. En effet chaque projet de développement logiciel est caractérisé par ses propres circonstances telles que le modèle d'affaires adopté, les contraintes techniques et budgétaires, la méthodologie de travail utilisée…etc.

Pour cela, il ne faut pas espérer l'apparition future d'un guide d'évaluation de la qualité logicielle applicable à tous les projets de développement, quels que soient leurs contextes.

Toutefois, plusieurs travaux de recherche [5] [6] [7] [8] [9] [12] ont proposé des méthodes d'évaluation de la qualité logicielle telle que :

- L'évaluation de la qualité basée sur le calcul des métriques;
- L'évaluation de la qualité basée sur la visualisation du code source;
- L'évaluation de la qualité basée sur l'avis d'expert;
- L'évaluation de la qualité basée sur les tests.

1.5.1. Évaluation de la qualité basée sur calcul des métriques

Dans l'industrie du logiciel, il est toujours très difficile d'exprimer le degré de conformité d'un logiciel informatique par rapport à ses critères de qualité, et de prouver objectivement sa capacité à satisfaire les attentes de ses parties prenantes. Cependant, la méthode décrite dans le cadre de cette section, est une méthode d'évaluation quantitative, qui vise par le biais d'une analyse structurelle du code source et le calcul de plusieurs métriques (définies selon des conventions standards) de déduire des indicateurs liés à la qualité du logiciel (du point de vue interne) et de prédire indirectement les attributs de qualité externes du logiciel.

La mesure de la qualité structurelle du code source se base principalement sur une multitude de métriques, telles que le nombre de lignes de codes, le couplage afférent/efférent, la densité des commentaires et la complexité cyclomatique. En effet, ces métriques permettent d'évaluer la qualité logicielle indirectement, via la mesure des attributs de qualité qui correspondent à des sous-caractéristiques (tel que, la maintenabilité, la fiabilité et la sécurité du logiciel) ou à des caractéristiques (tel que, la maturité, la disponibilité, la tolérance aux pannes et l'interopérabilité).

Selon (Pierre Mengal 2013), ces métriques sont généralement classées en deux grandes catégories :

- **Les métriques primitives** : cette catégorie englobe l'ensemble des métriques permettant de mesurer les propriétés de base du code source, il s'agit de métriques telles que:
    - le nombre de lignes de code;
    - la complexité cyclomatique;
    - la profondeur d'héritage;
    - le nombre de sous-classes;
    - le nombre de méthodes;
    - le nombre de méthodes héritées.

- **Les métriques de conception**. Ce type de métrique permet de déterminer si le code source du logiciel est conforme aux principes de conception préalablement définis. Par exemple :
    - les métriques de couplage ;
    - les métriques de cohésion de classes;
    - les métriques d'architecture de packages comme les métriques de couplage afférent et efférent.

L'évaluation de la qualité via le calcul des métriques est une méthode très utilisée, car elle permet d'obtenir des indicateurs sur la qualité de l'architecture et le code source du logiciel. Cette méthode permet aussi de comparer la qualité de plusieurs logiciels, via la comparaison numérique de leurs résultats de métriques.

Le calcul de ces métriques peut en partie être réalisé automatiquement par une grande variété d'outils de mesure proposés sur le marché (par exemple, JDepend, NDepend, Eclipse Metrics, Crap4J). Ces outils peuvent être utilisés manuellement par les programmeurs afin d'évaluer la qualité de leurs codes sources durant le processus de développement du logiciel, ou automatiquement dans le cadre d'un processus d'évaluation et d'intégration continues. Toutefois, il est important de bien choisir les métriques à utiliser, car la pertinence de chacune de ces métriques peut varier d'un logiciel à un autre, dépendamment de plusieurs

facteurs, tels que la taille du projet, sa complexité ainsi que le langage de programmation et les technologies utilisées.

Bien que les métriques obtenues à l'aide de l'analyse structurelle du code source soient qualifiées d'utiles pour déterminer la qualité d'un logiciel, leur usage pourrait avoir une certaine limitation. Car étant donné la grande diversité des types de logiciels et le choix technologique et architectural utilisé dans les logiciels modernes, l'utilisation de certaines métriques peut facilement devenir un moyen subjectif pour valoriser ou dévaloriser le logiciel. Par exemple l'indicateur de densité des commentaires dans le code source, peut être un signe de mauvaise qualité, alors qu'il convient de prendre en considération d'autres facteurs pour juger de la pertinence de ce résultat, car l'utilisation des noms significatifs pour nommer les classes, les méthodes, les variables et les évènements dans un logiciel peut justifier la densité faible des commentaires. En outre la profondeur de l'arborescence d'héritage des classes ne peut en aucun cas être un critère absolu de la qualité du code source, car dans quelques projets de développement logiciel, la profondeur élevée de l'arborescence d'héritage peut être justifiée par l'utilisation d'une architecture logicielle qui favorise la réutilisation des classes et les composantes du logiciel.

1.5.2. Évaluation de la qualité basée sur la visualisation du code source

L'évaluation de la qualité logicielle, est l'une des tâches les plus importantes et cruciale du processus de développement, vue qu'elle permet de déterminer si oui ou non le logiciel est en conformité avec les attentes et les spécifications. Toutefois cette tâche n'est pas toujours réalisée par les mêmes personnes ayant travaillé sur le projet. Malheureusement, cela peut facilement influencer l'objectivité des résultats obtenus, notamment si la documentation du projet n'a pas été maintenue à jour (par exemple le document de spécification des exigences et le document d'architecture du logiciel).

Donc, afin de mener une évaluation objective de la qualité du logiciel, l'évaluateur doit d'abord effectuer une rétro-ingénierie du logiciel à évaluer, dans le but d'illustrer

graphiquement l'anatomie du logiciel du point de vue structurel (architecture, conception et code source du logiciel) et du point de vue dynamique (comportement et interaction entre les composantes du logiciel). Pour ce faire l'évaluateur peut choisir l'une des deux méthodes suivantes :

- **La rétro-ingénierie manuelle du logiciel :** cette méthode consiste à visualiser manuellement le contenu des fichiers du code source du logiciel, dans l'objectif de comprendre son aspect statique (modélisation des classes, méthodes, propriétés, packages,…etc.), et de visualiser manuellement les interfaces du logiciel afin de comprendre son aspect dynamique (comportement).

- **La rétro-ingénierie automatique via un outil spécialisé :** cette méthode implique l'utilisation d'un outil spécialisé dans la rétro-ingénierie des logiciels (par exemple, Eclipse et Omondo) afin de générer automatiquement des modèles (par exemple UML) qui illustrent graphiquement l'aspect structurel du logiciel (par exemple, le diagramme de class, le diagramme d'objets, le diagramme de composants, et le diagramme de package), ainsi que l'aspect dynamique et comportemental du logiciel (par exemple, le diagramme de cas d'utilisation, diagramme de séquence et le diagramme d'activité).

L'illustration graphique de l'aspect statique et dynamique d'un logiciel sous forme de modèles, offre à l'évaluateur une vue globale de toutes les composantes du logiciel. Chose qui lui permet de vérifier que toutes les fonctionnalisées demandées par le client ont été implémentées et de valider aussi qu'elles ont été développées de la bonne façon.

Malheureusement, l'utilisation de cette méthode n'est pas adaptée aux logiciels volumineux, mais elle peut être appliquée pour un échantillon représentatif de classes ou de packages. En outre, l'utilisation de cette méthode de visualisation peut être combinée avec une autre méthode telle que le calcul des métriques. Cela offre à l'évaluateur l'avantage de comparer les résultats obtenus dans chacune des méthodes afin de justifier, de renforcer ou de critiquer les résultats obtenus via le calcul des métriques.

1.5.3. Évaluation de la qualité basée sur l'avis d'expert

Comme il a été mentionné précédemment, la qualité logicielle est perçue différemment selon chaque perspective (client, utilisateurs, spécialiste TI…etc.). Pour cela, l'implication d'un expert (tel que, le client, l'utilisateur, et le spécialiste du TI) dans le processus d'évaluation de la qualité peut augmenter considérablement la pertinence des résultats obtenus. En effet, ces experts ont comme mission de vérifier et de valider constamment le logiciel, tout long de son cycle de développement. Par exemple, pour chaque fonctionnalité ou module développé le client doit donner son avis de point de vue valeur ajoutée, l'utilisateur doit donner son avis du point de vue utilisation, et finalement le spécialiste génie logiciel, doit mesurer continuellement la qualité du logiciel en cours de développement du point de vue interne.

Il est important de noter que la perception de la qualité et l'avis des experts impliqués dans l'évaluation de la qualité du logiciel durant son cycle de développement, peuvent changer au fil du temps, notamment à cause des demandes de changements et les régressions qui peuvent survenir à cause des nouvelles fonctionnalités développées. Toutefois, leurs points de vue représentent des bonnes références pour estimer la qualité finale du logiciel.

Malheureusement, dans les projets de grande envergure, l'implication de toutes les parties prenantes dans le processus de l'évaluation de la qualité du logiciel durant son processus de développement, est une tâche impossible à faire. Cependant, il est possible de nommer une ou plusieurs personnes « Pilotes » qui vont représenter les personnes ayant la même perception de la qualité.

1.5.4. Évaluation de la qualité basée sur les tests

L'évaluation de la qualité logicielle via les tests, est une technique très utilisée dans le monde du génie logiciel, car elle permet de vérifier que toutes fonctionnalités demandées par le client du point de vue fonctionnel et non fonctionnel ont été implémentées dans le logiciel et de valider le résultat de chacune d'entre elles.

Les tests du logiciel sont effectués selon plusieurs niveaux (Wikipédia, 2016):

- **Test de composants (test unitaire) :** Ce type de test est généralement fait par le programmeur, et il a comme but de vérifier le bon fonctionnement d'une portion précise du logiciel. Par exemple, l'exécution d'une méthode de classe avec plusieurs combinaisons de paramètres, afin de vérifier et de valider son bon fonctionnement. Ce type de test est très couteux, notamment dans les logiciels volumineux. Pour cela il est possible d'appliquer ce type de test sur seulement un échantillon bien précis du code source, choisit en fonction des facteurs pertinents (par exemple, la criticité).

- **Test d'Intégration** : Contrairement au test unitaire, ce type de test vise à tester le fonctionnement de chaque module, en faisant abstraction des portions de code source qui le constituent.

- **Test système (test fonctionnel)** : Ce type de test « boite noire » a pour objectif d'évaluer la conformité des fonctionnalités développées aux exigences fonctionnelles et non fonctionnelles spécifiées dans le document des exigences du logiciel.

- **Test d'acceptation** : Le test d'acceptation est un test plus formel effectué à la fin de projet, et généralement réalisé par le client-utilisateur, dans le but de s'assurer que le logiciel final répond à toutes les attentes.

Toutefois, vu l'importance et la criticité du test d'acceptation dans le projet, étant donné qu'il implique la présence de toutes les parties prenantes incluant le client, pour approuver l'acceptation du logiciel. Il est très fortement recommandé d'investir le plus dans les trois

premiers tests, afin d'éliminer ou de réduire au maximum les défauts que le test d'acceptation peut détecter.

1.6. Conclusion

Ce chapitre nous a permis de mettre en évidence la pertinence de l'évaluation de la qualité logicielle, ainsi que les différentes perspectives, et critères de la qualité, et ses méthodes d'évaluations les plus répondues. Le prochain chapitre présente la démarche d'évaluation et d'amélioration du logiciel PACIQ.

CHAPITRE 2

ÉVALUATION ET AMÉLIORATION DE LA QUALITÉ DU LOGICIEL PACIQ

2.1. Introduction

Le présent chapitre vise à décrire de manière exhaustive et détaillée, la démarche d'évaluation de la qualité du logiciel PACIQ, ainsi que les améliorations apportées sur le code source du point de vue interne et externe.

2.2. Démarche et méthodologie de travail

PACIQ est un logiciel qui a été développé initialement en 2014 par plusieurs étudiants de l'ÉTS. Par la suite, plusieurs modifications ont été apportées sur son code source, suite à des demandes de changement et à des défauts identifiés par la cliente durant la phase de test d'acceptation. Malheureusement la documentation fonctionnelle et technique du logiciel n'avait pas été maintenue à jour, notamment en termes de critères de qualité, qui n'avaient pas été spécifiés sous forme d'exigences non-fonctionnelles dans le document de SRS initial.

Conséquemment, il a été nécessaire, d'évaluer le logiciel et d'identifier des améliorations tout en impliquant la cliente dans ces activités. Cette approche permet d'éviter toute difficulté potentielle qui peut survenir à cause d'une mauvaise compréhension des attentes concernant la qualité du logiciel, ou la présence d'une ambigüité causée par l'incohérence de la documentation du projet par rapport aux fonctionnalités implémentées dans le logiciel PACIQ.

2.2.1. Utilisation de la norme ISO 25000

L'évaluation de la qualité logicielle effectuée dans le cadre de ce projet, est une évaluation multi-perspective qui couvre les deux perspectives suivantes :

- **Le point de vue externe** : ce point de vue décrit la perspective transcendantale de la qualité qui reflète le point de vue personnel des clients et des utilisateurs du logiciel PACIQ, ainsi que sa conformité aux spécifications écrites dans le SRS;
- **Le point de vue interne** : ce point de vue s'intéresse à l'aspect structurel et technique de PACIQ, notamment son architecture, sa conception et la programmation de son code source.

Afin d'effectuer une évaluation complète et une amélioration ciblée de la qualité du logiciel PACIQ, tout en remédiant aux problèmes dérivés de l'absence de critères qualité dans le document de spécification des exigences, la norme ISO 25000 a été consultée car elle offre un guide d'accompagnement, pour l'évaluation et la définition des critères de qualité d'un logiciel. Les étapes recommandées sont, comme suit:

- **Étape de définition des exigences de qualité :** Avant d'entreprendre toute action, il a semblé nécessaire d'organiser plusieurs séances d'élicitation d'exigences avec la cliente, dans le but de mieux comprendre et vérifier sa perception de la qualité du logiciel PACIQ en ce moment.
- **Étape de choix du modèle de qualité :** durant cette étape, le modèle de qualité proposé par la norme ISO 25000 a été utilisé pour la démarche d'évaluation de la qualité. Ce dernier décrit plusieurs caractéristiques et sous-caractéristiques de la qualité, couvrant toutes les perspectives internes et externes citées précédemment, de la façon suivante (voir tableau 2.1):

Caractéristique de qualité	Sous-caractéristique de qualité
Efficacité du rendement	- Comportement vis-à-vis du temps - Utilisation des ressources
Aptitude fonctionnelle	- Exhaustivité fonctionnelle - Exactitude fonctionnelle - Pertinence fonctionnelle
Compatibilité	- Coexistence - Interopérabilité
Facilité d'utilisation	- Pertinence - Facilité d'apprentissage

	- Opérabilité - Protection de l'utilisateur à faire une erreur - Interface utilisateur esthétique - Accessibilité
Fiabilité	- Maturité - Disponibilité - Tolérance aux pannes - Rétablissement
Sécurité	- Confidentialité - Intégrité - Non-réfutation - Responsabilisation - Authenticité
Maintenabilité	- Modularité - Réutilisabilité - Modifiabilité - Testabilité
Portabilité	- Adaptabilité - Installabilité - Remplaçabilité

Tableau 2.1 Le modèle de qualité proposé par la norme ISO 25000

- **Étape de détermination des métriques et de conduite des évaluations :** les métriques utilisées pour évaluer la qualité du logiciel PACIQ sont des métriques liées aux méthodes d'évaluation de la qualité. Ainsi, cette approche de mesure est basée principalement sur l'avis personnel de chacun des experts impliqués, comme suit:
    - **Évaluation basée sur l'avis d'experts :** cette méthode nous a permet de mesurer la conformité du logiciel aux attentes, via l'organisation de plusieurs séances de de revues « Walk through» avec le client-utilisateur (expert du point de vue externe du logiciel) ce dernier a donné son avis sur toutes les fonctionnalités déjà développées, ainsi que sur toutes les améliorations apportées à la fin du projet. La deuxième revue effectuée dans le cadre de cette méthode, est une revue « Desk Check » que j'ai réalisée moi-même en tant qu'expert technique, cette revue m'a permet d'exprimer mon avis sur le logiciel du point de vue interne.

- **Évaluation basée sur les tests :** Cette méthode a été utilisée afin de vérifier l'implémentation de chaque fonctionnalité demandée par le client, et de valider les résultats de leurs exécutions. Ces tests ont été effectués selon deux niveaux, d'abord les tests d'intégration réalisés par moi-même dans l'environnement de développement, ensuite les tests fonctionnels réalisés par le client dans l'environnement de Test.

2.2.2. Utilisation des listes de vérification

Afin de mieux structurer et simplifier l'évaluation de la qualité logicielle de PACIQ, des listes de vérifications « Checklists » ont été conçues et ont permis d'évaluer l'ensemble du logiciel avec une même approche:

- Liste de vérification pour l'interface utilisateur (annexe I);
- Liste de vérification pour l'architecture et le code source du logiciel (annexe II);
- Liste de vérification pour la base de données (annexe III);
- Liste de vérification pour la sécurité du logiciel (annexe IV);

2.3. Résultat d'évaluation de la qualité du logiciel PACIQ

2.3.1. Efficacité du rendement

a. Comportement vis-à-vis du temps

PACIQ-EVAL-1 L'utilisation du SQL dynamique

Le logiciel PACIQ utilise des instructions « SQL dynamiques » afin d'interroger sa base de données. Cela peut malheureusement peut causer une dégradation de la performance du logiciel. Contrairement aux procédures stockées, le SGBDR « Microsoft SQL Server » recompile les requêtes SQL dynamiquement à chaque appel et génère à chaque fois un nouveau plan d'exécution.

Afin de remédier à ce problème, il est recommandé de remplacer l'utilisation du « SQL dynamique » par des procédures stockées.

PACIQ-EVAL-2 L'absence de l'utilisation des Index SQL

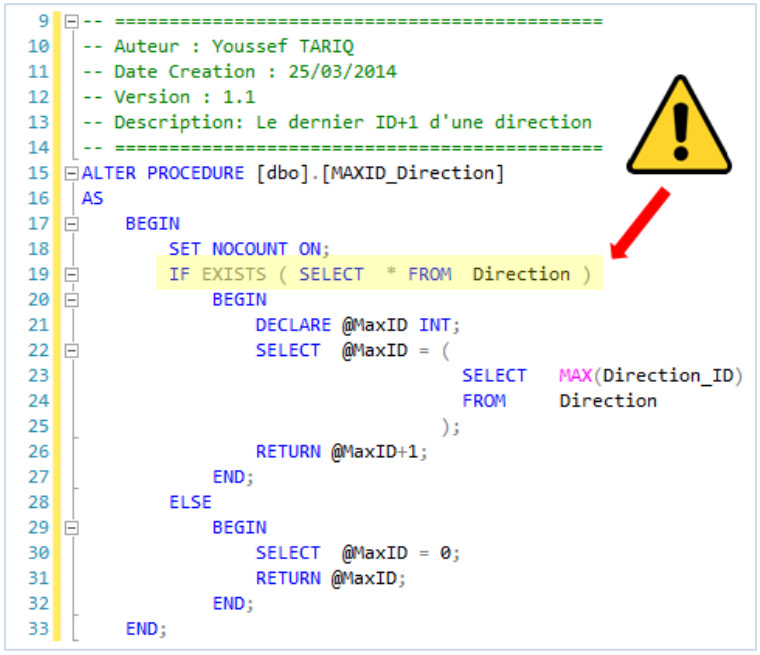
Aucun index n'est créé sur les colonnes des tables de la base de données du logiciel PACIQ. Cela peut augmenter significativement le temps d'exécution des requêtes.

Donc, pour assurer une meilleure performance durant l'exécution des requêtes, il est important de créer des index sur les colonnes les plus utilisées dans les jointures.

PACIQ-EVAL-3 Utilisation de l'instruction « SELECT » sans clause « WHERE »

Dans le but de vérifier si une table contient des enregistrements, quelques procédures stockées incluent des instructions qui interrogent tout le contenu de la table inutilement, ce qui peut diminuer considérablement la performance de la base de données, notamment si cette table contient beaucoup d'enregistrements.

Il est recommandé d'éliminer ce type d'instructions de toutes les procédures stockées de la base de données. Par exemple : la procédure stockée suivante vise à récupérer la valeur maximale de la colonne « **Direction_ID »** de la table « **Direction »,** pour l'incrémenter et l'utiliser comme clé primaire pour le prochain enregistrement**.** Donc afin d'éliminer l'utilisation de l'instruction « **SELECT * FROM Direction** » utilisée pour vérifier que la table « **Direction** » contient des enregistrements :



```
-- ============================================
-- Auteur : Youssef TARIQ
-- Date Creation : 25/03/2014
-- Version : 1.1
-- Description: Le dernier ID+1 d'une direction
-- ============================================
ALTER PROCEDURE [dbo].[MAXID_Direction]
AS
    BEGIN
        SET NOCOUNT ON;
        IF EXISTS ( SELECT  * FROM  Direction )
            BEGIN
                DECLARE @MaxID INT;
                SELECT  @MaxID = (
                                  SELECT   MAX(Direction_ID)
                                  FROM     Direction
                                 );
                RETURN @MaxID+1;
            END;
        ELSE
            BEGIN
                SELECT  @MaxID = 0;
                RETURN @MaxID;
            END;
    END;
```

Figure 2.1 Exemple d'utilisation de l'instruction « SELECT » sans clause « WHERE »

Il faut simplement changer le code de la procédure stockée par les instructions suivantes :

```
DECLARE @ID AS INT

SELECT TOP 1 @ID = MAX(Direction_ID)
FROM    Direction

--Si la table Direction ne contient aucune ligne nous retourons 0
SELECT CASE WHEN ISNULL(@ID,0) > 0 THEN @ID + 1 ELSE 0 END
```

Figure 2.2 Solution pour le problème d'utilisation de l'instruction « SELECT » sans clause « WHERE »

PACIQ-EVAL-4 PACIQ est configuré pour être déployé seulement en mode « Debug »

Le site web PACIQ est configuré dans le but d'être déployé seulement en mode « Debug ». Cela peut causer une dégradation de la performance du site web, car cette configuration doit être utilisée seulement dans l'environnement de développement pour déboguer l'application.

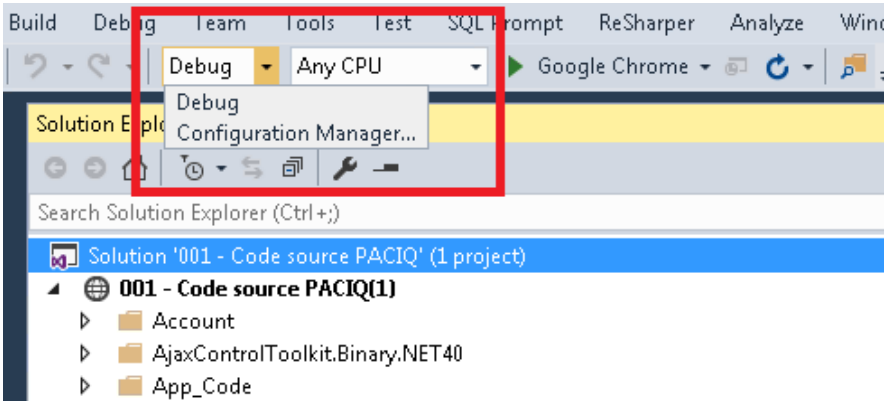


Figure 2.3 Utilisation du mode « Debug » pour le déploiement du site web PACIQ

Il est recommandé de créer un nouveau mode de compilation par défaut du projet PACIQ appelé « **Release** » et de passer la variable «**Debug** » à « **False** » dans le fichier de configuration (Web.config) du site web, cela permet d'indiquer que le site est désormais en production ou test.

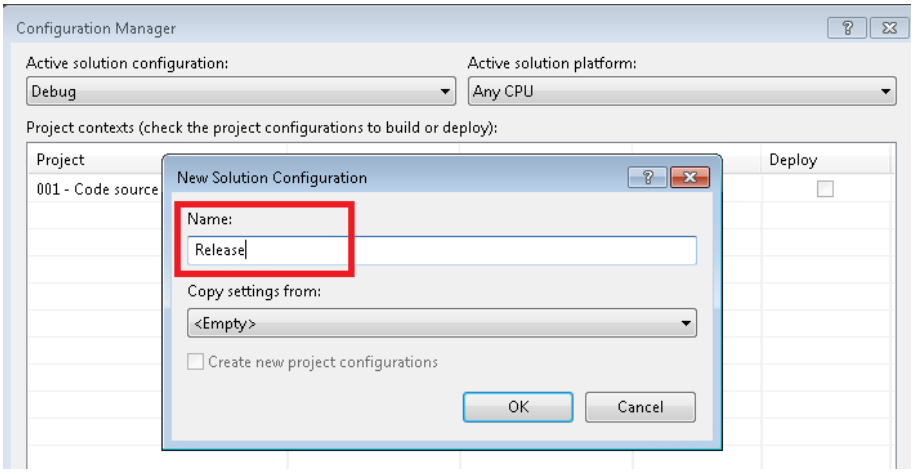


Figure 2.4 Solution pour le problème d'utilisation du mode « Debug » pour le déploiement du site web PACIQ

b. Utilisation des ressources

PACIQ-EVAL-5 Ouverture de plusieurs connexions SQL pour interroger la base de données

L'ouverture de plusieurs connexions pour interroger la base de données durant le cycle d'utilisation du logiciel PACIQ, peut mener à une augmentation du délai d'attente pour obtenir une connexion du groupe de connexions de Microsoft SQL serveur, ou encore à une exception SQL de type « *Timeout expired. The timeout period elapsed prior to obtaining a connection from the pool* ».

Par conséquent, il est fortement recommandé :

- Utiliser le patron de conception « Singleton » pour utiliser une seule connexion par utilisateur durant le cycle d'utilisation de l'application.
- Ajouter le paramètre «Max Pool Size » au niveau de la chaîne de connexion de la base de données PACIQ, située dans le fichier de configuration du site web (Web.config), tout en lui attribuant une valeur supérieure à 100 (valeur par défaut du paramètre «Max Pool Size ») et inférieure au nombre maximal de connexions simultanées autorisées dans le groupe de connexion de Microsoft SQL 2008 R2.

PACIQ-EVAL-6 Les pages web du site web PACIQ sont volumineuses

L'utilisation de l'assistant de création des pages web, intégré à l'outil de développement « Microsoft Visual Studio 2010», a malheureusement mené à la création de plusieurs pages Web de tailles volumineuses. Ces pages prennent beaucoup de temps pour s'afficher sur les fureteurs de téléphones intelligents et consomment plus de bande passante qu'une page HTML optimisée.

Il est recommandé de passer en revue toutes les pages web (*.aspx) de PACIQ afin de supprimer les balises HTML générées inutilement.

2.3.2. Aptitude fonctionnelle

a. Exhaustivité fonctionnelle

PACIQ-EVAL-7 Plusieurs défauts fonctionnels ont été identifiés par le client

Suite à des tests fonctionnels réalisés par le client, plusieurs défauts ont été identifiés et enregistrés dans la plateforme de création de billets « Bug Net » comme suit :

Abderrahmane Elmoul | Logout

BugNET Issue Tracker

Issue Summary For Abderrahmane Elmoul

Home | Search | My Issues | Admin

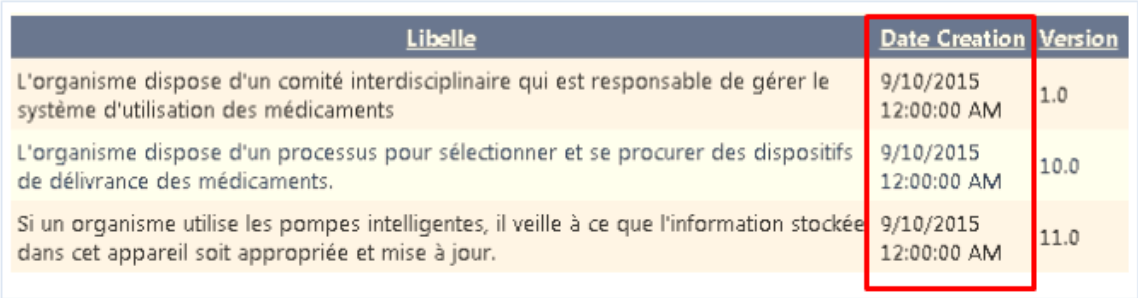
Id 1	Title	Project	Owner	Assigned	Status	Priority
PACIQ-7	Changer le "Planetree" par "Approche Humaniste" dans le tableau de bord global	PACIQ	Isabelle Olivier	Abderrahmane Elmoul		Normal
PACIQ-10	Enlever le logo d'Agrément Canada et placer le logo du CHU Sainte-Justine et de ETS seulement	PACIQ	Isabelle Olivier	Abderrahmane Elmoul		
PACIQ-11	Modifier la page accueil	PACIQ	Isabelle Olivier	Abderrahmane Elmoul		
PACIQ-13	Problème de modification des cahier de normes	PACIQ	Isabelle Olivier	Abderrahmane Elmoul		Normal
PACIQ-14	Données référence: normes: afficher les normes par programme accréditation et cahier de normes	PACIQ	Isabelle Olivier	Abderrahmane Elmoul		Normal
PACIQ-16	Ajuster les champs écriture dans donnée de référence critèere	PACIQ	Isabelle Olivier	Abderrahmane Elmoul		Normal
PACIQ-17	Ajouter une fonctionnalité pour retrouver rapidement les divers critères en lien avec les programme accréditation, cahier de normes, normes.	PACIQ	Isabelle Olivier	Abderrahmane Elmoul		Normal
PACIQ-18	relation entre les critères pouvoir partir d'un programme d'accréditation. pas seulement celui agrément canada	PACIQ	Administrator	Abderrahmane Elmoul		Normal
PACIQ-19	plan amélioration, création, zone de texte pour plan stratégique qui est plus long que la zone de visionnement	PACIQ	Isabelle Olivier	Abderrahmane Elmoul		
PACIQ-21	Demande pour ajouter des champs pour choisir deux autres responsable pour le même plan d'amélioration	PACIQ	Isabelle Olivier	Abderrahmane Elmoul		Normal
PACIQ-22	Céation de plusieurs activités d'amélioration pour un objectif.	PACIQ	Isabelle Olivier	Abderrahmane Elmoul		
PACIQ-23	Indicateur pouvoir inscrire un indicateur pas seulement un menu déroulant	PACIQ	Administrator	Abderrahmane Elmoul		
PACIQ-24	Aucun plan amélioration n'apparait dans la liste des plans amélioration. de plus comment allons nous faire la sélection des plans selon les directions services etc...	PACIQ	Isabelle Olivier	Abderrahmane Elmoul		
PACIQ-25	Aucun rapport ne fonctionne les données sont statique et ne changement jamais	PACIQ	Isabelle Olivier	Abderrahmane Elmoul		
PACIQ-26	Bug lors du changement du mot de passe pour un utilisateur	PACIQ	Isabelle Olivier	Abderrahmane Elmoul		

Figure 2.5 Liste des défauts fonctionnels identifiés par le client

b. Exactitude

PACIQ-EVAL-8 Les dates sont affichées et stockées dans un format invalide

Les dates sont affichées et stockées dans un format invalide, de manière à ce que la partie des heures est toujours égale à « 12:00:00 AM » (exemple : 9/10/2015 12:00:00 AM).



Libelle	Date Creation	Version
L'organisme dispose d'un comité interdisciplinaire qui est responsable de gérer le système d'utilisation des médicaments	9/10/2015 12:00:00 AM	1.0
L'organisme dispose d'un processus pour sélectionner et se procurer des dispositifs de délivrance des médicaments.	9/10/2015 12:00:00 AM	10.0
Si un organisme utilise les pompes intelligentes, il veille à ce que l'information stockée dans cet appareil soit appropriée et mise à jour.	9/10/2015 12:00:00 AM	11.0

Figure 2.6 Problème de stockage des dates dans un format invalide

Il est recommandé de :

- Formatter les dates sous la forme « YYYY-MM-dd ».
- Changer le type des variables et des colonnes de dates ayant un type « varchar » par le type « Date » ou « SmallDateTime ».

PACIQ-EVAL-9 Les clés primaires de quelques tables sont incrémentées manuellement

Afin d'ajouter un nouveau plan d'amélioration ou une direction dans la base de données, le logiciel PACIQ exécute deux procédures stockées (**dbo.MAXID_Plan_action** et **dbo.MAXID_Direction**) qui récupèrent la valeur maximale de la clé primaire de la table, qui sera par la suite incrémentée manuellement et utilisée comme clé primaire pour le prochain enregistrement. Cette approche peut engendrer des conflits lorsque deux ou plusieurs utilisateurs essaieront simultanément d'ajouter de nouveaux plans d'amélioration ou des directions.

Afin d'éviter le potentiel de conflits d'insertion, il est fortement recommandé d'utiliser l'incrémentation automatique de la clé primaire des tables, en attribuant la propriété (IDENTITY(1,1)) aux clés primaires des tables.

PACIQ-EVAL-10 Les messages du logiciel PACIQ sont parfois incorrects ou non pertinents

Le logiciel PACIQ affiche, dans quelques cas, des messages d'erreurs incorrects ou non pertinents. Par exemple : lorsqu'un utilisateur clique sur le bouton « modifier » ou « supprimer » un enregistrement dans la page « Indicateurs » sans sélectionner une ligne dans

la grille de données, le logiciel affiche le message incorrect « **Veuillez saisir un indicateur ! »**

Afin de remédier à ce problème, il est recommandé de tester toutes les fonctionnalités du logiciel dans le but d'améliorer la pertinence et l'exactitude des messages.

2.3.3. Compatibilité

a. Coexistence

PACIQ-EVAL-11 PACIQ peut coexister avec d'autres systèmes dans un environnement commun et partagé

L'utilisation d'une base de données « Microsoft SQL Server » afin de stocker les données ainsi que l'utilisation de la technologie « Microsoft ASP.NET » pour développer le code source du site Web du projet permet à l'application PACIQ de coexister avec d'autres applications Web dans le même serveur « IIS » et avec d'autres bases de données dans la même instance SQL. Ceci est une bonne pratique alors il n'y a pas de recommandation nécessaire.

2.3.4. Facilité d'utilisation

a. Facilité d'apprentissage

PACIQ-EVAL-12 Le logiciel PACIQ ne nécessite pas beaucoup d'efforts d'apprentissage

Grâce à la simplicité du logiciel PACIQ, une formation d'une demi-journée est suffisante pour qu'un utilisateur de l'hôpital Saint-Justine, puisse utiliser toutes ses fonctionnalités de manière efficace et efficiente dans le but d'atteindre ses objectifs métiers.

b. Protection de l'utilisateur à commettre une erreur

PACIQ-EVAL-13 PACIQ exécute toutes les opérations sans confirmation de l'utilisateur

Le logiciel PACIQ n'offre à ses utilisateurs aucun moyen pour confirmer ou annuler leurs opérations, notamment lors de la suppression des données (par exemple : la suppression des critères, des activités d'améliorations, des indicateurs…etc.).

Afin de remédier à ce problème, il est recommandé d'ajouter des boites de confirmation (c.-à-d. des Popups) pour offrir la possibilité aux utilisateurs de confirmer ou d'annuler leurs choix avant de procéder avec l'opération.

PACIQ-EVAL-14 Absence de la validation de toutes les données envoyées par l'utilisateur

Le logiciel PACIQ ne permet pas de valider toutes les données provenant de l'utilisateur. Par exemple : l'utilisateur peut créer un plan d'amélioration en choisissant une date de création dans le passé.

Il est donc recommandé d'ajouter des règles et des mécanismes de validation, côté client, (c.-à-d. dans le JavaScript) pour que chaque donnée envoyée par l'utilisateur soit validée.

PACIQ-EVAL-15 Suppression physique des enregistrements de la base de données

Le logiciel PACIQ supprime physiquement les enregistrements des tables de la base de données, ce qui ne permet pas de récupérer les données affectées et de rétablir l'état désiré en cas d'erreur.

Pour remédier à ce problème, il est recommandé d'ajouter deux colonnes au niveau de toutes les tables concernées par les opérations de suppression:

- StartDate de type SmallDateTime.
- EndDate de type SmallDateTime.

Ces deux colonnes seraient utilisées afin d'activer ou désactiver une ligne au lieu de la supprimer physiquement de la table afin de protéger l'utilisateur en cas d'erreur, de retracer l'historique des suppressions de données et de créer des rapports avec des informations intègres et cohérentes.

c. Interface utilisateur esthétique

PACIQ-EVAL-16 Le site web PACIQ n'offre pas des écrans adaptatifs (Responsive)

Le logiciel PACIQ n'est pas un site Web adaptatif qui peut offrir à ses utilisateurs un confort visuel durant son utilisation avec différents types de fureteurs et appareils mobiles, sans avoir besoin d'utiliser la fonctionnalité du zoom arrière/avant ou la barre de défilement.

La solution serait d'utiliser un cadriciel d'interface optimisé pour les appareils mobiles qui se base sur le langage HTML5 et CSS3 tel que : JQuery Mobile, Kendo Ui, Bootstrap ou Angular JS.

PACIQ-EVAL-17 Le choix des couleurs et des contrôles HTML

L'utilisation du style par défaut des pages « ASP.NET » fournit avec la version utilisée de l'outil de développement « Microsoft Visual Studio 2010 », mène à la création d'un site Web ayant une apparence très différente à celle du site Web officiel de l'hôpital Sainte-Justine, notamment en termes de choix des couleurs (c.-à-d. couleurs claires pour le texte et sombre pour l'arrière-plan) qui influencent la lisibilité textes ainsi que les types de contrôles HTML utilisés.

Afin de suivre le même standard de développement Web, il est recommandé de changer le fichier de style du site web PACIQ pour utiliser les mêmes couleurs du site Web de l'hôpital Sainte-Justine.

PACIQ-EVAL-18 La largeur des colonnes dans les grilles de données ne s'adapte pas au contenu

La figure suivante montre un exemple de problème présent dans toutes les grilles de données du site web PACIQ. Ce problème consiste à ce que la largeur des colonnes de la grille de données ne s'ajuste pas en fonction de la taille du contenu des cellules, mais plutôt au titre de la colonne qui se génère automatiquement à partir du nom de champ associé dans la base de données (Exemple Norme_ID, Cahier_Norme_ID… etc.).

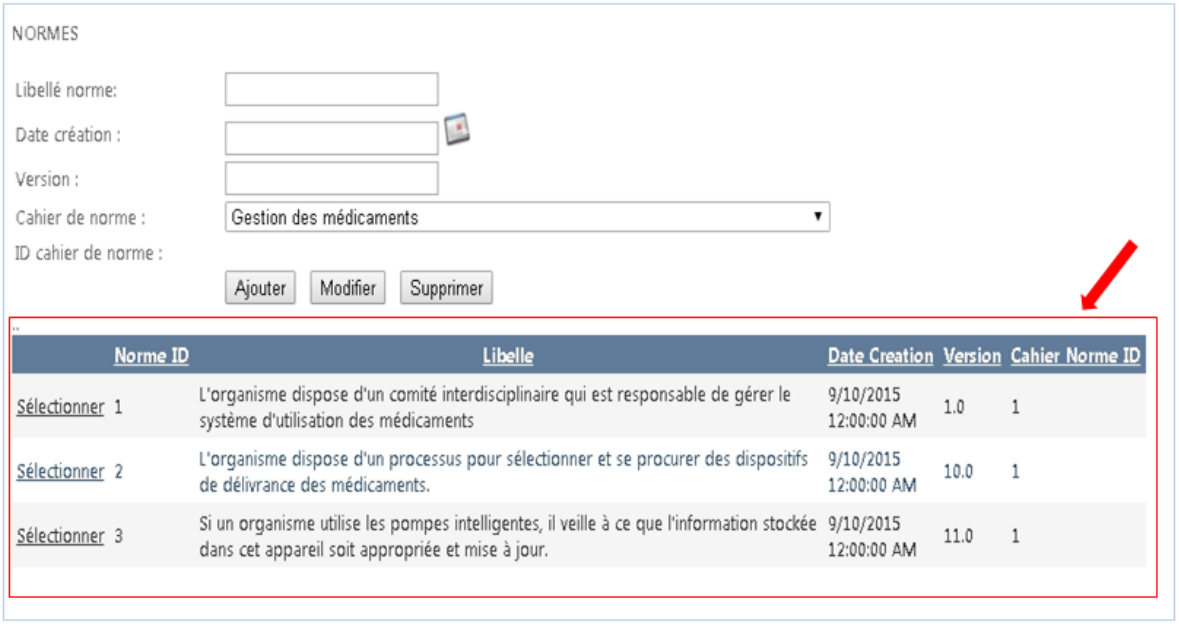


Figure 2.7 Problème de largeur des colonnes dans les grilles de données

Il est donc recommandé de :

- Utiliser les fichiers de ressources pour stocker tous les titres de colonnes et des champs de texte.
- Attribuer des titres réduits et des abréviations pour les colonnes, afin d'optimiser l'utilisation de l'espace dans les grilles de données.

PACIQ-EVAL-19 Les formulaires de création et de modification sont affichés en permanence dans les pages web

Les formulaires de création et de modification de données sont toujours affichés au niveau des pages de consultation de normes, de programme d'accréditation, de plans stratégiques, d'indicateurs… etc. sans que l'utilisateur choisisse une option pour modifier ou créer des données. Cela force l'utilisateur à avoir recours au défilement vertical inutilement.

La solution serait de cacher par défaut les formulaires de création et de modification, ensuite ajouter:

- Un bouton « **Modifier** » au niveau de chaque ligne dans les grilles de données, pour transformer la ligne sélectionnée en mode d'édition.
- Un bouton « **Supprimer** » au niveau de chaque ligne dans les grilles de données, pour supprimer la ligne courante.

- Et finalement un bouton « **Ajouter** » qui affichera le formulaire de création de données.

PACIQ-EVAL-20 Le logo de site web ne s'affiche pas proprement

Le logo du centre hospitalier universitaire mère enfant du CHU sainte-Justine utilisé dans le site web PACIQ ne s'affiche pas proprement.

Figure 2.8 Problème d'affichage du logo de PACIQ

Il est recommandé d'utiliser une autre image « *.PNG » pour afficher le logo du site web.

2.3.5. Fiabilité

a. Rétablissement

PACIQ-EVAL-21 Suppression physique des enregistrements de la base de données

La suppression physique des enregistrements des tables dans la base de données PACIQ ne permet pas de récupérer les données affectées et de rétablir l'état désiré en cas d'erreur.

Pour remédier à ce problème, il est recommandé d'ajouter deux colonnes au niveau de toutes les tables concernées par la suppression :

- **StartDate** de type SmallDateTime.
- **EndDate** de type SmallDateTime.

Ces colonnes seront utilisées afin d'activer ou désactiver une ligne au lieu de la supprimer physiquement de la table protéger l'utilisateur en cas d'erreur, de retracer l'historique de suppression des données et de créer des rapports avec des informations intègres et cohérentes.

PACIQ-EVAL-22 Absence de l'utilisation d'un outil de gestion de configuration

Le code source du projet PACIQ n'est pas maintenu dans un outil de gestion de configuration. Conséquemment, tout changement effectué sur un fichier du code source écrasera irrévocablement son ancienne version. Cela éliminera la possibilité de rétablir une ancienne version cohérente du code source.

Il est fortement recommandé, d'utiliser un outil de gestion de la configuration pour tous les produits du logiciel (c.-à-d. code source, document de spécification des exigences, document de maintenance… etc.) tell que TFS, SourceSafe ou SVN, afin de conserver l'historique des versions et de retracer les changements.

PACIQ-EVAL-23 Les erreurs et les exceptions du logiciel ne sont pas enregistrées

PACIQ ne permet pas de garder une trace des différents messages d'erreurs et d'exceptions rencontrées lors de son utilisation.

Il est recommandé, de créer une table « tError » dans la base de donnes du logiciel, qui sera utilisée pour sauvegarder toutes les informations sur les erreurs générées. Cela permettra d'identifier facilement la cause des erreurs et leurs occurrences afin de les corriger rapidement.

b. Disponibilité

PACIQ-EVAL-24 Le site web PACIQ roule sur un « AppPool » partagé avec d'autres applications

Le site web PACIQ partage actuellement son « AppPool » avec une autre application appelée « BugNet » qui est hébergée sur le même serveur web « IIS ». Par conséquent si le processus « w3wp.exe » arrête à cause de l'application « BugNet », PACIQ arrêtera aussi. En outre, à chaque fois « BugNet » sera déployée le « AppPool » partagé redémarrera, de façon que tous les utilisateurs de PACIQ perdront leurs sessions.

Afin de garantir une meilleure disponibilité de service, il est recommandé de créer un « AppPool » qui sera utilisé seulement par le site web PACIQ.

2.3.6. Sécurité

a. Confidentialité

PACIQ-EVAL-25 Le nom d'utilisateur de base de données et son mot de passe sont stockés en clair dans le fichier de configuration Web.config

L'authentification du logiciel PACIQ à la base de données se fait par le biais d'un nom d'utilisateur et d'un mot de passe, stockés en clair dans la chaîne de connexion du fichier de configuration « web.config ». Il suffit donc qu'un attaquant utilise un analyseur de paquets (c.-à-d. un sniffer) pour récupérer cet utilisateur ainsi que son mot de passe, ou tout simplement accéder au répertoire du site Web pour les récupérer.

Il est donc fortement recommandé de changer cette méthode d'authentification par une « authentification Windows intégrée », en précisant un nom d'utilisateur réseau dans la section d'authentification du « AppPool » de l'application et d'attribuer les droits nécessaires pour cet utilisateur sur la base de données PACIQ.

PACIQ-EVAL-26 Les utilisateurs ont tous les mêmes droits d'accès au site web PACIQ

Actuellement le logiciel PACIQ ne permet pas de gérer ses utilisateurs à l'aide de groupes de sécurité, de manière à limiter le droit d'accès à ses données selon des niveaux.

Il est recommandé d'ajouter la notion des groupes de sécurité dans l'application PACIQ en ajoutant deux tables « tGroup » et « tEmployeeGroup » dans la base de données de PACIQ.

b. Intégrité

PACIQ-EVAL-27 Absence de l'utilisation d'un outil de gestion de configuration

Le code source du logiciel PACIQ n'est pas maintenu dans un outil de gestion de configuration chose qui ne garantit pas l'intégrité de ses fichiers.

Il est recommandé d'utiliser un outil de gestion de configuration telle que TFS, SourceSafe ou SVN.

PACIQ-EVAL-28 Le mode de validation des requêtes spécifique à ASP.NET est désactivé

Le mode de validation des requêtes HTTP spécifique à ASP.NET est désactivé explicitement à partir du fichier de configuration de l'application (**requestValidationMode = FALSE**), cela crée des vulnérabilités exploitables via des attaques « XSS ».

Pour cela, il est recommandé d'activer le mode de validation (**requestValidationMode = TRUE**) dans le web.config de l'application, afin d'empêcher les attaquants d'injecter du contenu malicieux dans les pages du site web.

c. Non-réfutation

PACIQ-EVAL-29 Absence de la traçabilité d'accès au site web

Le logiciel PACIQ ne permet pas de conserver l'historique des accès à ses fonctionnalités, chose qui rend impossible de tracer l'historique des opérations effectuées par les utilisateurs.

La solution serait d'ajouter une table « tLog » qui sera alimentée à chaque fois un utilisateur accède à une page ou utilise une fonctionnalité du site web.

PACIQ-EVAL-30 Absence de la traçabilité de modification des données

Les informations concernant le nom d'utilisateur ayant modifié un enregistrement dans la base de données ainsi que « timestamp » ne sont pas conservées. De sorte que tous les utilisateurs peuvent facilement nier avoir modifié une donnée dans une table.

Pour remédier à ce problème, il est recommandé d'ajouter quatre colonnes dans chaque table :

Nom de la colonne	Type	Description
CreatedBy	Varchar(100)	Nom d'utilisateur ayant créé l'enregistrement
CreatedOn	SmalDateTime	Moment de création de l'enregistrement
LastUserModified	Varchar(100)	Nom d'utilisateur ayant modifié l'enregistrement
LastDateModified	SmalDateTime	Moment de la dernière modification de l'enregistrement

Tableau 2.2 Solution pour la traçabilité des modifications de données

d. Authenticité

PACIQ-EVAL-31 Aucun certificat SSL n'a été configuré pour le site web PACIQ

Aucun certificat SSL n'a été configuré pour sécuriser l'utilisation du site web PACIQ de l'extérieur de l'hôpital.

Il est recommandé de configurer un certificat SSL pour le site web PACIQ, afin d'assurer une connexion sécurisée entre le serveur web PACIQ et les navigateurs de ses utilisateurs.

2.3.7. Maintenabilité

a. Modularité

PACIQ-EVAL-32 Absence de la séparation des différentes couches de l'application

Toutes les fonctions métiers ainsi que celles d'accès aux données ont été placées dans le « Code Bhind » des pages web (couche présentation). Donc le couplage est très fort entre les différentes couches du logiciel, ce qui réduit considérablement sa modularité.

Il est donc recommandé de séparer le projet PACIQ en trois couches comme suit :

- PACIQDAL : ce projet contiendra toute la logique d'accès à la base de données.
- PACIQBAL : ce projet regroupera toute la logique d'affaires du logiciel.
- PACIQPresentation : ce projet englobera toute la partie d'interface utilisateur (UI).

PACIQ-EVAL-33 Le couplage est très fort entre le logiciel PACIQ et son module de sécurité

La sécurité de l'application PACIQ est gérée à partir des informations d'authentification (noms d'utilisateurs et mots de passe) stockées dans une table « Employee » de la base de données de l'application, par conséquent ce module de sécurité est fortement couplé à l'architecture interne de l'application, chose qui complexifie son remplacement ainsi que sa réutilisation dans d'autres applications.

Pour remédier à ce problème il est recommandé de :

- Créer une base de données centrale pour gérer les utilisateurs.
- Créer un projet « .NET » indépendant pour gérer l'authentification à partir de cette base de données centrale.

b. Réutilisabilité

PACIQ-EVAL-34 La déclaration répétée des variables identiques réduit la réutilisabilité du logiciel

Le logiciel PACIQ manipule toutes les données affichées, ajoutées, modifiées ou stockées dans des variables de session ou des variables déclarées de façon locale dans les méthodes du « Code Bhind ». Cela malheureusement ne favorise pas la réutilisation.

La solution serait d'utiliser un modèle orienté objet (classes, objets de classes, propriétés de classe, méthodes… etc.). Afin de factoriser le code en ensembles logiques permettant de manipuler uniquement des objets de classes.

PACIQ-EVAL-35 Les libellés et les messages de l'application sont écrits en dure dans les fichiers sources

Les libellés des champs de données ainsi que les descriptions des messages d'erreurs sont dupliqués dans toutes les pages de l'application, ce qui ne permet pas de les réutiliser dans des nouvelles pages.

Par conséquent il est recommandé d'utiliser les fichiers ressources pour stocker toutes les descriptions et messages d'erreurs dans le but de les réutiliser ultérieurement.

c. Modifiabilité

PACIQ-EVAL-36 Les noms des éléments HTML du site web ne sont pas significatifs

Les noms d'éléments HTML (boutons, grilles de données, listes déroulantes…etc.) et des sources de données du site web PACIQ sont générés automatiquement avec des noms qui ne sont pas significatifs, chose qui complexifie l'identification des instructions à changer lors d'une modification du code source.

La solution serait de définir et d'appliquer une convention pour tous les composants du site PACIQ. Cela permettra d'identifier rapidement les instructions à modifier ainsi que déduire le type de chaque élément à partir de son nom.

Exemples :

- **cbxNorme** : Pour nommer une liste déroulante de la norme.
- **txtCritere** : Pour nommer le champ de texte du critère.
- **dsAmeliorationPlan** : pour nommer la source de données des plans d'améliorations.
- **btnSave** : pour nommer un bouton de sauvegarde.

PACIQ-EVAL-37 Les objets de la base de données ne respectent pas une nomenclature

Les tables et les vues du logiciel PACIQ ne respectent pas une convention de nommage qui pourrait faciliter la recherche d'un élément dans la base de données.

Il est nécessaire d'utiliser une convention de nommage standard pour nommer les tables, les vues les procédures stockées…etc.

Exemple :

- **sEmployeeAdd** pour nommer une procédure stockée permettant d'ajouter un employé dans la base de données.
- **sEmployeeDelete** pour nommer une procédure stockée permettant de supprimer un employé.
- **tEmployee** : pour nommer une table d'employées.
- **vEmployee** pour nommer une vue pointant sur la table t'employée.

PACIQ-EVAL-38 Les méthodes et les évènements du « code Bhind » sont très longs

Les méthodes et les évènements situés dans le « code Bhind » des pages web, sont très longs et incluent beaucoup de logique dupliquée. Ainsi chaque modification apportée au code source peut potentiellement engendrer des régressions. En outre, afin de corriger une erreur il est nécessaire de parcourir toutes les classes du logiciel afin d'identifier les instructions similaires affectées par le même défaut, au lieu d'effectuer la correction dans un emplacement central.

Il faut réduire la taille des méthodes de classes en les décomposant sous la forme de plusieurs méthodes ayant une complexité cyclomatique réduite. Ensuite, regrouper les méthodes dupliquées dans des classes « Helpers ».

PACIQ-EVAL-39 Les descriptions des libelles et des messages d'erreurs sont codées en dure

Les libellés et les messages d'erreurs du logiciel sont codés en dure dans le code des pages web (code Bhind et Code HTML), il est nécessaire de parcourir toute l'application pour changer les descriptions similaires.

Il est donc recommandé de centraliser les messages d'erreurs et les libellés des champs dans des fichiers ressources.

PACIQ-EVAL-40 La solution PACIQ englobe plusieurs répertoires contenant des pages web dupliquées

La solution du site web PACIQ englobe plusieurs répertoires contenant un contenu dupliqué (Exemple : il existe deux répertoires nommés « Direction » contenant des pages identiques (CahierNorme.aspx, Criteres.aspx… etc.)), cela complexifie l'identification des pages qui correspondent à un module donné.

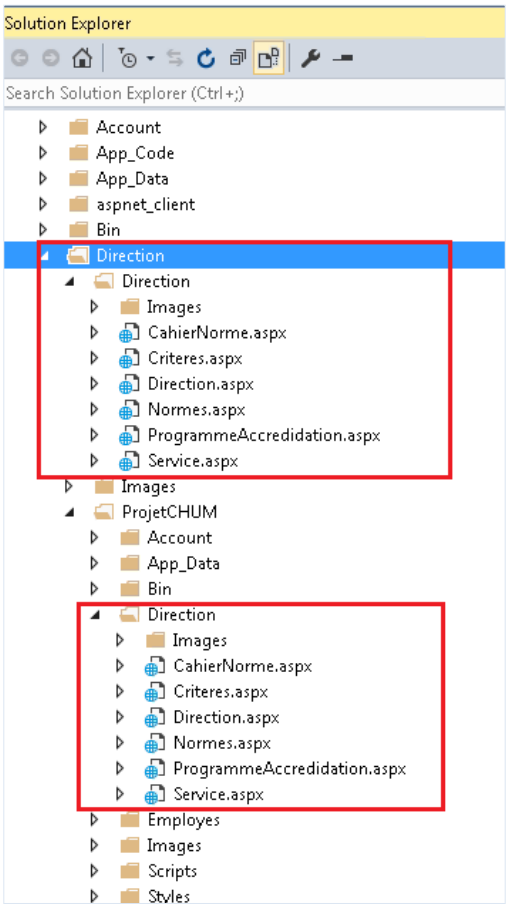


Figure 2.9 Duplication des répertoires identiques dans l'arborescence du logiciel PACIQ

Il est recommandé de créer un répertoire pour chaque module du projet PACIQ dans lesquelles toutes les pages, les fichiers de style et les scripts du module seront placés sans duplication.

PACIQ-EVAL-41 Les instructions de style des pages web ont été regroupées dans un seul fichier « CSS ».

Un seul fichier de style (*.css) a été créé pour toutes les pages de l'application. Par conséquent, pour modifier le style d'un élément dans une page, il est nécessite de parcourir le contenu de ce fichier en entier afin d'identifier la ligne à modifier.

Il est recommandé de :

- Créer un fichier CSS central contenant le style générique de l'application.
- Créer un fichier CSS par page en utilisant l'héritage, pour remplacer le style générique en cas de besoin.

PACIQ-EVAL-42 Les références des bibliothèques utilisées dépendent de la configuration logicielle de la machine de développement

Les références vers les bibliothèques (*.dll) utilisées dans le projet PACIQ dépendent fortement de la configuration logicielle de la machine de développement. Ainsi toute modification apportée sur une version d'une bibliothèque utilisée, causera un problème de compilation pour les autres développeurs.

La solution serait de créer un répertoire « **SharedLibrary** » dans la solution du projet, qui sera utilisé pour conserver et centraliser toutes les bibliothèques « **Third Party** » utilisées, de sorte que toutes les références de DLL pointeront vers ce répertoire.

PACIQ-EVAL-43 Le code source PACIQ contient beaucoup de codes morts

Le code source du site web PACIQ contient beaucoup de code mort, par exemple des fichiers de classes et des pages web (*.aspx) non utilisés, du code source commenté… etc.

La solution c'est d'effectuer un ménage pour supprimer le code source non utilisé.

PACIQ-EVAL-44 Les éléments du menu de l'application sont créés statiquement

Les éléments de l'arborescence du menu générale de l'application sont créés statiquement (codés en dure). Donc toute modification d'une URL ou d'une description nécessite le déploiement complet de l'application.

Pour remédier à ce problème, il est recommandé de créer une table contenant toutes les informations relatives aux pages de l'application, qui sera utilisée comme source de données pour créer dynamiquement le menu de l'application.

d. Testabilité

PACIQ-EVAL-45 Absence de la séparation des couches du logiciel PACIQ

Les méthodes développées dans le code source du projet PACIQ, incluent toute la logique de la couche présentation, métier et accès au donnés, chose qui peut complexifier le développement et l'exécution des tests unitaires et d'intégrations.

Il est recommandé donc, de séparer la logique de la couche présentation, métier et accès aux données.

2.3.8. Portabilité

a. Adaptabilité

PACIQ-EVAL-46 L'adaptation à l'environnement d'opération (Dev, Test et Production) est impossible

Le code source PACIQ ne contient aucune logique permettant de déterminer dans quel environnement l'application fonctionne (Développement, Test, QA, Production).

Il est recommandé donc d'ajouter une clé à l'intérieur de la balise « AppSetting » du fichier de configuration de l'application « Web.config » pour déterminer l'environnement sur lequel l'application fonctionne.

PACIQ-EVAL-47 L'adaptabilité à l'environnement technique est garantie

PACIQ a été développé en utilisant la version 4.0 du Framework .NET, par conséquent l'hébergement de ce site web peut se faire sur toutes les versions du serveur web IIS supportant la version 4.0 du framwork.Net.

b. Installabilité

PACIQ-EVAL-48 PACIQ peut facilement s'installer sur un serveur web

Le logiciel PACIQ a été développé sous forme d'un site web Aspx.NET qui s'installe facilement sur un serveur web. Actuellement le déploiement de l'application dans l'environnement de test a été fait seulement en copiant le code source du site web dans le répertoire (C:\Programme\inetpub\www\PACIQ).

Toutefois il est recommandé d'utiliser l'outil « WebDeploy » pour faciliter le déploiement de l'application cela permet également de tracer toutes les opérations de déploiement effectuées.

c. Remplaçabilité

PACIQ-EVAL-49 PACIQ est un logiciel développé sur mesure qui n'est pas facilement remplaçable

Le site web PACIQ est un logiciel développé sur mesure par des étudiants de l'École Supérieur de Technologie, donc seulement une solution développée sur mesure respectant les exigences fonctionnelles et non fonctionnelles spécifiées par le client, peut remplacer le site web PACIQ.

2.4.Amélioration de la qualité du logiciel PACIQ

Suite à l'évaluation de la qualité réalisée dans la première étape de ce projet, et qui a permet d'identifier plusieurs défauts techniques et fonctionnels dans le logiciel PACIQ. J'ai procéder à une évaluation de la criticité de chaque faiblesse, par rapport à l'impact de l'amélioration proposée, afin de réaliser un plan d'action qui tient en compte plusieurs facteurs tels que, l'urgence de chaque problème, l'interdépendance des problèmes, la

pertinence de l'amélioration proposée, ainsi que la durée du projet. Par conséquent, j'ai décidé en collaboration avec le client de procéder de la manière suivante :

- D'abord, j'ai corrigé tous les défauts fonctionnels soulevés par le client, afin de lui donner suffisamment le temps, pour vérifier et valider les corrections apportées;
- Ensuite, j'ai corrigé tous les défauts critiques du point de vue interne du logiciel.

Numéro d'évaluation	Description du point d'évaluation	Criticité du défaut	Impact du changement
PACIQ-EVAL-1	L'utilisation du SQL dynamique	Moyenne	Élevé
PACIQ-EVAL-2	L'absence de l'utilisation des Index SQL	Moyenne	Faible
PACIQ-EVAL-3	Utilisation de l'instruction « SELECT » sans clause « WHERE »	Moyenne	Faible
PACIQ-EVAL-4	PACIQ est configuré pour être déployé seulement en mode « Debug »	Faible	Faible
PACIQ-EVAL-5	Ouverture de plusieurs connexions SQL pour interroger la base de données	Faible	Moyen
PACIQ-EVAL-6	Les pages web du site web PACIQ sont volumineuses	Faible	Moyen
PACIQ-EVAL-8	Les dates sont affichées et stockées dans un format invalide	Moyenne	Faible
PACIQ-EVAL-9	Incrémentation manuelle de la clé primaire des tables	Très grave	Élevé
PACIQ-EVAL-10	Les messages du logiciel PACIQ sont parfois incorrects ou non pertinents	Faible	Faible
PACIQ-EVAL-13	PACIQ exécute toutes les opérations sans confirmation de l'utilisateur	Moyenne	Faible
PACIQ-EVAL-14	Absence de la validation de toutes les données envoyées par l'utilisateur	Moyenne	Moyen

PACIQ-EVAL-15	Suppression physique des enregistrements de la base de données	Grave	Moyen
PACIQ-EVAL-16	Le site web PACIQ n'offre pas des écrans adaptatifs (Responsive)	Faible	Élevé
PACIQ-EVAL-17	Le choix des couleurs et des contrôles HTML	Faible	Moyen
PACIQ-EVAL-18	La largeur des colonnes dans les grilles de données ne s'adapte pas au contenu	Faible	Moyen
PACIQ-EVAL-19	Les formulaires de création et de modification sont affichés en permanence dans les pages web	Faible	Moyen
PACIQ-EVAL-20	Le logo de site web ne s'affiche pas proprement	Faible	Faible
PACIQ-EVAL-21	Suppression physique des enregistrements de la base de données	Grave	Moyen
PACIQ-EVAL-22	Absence de l'utilisation d'un outil de gestion de configuration	Très grave	Faible
PACIQ-EVAL-23	Les erreurs et les exceptions du logiciel ne sont pas enregistrées	Moyenne	Faible
PACIQ-EVAL-24	Le site web PACIQ roule sur un « AppPool » partagé avec d'autres applications	Moyenne	Faible
PACIQ-EVAL-25	Le nom d'utilisateur de base de données son mot de passe sont stockés en clair dans le fichier Web.config	Grave	Faible
PACIQ-EVAL-26	Les utilisateurs ont tous les mêmes droits d'accès au site web PACIQ	Moyenne	Moyen
PACIQ-EVAL-27	Absence de l'utilisation d'un outil de gestion de configuration	Grave	Faible
PACIQ-EVAL-28	Le mode de validation des requêtes	Faible	Faible

	spécifique à ASP.NET est désactivé		
PACIQ-EVAL-29	Absence de la traçabilité d'accès au site web	Faible	Faible
PACIQ-EVAL-30	Absence de la traçabilité de modification des données	Moyenne	Faible
PACIQ-EVAL-31	Aucun certificat SSL n'a été configuré pour le site web PACIQ	Moyenne	Élevé
PACIQ-EVAL-32	Absence de la séparation des différentes couches de l'application	Faible	Élevé
PACIQ-EVAL-33	Le couplage fort du module de sécurité du logiciel PACIQ	Faible	Élevé
PACIQ-EVAL-34	La déclaration répétée des variables identiques réduit la réutilisabilité du logiciel	Faible	Élevé
PACIQ-EVAL-35	Les libellés et les messages de l'application sont écrits en dure dans les fichiers sources	Faible	Faible
PACIQ-EVAL-36	Les noms des éléments HTML du site web ne sont pas significatifs	Faible	Moyen
PACIQ-EVAL-37	Les objets de la base de données ne respectent pas une nomenclature	Faible	Moyen
PACIQ-EVAL-38	Les méthodes et les évènements du « code Bhind » sont très longs	Faible	Élevé
PACIQ-EVAL-39	Les descriptions des libelles et des messages d'erreurs sont codées en dure	Faible	Faible
PACIQ-EVAL-40	La solution PACIQ englobe plusieurs répertoires contenant des pages web dupliquées	Faible	Moyen
PACIQ-EVAL-41	Les styles CSS des pages web ont été regroupés dans un seul fichier	Faible	Moyen

	« CSS ».		
PACIQ-EVAL-42	Les références des bibliothèques utilisées dépendent de la configuration de la machine de développement	Moyenne	Faible
PACIQ-EVAL-43	Le code source PACIQ contient beaucoup de codes morts	Faible	Moyen
PACIQ-EVAL-44	Les éléments du menu de l'application sont créés statiquement	Moyenne	Moyen
PACIQ-EVAL-45	Absence de la séparation des couches du logiciel PACIQ	Faible	Élevé
PACIQ-EVAL-46	L'adaptation à l'environnement d'opération est impossible	Moyenne	Moyen

Tableau 2.3 Matrice de criticité des faiblesses et d'impacts d'améliorations

2.3.1. Corrections des défauts fonctionnels identifiés par le client

- **Correction du problème de modification des cahiers de normes :** Cette correction avait comme objectif de modifier la logique de modification des cahiers de normes, et qui empêche la mise à jour des champs lorsque l'utilisateur change des caractères « majuscule » par « minuscule » ou l'inverse. En effet j'ai appliqué cette correction sur tous les formulaires de modification de l'application.

- **Ajout de deux responsables pour le même plan d'amélioration :** Cette correction avait comme but de modifier les pages de création, de modification ainsi que les rapports, afin de permettre aux utilisateurs d'assigner le même plan d'amélioration et ses activités à trois responsables en même temps au lieu d'un seul

- **Création de plusieurs activités d'amélioration pour un objectif :** Cette correction a permet aux utilisateurs d'insérer et de modifier plusieurs activités d'amélioration pour un même objectif.

- **Ajout de la possibilité d'inscrire un indicateur de qualité non inclus dans la liste des indicateurs :** Cette correction a offert l'avantage aux utilisateurs de choisir des indicateurs non encore inscrits dans les données de références des indicateurs

- **Affichage et modification et plans d'amélioration :** Cette correction avait deux objectifs, d'aborder la correction du problème critique qui empêche l'insertion des plans d'action dans la base de données. Ensuite, la modification de la page d'affichage des plans d'action afin de permettre aux utilisateurs de choisir un plan d'amélioration à modifier.

- **Adaptation des champs à la largeur des pages :** Cette correction consistait à modifier toutes les pages de l'application PACIQ et leurs fichiers de style (*.CSS), afin de modifier les largeurs fixes attribuées aux éléments du HTML, par une largeur dynamique qui prend en considération toutes les dimensions des écrans utilisées.

- **Correction du problème de synchronisation des rapports avec le logiciel PACIQ :** Cette correction avait comme objectif de corriger tout le processus (Packages SSIS) qui permet de synchroniser les données de la base de données principale du logiciel vers la base de données des rapports.

- **Correction du problème de changement de mot de passe dans le logiciel PACIQ :** Cette correction consistait à modifier l'écran de modification de mot de passe afin de permettre aux utilisateurs de saisir un nouveau mot de passe

En plus de ces changements importants, j'ai corrigé aussi d'autres petits problèmes, tels que la modification des descriptions des zones de textes, la modification du style de la page d'accueil, l'augmentation de la performance lors du premier chargement…etc.

2.3.2. Corrections des défauts critiques du point de vue interne du logiciel

En plus de la correction des défauts fonctionnels identifiés par le client, j'ai apporté aussi les améliorations proposées pour les points d'évaluations suivants :

Numéro d'évaluation	Description du point d'évaluation	Statut de l'amélioration
PACIQ-EVAL-3	Utilisation de l'instruction « SELECT » sans clause « WHERE »	Complété
PACIQ-EVAL-4	PACIQ est configuré pour être déployé seulement en mode « Debug »	Complété
PACIQ-EVAL-5	Ouverture de plusieurs connexions SQL pour interroger la base de données	Complété
PACIQ-EVAL-8	Les dates sont affichées et stockées dans un format invalide	Complété
PACIQ-EVAL-10	Les messages du logiciel PACIQ sont parfois incorrects ou non pertinents	Complété
PACIQ-EVAL-18	La largeur des colonnes dans les grilles de données ne s'adapte pas au contenu	Complété
PACIQ-EVAL-23	Les erreurs et les exceptions du logiciel ne sont pas enregistrées	Complété
PACIQ-EVAL-24	Le site web PACIQ roule sur un « AppPool » partagé avec d'autres applications	Complété
PACIQ-EVAL-26	Les utilisateurs ont tous les mêmes droits d'accès au site web PACIQ	Complété

PACIQ-EVAL-29	Absence de la traçabilité d'accès au site web	Complété
PACIQ-EVAL-38	Les méthodes et les évènements du « code Bhind » sont très longs	Complété
PACIQ-EVAL-40	La solution PACIQ englobe plusieurs répertoires contenant des pages web dupliquées	Complété
PACIQ-EVAL-42	Les références des bibliothèques utilisées dépendent de la configuration de la machine de développement	Amélioration complétée
PACIQ-EVAL-43	Le code source PACIQ contient beaucoup de codes morts	Amélioration complétée

Tableau 2.4 Points d'évaluation corrigés

2.5.Conclusion

Ce chapitre a permis présenter ma démarche d'évaluation et ses différents résultats ainsi que l'ensemble d'amélioration apporté sur le code source du logiciel PACIQ dans le but d'améliorer sa qualité. Le prochain chapitre vise à présenter l'impact de ces améliorations sur la qualité du logiciel, ainsi que les enjeux rencontrés durant le projet.

CHAPITRE 3

RÉSULTATS ET SYNTHÈSE DU TRAVAIL RÉALISÉ

3.1 Introduction

L'objectif de ce chapitre consiste à présenter d'abord l'impact des améliorations apportées sur le code source du logiciel PACIQ au niveau de sa qualité initiale, ensuite décrire les enjeux rencontrés lors de la réalisation du projet et discuter des solutions proposées et mises en œuvre.

3.1. Résultats de l'amélioration du logiciel PACIQ

Bien que le nombre d'anomalies identifiées lors de l'évaluation de la qualité est relativement élevé par rapport aux différentes corrections et améliorations apportées sur le code source du logiciel PACIQ, l'importance relative des changements effectués a augmenté considérablement le niveau de qualité finale de la nouvelle version du prototype logiciel PACIQ, et ce notamment d'un point de vue externe. L'utilisatrice principale a insistée sur l'importance de corriger tous les défauts qu'elle a identifiée durant la phase de test d'acceptation. Durant ces tests, elle a aussi identifié des défauts ainsi que les améliorations fonctionnelles qui ont impliqué des ajustements aux règles d'affaires. Conséquemment, cette nouvelle version du logiciel peut être considérée maintenant de bonne qualité par rapport à sa version précédente du point de vue client-utilisateur mais aussi du point de vue interne (c.-à-d. architecture, conception et programmation du code source).

3.2. Enjeux rencontrés et solutions mises en œuvre

L'évaluation de la qualité du logiciel PACIQ et son amélioration n'était pas un objectif facile à atteindre, dans un projet de courte durée, notamment à cause des problématiques suivantes :

D’abord, le logiciel PACIQ, a été développé dans le cadre d’un processus de développement Agile à la suite de plusieurs itérations effectuées par plusieurs étudiants. Cela a eu comme impact une certaine incohérence entre les modules du logiciel au niveau architectural, et au niveau de la programmation (c.-à-d. du code source) - étant donné que chaque programmeur avait son propre style de programmation et sa propre méthode de travail. Pour résoudre ce problème, l’évaluation a été effectuée par la visualisation statique du code source dans le but de mieux comprendre l’anatomie du code source existant du logiciel et son architecture. En outre, le logiciel PACIQ a été approché comme un ensemble de modules avec l’objectif de mesurer la qualité de chaque module, de façon séparée, et d’en déduire une qualité globale.

Par la suite il a été observé que la cliente du logiciel PACIQ n'a pas adhérée convenablement au processus d’évaluation et d’amélioration établi dans ce projet. Elle n’avait pas testé toutes les corrections apportées ainsi que toutes les nouvelles fonctionnalités développées. La solution à cette problématique a été la planification et la tenue de plusieurs réunions de revue « Walkthrough» afin de l’encourager à vérifier et valider tous les changements ensemble.

Finalement, les demandes de changements reçues de la cliente, durant le projet, ont influencées le plan d’action initial. Il a été nécessaire de changer la priorité des améliorations plusieurs fois. La solution proposée dans ce cas, a été de réévaluer l’impact de chaque nouvelle demande de changement sur le calendrier du projet et le plan d’action initial. De cette manière une discussion concernant la pertinence d’un changement était faite de manière structurée avec la cliente de manière à ce qu’elle décide si la priorité assignée est correcte.

CONCLUSION

Ce rapport a présenté une synthèse de la démarche d'évaluation et d'amélioration réalisée dans le cadre de l'amélioration du prototype logiciel PACIQ. Il a aussi présenté les enjeux rencontrés ainsi que les solutions proposées et mises en œuvre. Ainsi, ce projet a pu démontrer l'importance et la pertinence de l'utilisation d'en processus d'évaluation de la qualité dans les projets de développement logiciel. Ce projet appliqué a permis de détecter des défauts et d'identifier des faiblesses à ce logiciel selon deux perspectives - interne et externe.

La prochaine étape de ce projet serait de continuer la correction et l'amélioration de la version actuelle du logiciel PACIQ. Parfois les programmeurs vont préférer refaire complètement le logiciel qui possède un grand nombre de défauts. Il n'est simplement pas raisonnable de tout recommencer le développement du logiciel.

ANNEXE I

Liste de vérification pour l'interface utilisateur

Liste de vérification	
Code de la règle	**Description de la règle**
CHK_UI_01	Les couleurs des pages web sont bien choisies
CHK_UI_02	le logo du site web est bien placé
CHK_UI_03	Les liens des pages web sont faciles à discerner
CHK_UI_04	Les pages web sont « Responsive » et s'adaptent aux différents formats et largeurs d'écrans
CHK_UI_05	Les caractères ont une taille suffisante
CHK_UI_06	les pages web possèdent la même uniformité
CHK_UI_07	Le site web contient une page « Plan de site »
CHK_UI_08	Les messages d'erreurs sont pertinents
CHK_UI_09	La mise à jour des données ne nécessite pas le chargement de la page
CHK_UI_10	Les pages web s'ouvrent rapidement

Tableau 3.1 Liste de vérification pour l'interface utilisateur

ANNEXE II

Liste de vérification pour l'architecture et le code source du logiciel

Liste de vérification	
Code de la règle	**Description de la règle**
CHK_PROG_01	Les classes, objets, variables méthodes respectent une nomenclature
CHK_PROG_02	La complexité cyclomatique des méthodes n'est pas élevée
CHK_PROG_03	Le couplage est faible entre les modules du logiciel
CHK_PROG_04	La cohésion des classes et des méthodes est normale
CHK_PROG_05	Les instructions utilisées dans plusieurs endroits sont réutilisées
CHK_PROG_06	Les commentaires du code source sont pertinents
CHK_PROG_07	Le code source ne contient pas de code mort
CHK_PROG_08	Des tests unitaires sont créés pour chaque méthode
CHK_PROG_09	Les objets volumineux sont détruits explicitement dans le code source
CHK_PROG_10	Les patrons de conception utilisés sont pertinents

Tableau 3.2 Liste de vérification pour l'architecture et le code source du logiciel

ANNEXE III

Liste de vérification pour la base de données

Liste de vérification	
Code de la règle	**Description de la règle**
CHK_DB_01	Les tables de la base de données contiennent des « indexes »
CHK_DB_02	Les procédures stockées n'utilisent pas du « SQL dynamique »
CHK_DB_03	Les champs retournés par les procédures stockées sont nommées
CHK_DB _04	Les curseurs ne sont pas utilisés dans procédures stockés
CHK_DB_05	Les données sont stockées dans des types de données adéquats
CHK_DB_06	Les procédures stockées n'utilisent pas des transactions
CHK_DB_07	Les opérations de suppression suppriment logiquement les données
CHK_DB_08	Les opérations de sélection utilisent des clauses « Where »
CHK_DB_09	Un backup automatique est planifié pour la base de données
CHK_DB_10	Les éléments de la base de données respectent une nomenclature

Tableau 3.3 Liste de vérification pour la base de données

ANNEXE IV

Liste de vérification pour la sécurité du logiciel

Liste de vérification	
Code de la règle	**Description de la règle**
CHK_SEC_01	Les mots de passe ne sont pas envoyés en clair dans le réseau
CHK_SEC_02	Le logiciel permet d'authentifier ses utilisateurs
CHK_SEC_03	La vulnérabilité « d'injection SQL » est corrigée dans le logiciel
CHK_SEC_04	La vulnérabilité « d'injection de commande » est corrigée dans le logiciel
CHK_SEC_05	La vulnérabilité « XSS » est corrigée dans le logiciel
CHK_SEC_06	La vulnérabilité « Cross Site Request Forgery » est corrigée dans le logiciel
CHK_SEC_07	La complexité des mots de passe est suffisante
CHK_SEC_08	L'expiration des mots de passe du logiciel est gérée
CHK_SEC_09	Le logiciel utilise le protocole HTTPS pour chiffrer les communications avec le serveur web
CHK_SEC_10	L'intégrité des données est assurée dans le logiciel
CHK_SEC_11	La confidentialité des données est assurée dans le logiciel

Tableau 3.4 Liste de vérification pour la sécurité du logiciel

BIBLIOGRAPHIE

[1] Assurance Qualité Logicielle (Alain April, Claude Y Laporte, 2011)

[2] Kitchenham, B., & Pfleeger, S. (1996). Software quality: the elusive target. Software, IEEE, 13(1), 12–21;

[3] Experiences from performing software quality evaluations via combining benchmark-based metrics analysis, software visualization, and expert assessment (Aiko Yamashita, 29 Sept. Oct. 2015);

[4] An experience report for software quality evaluation in highly iterative development methodology using traditional metrics (Kumi Jinzenji; Takashi Hoshino; Laurie Williams; Kenji Takahashi, 2013);

[5] MÉTRIQUES ET CRITÈRES D'ÉVALUATION DE LA QUALITÉ DU CODE SOURCE D'UN LOGICIEL (Pierre Mengal 2013)

[6] A Software Quality Evaluation Method Using the Change of Source Code Metrics(M. Nakamura,T. Hamagami, 2012);

[7] The Software Quality Evaluation Method Based on Software Testing(Liu Wen-Hong, Wu Xin, 2012);

[8] Metrics and Antipatterns for Software Quality Evaluation (Francesca Arcelli Fontana, Stefano Maggioni, 2012);

[9] Évaluation de la Qualité des Applications web : État de l'Art (Ghazwa Malak, Nadir Belkhiter, Mourad Badri, Linda Badri, 2001,2002)

[10] Experiences from performing software quality evaluations via combining benchmark-based metrics analysis, software visualization, and expert assessment(Aiko Yamashita, 2012);

[11] A Quantitative Software Quality Evaluation Model for the Artifacts of Component Based Development(Kilsup Lee, Sung Jong Lee, 2005)

[12] A Software quality evaluation method using the change of source code metrics (Mitsuhiro Nakamura, Tomoki Hamagami, 2012)

[13] Garvin, D. A. (1988). Managing Quality - the strategic and competitive edge. New York, NY: Free Press [u.a.];