RAPPORT FINAL
PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
DANS LE CADRE DU COURS LOG792

MQMM
MANUFACTURING QUALITY MANAGEMENT MODULE

NOÉMIE CLOUTIER
CLON10618906
PIER-OLIVIER FAUCHER
FAUP01029005
CLÉMENT ROCHON
ROCC12118909
PHILIPPE R.TESSIER
RTEP08099005

DÉPARTEMENT DE GÉNIE LOGICIEL ET DES TI

Professeur-superviseur

ALAIN APRIL

MONTRÉAL, 15 AOÛT 2017
ÉTÉ 2017

REMERCIEMENTS

Nous tenons à remercier toutes les personnes qui ont contribué à la réalisation de notre projet final d'études et la rédaction de ce rapport.

Tout d'abord, merci à Netsen pour nous avoir permis de réaliser ce projet avec eux et particulièrement à Alpha Diallo et Marcel Tawe pour leurs disponibilités. Ils ont toujours essayé de régler nos problèmes le plus rapidement possible malgré leurs horaires chargés.

Enfin, nous voulons remercier notre professeur-superviseur, Alain April pour ses nombreux conseils, sa présence constante et sa flexibilité tout au long du projet.

RAPPORT FINAL

NOÉMIE CLOUTIER
CLON10618906
PIER-OLIVIER FAUCHER
FAUP01029005
CLÉMENT ROCHON
ROCC12118909
PHILIPPE R.TESSIER
RTEP08099005

RÉSUMÉ

Le but de ce rapport est de décrire le processus d'analyse et de conception de la création d'un système d'audit. Notre module doit se greffer à un système déjà existant utilisant la maîtrise statistique des procédés pour assurer la qualité de pièces automobiles produite dans une usine. Ce rapport s'adresse aux étudiants ou aux employés futurs qui pourront travailler sur ce module.

Les analyses que l'on trouve dans ce document sont les résultats finaux du projet. Les modifications et les améliorations seront justifiées dans le paragraphe suivant lesdites analyses.

Finalement, nous présenterons les résultats de notre projet ainsi que la rétrospective que nous avons faite à la fin du projet.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

Dans le document suivant, des abréviations, sigles et acronymes seront utilisés amplement. Pour faciliter la lecture, il sera possible de se référer au tableau ci-dessous pour comprendre la signification.

Abréviations, sigles et acronymes	Signification
BD	Base de données
BO	Business Object
CSS	Cascading Style Sheets
C#	Langage de programmation C-Sharp
DAO	Data Access Object
DTO	Data Transfer Object
ERM	Entity Relationship Model
HC	Donnée hors contrôle
HTML	Hyper Text Markup Language
IDE	Integrated Development Environment
QA	Assurance Qualité
QMP	Quality Management Process
MQMM	Manufacturing Quality Management Module
MVC	Modèle Vue Contrôleur
NC	Donnée non conforme
ORM	Object-relational mapping.
PDG	Président-directeur général
PFE	Projet de fin d’études
PO	Product Owner
R&D	Recherche et développement

SQL	Langage utilisé pour communiquer avec les bases de données
SPC	Statistical Process Control
UI	Interface
UX	Expérience utilisateur
VPN	Virtual private network

INTRODUCTION

La compagnie NetSen Group est une entreprise œuvrant dans le domaine des technologies de l'information. Elle propose, entre autres, un logiciel permettant d'analyser des données recueillies par une machine à mesurer tridimensionnelle. Le système permet de repérer les données hors contrôles et non conformes. Par contre, il ne permet pas d'entreprendre des actions correctives pour régler ces irrégularités.

Dans le cadre du projet de fin d'études, la compagnie Netsen a soumis un projet d'étude de conception d'un module d'analyse et de contrôle statistique de procédé à travers des robots de mesure dans l'industrie automobile. L'objectif de cette proposition est d'ajouter un module de gestion des audits. De plus, le client avait demandé d'améliorer les performances du système existant.

Le module permettra de déclencher une action corrective à partir d'une non conformité ou d'un hors contrôle et de gérer des audits. Dans un premier temps, nous discuterons du contexte de ce projet, du système existant ainsi que de la problématique du projet. Ensuite, nous expliquerons la méthodologie que nous avons employée lors du projet et le plan de travail que nous avons élaboré. Par la suite, nous décrirons le processus de conception et prototype que nous avons produit. Finalement, nous reviendrons sur les apprentissages qui sont sortis de ce projet et les recommandations que nous avons pour le projet.

CHAPITRE 1

MISE EN CONTEXTE

1.1 PRÉSENTATION DU CLIENT

Notre client dans ce projet est NetSen Group. Cette entreprise de Montréal oeuvre dans le domaine des technologies de l'information et la gestion des processus. Ils offrent des services de gestion des processus métiers à ses clients. Ils possèdent trois produits, soit un système d'information pour les hôpitaux (SIH), un système d'information sur la santé (DHIS 2) et un système de contrôle de la qualité pour les manufacturiers (Cloud SPC4 CMM). C'est ce dernier produit auquel doit se greffer notre système. Il s'agit d'une PME et leurs bureaux se trouvent à Montréal. Le promoteur du projet est Alpha Diallo, le PDG de Netsen.

1.2 SYSTÈME EXISTANT

Le système existant, nommé SPC4CMM, est un logiciel permettant d'analyser des données recueillies par une machine à mesurer tridimensionnelle. Avec ces données et les spécifications définies par le client, le système génère des rapports pour déterminer les données hors contrôles et non conformes de pièces manufacturées.

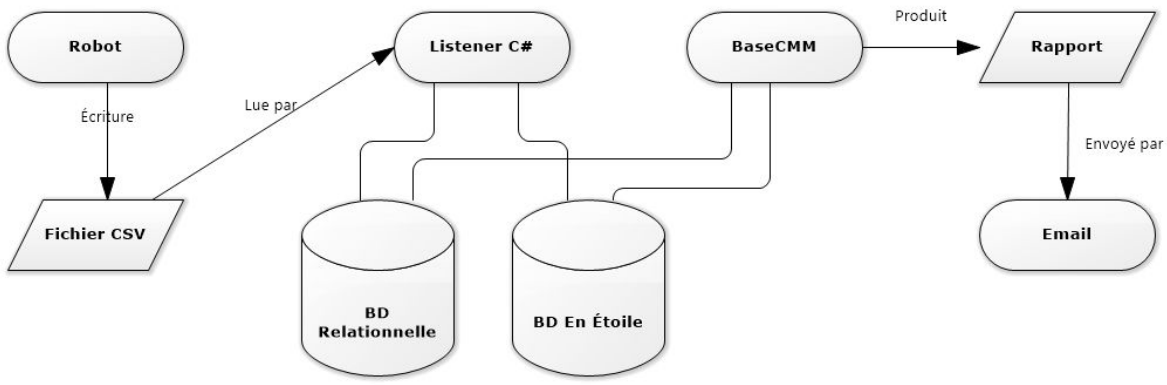

Figure 1. *Architecture fonctionnelle - Application SPC4CMM*

Lors de la session d'hiver 2017, des étudiants avaient comme objectifs de générer des alertes et graphiques en fonction des résultats de pièces non conformes ou hors contrôles. Ces graphiques devaient être générés dans des rapports et envoyés au besoin par courriel.

Lors du lancement du projet, nous avons eu accès au rapport final de ces étudiants. Vous pouvez le consulter à l'Annexe II de ce rapport. Nous l'avons joint parce qu'il a influencé les décisions du client et qu'il nous a permis de comprendre les détails du contexte du projet. Dans ce rapport, les élèves soulevaient entre autres la difficulté d'installer un poste de développement de SPC4CMM. Les figures que l'on y retrouve nous permettent de comprendre rapidement la structure de l'application existante.

1.2.1 ARCHITECTURE EXISTANTE

L'application existante utilise l'ORM *nHibernate* ainsi que le patron DAO – DTO – BO (*DataBase Interaction DAO/DTO Design Pattern*). Ce patron est utilisé pour faire la liaison entre le modèle de données et le modèle d'affaires pour qu'ensuite des vues soient générées. Cette architecture est utilisée en tandem avec le logiciel libre *nHibernate*. Ce cadriciel permet d'utiliser un fichier de zapping pour lier un élément du modèle d'affaires à sa représentation dans la base de données. Les fichiers de DTO servent à contenir les éléments du modèle d'affaires et à effectuer différents types de formatage pour diverses parties de l'application. Les fichiers DAO sont présents puisqu'ils sont nécessaires, mais ne possèdent aucune fonctionnalité les différenciant les uns des autres. Les fichiers BO eux utilisent les précédents éléments dans le but d'effectuer les transactions à la base de données. Pour ce projet, l'architecture existante est une importante contrainte, car il est important de ne pas briser ou contourner l'architecture existante d'un produit logiciel.

1.3 PROBLÉMATIQUE

La problématique du projet nous a été expliquée par le promoteur et notre superviseur lors de la première rencontre de travail. Vous trouverez les documents qui nous ont été fournis à

l'Annexe III. Le cas d'utilisation principal est que pour l'instant, lorsqu'une NC ou HC est soulevée par le système SPC4CMM, il n'y a pas de moyen structuré de résoudre ce problème ou même de garder des traces des actions prises lors de la résolution du défaut. De plus, le système existant a beaucoup de problèmes de performance lors de la génération des rapports de non conformité et de contrôle des pièces manufacturées.

1.4 ENVIRONNEMENT DE DÉVELOPPEMENT

Pour le suivi du projet, nous allons utiliser la plupart des outils qui sont déjà implantés dans cette compagnie. Au lieu de configurer nos machines personnelles, nous avions un accès à distance vers une VM qui contenait déjà tous les logiciels nécessaires au développement. Malgré que cela a simplifié l'installation, nous avons vécu beaucoup de problèmes avec celles-ci. Voici ce que nous avons identifié comme environnement de développement et la liste des outils utilisés pour ce projet:

1.4.1 PILE TECHNOLOGIQUE

ASP.NET MVC : Framework C# axé sur les modèles permettant de construire des sites web dynamiques.

Bootstrap et Materialize : Frameworks HTML, CSS et JS permettant de construire des sites réactifs. On l'utilise lors de la création des interfaces.

Mercurial : Protocole de gestion de code source.

NHibernate : ORM pour créer lié les entités à la base de données.

SQL Server : Base de données relationnelle

1.4.2 OUTILS DE GESTION DE PROJET

Antidote : Correction de fautes d'orthographe et révision de la grammaire.

Appear.in : Permettre les rencontres hebdomadaires à distance.

Google Drive : Écrire, formater et partager les livrables.

Jira : Gérer les tâches du projet et permettre au client de prioriser celles-ci.

Slack : Communiquer entre les membres de l'équipe ainsi qu'avec le client.

1.4.3 OUTILS DE DÉVELOPPEMENT

BitBucket : Service web d'hébergement de code source.

Microsoft Visual Studio : IDE utilisé pour produire le code.

Photoshop : Éditer des images et produire des croquis d'interfaces utilisateurs

SQL Server Management Studio : Gérer et analyser la base de données.

Software Idea Modeler : Créer des diagrammes et des organigrammes.

SourceTree : Interface pour simplifier l'utilisation de Mercurial.

VM : Permettre aux développeurs d'utiliser l'environnement de développement.

CHAPITRE 2

ANALYSE

Cette section du rapport liste les éléments qui ont été déterminés au début de la phase d'analyse du projet. Les différentes phases seront expliquées plus bas, puisqu'elles ont été définies lors de l'étape d'analyse du projet. Le but de l'analyse est de comprendre les documents d'exigences fournis par le client et de définir quels sont les objectifs et comment les atteindre.

2.1 OBJECTIFS

Suite aux lancements du projet, nous avons divisé celui-ci en deux parties : le MQMM et l'amélioration de la performance des rapports. Les deux parties sont indépendantes et le but est de s'assurer qu'un prototype d'interface du MQMM est livré et d'utiliser le temps qui reste, si possible, pour explorer des améliorations possibles de performance.

Pour le MQMM, nous avons identifié deux rôles d'utilisateurs: l'administrateur et l'utilisateur. Les objectifs du module utilisateur est de: gérer des incidents, traiter des actions et des audits. Ce module pourra déclencher une action corrective à partir d'une non conformité ou d'un hors contrôle. Cette action corrective permettra de suivre la résolution d'un problème jusqu'à sa correction. Deuxièmement, ce module pourra créer un audit et une action indépendamment d'une NC ou d'un HC. Troisièmement, ce module devra être capable de gérer le profil des utilisateurs. L'objectif du sous-module administrateur est de pouvoir modifier les champs (c.-à-d. les items dans les listes déroulantes) qui seront offerts à l'utilisateur. Pour ces deux rôles, il sera possible de consulter un tableau de bord qui offrira une vue d'ensemble aux différentes tâches en cours et fonctionnalités du MQMM.

Quant à l'amélioration de la performance des rapports existants, la mesure de l'amélioration de la performance est le temps de production en milliseconde. L'objectif serait de proposer

des avenues pour améliorer la performance et mettre en place une fonction de mesure de la variation de la performance. Nous ne nous fixons pas comme objectif d'implanter la ou les solutions que nous avons proposées, car certaines de ces solutions pourraient être complexes à implanter dans le système existant. Au fil du projet, par manque de ressources, nous avons abandonné cet objectif petit à petit en focalisant seulement sur la réalisation du prototype de MQMM. Le détail de ces décisions sera expliqué dans la section retour sur les objectifs.

2.2 RÔLES

Pour établir les rôles de l'équipe, nous nous sommes basés sur la méthodologie agile. Comme nous allons tous participer au développement, nous nous sommes seulement attribué les rôles satellites aux développeurs. Le but de ce type d'organisation est que l'équipe soit autogérée. Ce type d'organisation en étoile, nous permettra de partager l'information plus facilement et d'augmenter la coopération entre les membres de l'équipe.

Prénom	Rôle(s)	Responsabilités
Philippe Tessier	*Scrum Master*	Le *Scrum Master* aide l'équipe à travailler de façon autonome et l'assiste de façon à maximiser la valeur créée par celle-ci. - Planifier et écrire les comptes-rendus des rencontres hebdomadaires. - Assurer la bonne communication avec le client.
Marcel Tawe	*Project Owner*	Le *Project Owner* est responsable de maximiser la valeur du produit et le travail de l'équipe de développement. Dans notre cas, il s'agit de notre client. - Développer et maintenir une vision du produit et une stratégie de marché. - Comprendre le domaine d'activité du produit.
Pier-Olivier Faucher	*Lead QA*	Le *Lead QA* est responsable de définir et de s'assurer du respect des critères de qualité du produit. Il doit faire le suivi des fonctionnalités du produit en conformité avec le cahier des

		charges - Établir les critères de qualité du MQMM. - Établir les critères d'évaluation des solutions d'amélioration des rapports. - Définir les critères d'acceptation du projet.
Clément Rochon	*Lead R&D*	Le *Lead R&D* est responsable de choisir quelle technologie sera exploré pour améliorer les rapports de l'application déjà en place. - Rechercher les solutions possibles pour l'amélioration de la performance. - Explorer ces solutions et évaluer la performance des différentes solutions. - Si possible, appliquer les améliorations les solutions à la technologie présente.
Noemie Cloutier	*Lead Developer*	Le *Lead Developer* est responsable de gérer la planification et la conception du MQMM. - Connaître et analyser la structure du produit existant. - Estimer les tâches de développement du MQMM. - Partager ses connaissances avec les autres développeurs.

Tableau 1. *Rôles*

2.3 PHASES DU PROJET

Pour mesurer l'avancement du projet, nous avons décidé de séparer le projet en différentes phases. Certaines des phases se chevauchent puisque nous avons deux objectifs distincts pour le projet. Lors de notre proposition, nous avions deux autres phases, la phase de *grooming* que nous avons inclus dans l'analyse et la phase de QA que nous avons abandonné par manque de temps. Voici donc les phases que nous avons établies:

- Analyse
- Conception
- Rétrospective

La phase d'analyse est celle qui englobe entre autres la proposition. Elle sert à établir les objectifs du projet et les risques de celui-ci. C'est aussi durant cette phase que nous nous familiarisons avec les environnements de développement pour être prêts à évaluer les tâches qui mèneront à nos objectifs. À travers cette phase d'analyse, nous avons une sous-phase de *grooming*. Celle-ci sera assez courte dans le projet. Elle consiste à déterminer les tâches qui nous mèneront aux objectifs établis par l'équipe dans la phase d'analyse. Toutes les familiarisations qui ont eu lieu dans la première phase servent à obtenir l'estimation la plus exacte possible des travaux à accomplir. Après cette phase, le *Product Owner* se doit de prioriser les tâches que nous avons déterminées pour nous assurer de maximiser la productivité de l'équipe de développement.

Durant la phase de conception, l'objectif sera pour nous de déterminer chaque semaine quels sont les objectifs pour l'équipe et le *Scrum Master* aura pour tâche de s'assurer que les développeurs ont tous les outils dont ils ont besoin pour atteindre leurs objectifs.

Finalement, on trouve la rétrospective. Cette phase permet de réfléchir aux processus utilisés lors du projet et aussi de vérifier si nous avons atteint les objectifs que nous nous sommes fixés au départ. C'est durant cette phase que ce rapport a été produit.

Dans ce projet, nous avons décidé d'adopter la méthodologie agile. Par contre, comme nous avons tous peu de cases horaires pour nous rencontrer, nous avons décidé de réduire les activités. Ainsi, nous aurons une rencontre par semaine qui nous servira de mise à jour. Le compte-rendu de ces rencontres nous permettra de garder la trace des tâches accomplies ainsi que de facilement suivre l'avancement du projet de semaine en semaine. Si une autre rencontre est nécessaire durant la semaine, nous avons prévu une autre case horaire pour cette deuxième réunion.

2.4 ARTÉFACTS

Pour lister les artéfacts qui seront réalisés pour ce projet, nous nous sommes basés sur les requis de l'école principalement. À ces livrables, nous avons ajouté le *backlog* qui contiendra l'ensemble des tâches qui selon nous constitue le projet. De plus, nous avons ajouté le rapport d'amélioration de la performance, ainsi que le MQMM, représentant les deux objectifs principaux du projet représenté plus haut. Vous pourrez trouver des références vers chacun des artéfacts à l'Annexe I.

Nom de l'artéfact	Phase	Description
Proposition	Analyse	Document descriptif du projet et des objectifs.
Document *Workflow*	Analyse	Document fonctionnel du module MQMM. Inclus la navigation, les interfaces et des cas d'utilisation.
Backlog	Analyse	Ensemble des tâches estimées pour atteindre les objectifs.
Modèle relationnel	Conception	Contient le diagramme de la base de données, les scripts SQL et une courte explication sommaire des entités.
Rapport d'étape	Conception	Contient le backlog et le progrès du projet.
MQMM	Conception	Représente le code à livrer pour le MQMM.
Rapport des rencontres hebdomadaires	Rétrospective	Regroupement des comptes-rendus des réunions hebdomadaires.
Présentation orale	Rétrospective	Présente les éléments techniques du projet aux autres élèves
Rapport final	Rétrospective	Rétrospective et évaluation du projet.

Tableau 2. *Artéfacts*

Au cours du projet, nous avons décidé de retirer l'activité d'étude des problèmes de performance des rapports. Deux raisons principales justifient cette décision. La première est l'accumulation de retard et aussi par la découverte d'artéfact qui fait en sorte que nous n'avions plus le temps nécessaire pour compléter cette partie du projet. La deuxième raison est l'absence d'un environnement de test. Au cours du projet, nous nous sommes rendu compte qu'il était très difficile de répliquer l'environnement de production pour tester le système SPC4MM. Ceci peut-être une des raisons qui explique la lenteur du système existant, puisqu'il est très difficile de répliquer cet environnement. Conséquemment, les développeurs actuels peuvent difficilement prédire la performance de leur code, car ils n'ont pas l'environnement d'essai avec le volume réel de données, etc. Ainsi, il aurait été difficile pour nous de faire des essais et ainsi de mesurer l'amélioration de la performance de façon rigoureuse.

Suite à la demande du client, nous avons produit un document présentant le modèle relationnel du MQMM afin de permettre plus de visibilité sur notre proposition, et ce, malgré que le document *Workflow* expliquait déjà plusieurs de ces aspects. Comme le tableau suivant le démontre, nous semblons avoir un peu trop de livrables pour le temps alloué au projet, ce qui a fortement diminué le temps que nous avons pu consacrer à la conception.

2.5 PLAN DE TRAVAIL

Suite à la définition des livrables, nous avons établi un plan de travail. Nous avons choisi de ne pas estimer, en heures, les tâches puisque nous avons adopté une structure agile. Ainsi, nous avons plutôt choisi de prioriser les tâches dans le *backlog* pour établir les priorités. Le tableau 3 présente le plan de travail contenant les différentes *stories* que nous avons créées ainsi que les livrables et leur date de remise.

Phase	Commence	Termine	Livrable(s)/Artéfacts
Analyse	2017-05-11	2017-06-13	Proposition

			Document *Workflow* *Backlog*
Conception	2017-06-13	2017-07-08	Modèle relationnel Rapport d'étape MQMM
Rétrospective	2017-06-25	2017-07-12	Rapport des rencontres hebdomadaires Rapport final Présentation orale

Tableau 3. *Plan de travail - Phases*

#	Commence	Termine	Tâches/Jalon	Livrable(s)/Artéfacts	Responsable
1.1	2017-05-11	2017-05-11	Rencontre initiale avec le chargé de projet	Rapport hebdomadaire	Tous
1.2	2017-05-17	2017-05-17	Lancement du projet	Fiche de renseignements Rapport hebdomadaire	Tous
1.3	2017-05-23	2017-05-23	Rencontre Scrum – Équipe	Rapport hebdomadaire	Tous
1.4	2017-05-30	2017-05-30	Rencontre Scrum – Équipe	Rapport hebdomadaire	Tous
1.5	2017-06-06	2017-06-06	Rencontre Scrum – Équipe	Rapport hebdomadaire	Tous
1.6	2017-06-13	2017-06-13	Rencontre Scrum – Équipe	Rapport hebdomadaire	Tous
1.7	2017-06-20	2017-06-20	Rencontre Scrum – Équipe	Rapport hebdomadaire	Tous
1.8	2017-06-27	2017-06-27	Rencontre Scrum – Équipe	Rapport hebdomadaire	Tous
1.9	2017-07-04	2017-07-04	Rencontre Scrum – Équipe	Rapport hebdomadaire	Tous
1.1	2017-07-11	2017-07-11	Rencontre Scrum – Équipe	Rapport hebdomadaire	Tous
1.11	2017-07-18	2017-07-18	Rencontre Scrum – Équipe	Rapport hebdomadaire	Tous
1.12	2017-07-25	2017-07-25	Rencontre Scrum – Équipe	Rapport hebdomadaire	Tous
1.13	2017-08-01	2017-08-01	Rencontre Scrum – Équipe	Rapport hebdomadaire	Tous
1.14	2017-08-08	2017-08-08	Rencontre Scrum – Équipe	Rapport hebdomadaire	Tous
2	2017-05-30	2017-06-13	Remise de la proposition de projet	Proposition de projet	Philippe
3.1	2017-06-06	2017-06-13	Création du backlog	Backlog	Philippe
3.2	2017-06-06	2017-06-13	Validation des Ui et du Workflow	Backlog	Philippe
4.1	2017-07-04	2017-07-11	Rédaction du rapport d'étape	Rapport d'étape	Philippe
4.2	-	2017-07-11	Remise du rapport d'étape	Rapport d'étape	Philippe
5.1	2017-06-27	2017-08-08	Coding	MQMM	Tous
5.1.1	2017-06-27	2017-08-08	CMM-87 - As a User, I can create, read, update and delete an audit	MQMM	Tous
5.1.2	2017-06-27	2017-08-08	CMM-88 - As a User I can create, read, update and delete an incident	MQMM	Tous
5.1.3	2017-06-27	2017-08-08	CMM-89 - As a User I can create, read, update and	MQMM	Tous

			delete an action		
5.1.4	2017-06-27	2017-08-08	CMM-90 - As an Admin, I can manage the user's fields	MQMM	Tous
5.1.5	2017-06-27	2017-08-08	CMM-91 - As an Admin, I can manage the roles of users within these QMP (audit, incident, action) -	MQMM	Tous
5.1.6	2017-06-27	2017-08-08	CMM-92 - As a User, I can use the dashboard to view the QMP I'm a part of (audit, incident, action) -	MQMM	Tous
5.1.7	2017-06-27	2017-08-08	CMM-93 - As a Admin, I can use the dashboard to view the QMP I manage.	MQMM	Tous
5.2	-	2017-08-08	Remise du code	MQMM	Noémie
5.3	-	2017-08-11	Présentation	Présentation	Tous
6.1	2017-08-08	2017-08-11	Préparation de la présentation	Présentation	Tous
6.2	-	2017-08-11	Présentation	Présentation	Tous
7.1	2017-07-25	2017-08-15	Rédaction du rapport final	Rapport final	Philippe
7.2	-	2017-08-11	Remise du rapport final	Rapport final	Philippe

Tableau 4. *Plan de travail*

Notre plan de travail a été respecté, malgré que nous avons dû retirer la remise du rapport d'amélioration de performance puisque nous avons abandonné cet objectif. De plus, nous avons commencé la rédaction du rapport final plus tôt lorsque la remise a été fixée à la mi-août et que la rédaction de celui-ci faciliterait beaucoup la préparation de la présentation finale.

2.6 RISQUES

Le tableau 5 présente la liste des risques identifiés suite à l'analyse. La plupart des risques identifiés se situent au niveau des communications avec les clients et le manque de documentation sur le projet actuel.

Risque	Impact	Probabilité	Mitigation / atténuation
Manque d'expertise dans les systèmes de gestion de workflow	Moyen	Élevé	Recherche sur le fonctionnement d'application similaire (Sharepoint, Benz Info Solution)

Divergence entre les besoins du client et le travail réalisé	Élevé	Moyen	Rencontre hebdomadaire et approbation des propositions par le client.
Manque d'expertise dans le ORM nHibernate	Moyen	Moyen	Lecture de document et procéder de façon itérative.
Insatisfaction du client pour le code produit	Faible	Élevé	Revue de code par les paires et aussi par un représentant du client.

Tableau 5. *Risques*

Au cours du projet, nous avons apporté quelques modifications à la liste des risques. Tout d'abord, la *Divergence entre les besoins du client et le travail réalisé* et *l'Insatisfaction du client pour le progrès du code du prototype* ont augmenté au cours du projet. Nous avons eu plus de difficulté à communiquer avec eux et à leur faire comprendre notre proposition de conception initiale. De plus, il y avait une certaine divergence entre les méthodes proposées par le PO et les méthodes que nous pensions utiliser. Ensuite, nous avons ajouté le *Manque d'expertise dans la technologie ORM nHibernate* puisqu'aucun d'entre nous n'avait travaillé avec ce logiciel libre auparavant et son utilisation dans le projet était centrale et complexe.

CHAPITRE 3

CONCEPTION

3.1 ENTITÉS

Voici la description des entités que nous avons identifiées suite à notre compréhension de la logique d'affaires désirée. Pour chacune de ces entités, nous décrirons leurs rôles et leurs interactions avec les autres entités. De plus, nous listerons une courte description des champs que cette entité pourrait contenir.

3.1.1 QUALITY MANAGEMENT PROCESS

Le Quality Management Process, aussi appelé QMP, est l'entité centrale du module MQMM. Il représente la gestion d'un événement de gestion de la qualité et tient compte de l'avancement du processus. Dans le document d'analyse de Improver qui nous avait été donné, le QMP est équivalent à l'audit. Cette nomenclature est plus proche des termes d'assurance qualité utilisés et nous a été proposée par le client.

Le QMP sera un contenant pour les événements d'un processus d'assurance qualité typique. Par exemple, dans le cas d'une non conformité, on créerait un QMP contenant un incident qui serait lié à cette non conformité. Ensuite, on pourrait ajouter une action de type corrective à celui-ci. Finalement, on pourrait ajouter une validation pour valider qu'il s'agisse de la bonne action corrective proposée. Voici quelques événements identifiés à l'aide du document de spécifications fourni : l'action, l'incident, la validation, la vérification et l'analyse. Dans un cas idéal, tous les événements hériteraient de la même entité.

Pour la conception du module MQMM, il serait intéressant de pouvoir créer des gabarits de QMP qui contiendrait des événements prédéterminés (c.-à-d. créés par les utilisateurs) facilitant ainsi le travail de suivi des problèmes qualité :

Date de création : DateTime
Date d'ouverture : DateTime
Date de modification : DateTime

Date de fermeture : DateTime
Date d'archivage : DateTime
Responsable : Utilisateur
Documents : List<Document>
Événements : List<Événement>
Processus : List<Processus>
Procédures : List<Procédure>
Priorité : Decimal
Description : String
Commentaire : String
État :

- Brouillon : sans date d'ouverture
- En cours : avec une date d'ouverture
- Fermé : avec une date de fermeture
- Archivé : avec une date d'archivage

3.1.2 ACTION

Une action représente une prise d'action par les utilisateurs du système. L'action en question peut être corrective ou simplement un autre type d'action. La majorité du *workflow* de l'action se fait à l'extérieur du système, mais cet événement sert à garder les traces de ce qui a été fait. Par exemple, un utilisateur demande à quelqu'un d'aller calibrer une machine.

Date de création : DateTime
Date de modification : DateTime
Date de réalisation : DateTime
Description : String
Commentaire : String
Chargé : Utilisateur
État :

- En cours : avec une date d'ouverture
- Terminé : avec une date de réalisation

3.1.3 INCIDENT

Les incidents représentent des événements, par exemple, une pièce manufacturée possède une ou plus de ses dimensions hors contrôles (HC), une pièce possède une non conformité (NC). Un incident peut aussi être un accident sur le lieu de travail, une plainte de client ou un problème avec un processus ou un équipement. Lorsqu'il s'agit d'une NC ou d'un HC, les pièces, processus et procédés affectés sont indiqués dans la description de l'incident. La sévérité et la priorité de l'incident pourraient être indiquées par le type d'incident au lieu d'une table séparée:

Date de création : DateTime
Date de modification : DateTime
Date d'ouverture : DateTime
Date de fermeture : DateTime
Description : String
Commentaire : String
Chargé : Utilisateur
Sévérité : Niveau de sévérité
Type : NC, HC ou autre
Incident : Lien vers NC ou HC
État :

- En cours : avec une date d'ouverture
- Terminé : avec une date de fermeture

3.1.3.1 NON CONFORMITÉ

La non conformité n'est qu'une information supplémentaire concernant un incident. Il s'agit d'un incident qui est lié à une non conformité. Comme les non conformités ne sont pas sauvegardées dans la base de données, nous pourrions créer une entité contenant des détails sur ladite non conformité.

3.1.3.1 HORS CONTRÔLE

Comme pour la non conformité, la notion de hors contrôle est aussi un incident. Il s'agit d'un incident qui a été lié à un hors contrôle.

3.1.4 VALIDATION

La notion de validation est un événement associé à un QMP. La validation sert à valider les informations qui se trouvent dans un QMP spécifique. Par exemple, si un utilisateur ajoute une analyse à un QMP, il peut vouloir la faire valider par un autre utilisateur. Il peut ainsi créer une validation.

Date de création : DateTime
Date de modification : DateTime
Date de validation : DateTime
Chargé : Utilisateur
Critère d'évaluation : List<String>
Commentaire : String
Valide : Boolean
État :

- En cours : sans date de validation
- Valide: si valide égal 1
- Invalide: si invalide égal 0

3.1.5 VÉRIFICATION

La notion de vérification est un événement associé à un QMP. La vérification sert à indiquer si une action a bien été exécutée ou si l'incident a été corrigé. Elle contient le même genre d'information que la validation.

Date de création : DateTime
Date de modification : DateTime
Date de vérification : DateTime
Chargé : Utilisateur
Critère d'évaluation : String
Commentaire : String

Valide : Boolean
État :

- En cours : sans date de vérification
- Valide : si valide égal 1
- Invalide : si invalide égal 0

3.1.6 ANALYSE

L'analyse est un événement associé à un QMP. Elle contient le même genre d'information que la vérification et la validation. Dans le cas d'une analyse plus poussée, il serait possible d'ajouter des champs associés aux résultats possibles de cette analyse. De plus, il serait intéressant d'offrir la possibilité aux administrateurs de créer différents types d'analyse requérants certains champs en particulier.

Date de création : DateTime
Date de modification : DateTime
Date de réalisation : DateTime
Description : String
Commentaire : String
Chargé : Utilisateur
Causes : List<Cause> (description)
État :

- En cours : Avec une date de création
- Terminé : Avec une date de réalisation

3.1.6.1 CAUSE

Les causes peuvent être associées à une analyse et représentent ce qui a provoqué un incident. Notons que les causes pourraient également être un champ texte à l'intérieur de l'analyse au lieu d'être une table séparée.

3.1.7 DOCUMENT

La nécessité d'ajouter un document ou un lien à une entité permet de mieux expliquer un QMP. Par exemple, si un client remplit un formulaire de plainte, ce formulaire pourra être

attaché directement à l'incident. Dans la conception actuelle proposée, les documents sont implémentés à chaque événement, mais ils pourraient aussi se trouver directement dans le QMP.

3.2 CAS D'UTILISATION

Cette section sert à décrire les cas d'utilisation, les plus simples, pour chacun des types d'événements identifiés pour le MQMM. Cela permet de mieux comprendre comment le système opèrera pour les utilisateurs. Les incidents et les QMP sont des cas d'utilisation à part entière tandis que les autres sont des sous-cas qui font partie du cas d'utilisation de la création d'un QMP.

3.2.1 CU1 : CRÉER UN INCIDENT

Ce cas d'utilisation se produit lorsqu'un utilisateur est en train de consulter une non conformité (NC) ou un hors contrôle (HC) et qu'il décide de créer un QMP pour celui-ci.

1. Le responsable consulte un incident
2. Le responsable crée un incident associé à cet incident
    a. Le système crée un QMP
    b. Le système crée un incident de type NC ou HC
3. Le responsable ajoute une action (voir action)
    a. Le système crée l'action dans ce QMP
4. Le responsable clôt l'action
5. Le responsable clôt l'incident

3.2.2 CU2 : CRÉER UN QMP

Ce cas d'utilisation se produit lorsqu'un utilisateur décide de créer un QMP à partir de zéro.

1. Le responsable crée un QMP
    a. Le responsable ajoute une action (voir action)
    b. Le responsable ajoute une non conformité (voir incident)
    c. Le responsable ajoute un hors contrôle (voir incident)
    d. Le responsable ajoute une validation (voir validation)
    e. Le responsable ajoute une vérification (voir vérification)
    f. Le responsable ajoute une analyse (voir analyse)

2. Le responsable clôt le QMP

Voici la liste des événements que peut contenir un QMP :

3.2.2.1 CU2.1 : CRÉER UNE ACTION

1. Le responsable crée une action corrective
    a. Le système crée une action dans ce QMP
    b. Le responsable ajoute un chargé
    c. Le système notifie le chargé par courriel
2. Le chargé réalise l'action
3. Le responsable clôt l'action

3.2.2.2 CU2.2 : CRÉER UN INCIDENT

1. Le responsable crée un incident
    a. Le système crée un incident dans ce QMP
2. Le responsable choisit le type d'incident (NC ou HC)

3.2.2.3 CU2.3 : CRÉER UNE VALIDATION

1. Le responsable crée une validation
    a. Le système crée une validation dans ce QMP
2. Le responsable valide ou invalide

3.2.2.4 CU2.4 : CRÉER UNE VÉRIFICATION

1. Le responsable crée une vérification
    a. Le système crée une vérification dans ce QMP
    b. Le responsable ajoute un chargé
    c. Le système notifie le chargé par courriel
2. Le chargé valide ou invalide

3.2.2.5 CU2.5 : CRÉER UNE ANALYSE

1. Le responsable crée une analyse
    a. Le système crée une analyse dans ce QMP
    b. Le responsable ajoute un chargé
    c. Le système notifie le chargé par courriel
2. Le chargé ajoute des causes
    a. Le système notifie le responsable par courriel

3.3 WIREFRAME

3.3.1 PAGE D'ACCUEIL

L'interface de la page d'accueil sera assez simple. L'objectif est de présenter aux utilisateurs, dès le départ, les informations qu'ils veulent visionner le plus souvent possible.

Figure 2. *Wireframe : Page d'accueil*

La section des mesures (c.-à-d. les Kpi) et des graphes servira à afficher des statistiques pour permettre aux usagers d'évaluer rapidement la santé du système de gestion de la qualité pour une usine. Le tableau 6 présente des exemples de données que l'on peut afficher aux usagers.

Métriques/Graphes	Unités	Type
Nombre d'audits, d'actions correctives et/ou d'incidents ouverts et fermés	Nombre par semaine ou par mois	Data blocks
Nombre d'audits, d'actions	Nombre par semaine ou par	Line graph, donut

correctives et/ou d'incidents	mois	chart

Tableau 6. *Exemples de métriques et de graphes*

La section tableau contient la liste des QMP ordonnable selon les paramètres choisis par l'usager. L'information la plus primordiale est l'état de celui-ci. Lorsqu'un utilisateur clique sur l'un des QMP, il peut consulter la vue utilisateur décrite plus bas. Le tableau contient aussi les boutons pour ajouter un audit, un incident et une action. Pour la création des tableaux et des graphes, nous recommandons l'utilisation de la librairie Kendo UI. Vous pouvez trouver plus d'information à ce sujet en annexe.

3.3.2 UTILISATEUR

L'interface de l'utilisateur du MQMM aura la configuration présentée ci-dessous. Cette interface présente un QMP. On y retrouve deux colonnes. L'une contiendra le *workflow* de l'audit et l'autre le formulaire actuel (c.-à-d. les détails et informations du QMP).

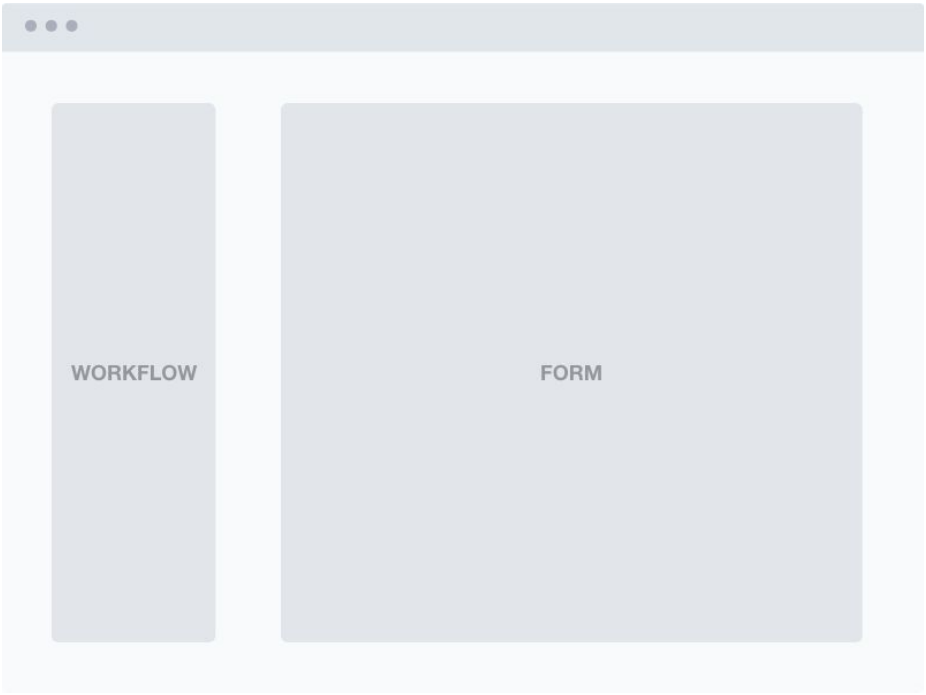

Figure 3. *Wireframe : Utilisateur*

La colonne *workflow* représente les différents événements qui constituent le QMP. Ainsi, il sera facile pour l'utilisateur de comprendre les étapes (c.-à-d. le workflow) de l'audit d'un seul coup d'oeil. C'est aussi cette section qui sert de navigation pour l'utilisateur. En cliquant sur un des événements du *workflow*, l'utilisateur peut accéder au formulaire de cet événement, mais aussi de détailler le *workflow* associé à cet élément.

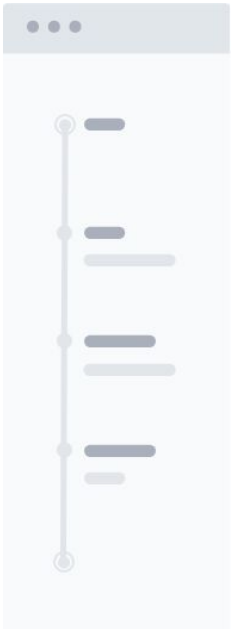

Figure 4. *Wireframe : Workflow*

À la droite de chaque élément, un cercle contenant une icône représente l'état de celui-ci. De plus, la couleur aide l'usager à comprendre l'impact de l'état rapidement. Chaque *workflow* commence par l'icône le début de l'événement décrit plus bas. Pour l'icône de fin, on le trouve seulement lorsque l'événement est clôt. L'usager peut voir le nom de l'état en survolant l'icône avec sa souris. De plus, l'état se trouvera aussi sur le formulaire sous forme textuelle.

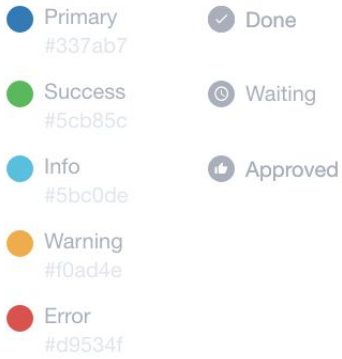

Figure 5. *Wireframe : États*

Quant à elle, la colonne *formulaire* changera constamment selon l'élément du *workflow* qui est sélectionné. Les différents formulaires sont décrits dans la section *formulaires* du document. L'interface des formulaires doit représenter une interface simple et indépendante de l'audit.

La figure 6 présente un exemple de *workflow* pour un des éléments :

Figure 6. *Wireframe : Workflow UX*

3.4 PROTOTYPE

Le prototype proposé doit s'insérer dans l'architecture de l'application SPC4CMM déjà utilisée par Netsen. Nous avons créé une section, dans le menu principal, pour le nouveau module MQMM. Cette section ne possède pas de fonctionnalités autres que de rediriger l'utilisateur vers les sections *Utilisateur* et *Administrateur*.

Figure 7. *Prototype : Menu*

3.4.1 UTILISATEUR

Le formulaire QMP est le premier écran qui apparaît lorsque la section *Utilisateur* est sélectionnée. On y retrouve l'interface présenté dans la section *wireframe* du même nom. Il contient les deux sections distinctes du workflow et du formulaire.

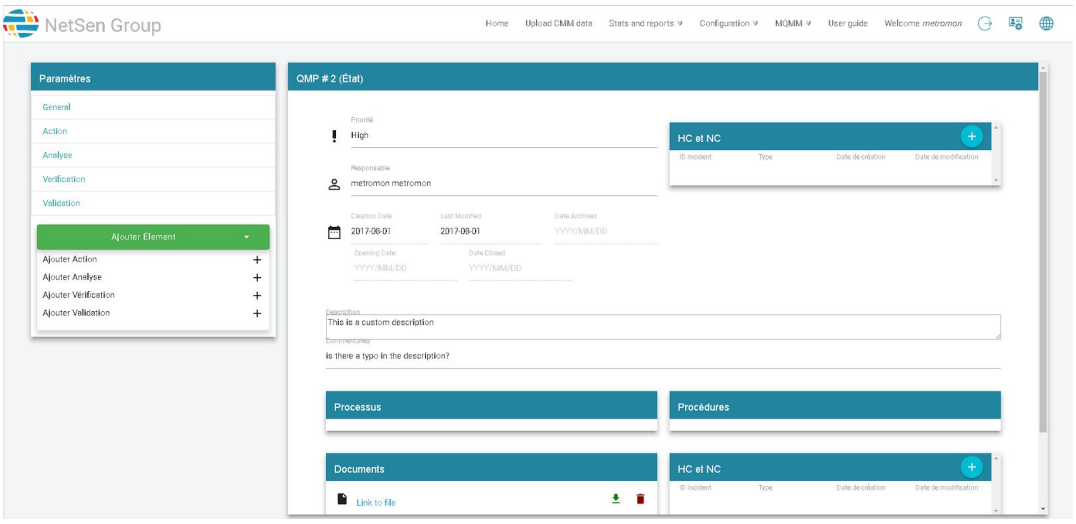

Figure 8. *Prototype : QMP*

À des fins de démonstration, la section *workflow* est remplie de façon statique, pour que chaque formulaire produit puisse y apparaître, tout en étant représentatif du produit final : L'ordre dans lequel les liens apparaissent représente l'ordre dans lequel les formulaires doivent être complétés et un clic sur le lien va modifier le formulaire. La section *workflow* contient également les boutons nécessaires pour ajouter de nouvelles étapes dans le *workflow* (par exemple, si un QMP requiert plusieurs actions pour être complété). Cependant, puisque le but de l'application est d'aider à faire de la gestion de qualité, il n'est pas possible de retirer des étapes du workflow. Il est toutefois possible d'archiver des étapes (typiquement lorsque celle-ci est complétée).

Le formulaire QMP contient toutes les données de haut niveau reliées à un événement : les dates importantes, le niveau de priorité, le nom de la personne responsable, une description du problème, un commentaire, ainsi que la liste de non conformités (NC) et de hors contrôle (HC) ayant déclenché ce QMP. Il est possible de manuellement ajouter des éléments à cette liste de HC et NC si jamais la liste n'est pas complète. Les sections du formulaire intitulées *Processus*, *Procédures* et *Documents* servent à lier des documents au QMP dans l'éventualité où il s'agit d'un problème connu ayant une solution documentée.

Les formulaires *Action, Analyse, Validation* et *Vérification* contiennent sensiblement les mêmes informations soit : les dates importantes, nom du responsable, niveau de priorité, description et commentaire. Ces formulaires sont accessibles en cliquant sur leur lien respectif dans la section *workflow* (qui n'est pas présentée dans les captures d'écran, mais toujours présente dans l'application).

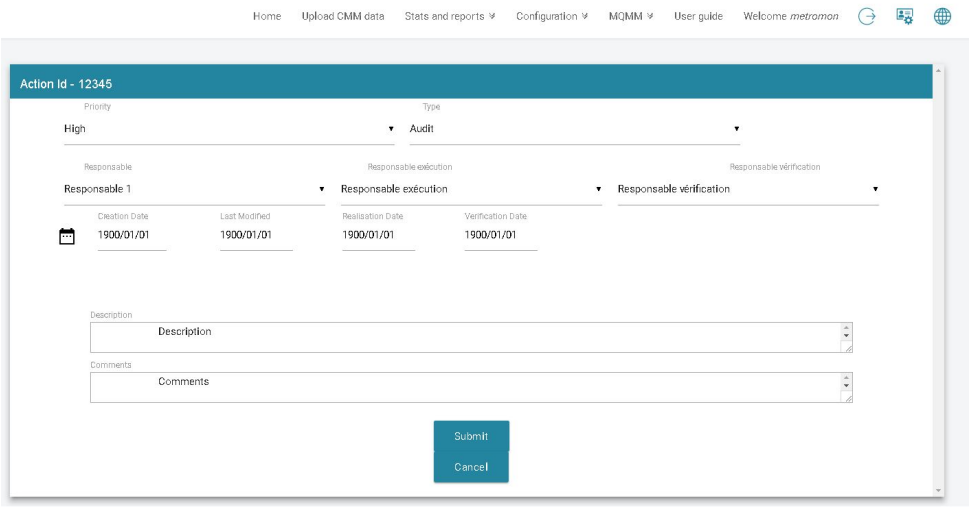

Figure 9. *Prototype : Action*

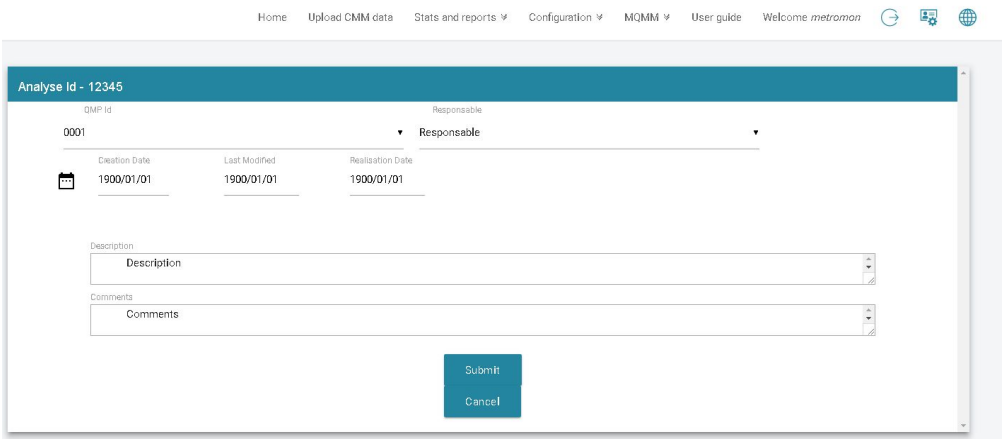

Figure 10. *Prototype : Analyse*

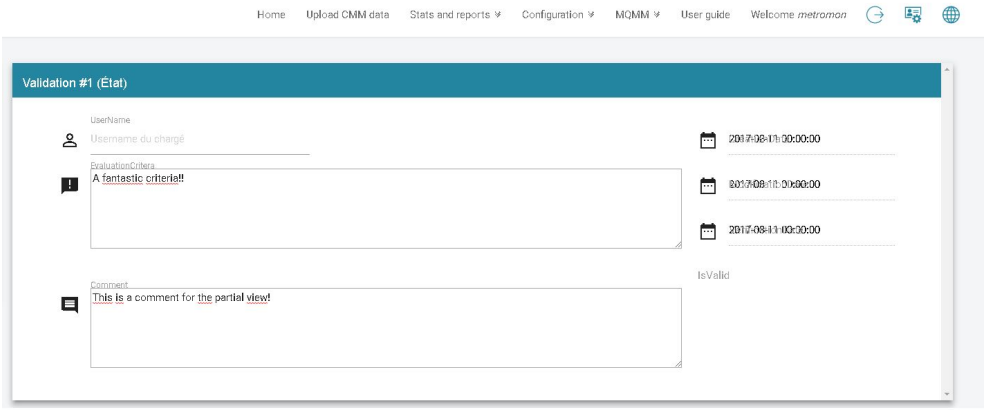

Figure 11. *Prototype : Validation*

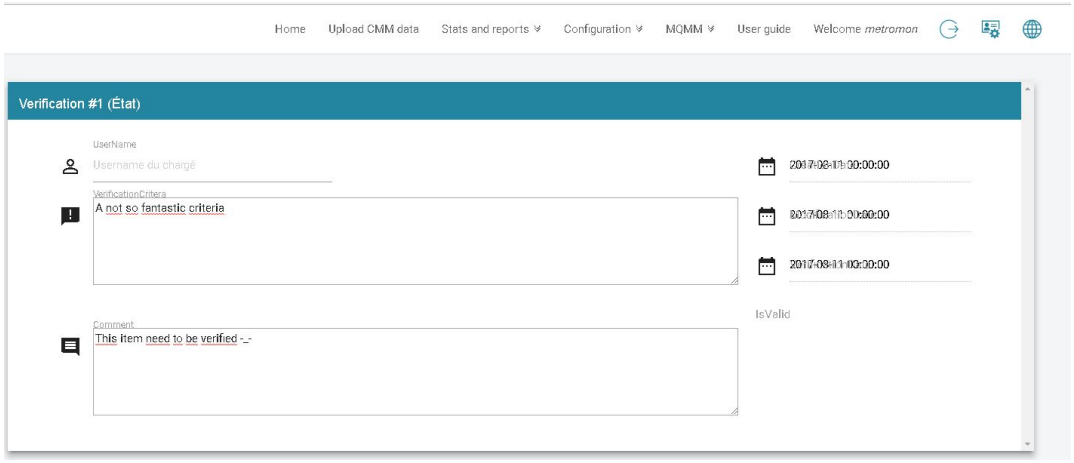

Figure 12. *Prototype : Vérification*

3.4.2 ADMINISTRATEUR

La section *Administrateur* permet de configurer les options accessibles depuis les formulaires. Comme chaque écran de cette section est structuré de la même façon et permet d'exécuter les mêmes tâches, une seule capture d'écran est utilisée pour représenter toute cette section.

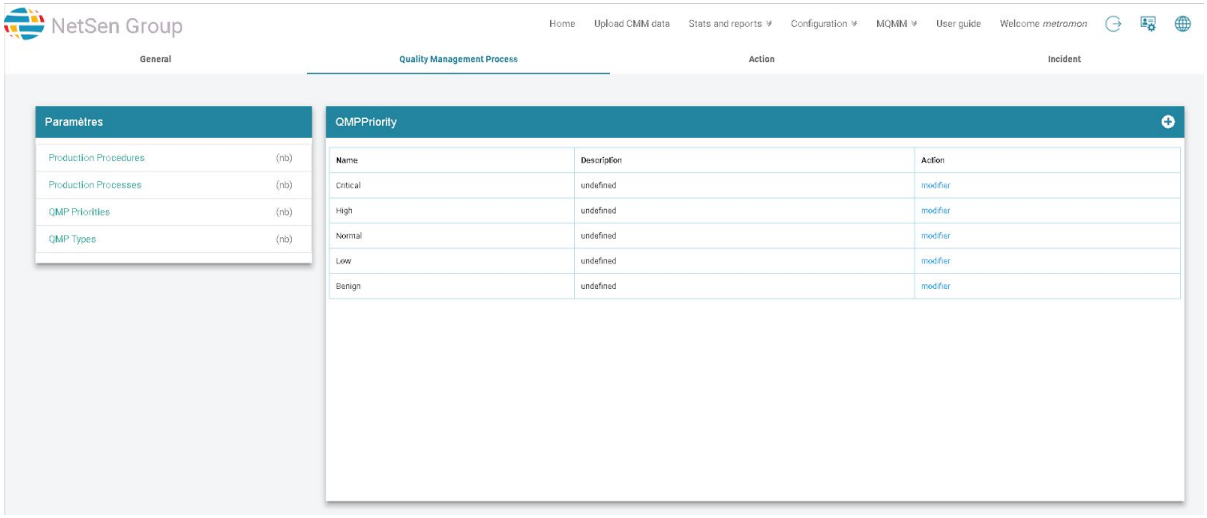

Figure 13*. Prototype : Exemple d'écran administrateur*

Cette section comporte quatre onglets représentant une section générale, ainsi que différentes entités présentes dans MQMM (QMP, Actions et Incidents). Encore une fois, l'écran est divisé en deux sections : *workflow* et *formulaire*. La section *workflow* permet de naviguer entre les différentes options disponibles pour un formulaire donné. Le contenu peut changer, mais le principe reste le même : en cliquant sur une entrée du *workflow,* le contenu de la section *formulaire* est modifié pour refléter ce choix.

La section *formulaire* contient un tableau présentant les options qui seront disponibles aux utilisateurs. Ces options ont uniquement deux propriétés à afficher : le nom et une description. La troisième colonne du tableau permet de modifier les entrées, mais pour éviter de créer des incohérences dans le reste du prototype, il n'est pas possible de supprimer des entrées. Il est cependant possible d'ajouter de nouvelles entrées en fournissant un nom et une description.

3.5 BILAN DE CONCEPTION

Lors de la phase de conception, nous avions comme objectif de réaliser un prototype de MQMM pour obtenir des commentaires du client. Nous avons passé beaucoup plus de temps que prévu sur la conception du prototype logiciel. En effet, il nous a fallu beaucoup d'efforts pour réussir à simplifier les spécifications que le client nous avait fournies.

À la suite de notre étude des documents de spécifications ainsi que de notre proposition, nous avons élaboré une première ébauche de conception avec le document *Workflow* qui représente dans un premier temps notre vision préliminaire des fonctionnalités que le projet MQMM devait représenter à l'aide de cas d'utilisation et de diagrammes de séquence. De plus, plusieurs esquisses des fonctionnalités de la page d'accueil et du module de configuration des utilisateurs y ont été offerts au client.

Ensuite, la conception d'un modèle relationnel a été réalisée afin de clarifier les liens entre les différentes entités du module MQMM ainsi que le rôle représenté par les objets du schéma. C'est à partir de ce modèle relationnel que nous avons réalisé les différentes classes d'accès aux données et effectuer le *mapping* dans notre projet.

Lors de la conception des formulaire de notre module, nous avons utilisé le modèle relationnel ainsi que les spécifications offerts par le client afin de créer des formulaires adéquats. À des fins d'usabilité et de faire plus facilement le suivi des étapes du QMP, nous avons décidé lors de la conception d'ajouter le menu latéral gauche contenant les éléments présents dans notre module. Afin de protéger les données, nous n'avons pas ajouté de fonctionnalité de suppression dans notre module.

En plus des documents de conception que nous avons produits, nous avons aussi fourni au client un prototype fonctionnel dans lequel nous avons effectué quelques accès à la base de données pour faire une preuve de concept. Avec le prototype, nous avons aussi créé un script SQL pour générer la base de données nécessaire pour faire fonctionner le prototype. Dans le prochain chapitre, nous reviendrons sur nos objectifs et sur notre rétrospective du projet.

CHAPITRE 4

RÉTROSPECTIVE

4.1 RETOUR SUR LES OBJECTIFS

Comme nous l'avons expliqué au début de ce rapport, nous avions initialement deux principaux objectifs. Le premier était de concevoir un prototype de MQMM. Ensuite, nous voulions explorer des améliorations possibles pour l'application existante qui a des problèmes de performance lors de la production des rapports de qualité pour une usine complète.

Durant le projet, nous avons rapidement constaté qu'il serait difficile de compléter ces deux objectifs. Comme notre objectif premier était le MQMM, nous avons concentré nos efforts sur celui-ci. Au cours de la conception, après avoir remis le document *Workflow*, le client nous a demandé de produire un document de ERM. Cela a aussi ajouté à la charge de travail que nous avons dû produire avant de commencer la conception.

Finalement, comme il a été décrit, nous avons produit un prototype de haut niveau. Lors de réunion avec le client, nous avons remarqué qu'ils étaient hésitants avec la proposition initiale que nous lui avions remise. Il semblait risqué pour nous d'essayer de livrer un produit qui pourrait être très loin des attentes du client. De plus, après quelque temps, nous avons constaté que la structure du logiciel existant SPC4CMM était très complexe et qu'ajouter des fonctionnalités s'avèrerait plus complexe que nous l'avions anticipé. Ainsi, nous avons pris la décision de produire un prototype de haut niveau intégré au système existant. Le client pourra ainsi plus facilement évaluer si celui-ci correspond à ses besoins. S'il pense que le prototype effectue bien ce qu'il désire, il pourra compléter le développement lors d'une prochaine itération avec une autre équipe.

4.2 POST-MORTEM

La dernière phase du projet était la rétrospective. Durant cette phase, nous nous sommes réunis pour réfléchir au projet et nous avons tenté d'évaluer les processus utilisés lors du projet. Voici donc certaines des conclusions auxquelles nous sommes arrivés suite à cette réunion.

Tout d'abord, nous nous sommes questionnés sur les processus utilisés. Comme nous l'avons expliqué auparavant, nous avions décidé de nous rencontrer une fois par semaine. Cette réunion durait une heure et elle servait à faire le point avec le client et notre superviseur. Nous pensons que cette réunion aurait dû être écourtée pour nous laisser du temps pour se rencontrer sans le client et produire des livrables. De plus, nous avons à plusieurs reprises planifié une deuxième réunion pour avancer le projet. Nous aurions dû prévoir cette réunion chaque semaine pour nous assurer d'avoir un moment pour partager nos connaissances.

Ensuite, pour ce qui est de notre équipe, plusieurs d'entre nous ont eu de la difficulté à fournir les efforts attendus durant le projet. Il a aussi été difficile de gérer la différence dans les efforts que chacun fournissait au projet.

Finalement, le client était très généreux de son temps, ils nous offraient sans cesse leur aide et tentaient de nous répondre le plus rapidement possible. Malgré cela, les machines virtuelles étaient très instables. Il est certain que cela nous a sauvé du temps lors du démarrage du projet, mais cette instabilité a beaucoup ralenti le développement. Même si le client réglait le problème le plus rapidement possible, la fenêtre que nous avions pour travailler sur le projet était souvent passée. Nous croyons aussi que l'architecture existante de SPC4CMM a aussi beaucoup ralenti le développement, nous en reparlerons dans nos recommandations.

CONCLUSION

En conclusion, la compagnie Netsen a soumis un projet de fin d'études visant la conception d'un module d'analyse et de contrôle statistique de procédé provenant de robots de mesure dans l'industrie automobile. Notre objectif était de concevoir un module de gestion des audits et de l'intégrer au produit existant nommé SPC4CMM. De plus, si le temps le permettait, le client aurait aimé améliorer les performances des rapports du système existant.

Nous croyons que les concepts du prototype MQMM permettra de développer un futur système de gestion des audits facilement adaptable à une autre industrie et facilement extensible. De plus, les interfaces prototypées sont très conviviales. Finalement, notre expérience avec l'architecture du logiciel existant SPC4MMM nous a permis d'élaborer des idées pour améliorer ce logiciel sans avoir fait une étude exhaustive.

Même si nous ne sommes pas arrivés à un produit fini (c.-à-d. un logiciel complet et fonctionnel), nous sommes persuadés que le prototype conçu et sa documentation permettront à Netsen de mieux comprendre les enjeux d'un tel module et de prendre des décisions éclairées sur son futur développement. Il ne faut pas oublier que ce projet est avant tout une activité pédagogique et nous avons beaucoup appris concernant ce domaine d'affaires, le contexte d'entreprise, l'architecture du logiciel existant et les technologies ASP.NET que nous avons dû apprendre.

RECOMMANDATIONS

À la suite de la réalisation du projet et des retours du client, nous avons une liste de recommandations. Ces recommandations sont basées sur deux aspects généraux, l'amélioration de la performance et l'amélioration de l'adaptabilité (c.-à-d. la facilité de faire évoluer le logiciel SPC4CMM).

ORM ET ARCHITECTURE TROIS TIERS

L'approche de conception de Netsen qui a été décrite au début du rapport possède un avantage au niveau de la séparation des rôles, de la persistance des instances d'objets ainsi que du *caching* des requêtes. La séparation des rôles vise une plus fine granularité quant aux actions que chacun des éléments du logiciel effectue afin de séparer la logique d'affaire de la logique système. La persistance ainsi que le *caching* permettent de garder en mémoire, autant que possible, les données de la BD et ainsi réduire la quantité de transactions effectuées entre le système et la base de données.

Cependant, cette approche de conception désuète comporte un certain lot de désavantages. Tout d'abord, le patron DAO – DTO – BO augmente la confusion quant à la responsabilité de chacune des parties du logiciel. Par exemple, le DTO est censé contenir la logique d'insertion, de modification et de suppression des données. Dans ce contexte, c'est le BO qui s'en occupe. De plus, le DTO duplique des champs du fichier de mapping modèle qui est lui aussi une duplication des champs du modèle (pur). Dans ce cas à quoi sert le DAO ? Selon la documentation, le DAO est sensé faire les connexions à la BD (ce que présentement, le DTO effectue), le DTO est sensé servir d'entité de transfert et le BO est celui qui s'en sert, mais cela soulève une autre question, le modèle (pur) sert à quoi dans cette situation? De plus, tous ces éléments sont générés par une architecture de fabriques et de fabriques de fabriques.

Bien qu'il y ait un certain avantage au niveau de la séparation des rôles (separation of concerns), nous craignons que cette approche limite la mise à l'échelle de l'application. Selon les plans du client, l'application est destinée à grossir et à devenir un système de gestion de la qualité et des changements génériques dans le but de desservir une gamme plus diverse d'industries. L'application devra donc être facilement modifiable par les développeurs, mais aussi offrir la flexibilité de configuration par les clients eux-mêmes.

Actuellement, l'ajout ou la modification de fonctionnalités nécessite une quantité considérable de modifications au code (c.-à-d. au modèle, au DAO, au DTO, au BO, aux ViewModels, etc.). Ces modifications multiples et parfois même redondantes vu la quantité de code *BoilerPlate*, diminue la productivité et augmente la probabilité d'erreurs. D'autres avantages, tels que la persistance des entités, tendent à être grandement diminués par l'utilisation d'autres méthodes et/ou d'autres architectures.

Il faut donc que Netsen se questionne sur les avantages de l'utilisation de cette architecture et connaisse ses inconvénients. Puisque cette architecture a le défaut de nécessiter beaucoup de modifications lors d'un changement, il serait intéressant de revoir l'utilisation du patron DAO – DTO – BO. Premièrement, la version utilisée de nHibernate est obsolète. Netsen ne profite donc pas de l'évolution des techniques de conception qui pourraient aider à résoudre cette problématique. Ainsi, il serait intéressant pour Netsen d'examiner une approche pour garder les fonctionnalités de l'ORM par l'utilisation de métadonnées dans les entités que l'on veut *mapper* à la base de données. Avec cette approche, les fonctionnalités nécessaires aux accès à la base de données et au transport de données seraient automatiquement prises en charge par le cadriciel. Cette approche supprimerait aussi la nécessité d'écrire et de maintenir les fichiers de *mapping*, les classes d'accès aux données ainsi que les fichiers de transport des données. La bonne nouvelle est que la version plus récente de *nHibernate* utilise cette méthode. Il serait ainsi encore possible d'écrire des DAO et DTO, mais il faut savoir que leur

valeur ajoutée n'est pas si intéressante si ça implique plus de code à modifier s'il y a des changements dans les éléments du modèle.

ARCHITECTURE ORIENTÉE SERVICES

Une autre recommandation serait de tirer avantage d'une architecture orientée services. L'architecture actuelle devrait progressivement prendre avantage d'approche plus récente de conception. Dans cette nouvelle architecture, chaque composante (c.-à-d. nommée service) sont des entités pouvant œuvrer indépendamment et qui communique entre elles selon des protocoles et portes d'entrée définie (c.-à-d. des messages). Chaque service peut être composé d'autres sous-services. Ces sous-services peuvent utiliser une technologie complètement différente et même hébergée à des endroits différents.

Les utilisateurs peuvent ainsi interagir avec ces services par l'entremise de diverses applications qui peuvent être indépendantes du serveur sur lequel les services sont situés. Cette particularité permet de déléguer aux clients tout le traitement d'affichage. La séparation de l'application en multiples services offrirait des avantages pour la performance, car elle permettrait une meilleure mise à l'échelle. Si de nouvelles fonctionnalités doivent être ajoutées, de nouveaux services peuvent être créés. De plus, la séparation en services serait un excellent commencement dans le but de migrer vers AWS ou Azure.

Tout d'abord, il serait préférable de séparer totalement le traitement d'affichage du reste de l'application. Il existe de nombreux cadriciels web pouvant aider à cette tâche, Angular.js, Backbone.js, React.js et Ember.js pour ne nommer que ceux-là. Cette séparation permet également d'utiliser des technologies indépendamment de celles utilisées pour les services aussi que les technologies utilisées par les autres applications clients. Par exemple, si un client désire utiliser une page internet pour accéder à l'information et aux fonctionnalités du système, le tout est possible sans que le système ait à le gérer. Si, chez ce même client, des employés voulaient accéder à l'application à partir d'un téléphone Android sans avoir à

passer par un fureteur? C'est également possible sans même modifier les services. Vous voulez avoir accès aux fonctionnalités à partir d'un jeu vidéo?

Pour les autres parties de l'application existante, une partie du travail est déjà effectuée. Vu la bonne séparation des rôles, il serait relativement facile de déterminer quels segments formeraient un service indépendant. Une piste d'idées serait :

- Mailing Service
- User and Permission Service
- Quality Management Service
- Reporting service
- Logging Service
- Mesure Service
- …

SYSTÈME DE GABARIT

Une autre idée pour l'amélioration potentielle de la section gestion de la qualité et du changement serait l'utilisation de formulaire avec champs arbitraire au même principe qu'un formulaire *Google Form*. Cette fonctionnalité donnerait une plus grande adaptabilité à l'application.

Donnons par exemple un type de processus de gestion de la qualité (QMP). Nous pourrions offrir aux utilisateurs (et aux clients) la possibilité de modifier les champs à remplir et possiblement avoir des actions par défaut à effectuer pour compléter le QMP lorsqu'un type spécifique est sélectionné. Ce principe est utilisé dans d'autres systèmes de gestion industrielle. Nous ne connaissons pas très bien les techniques et les bonnes pratiques utilisées pour créer des champs arbitraires. Nous supposons l'utilisation de base de données NoSQL (Cassandra, MongoDB, DynamoDB, etc.) dans le stockage de l'information, car ces bases de données n'impliquent pas une rigidité quant à leur schéma de tables. Cette approche

sacrifierait de l'efficacité au niveau de la recherche au profit d'une meilleure évolutivité et potentiellement de parallélisation. Il faut donc se questionner sur les avantages et les désavantages.

Azure et AWS offrent des services de base de données NoSQL (DynamoDB pour AWS et CosmosDB pour Azure) qui s'agencent bien avec d'autres de services offerts pour de l'analyse. Ça ne serait pas une mauvaise idée d'analyser les autres services avant de faire un choix entre ces deux fournisseurs.

SOMMAIRE

Nous croyons que plusieurs améliorations permettraient d'améliorer l'adaptabilité de l'application actuelle de Netsen. Pour ce projet de fin d'études, comme la durée du projet était très courte, une meilleure adaptabilité nous aurait permis d'arriver à un prototype logiciel plus complet. C'est pourquoi nous recommandons de revoir l'architecture *N-Tiers* actuelle et de mettre à jour l'ORM qui est obsolète. Ensuite, la séparation en services et microservices donnerait une meilleure évolutivité à l'application et améliorerait les performances une fois lancée sur Azure ou sur AWS. Finalement, développer une façon de laisser les utilisateurs modifier eux-mêmes les champs disponibles améliorerait adaptabilité à l'application et cela permettrait au client de se démarquer de leurs compétiteurs.

RÉFÉRENCES

Bootstrap. « Documentation ». En ligne. <http://getbootstrap.com/>. Consulté le 1er août 2017.

Dojo. « Dojo Documentation». En ligne. <https://dojotoolkit.org/documentation/>. Consulté le 1er août 2017.

InfoQ. 2016. « Characteristics of a Great Scrum Team ». En ligne. <https://www.infoq.com/articles/great-scrum-team>. Consulté le 6 juin 2017.

KendoUI. « Documentation». En ligne. <http://docs.telerik.com/aspnet-core/introduction>. Consulté le 1er août 2017.

NHibernate. « Documentation ». En ligne. <http://nhibernate.info/doc/>. Consulté le 1er août 2017.

NHibernate. « Versions ». En ligne <https://en.wikipedia.org/wiki/NHibernate>. Consulté le 3 août 2017

Netsen Group Inc. « Netsen Group Inc.». En ligne <http://netsengroup.com/>. Consulté le 8 août 2017

Telerik. « Telerik Demos». En ligne. <http://www.telerik.com/support/demos>. Consulté le 1er août 2017.

ANNEXE I

ARTÉFACTS

PROPOSITION

WORKFLOW

BACKLOG

MODÈLE RELATIONNEL

RAPPORT D'ÉTAPE

RAPPORT DES RENCONTRES

ANNEXE II

RAPPORT FINAL - HIVER 2017

Analyse statistique - Intervalles qualités - Robots de Fabrication CMM

Cliquez sur le lien

ANNEXE III

SPÉCIFICATIONS - PROJET MQMM

Cliquez sur le lien