SUPINFO Ecole Supérieure d'Informatique
455 Promenade des Anglais
06299 Nice

Thales Underwater Systems
525 ROUTE DES DOLINES
06560 VALBONNE (Sophia-Antipolis)

Mémoire de fin d'études pour l'obtention du diplôme de

Master Of Science

Expert en Informatique et Systèmes d'Information

Comment optimiser une application codée selon un paradigme procédural vers une programmation orientée objet dans le contexte contraint d'un projet déjà amorcé en entreprise ?

Présenté par **Romain DALICHAMP**

Session 2013 – 2014

Sous la direction de

DELAIGUE Marc-Henri, tuteur en entreprise

MONTES Michaela

1 TABLE DES MATIERES

2 REMERCIEMENTS

Je remercie Monsieur Xavier LABOURDETTE et Monsieur Bernard CHESTA pour m'avoir accueilli au sein du service Supports Technique et Produits et considéré comme un membre à part entière de l'équipe CSS.

Je tiens à remercier mon tuteur en entreprise Marc-Henri DELAIGUE, responsable fonctionnel du projet et responsable de mon apprentissage en entreprise pour m'avoir orienté sur ce projet durant toute la période de mon apprentissage et m'avoir accordé sa confiance concernant les décisions techniques que j'ai choisi de mettre en place.

Je tiens également à remercier Michaela MONTES, responsable technique du projet pour m'avoir conseillé et apporté son aide précieuse face à toutes les difficultés rencontrées.

Je remercie également tous les membres du Service Soutien et Service Client pour leur accueil.

J'aimerais remercier Monsieur Emmanuel OUYAHIA et Carole RIDEL du campus de Nice de l'école d'informatique SUPINFO ainsi que tous les professeurs qui nous ont assurés les cours pour leur présence, réactivité, enseignement et aide apporté tout au long de cette année scolaire.

Pour finir, je remercie tous les professeurs des années scolaires précédentes qui ont participés significativement à ma compréhension du monde de l'informatique et de l'entreprise.

3 UN PARCOURS ATYPIQUE POUR UN OBJECTIF INCHANGE

Depuis toujours motivé par le développement informatique, le collège a été une étape déterminante dans ma formation. Fortement encouragé à intégrer la section lycéenne « Science et Technologies de la Gestion », les possibilités d'intégration d'une école informatique post baccalauréat m'ont vraiment été restreintes. Ce qui m'a mené à une année non concluante au sein de l'Université de Nice qui requérait un niveau nettement supérieur à celui obtenu dans le domaine des mathématiques.

Déterminé à travailler dans le secteur de l'informatique, l'école privée « UFIP » (Université de Formation Inter-Professionnelle) m'a permis d'obtenir un premier diplôme, le Brevet de Technicien Supérieur Informatique de Gestion dont la deuxième année a été effectuée en alternance dans ma première entreprise : Monitoring Company.

Monitoring Company est une très petite entreprise de 9 personnes dans le secteur du Broadcast. J'y occupais le poste officiel de Web Designer (création des affiches pour les salons et représentations, intégration pour le site internet …) mais comme bien souvent dans une entreprise de petite taille, il est nécessaire de se diversifier ce qui amène également à améliorer et varier ses compétences dans de multiples domaines. J'ai donc mis en place la téléphonie IP de l'entreprise, développé un format générique de newsletters et effectué plusieurs tâches d'administration réseau.

L'année suivante, la licence professionnelle de l'IUT de Nice LPSIL: Licence professionnelle SYSTEMES INFORMATIQUES ET LOGICIELS - Spécialité IDSE: Informatique Distribuée et Systèmes d'Information d'Entreprise, est une formation que j'ai choisi d'effectuer en alternance chez Interbat Services. Spécialisée dans la dématérialisation d'appel d'offres, ma mission consistait en la création du Back Office du site internet principal de l'entreprise mais aussi également en le développement de fonctionnalité supplémentaire sur le Front office.

Ce site internet est architecturé selon la structure des marchés publics et leur fonctionnement. Il a donc été développé via un Framework « fait maison » par un développeur sur une durée de 4 ans. Les défis d'apprentissages ont été nombreux durant cette année : découverte, utilisation et amélioration d'un framework privé avancé, développement sur une version « de production » en temps réel, mise en place d'un outil de visionnage.

A la fin de ma 3ème année Post-Bac il m'était impensable d'arrêter mes études à ce niveau, je voulais faire un Master et, évidemment, en alternance.

C'est à ce moment que j'ai dût prendre une décision importante : signer à nouveau un contrat de professionnalisation avec la même entreprise Interbat Services dont je connaissais de façon assez complète le fonctionnement de l'entreprise, le framework ainsi que mon travail ou tenter le coup en postulant pour de nouvelles entreprises qui pourraient m'apporter une vision différente du monde du travail.

L'objectif visé était, après avoir travaillé dans 2 entreprises de petite taille, de décrocher un poste dans une entreprise à échelle nationale, voir européenne pour découvrir une nouvelle façon de travailler, de s'organiser mais aussi améliorer mes expériences professionnelles.

J'ai donc répondu à plusieurs offres d'emplois d'une multitude d'entreprises présentes sur Sophia-Antipolis telles Amadeus, Thales, Orange, AirFrance. Très rapidement deux entreprises m'ont contacté : Thales Underwater Systems et Orange. En raison d'une impossibilité d'embauche liée aux règles imposées par l'Organisme Paritaire Collecteur Agréé par Orange, mon choix s'est définitivement porté sur Thales qui était également mon choix prioritaire.

C'est ainsi qu'après mon embauche en contrat de professionnalisation chez Thales Underwater Systems j'entrais en première année de Master à SUPINFO.

Le choix de l'établissement de formation fût long et nécessita beaucoup de recherches et d'informations pour trouver une école qui remplisse tous les critères qui me correspondaient. L'école idéale délivrait évidemment des cours d'informatique, un diplôme reconnu par l'état et était donc inscrite au RNCP, le Répertoire National des Certifications Professionnelles, mais me permettait également de réaliser mes études en alternance sans avoir à repasser une nouvelle 3ème année dans cet établissement. Campus-ID, Epitech, Polytech, ITII, SUPINFO sont autant d'écoles reconnues dans la région mais seule SUPINFO était en accord avec ma recherche.

L'ensemble de ces années scolaires et d'alternance m'ont transformé aussi bien personnellement que professionnellement. Le sérieux, la rigueur, les compétences et relations acquis en entreprise m'ont servi dans mon apprentissage scolaire et, réciproquement, les connaissances acquises lors de ma formation m'ont été très utiles en entreprise.

Naturellement, la graduation de mon apprentissage scolaire s'est effectuée simultanément en entreprise et durant mes années de formations post Baccalauréat.
Si je devais la diviser sous forme de grandes étapes, elles pourraient être les suivantes :

- BTS et première année d'alternance, découverte de la programmation objet.
- Licence de Professionnalisation, apprentissage du design pattern « Modèle, Vue, Contrôleur » et des Framework les plus répandus
- Master 1 et 2, robotique, applications mobiles et conception d'un Framework privé

Il va donc sans dire que j'ai toujours tenté d'appliquer de la meilleure façon possible mes connaissances aux situations rencontrées en entreprise.

J'ai programmé un modèle de newsletter pour l'entreprise Monitoring-Company, ainsi que configuré toute la téléphone VOIP de l'entreprise. L'apprentissage de la programmation objet et la connaissance de l'entreprise m'ont alors orienté vers la fin de l'année dans le développement d'une application portable en JAVA permettant à tous les commerciaux de télécharger tous les documents de l'entreprise à partir de n'importe quel ordinateur équipé de JAVA.

Chez Interbat Services, mes nouvelles connaissances orientées « MVC » et frameworks de programmation m'ont facilité l'apprentissage du Framework privé de l'entreprise mais aussi son amélioration en profondeur lors du développement des fonctionnalités qui m'étaient attribuées. En plus du Back Office et du Front Office j'ai donc amélioré certaines parties des classes principales telles que la classe principale « Root ».

Finalement, ces deux premières entreprises m'ont permis d'acquérir l'expérience et la maturité nécessaire pour aboutir sur un projet représentant un véritable défi pour une personne en alternance : refactorer de façon optimale une application codée selon un paradigme procédural (appels successifs de procédures). En effet, l'application en partie développée par le stagiaire me précédant, était composée d'un code dit « procédural » et en contenait tous les désagréments : effets de bords nombreux, langages mélangés, développement à plusieurs très difficile etc… . C'est pourquoi en tant que développeur j'ai proposé et apporté une solution permettant de continuer le développement des fonctionnalités demandées tout en refactorant le code au fur et à mesure.

Mon expérience professionnelle est centrée principalement sur de la programmation JAVA et PHP impliquant obligatoirement l'utilisation de SQL, MySQL, Javascript, HTML, CSS mais j'ai également eu l'occasion pendant mes différentes années d'étude de découvrir d'autre langages tels que le Python, Ruby On Rails, C#, C++, XML et Android.

Le diagramme circulaire ci-dessous permet de rendre compte rapidement et facilement du pourcentage d'utilisation de chaque langage tout au long de mes études.

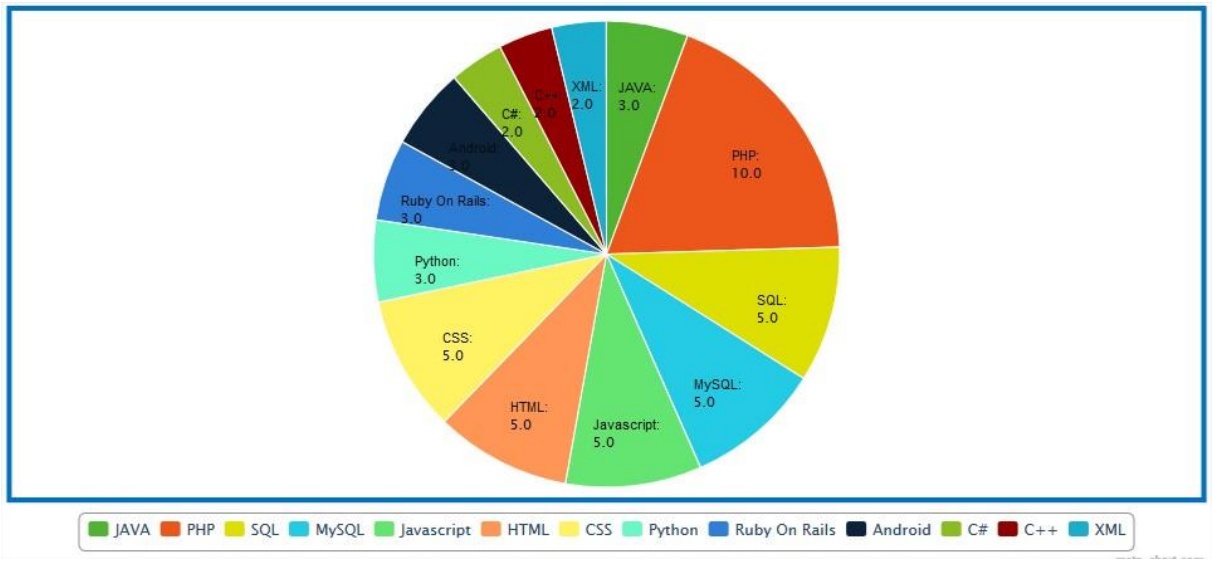


Figure 1 - Camembert Langages Programmation

De par mes expériences professionnelles, le PHP est le langage que j'ai le plus pratiqué. Cependant, dans le domaine du web, le fonctionnement des Frameworks les plus connus de chaque langage « côté serveur » est réalisé selon le même Design Pattern : MVC (Modèle, Vue, Contrôleur).

« Pylons » ou « Django » pour python, « On Rails » pour Ruby ou « Zend », « Symfony » et « Yii » pour PHP, tous ces Frameworks sont construits selon un fonctionnement relativement similaire composé de différents modules permettant :

- La réécriture d'URL
- L'utilisation de Template
- Le fonctionnement Objet avec une base de données
- L'héritage
- La gestion des erreurs

Le monde du web utilise les mêmes standards et langages du côté client quels que soient les langages de programmation utilisés du côté serveur. Css, HTML, Javascript, MySQL ou SQL ont donc été utilisés de façon récurrente. Toutes ces fonctionnalités communes permettent notamment l'apprentissage simple, facile et rapide d'un de ces langages web côté serveur à partir du moment où l'exercice a déjà été pratiqué une première fois.

Nous verrons au travers de ce mémoire comment j'ai pu mettre en place mes acquis au travers d'une problématique centrée sur ma dernière entreprise d'apprentissage, Thales Underwater Systems.

4 THALES, UNE ENTREPRISE INNOVANTE

Aujourd'hui côté à la bourse de paris, Thales est une multinationale qui a vu le jour en 2000 suite à la fusion de 3 sociétés pour n'en former qu'une seule, à l'origine : Thomson-CSF, Alcatel et Dassault Electronique.

Thales compte aujourd'hui un total de 65 000 salariés dont 73% d'ingénieurs et cadres dans 56 pays différents pour un chiffre d'affaires de 14,2 milliards d'euro.

Figure 2 - Carte mondiale

Parmi les entreprises en collaboration directe avec Thales nous trouvons :

- Conception et innovation IP, Cloud, accès fixe et mobile très haut débit
- Constructeur aéronautique Français
- Expert naval, Architecture, Construction, Ingénierie et Nucléaire
- Groupe Industriel & Technologique Français aéronautique, astronautique, défense et sécurité

L'Actionnariat de la société Thales est divisé en 3 grandes parties.

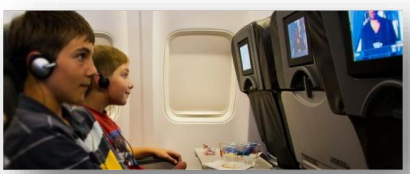
- État français *(Agence des Participations de l'Etat)* 26,63 %
- Dassault Aviation 25,53 %
- Flottant 47,84 % Autocontrôle 1,21 %
  Salariés 2,13 %

Figurant au classement des « Top 100 Global Innovators » du palmarès Thomson Reuters, les 100 entreprises les plus innovantes au monde, Thales prouve à chaque nouveau projet son expertise dans l'électronique et nouvelle technologie.

L'actualité Thales est actuellement en plein essor. Avec 65% de ventes à l'export et un objectif de 75% d'ici 2 ans, l'entreprise communique beaucoup sur ses nouvelles innovations. En voici quelques-unes très récentes :

- **Equipement internet** sur les avions

Avec le rachat de l'entreprise LiveTV (500 employés) au prix de 150 millions de dollars, Thales a récupéré deux compagnies américaines en tant que clientes (United Airlines et JetBlue) ainsi qu'une flotte de 450 appareils connectés.



L'objectif principal est de connecter 70% de la flotte mondiale d'ici 2025.

- **RAPIDFire** le nouveau canon de défense Anti-Aérienne

Dévoilé lors du salon de l'armement terrestre « Eurosatory », le système de défense anti-aérien permet de se défendre contre une attaque de drone là ou un système de défense « Classique » aurait été saturé, notamment par le nombre d'entités.

- **Sonar Flash** *(Folding Light Acoustic System for Helicopters)*

Le sonar Flash développé par Thales Underwater Systems de la gamme Compact Flash Sonics est un sonar actif à basse fréquence aéroporté.

Il équipe les hélicoptères de lutte anti-sous-marine permettant une grande portée de détection pour une utilisation vraiment rapide.

Figure 3 - Diversité Thales

Spécialisée dans les technologies de pointes, Thales est un groupe d'électronique constitué de plusieurs activités développées à l'échelle mondiale :

- Aérospatial
  - Avionique
  - Espace
- Transport
  - Systèmes de transport terrestres
- Défense & Sécurité
  - Système d'information et de communication sécurisés
  - Systèmes terrestres et aériens
  - **Systèmes de mission de défense**
- DCNS (société détenue à 35 % par le groupe)

Cette année, en 2014, Thales a également annoncé le 22 mai la reprise prochaine des activités dédiées de la société « Alcatel-Lucent » dans le domaine de la cyber-sécurité.

Les **Systèmes de Mission et de Défense** couvrent plusieurs filiales :

- Systèmes de Mission Aéroportés
- Systèmes de Combat de Surface
- Systèmes de Combat Electroniques
- **Systèmes de lutte sous la mer**

J'ai réalisé mon apprentissage au sein de la branche « Systèmes de lutte sous la mer », appelée Underwater Systems, et plus précisément au sein du service Supports Technique et Produits du site Sophipolitain de Thales

4.1 THALES UNDERWATER SYSTEMS (TUS)

Filiale à 100% Thales et spécialisée dans la vente de systèmes de défense et de lutte sous la mer, **Thales Underwater Systems** est implanté en France, en Angleterre, en Australie et aux Etats-Unis avec plus de 2000 personnes employées dans le monde.

Figure 4 - Bannière maritime

Avec une large gamme de produits pour sous-marins, bâtiments de surface, navire de lutte contre les mines, avions et hélicoptères, l'entreprise se tourne de plus en plus vers l'export. Les systèmes phares de l'innovation « TUS » sont principalement :

- **Systèmes aéroportés** : détection de sous-marins
- **Systèmes de bâtiments de surface** : détection de sous-marins
- **Systèmes de guerre des mines** : détection de mine(s) sous-marine(s)
- **Drones navals** : détection de mines de fond
- **Systèmes pour sous-marins** : sonars et téléphonie

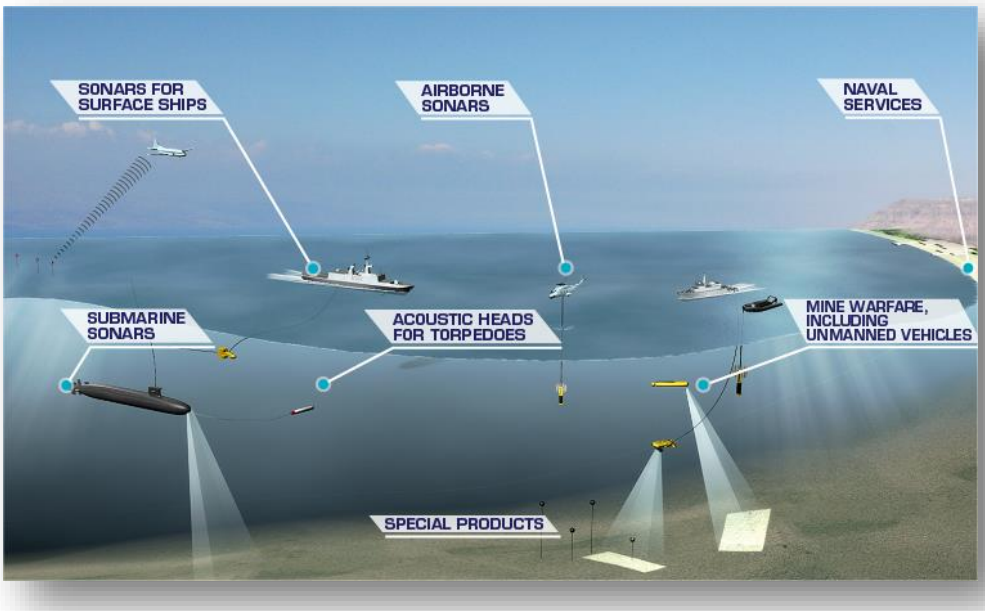


Figure 5 - Systèmes maritimes

Ci-dessus une réalisation graphique représentant les principaux matériels utilisés lors de missions de défense sous-marine.

4.2 ENTOURAGE PROFESSIONNEL DIRECT

Le domaine CSS, « Client Support and Services » a pour mission principale d'obtenir la satisfaction de nos clients tout au long du cycle de vie des systèmes de TUS SAS, dans le respect des objectifs de coûts, qualité et délais affichés par la Direction Générale ou inclus dans les contrats de soutien et services signés.

Pour réaliser cette mission et apporter son concours au succès de la société, le Domaine CSS s'articule autour de 5 activités principales :

4.2.1 Soutien Logistique Intégré (SLI)

Le service Soutien logistique Intégré qui assure, selon la spécialité dominante de chacun, les opérations préalables à la mise en service des produits, équipements et systèmes :

- Etudes logistiques
- Sûreté de fonctionnement
- Documentation et formation.

Les exigences du soutien logistique intégré sont prises en compte dès les phases de proposition et conception, pour optimiser le couple système opérationnel de soutien, assurer l'intégration optimale des éléments de soutien et la satisfaction finale du client.

4.2.2 Support Technique et Produit

Le service STP, Support Technique et Produits, est un service composé de 8 personnes. En charge de la résolution des problèmes techniques, C'est au travers de ce service que j'ai compris l'importance de la documentation dans l'entretien des systèmes vendus par Thales.

4.2.3 Offres de soutien / services et Business Développement

La partie offre de soutien, services et business développement consiste en l'arbitrage des conflits potentiels de marché entre les entreprises, la préparation et la mise à jour du plan de l'exploitation annuelle, la gestion des projets spécifiques, la participation à des évènements internationaux.

4.2.4 Programmes

Les « programs managers » ont pour mission la maîtrise, entre Thales et ses clients, des coûts, des délais, de la qualité et la technicité. Ils sont, via un budget qui leur est attribué, en relation permanente avec les autres équipes internes pour assurer le bon déroulement de la livraison d'un système. Ils sont présents tant au début du projet lors de la phase d'évaluation des besoins que lors de la livraison du produit au client.

4.2.5 Soutien Opérationnel

La partie soutien opérationnel du service STP consiste à apporter un soutien logistique, à entretenir et réparer tout système vendu à un client. Cette option de soutien est précisée dans les contrats et doit être assurée sur une période de 10 ans minimale.

4.2.6 Systèmes d'informations Spécifiques.

La gestion des données de soutien logistique est d'une importance cruciale pour assurer un Service Client efficace et optimisé tant sur le plan des obligations contractuelles que pour des besoins plus internes comme le suivi d'indicateurs de performances du soutien. Le projet sur lequel j'ai travaillé consistait justement en un développement d'une application Web connectée à différentes sources de données logistiques de Thales.

4.3 Structure d'Accueil

Thales est une entreprise internationale dont le système de fonctionnement est imposé selon plusieurs critères tels que la sensibilité des documents, la sécurité du groupe, la coordination des équipes ou encore les dépenses et moyens alloués. Dans mon cas, ces différents paramètres ont eu une forte influence sur mon travail et mes choix et décisions.

Tout d'abord, au niveau des ressources et connaissances techniques, étant le seul développeur du service, j'ai dû me reposer sur mes connaissances actuelles et de façon autodidacte, rechercher via internet ou la documentation disponible à Thales des solutions aux problèmes rencontrés.

Ensuite, pour la partie matériel et configuration réseau, des serveurs HTTP, test et production, ont bien entendu également été imposés car entretenus sur le réseau Thales et administrés via des configurations internes à l'entreprise. Donc les droits de connexion utilisateur, la puissance des serveurs, la taille de stockage, le débit de transfert et d'accès sont autant de paramètres qui sont difficilement modifiables.

Concernant l'environnement de développement, le poste informatique a été livré sans les logiciels nécessaires pour une telle application. Il a donc fallu faire des compromis et parfois attendre plusieurs mois pour prendre la bonne décision entre les logiciels idéaux et ceux testés et autorisés à l'installation par le groupe Thales. Le système d'exploitation sur le poste de travail : Windows XP, les navigateurs : Internet Explorer et Mozilla Firefox, ainsi que leur version ont également été imposés ce qui, notamment pour la portabilité de l'application web, ont parfois mené à des efforts de développement ou le redéveloppement d'une fonctionnalité complète.

4.3.1 Equipe de Travail

Au sein de l'équipe STP, constituée de huit personnes, deux sont situées en Bretagne et trois sont directement concernées par le même projet que moi.

Marc-Henri DELAIGUE, mon tuteur, est Spécialiste Ingénierie du Soutient. Son activité consiste en l'ingénierie des processus concernant le service client en incluant la définition fonctionnelle des systèmes d'information qui y sont liés.

Michaela Montes est ma responsable technique. Ingénieur Documentation Formation Client, elle m'a conseillé principalement autour des aspects SQL et Système d'Information. Son activité consiste en l'ingénierie de l'information client.

Responsable Centre de Compétences, **Bernard CHESTA** est en charge de l'équipe STP. Il est également le responsable de Marc-Henri et Michaela et donc, impliqué dans notre projet et dans toutes nos réunions.

Pour ma part, mon activité originelle consiste en le développement d'une application web depuis une spécification fonctionnelle rédigée par mon tuteur.

4.3.2 Fonctionnement de l'équipe

Articulée autour d'un projet commun chaque personne de l'équipe agit de façon directe sur l'évolution du projet. Dans un premier temps Marc-Henri donne les directives fonctionnelles, rédige la documentation et gère les ressources internes et externes impliquées dans le projet. A mon arrivée, il m'a remis une documentation fonctionnelle permettant de donner un axe, une direction à mes développements, permettant ainsi de prévoir en amont tout difficulté qui pouvait être rencontrée. Pour chaque développement ou problématique liée au Système d'Information, Michaela valide, corrige et décide des solutions techniques apportées. Finalement, Bernard contrôle au moyen de réunions hebdomadaires les avancées techniques et fonctionnelles du projet pour garantir sa réussite. Force de proposition et stricte dans l'organisation j'ai eu pour rôle d'évaluer et de développer chaque tâche fonctionnelle nécessaire à l'avancement du projet.

4.3.3 Communication

La communication au sein de l'équipe STP s'est effectuée principalement au sein de réunions d'avancement, de suggestions, ou de mails.

Les réunions d'avancement permettent d'informer des développements effectués, de se situer dans le planning mais également d'organiser la suite des tâches à effectuer. Elles sont effectuées de façon hebdomadaire par l'équipe au sein du service STP qui est concernée directement par le projet.

D'autres réunions étaient plus centrées sur la présentation du projet à une personne concernée par son utilisation future. Elles permettent de vérifier la bonne direction du projet et de récupérer les avis, remarques ou suggestions qui pourraient améliorer l'application.

Les réunions de présentation dévoilent une version stable de l'application à l'ensemble des personnes qui pourraient être concernées par son utilisation : dans une optique de promotion de l'application.

4.3.4 Conditions de travail

Le service CSS / STP n'intègre aucun autre développeur qui lui soit propre. Habituellement, la plupart les développements Système d'Information sont sous-traités à la Société de Services GFI Informatique. Je n'ai donc pas bénéficié de l'expérience d'une équipe de développeurs implantée dans le service depuis plusieurs années ce qui a fortement orienté mes décisions et ma façon de travailler qui a de par ce fait été beaucoup orientée conception. J'ai notamment été très influencé par la structure existante de l'entreprise : les logiciels autorisés, les versions des logiciels, les règles de sécurité, les processus des demandes sont tout autant de paramètres imposés que j'ai découverts au jour le jour. La pérennité des décisions a également été un facteur important à prendre en compte en amont de chaque nouvelle recherche. La mise en place d'un grand nombre de fonctionnalités a nécessité jusqu'à présent une implication systématique d'une ou plusieurs personnes pendant un certain temps, ce qui engendre des coûts et une marge d'erreur relativement faible. Les recherches ont donc souvent été privilégiées aux essais multiples. Ces personnes impactées par les choix et décisions concernant les développements peuvent être :

- Thales Global Services pour l'infrastructure

Le Système d'Information au sein de Thales Global Services a pour but de gérer l'infrastructure informatique au sein du groupe, ainsi que les applications, les droits utilisateurs, les serveurs, les bases de données, les machines virtuelles, les sauvegardes des données ainsi que la qualité du réseau.

- Les prestataires logistiques

En charges de la préparation des données pour une affaire et de l'utilisation de l'application développée, les prestataires sont directement impactés par toute nouvelle amélioration et modification.

- Relations internes

Pour résoudre certaines problématiques, trouver les solutions ou prendre les meilleures décisions possibles il a souvent été nécessaire de faire appel aux connaissances d'autres

personnes en interne notamment pour les aspects : sécurisation des données, utilisation du réseau et mise en production.

- Formation et connaissances

La capitalisation de la connaissance est fondamentale chez TUS. Toute formation supplémentaire y est encouragée, j'ai bénéficié de plusieurs jours complets et pour découvrir certaines des spécificités métier de Thales : Environnement Opérationnel Sonar, Palma. Aussi avant la fin de mon contrat, j'ai formé un employé TUS, David dans le but de transmettre la connaissance technique acquise tout au long du projet.

4.3.5 Répartition des tâches

Les tâches qui m'ont été attribuées pour ce projet en entreprise sont articulées autour de quatre axes principaux

Le développement d'une application web intranet dans un temps imparti selon les priorités fixées dans le temps au sein de mes deux années d'apprentissage

Etre force de proposition. Un mélange de l'ensemble des connaissances fonctionnelles et de connaissances techniques est souvent nécessaire pour trouver la bonne solution du premier coup. La documentation technique contenant toutes les demandes fonctionnelles ayant déjà été établie, il est à ma charge d'être force de proposition et de proposer le contournement d'un problème ou une meilleure solution.

La gestion des réunions est un des axes d'amélioration les plus important de mon apprentissage. Planifier dans le temps de façon hebdomadaire, préparer les principaux sujet de discussion en fonction de leur pertinence et priorités, animer pour déclencher des débats sur les points bloquants et confirmer les futures évolutions, puis rédiger les comptes rendus des réunions a au final pris autant d'importance que les développements. Bien que moins quantitatif que les développements, sans ce suivi et les directions données pendant ces réunions, le projet ne serait sans doute pas aussi avancé.

La planification est une étape cruciale de tout développement supplémentaire. Toute tâche évaluée en difficulté ou temps peut ensuite être priorisée avec les autres. Sans cette évaluation la visibilité du projet en entier ne serait limité qu'a la tâche en cours jusqu'à sa finalisation.

Il m'a également été attribué des tâches « implicites » que tout développeur se doit de suivre comme l'écriture d'un code de la meilleure qualité possible, l'ajout de commentaires, la recherche d'informations et le suivi des actualités en matière de programmation.

4.3.6 Organigramme du service Supports Technique et Produits

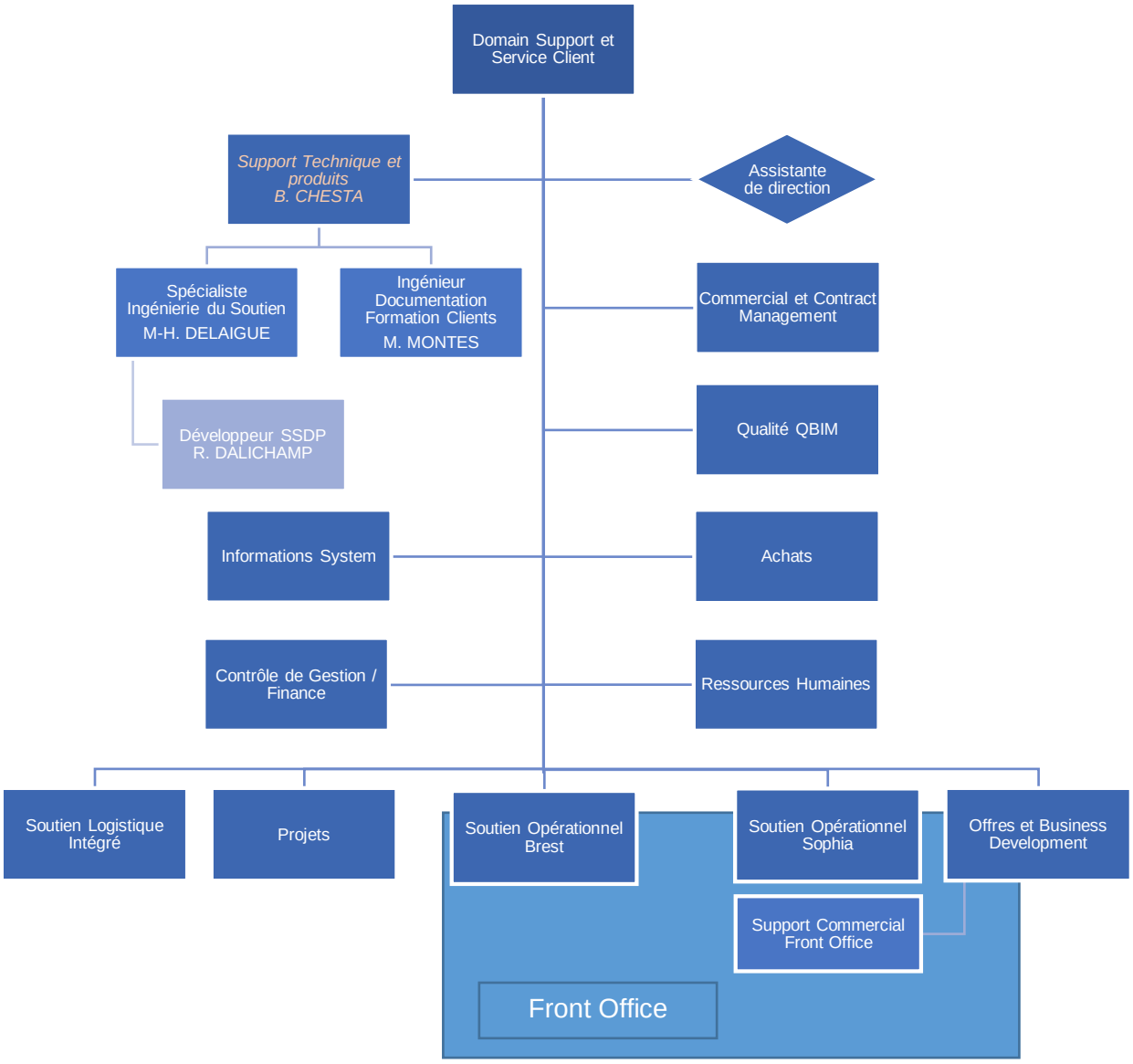


Figure 6 - Organigramme

Nous pouvons voir ci-dessus, le domaine support et service client est composé de plusieurs services.

Le service Support et technique produits est celui que j'ai intégré à mon arrivée chez Thales Underwater Systems.

Bernard CHESTA est le responsable Marc-Henri DELAIGUE qui est mon tuteur et Michaela MONTES, ma responsable technique.

Toutes les autres personnes des services sont dans mon entourage géographique, le même étage du bâtiment – mais ne travaillent pas directement sur le même projet que nous. Elles ne sont donc pas précisées de façon nominative dans cet organigramme.

5 UNE SITUATION ATYPIQUE

5.1 ENVIRONNEMENT DU PROJET

Arrivant chez Thales, et après une courte formation concernant le fonctionnement interne, les processus métiers et les règles de sécurités, je me suis vu attribuer pour mission durant la totalité de mes années d'apprentissage la conception et le développement d'une application nommée **SSDPM :** Sonar Support Data Package Maker.

De façon synthétique, **Sonar Support Data Package Maker** est une application développée en langage informatique PHP Hypertext Preprocessor. Elle est donc une application web dont l'utilisation se fait de façon client-serveur. Le but final et fonctionnel de cette application est de permettre la capitalisation des connaissances et l'utilisation en interne et de la documentation technique d'un système. Cette application doit permettre la mise à jour en temps réel par tous les collaborateurs mais également la remise ou la vente de modules (Packages) aux clients intéressés tels que les Marines de Défense de la France ou de pays étrangers. Copies d'écran en Annexe 2 et Annexe 3

Bien qu'une ébauche de la documentation technique de l'application soit existante et que je dispose de l'accès total aux sources, le temps d'apprentissage et de compréhension du projet fut relativement plus long que celui qui aurait été nécessaire lors de la reprise d'une application respectant les standards de programmation actuels.

Face à une application dont les bases avaient déjà été développées par un stagiaire précédent et au vu du planning mis en place concernant les améliorations et les avancées du développement, mon premier réflexe a été d'essayer de prendre en compte les raisons de l'existant et l'environnement dans lequel le projet a est situé.

Codée en programmation procédurale, paradigme basé sur le concept d'appel procédural. L'application n'avait pas été pensée pour permettre l'utilisation d'objets – via le paradigme de la programmation objets. De plus, ne respectant pas les standards web définis par le W3C, World Wide Web Consortium (W3.org) la compatibilité de l'application n'en était que plus réduite. Ces deux décisions prises en amont de mon arrivée vouaient l'application à une impossibilité de développement à plusieurs développeurs, à une maintenance difficile et à de nombreux effets de bord pour chaque amélioration supplémentaire développée.

Tenant compte de ces analyses, les premières actions effectuées ont été orientées apprentissage. En effet, mieux connaître l'application et le contexte dans lequel elle a été initialisée m'a permis d'avoir une vue globale sur sa finalité. Pour ce faire, le développement de nouvelles fonctionnalités de façon immédiate a été très utile. Que ce soit des retouches au

niveau du design ou l'intégration de scripts, pratiquer le code est une activité d'apprentissage qui m'a positionné à la place du développeur précédent.

De ce point de vue il a été plus facile de comparer l'existant avec mes connaissances personnelles acquises tout au long de mes études de mes expériences professionnelles.

Déjà, à ce niveau de découverte j'imaginais un axe d'amélioration possible à présenter à l'équipe dans le but de rendre un travail en fin de stage de meilleure qualité que celui trouvé précédemment. Ce qui, bien qu'en ayant trouvé une réalisation de très bonne qualité, aurait de toute manière été l'état d'esprit idéal à adopter lors d'une intégration à un projet de développement.

Ces huit premières semaines ont donc été consacrées à la remise en question et l'évaluation du travail existant en fonction de mes propres connaissances. Les possibilités d'actions n'étaient que très légères et mes connaissances évidemment encore non évaluées, la confiance de l'équipe ne s'acquérant que via la pratique et les réalisations.

A ce moment-là je n'avais qu'une vision très générale et peu expérimentée du devenir concret de l'application. *Comment mettre à profit mes connaissances à profit dans un cas déjà établi d'un projet qui ne demande qu'à avancer à partir de ses propres bases ?*

Naturellement, des priorités ont très vite été mises en place. Dans un premier temps l'installation complète d'un poste de travail fonctionnel, dans un deuxième temps l'évaluation en temps et l'organisation des tâches à effectuer, puis dans un troisième temps la conception des pratiques de développements qui me permettraient, plusieurs mois après, de réutiliser un maximum le code déjà écrit et qui allait l'être dans le but de gagner du temps par la suite.

5.2 SPECIFICITES DE L'ENTREPRISE

Ce premier état des lieux aurait sûrement été différent dans une entreprise de taille autre type PME (Petites et Moyennes Entreprises). Dans ce cas, toute décision attribuée sur plusieurs jours impacte directement ou indirectement plusieurs professionnels et donc leur travail de façon quantitative ou qualitative.

- Professionnels concernés

Tout d'abord, le service logistique est contraint au niveau des délais fixés par les commerciaux envers les clients. Les affaires – chez Thales, une affaire rassemble l'ensemble des informations concernant la vente, la conception, la production, l'installation, la configuration et le support d'un système pour frégate, sous-marin ou hélicoptère - ayant évidemment priorité sur le projet, nous devons nous adapter aux délais fixés par les autres équipes au sein de TUS, notamment à la remise de la documentation d'un système vendu à un client pour pourvoir fournir un accès à l'application aux équipes interne, aux commerciaux voire aux clients.

Pour chaque affaire, cette documentation est créée en amont. Notre application fonctionnant principalement suite à cette génération de donnée, il est indispensable d'identifier les zones d'activités annuelles – donc plusieurs mois à l'avance – de cette production pour permettre l'utilisation de l'application de façon stable et complète.

L'équipe du support est indirectement concernée par la création des packages. En effet, si la documentation concernant un système est ajoutée à temps et mise à jour, elle leur sera d'une grande aide.

Les commerciaux peuvent utiliser un package de documentation concernant une affaire de deux façons différentes : **l'utilisation** en mission ou la **vente**. Ils sont donc directement impactés par la bonne réalisation d'un package au préalable.

- **Problématiques liées à** la taille de l'entreprise

Comme affirmé précédemment, certaines recherches ont été induites par la taille de l'entreprise.
Thales est une entreprise développée à l'échelle mondiale et, qui plus est, dans le secteur de la défense. Il y a donc trois paramètres majeurs dont il faut tenir compte lors de toute recherche et décision :

- L'expérience

Existant depuis l'année 2000, le groupe. La finalité des systèmes créés par Thales est toujours à la pointe de la technologie mais elle ne va pas obligatoirement de pair avec l'utilisation de ressources innovantes puisque ces dernières doivent être testées et validées au niveau de la sécurité et ce à l'échelle du groupe avant d'être mises en place. Ainsi les versions des logiciels et des langages de programmation autorisées peuvent avoir un décalage de version.

L'ancienneté d'une entreprise de cette taille est également ressentie au travers des habitudes et méthodes de travail qui ont fait leur preuves et qui de par ce fait son figées et proposées à l'échelle du groupe. Il est donc parfois difficile de présenter et intégrer des nouvelles façons de travailler de la part d'un « apprenti » qui ne fait peut être « que passer » quelques mois comme beaucoup d'autres chaque année.

- La sécurité

En tant qu'entreprise dans le secteur de la défense, Thales Underwater se doit de gérer la sécurité de ses applications de façon générale. Certaines idées ou solutions n'ont donc pas vues le jour ou été mises en place de par le type de données qu'elles utilisaient.

L'utilisation d'une tablette Android à connexion 3G pour utiliser des données de l'entreprise est par exemple trop peu sécurisée pour le moment.

- Une gestion à l'échelle « Groupe »

Thales Underwater Systems est gérée par la multinationale « Thales Group ». Elle est donc soumise à certaines règles à cette échelle auxquelles elle ne peut déroger. Par exemple, tous les développements lourds doivent être sous-traités la société GFI.

Cette seule contrainte implique une externalisation des serveurs de développements, de test et de production ce qui a son tour implique une très faible possibilité d'action sur ces serveurs de par les droits d'accès ou même dans le choix du matériel et des logiciels.

Les recherches et décisions doivent donc être en adéquation constante avec l'existant du groupe. Cette affirmation semble logique et facile à réaliser, mais elle peut impliquer plusieurs semaines ou mois de travail, ce qui a été le cas dans le développement de l'application SSDPM, nous le verrons par la suite dans la partie décisionnelle en amont de la réalisation de certains modules.

5.3 Technologies en Rapport avec le Projet

Ces dernières années, les tendances dans le secteur de l'informatique et plus précisément dans le secteur du développement informatique sont à l'organisation, aux spécifications et à la réalisation de travail réutilisable et compréhensible par la communauté.

Il n'est donc pas surprenant de voir apparaître de nombreux « Wiki » internes dans des entreprises de toutes tailles qui permettent de stocker l'historique du développement d'une application sous la forme de documentation fonctionnelle, technique ou de cas pratiques.

Les outils de versionnage sont également de plus en plus utilisés. Ils permettent le travail à plusieurs développeurs mais aussi de

- Retrouver un « bug »
- Revenir à une ancienne version de l'application
- D'appliquer des « patch » correctifs
- D'utiliser et gérer plusieurs versions de développements

Les Environnements de Développement Intégrés ou IDE deviennent aussi indispensables. De plus en plus intelligents ils « comprennent » le code de tous les Framework existants et sont

configurables pour les spécificités souhaitées ce qui les rend assez dynamiques pour notamment permettre l'intégration d'un Framework privé.

Le Design Pattern « MVC » ou Modèle Vue Contrôleur est le plus utilisé dans le monde du web. Il permet de séparer de façon claire le code destiné à la base de données, le code destiné au traitement de la requête et le code destiné aux vues et design.
Les Framework les plus connus à utiliser ce design pattern dans le langage PHP sont :

- Symfony
- Yii Framework
- Zend Framework
- Joomla! Platform
- CakePHP

Tous les autres langages orientés web ont évidemment également leurs propres Framework que ce soit en Ruby, JavaScript, Java, Perl, Python, ASP .Net etc…

Concernant le langage de programmation PHP des nouveautés sont apparues dans la version 5.3 mais n'étaient pas utilisées dans l'application ce qui a d'ailleurs par la suite posé quelques petits problèmes de compatibilités. Il a donc été nécessaire d'en tenir compte.

Voici quelques-unes des nouveautés dans la version de PHP utilisée par l'application

- Les espaces de noms ont été ajoutés
    - Considérés comme des « Packages » en JAVA, les espaces de noms en PHP sont définis via le mot clé « namespace » et peuvent être utilisés via le mot « use ».
      Toutes les classes de cet espace de nom se retrouveront alors importées automatiquement.
- Fermetures et fonctions lambda
    - Permet d'assigner une fonction à une variable qui pourra ensuite être utilisée comme une fonction. Elle n'est pas assignée obligatoirement à une classe
- Déclaration des constantes hors d'une classe à l'aide du mot clé "const"
    - Permet de définir une constante sans avoir à utiliser la fonction « define »
- Accès dynamique aux méthodes statiques
- Exceptions imbriquées

Le refactoring (réusinage en français) qui peut être compris comme « amélioration du code existant » est un sujet d'actualité et qui coûte encore de nos jours beaucoup d'argent à certaines entreprises. N'étant pas aussi organisés dans le code, les développeurs formés il y a 10, 20 voire 30 ans, ne prenaient pas automatiquement en compte des facteurs qui paraissent intuitifs aujourd'hui. Beaucoup avaient été formés sur un paradigme de

programmation « procédurale » ce qui est visuellement et fonctionnellement beaucoup moins organisé que le paradigme « Objet ». Le refactoring peut permettre de gagner de l'argent ! Il permet de réduire significativement le nombre de ligne de code, donc de faciliter la maintenance et d'économiser des ressources pour la machine. Pour donner un ordre d'idée, après un refactoring complet d'une application il est courant d'obtenir un gain de performance pouvant aller de 10 à 30% pour une application web !

Pour information les premiers langages de haut niveau ont été créés à partir des années 1980, comme le C++ et le Perl. C'est à partir des années 1990 avec le gonflement de « l'ère internet » que des langages de types « RAD » Rapid Application Développement apparaissent. Orientés objets et fournis avec un environnement de développement intégré. Il est donc normal que la génération précédente de développeur ne maîtrise pas les mêmes concepts que ceux d'aujourd'hui d'autant plus que la communauté s'agrandit exponentiellement d'années en années.

Pour finir, il est important je pense de préciser que les premières spécifications formelles du langage PHP commencent à voir le jour. Pour un langage de programmation, la spécification est la source de référence pour sa syntaxe et son utilisation. Elle contient des informations détaillées sur tous les aspects du langage et définit un cadre pour son implémentation. Donc en cours d'élaboration et pas encore disponible (*Voir Webographie - Spécification Formelle PHP*), la seule et principale source de référence utilisée dans ce langage est la documentation qui elle est très bien fournie, et a une communauté participative (*Voir Webographie – Documentation PHP*).

Etre en veille technologique constante concernant les dernières innovations, les dernières améliorations ou les dernières spécifications des langages est une activité cruciale et indispensable pour un développeur. Le manquement à cette activité régulière peut très vite devenir critique concernant la sécurité de certains développements. Il est donc important en tant que développeur d'utiliser différents moyens de surveillance des nouveautés, ils peuvent être :

- Le site officiel

Pour le langage PHP la référence est php.net. Il permet de trouver la documentation technique complète nécessaire à l'utilisation de toutes les fonctions mais aussi et surtout des cas pratique d'utilisation postés en commentaires par les membres de la communauté. Toutes les mises à jour sont également détaillées sur ce site.

- Un « channel » IRC

Internet Relay Chat, est un protocole qui permet de dialoguer en temps réel avec d'autres utilisateurs. Les clients de ce protocole sont nombreux et gratuits ou payants. Ce qui est intéressant ici c'est, après la connexion à un serveur et un canal spécifique, la possibilité de dialoguer en temps réel avec beaucoup de passionnés experts dans le langage.

- Le forum de la communauté

Le forum de référence est developpez.net. Lorsque l'on bute sur un problème la solution est généralement ici.

- Lecteurs de flux RSS

Récupérer les flux RSS des sites spécialisés tels que (Human Coders, Ycombinator …) et plus généraux permet d'effectuer facilement et dans le détail une veille technologique. Il est aussi utile d'y inclure des sites tels que Apache, JavaWorld, W3.org qui ne concernent pas directement le langage PHP mais la sécurité ou le développement en général.

5.4 Etude et Analyse Personnelle du Contexte

Suite à la découverte progressive de ce contexte professionnel et grâce à une combinaison de mes connaissances actuelles, des recherches effectuées sur internet et des avis retournés par mes pairs (collègues de travail, collègues de formation et autres connaissances experts en développement) des points importants voire bloquants sont apparus indispensables à résoudre dans mes deux années d'alternances. Dans un premier temps, j'ai remarqué des manques concernant la gestion du projet qui sont pourtant habituellement trouvables dans tout développement réalisés :

- Les versions de l'application

Une sauvegarde des sources avait bien été effectuée, mais aucune sauvegarde d'une ou plusieurs versions n'a été créé, il était donc très difficile de reconstituer l'historique du code et les évolutions mineures et majeures qui ont eu lieu. Il n'est de plus, pas possible de récupérer une version antérieure si un bug bloquant venait à être décelé.

- Versionnage du code

Tout comme les versions de l'application, le code n'était pas soumis à un versionnage via des logiciels tels que Subversion ou GIT. De même il n'était donc pas possible de revenir en arrière lors d'un bug majeur. Ces outils sont très utiles, même indispensables, lorsqu'il y a plusieurs développeurs qui travaillent sur la même application mais ils peuvent également permettre de suivre des cas pratiques de développement et servir de mini tutoriels lors de la transmission du savoir ou de l'arrivée d'un nouveau développeur.

- Pas de différence de version « DEVELOPPEMENT », « TEST », « PRODUCTION »

A ce stade de développement, le fait qu'il n'y ait pas de version de production est normal, mais il n'y avait pas de séparation entre une version qui servirait au développeur, qui planterai régulièrement et une version de test qui permettrait aux personnes qui le souhaitent, ici mon tuteur, de tester l'application pour valider ou invalider un développement effectué. Dans un second temps, concernant la programmation et le code en lui-même, voici les problèmes que j'ai soulevés :

- Programmation selon un paradigme procédural

C'est évidemment la première chose que j'ai relevée immédiatement en ouvrant les premières pages du code source de l'application. Le paradigme procédural peut présenter certains avantages dans des situations bien précises comme par exemple lors de la réalisation de scripts afin de permettre la réutilisation du code, mais dans le cas du développement d'une application orientée web, la POO, programmation orientée objet, est un paradigme bien plus approprié.

- Effets de bords importants et multiples

Parce qu'il y a de nombreux appels à la base de donnée à faire et qu'il est nécessaire d'afficher des vues pour que l'utilisateur aie un rendu visuel de l'application et de son fonctionnement, la programmation procédurale est ici inappropriée car elle complique la lisibilité du code et produit de nombreux effets de bord en retournant des morceaux de codes HTML.

- Documentation technique impossible à réaliser

La documentation technique qui existe actuellement est une documentation qui représente l'application dans son contexte réseau. Aucune documentation technique n'existe pour le code source de l'application pour deux raisons : premièrement, il faudrait commenter chaque procédure ou fonction, deuxièmement, il n'y a pas de logique récurrente à expliquer et détailler pour une prise en main rapide, facile et habituelle.

- Maintenance difficile

C'est d'ailleurs cette logique récurrente qui rend la maintenance de l'application très difficile. Seul le développeur qui a écrit les lignes est capable de déduire de façon presque immédiate où peut se trouver l'erreur. Tout d'abord parce que le résonnement, le fonctionnement et les algorithmes de l'application lui son propre mais aussi parce qu'il en connaît l'historique.

Mélanger un code selon un paradigme procédural PHP (donc côté serveur) avec du code Html (HyperText Markup Language), Css (Cascading Style Sheets), Javascript (permet de dynamiser les pages web) et SQL (Structured Query Language) dans les même pages de script augmente de façon très importante la difficulté de lecture, compréhension et l'activité du débogage.

- Administration technique impossible

Dernier point important, l'administration technique de l'application n'est en l'état réalisable que par le développeur lui-même. Dans cette situation de paradigme procédural appliqué à une application web et manipulant des fichiers, il y a des paramètres qui servent à la configuration de l'application comme : les identifiants de connexion à la base de données, l'emplacement et le chemin des fichiers utilisés ou encore la connexion à un serveur FTP (File Transfert Protocole). Ces paramètres étaient disséminés dans le code aux endroits où ils étaient requis ce qui peut paraître suffisant si l'on ne pense pas à l'administration future de l'application. En effet il peut arriver que l'application soit migrée de serveur, que le serveur FTP change d'adresse ou que les identifiants de connexion soient modifiés et, dans une entreprise de la taille de Thales, il faut qu'une personne autre que les développeurs puisse modifier ces données de façon directe et simple. C'est pourquoi généralement à une application, web ou non, est associé un fichier de configuration contenant ces données au format par exemple INI (le format INI est une convention créée par Microsoft introduite par les systèmes d'exploitations Windows en 1985. Aujourd'hui Microsoft a remplacé ces fichiers en grande partie par le registre Windows, mais d'autres logiciels non-Microsoft continuent de l'utiliser).

Toutes ces raisons me laissent à penser que la conception de l'application aurait pu être améliorée dès son initialisation, mais le manque de temps et la nécessité de développement immédiat d'une application visuelle avaient sûrement dû prendre le pas sur la priorité des actions à effectuer.

Ainsi, ayant pris connaissance de toutes ces difficultés connues des développeurs et ayant pour conscience personnelle de rendre une solution à la fin de mon apprentissage qui ne les contienne pas, j'ai naturellement été amené à me poser une première question concernant cette application :

Tous ces points nécessitent des actions de refactoring ? Pour un développeur ils apparaissent comme indispensables à résoudre, mais comment convaincre mon tuteur et toute l'équipe concernée par le projet qui ne sont pas spécialistes en la matière qu'une partie importante de mon alternance – et donc du temps de développement - devra être consacrée à la résolution de ces problèmes qui jusqu'à présent n'avaient pas été relevés ?

Après avoir présenté lors de réunions ces différentes problématiques qui se trouvaient face à moi pour ne pas risquer d'être « bloqué » durant mes futurs développements et ne pas laisser un code sous cette forme à mon successeur présumé, j'ai pris en compte les différents enjeux qui m'ont été présentés :

- **Respect des objectifs dans les temps**

Avant mon embauche des objectifs ont été évalués dans le temps, datés et numérotés en semaine pour confirmer le bon avancement du projet. Un redéveloppement complet de l'application amènerait le projet et tous ses objectifs à être décalés dans le temps de plusieurs mois.

De plus, ces objectifs ont été fixés en fonction de dates correspondant à des lancement ou clôtures d'affaires qui peuvent servir de tremplins de communication et de financement pour l'application.

En Annexe 4 le diagramme de Gantt de planification du projet dans le temps

- **Enjeu monétaire & respect des objectifs**

Le projet de l'application SSDPM a besoin de financement pour continuer à exister, ils sont attribués en fonction des objectifs remplis ou non. Si un redéveloppement complet de l'application venait à être effectué, les objectifs d'avancement ne seront pas validés et les financements pourraient être annulés.

Cette expérimentation professionnelle est donc une prise de décision importante quant à l'urgence du projet qui sera très fortement influencée par un contexte professionnel déjà initié et dont l'ambition est d'avancer au maximum.

- **Vision technique du futur de l'application**

L'application se veut, à terme de mon contrat, être dynamique, documentée et relativement facile à prendre en main par un développeur de niveau ingénieur. Pour une application web, le type de programmation web généralement utilisé est la programmation orientée objet utilisant par exemple un patron de conception « Modèle, Vue, Contrôleur ». Nous verrons les détails de ce fonctionnement lors du déroulement de la problématique.

Ainsi, ces décisions nous amènent toutes indéniablement du côté du développement de l'application à une question passionnante du point de vue de la conception de code orienté objet :

Comment optimiser une application codée selon un paradigme procédural vers une programmation orientée dans le contexte contraint d'un projet déjà amorcé en entreprise ?

6 PROBLEMATIQUE

Comment optimiser une application codée selon un paradigme procédural vers une programmation orientée objet dans le contexte contraint d'un projet déjà amorcé en entreprise ?

6.1 REFORMULATION DE LA PROBLEMATIQUE DANS UN AUTRE CONTEXTE

Avant toute chose, simplifions réellement cette problématique avec un exemple simple complètement hors contexte. Reporter un problème complexe dans un autre domaine permet de faciliter sa compréhension et parfois même de trouver de nouvelles idées, solutions ou réponses (Voir par exemple l'anecdote d'Archimède pour le mot *Eurêka !)*.

Dans ce cas, reprenons tout d'abord tous les termes techniques de la question :

Paradigme : style fondamental de programmation informatique

Paradigme procédural : paradigme de programmation consistant à utiliser un ensemble de procédures pour réutiliser le code existant.

Programmation orientée objet : paradigme de programmation plus récent que le procédural consistant à utiliser des « objets » de programmation.

Contexte contraint d'un projet déjà amorcé en entreprise : projet débuté dont la phase d'initiation a déjà été effectuée, les objectifs sont fixés et doivent être remplis le plus rapidement possible.

Il s'agit donc dans cette problématique d'améliorer le code de l'application tout en continuant le développement des nouvelles fonctionnalités.

Reformulons maintenant la question dans un tout autre domaine en tant que chauffeur de taxi. Cet exemple n'est pas choisis au hasard, de nombreux professeurs utilisent pour apprendre la programmation objet à leurs élèves l'exemple d'une voiture, c'est donc un petit clin d'œil.

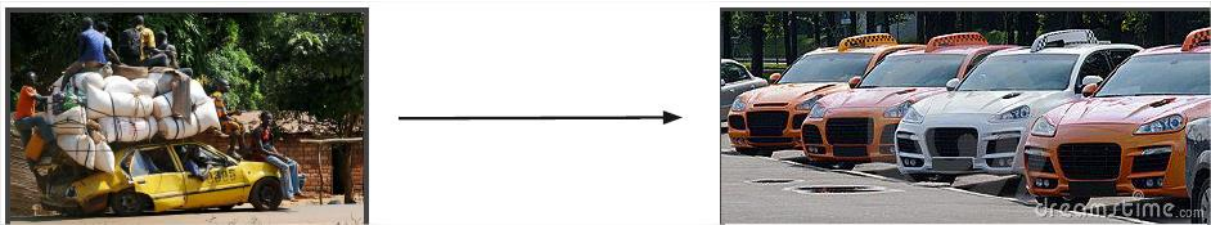


Figure 7 - Problématique en taxi

Comment, pour le chauffeur, modifier progressivement son taxi (image de gauche ci-dessus) pour obtenir au final un de ceux de la flotte de l'image de droite ci-dessus tout en continuant de travailler et gagner de l'argent ?

Il est maintenant évident que l'opération n'est réellement pas facile et qu'elle s'effectuera de façon progressive. Une grande réflexion préalable sera nécessaire ainsi qu'une très bonne connaissance en mécanique automobile. Les changements effectués au fur et à mesure ne doivent pas modifier le bon fonctionnement du taxi et, surtout, ne doivent pas empêcher le chauffeur de prendre des clients !

6.2 ELEMENTS DE REFLEXION, AXES D'AMELIORATION

Avec cette problématique vient donc une phase d'analyse qui permet de déterminer les éléments précis du contexte sur lequel il est nécessaire d'avoir une réflexion approfondie et d'effectuer un travail concret, c'est la partie « refactorisation » et « conception » du Framework.

Suite à de nouveaux développements pour l'application tels que la création d'un tableau spécifique, je maîtrise maintenant mieux l'application et, en parallèle de mes recherches et connaissances personnelles, certains points importants sont ressortis comme majeurs pour une transformation future.

Premier axe de réflexion, **la structure de l'application** n'est réalisée sous forme de dossiers nommés en rapport avec des termes métiers qui sont concrètement des « types » de matériel vendus par Thales. Habituellement, une application est structurée selon sa nature, une application web orientée utilisant un patron de conception M.V.C (« Modèle, Vue Contrôleur ») aura donc tendance à contenir uniquement quelques dossiers (Asset, Controlers, Views). Cette modification peut être relativement simple à très complexe selon la façon dont sont gérés les appels des fichiers.

Annexe 5 : Ancienne et nouvelle arborescence de l'application

Second axe majeur, **le mélange des langages**. Ici chaque page contient du Html, Javascript, Css, SQL et PHP, or tout développeur web sait qu'il est impératif pour garder une bonne lisibilité du code sur une application importante de séparer au maximum ces langages non seulement au niveau des lignes mais aussi dans des fichiers différents, voire dans des dossiers différents.

Troisième point, **la redondance du code**. Par exemple, l'application effectue différents appels à la base de données sur chaque page qui utilisent tous une nouvelle connexion à chaque fois ce qui augmente le nombre de lignes.

Quatrième et dernier point, **la configuration de l'application** est effectuée via des paramètres qui sont disséminés à des endroits stratégiques dans le code de l'application. Cette dispersion empêche leur modification simple par une personne inexpérimentée en développement mais qui aurait les compétences en administration réseaux nécessaires.

6.3 CONNAISSANCES SPECIFIQUES

Ces axes de réflexions impliquent logiquement des connaissances spécifiques pour les résoudre. Le fait d'avoir déjà programmé avec des Framework orientés MVC et d'avoir sût reconnaître leurs avantages face à une programmation procédurale dans cette situation m'a permis d'en ressortir une problématique mais ne signifie pas pour autant que j'avais toutes les connaissances nécessaires pour une telle réalisation. Ci-dessous un tableau récapitulatif des connaissances évaluées en début de parcours

Connaissance	Ok (Formation)	Ok (Exp. Pro.)	A acquérir
Paradigme procédural	X		
Paradigme fonctionnel	X		
Paradigme objet	X		
Patron de conception orienté Modèle, Vue, Contrôleur	X		
Fonctionnement Frameworks Open Source (Symfony, Yii)	X		
Fonctionnement Frameworks Privés		X	
Patrons de Conception	X		
Travail collaboratif		X	
Développements métiers		X	
Gestion de projet	X	X	X
Conception moteur de rendu de vues			X
Conception d'un module de réécriture d'url			X
Conception de la gestion des exceptions			
Utilisation d'un fichier de configuration			X
Refactoring			X
Sécurisation			X
Conception en Micro-Framework			X

La ligne « Gestion de Projet » est cochée dans les 3 catégories et c'est volontaire. J'ai en effet eu des cours de gestion de projet que j'ai réussi à appliquer partiellement dans mes expériences professionnelles, mais pas toutes. Certaines seront indispensables pour le travail à venir tel que notamment l'évaluation des tâches en temps.

Comme on peut le constater, j'avais les connaissances générales et déjà pratiqué tout ce qu'il fallait, cette fois-ci, concevoir pour cette application. Au niveau de la conception je n'avais cependant jamais réalisé de façon complète un moteur de rendu de vues ou un module de réécriture d'url pour un Framework ou Micro-Framework.

Je n'avais non plus jamais réalisé de refactoring à grande échelle sur plusieurs milliers de lignes. Qu'est-il nécessaire de supprimer ? D'ajouter ? De modifier ? De conserver ? Il y a-t-il une limite ? Et quand bien même je saurais quelle partie modifier, comment l'optimiser ? De plus, est-il nécessaire de tout optimiser ? Heureusement pour moi dans cette situation, le PHP est un langage que je connais depuis plusieurs années et c'est un de ceux dont je me suis servi pour apprendre à programmer. J'ai donc écrit dans ce langage du code brouillon, procédural, objet en MVC ou non ce qui m'a permis de comprendre relativement facilement pourquoi mon prédécesseur avait codé d'une manière ou d'une autre puis d'en faire le lien avec mes connaissances personnelles et les exemples trouvés sur internet pour finalement réussir à prendre les meilleures décisions possibles.

En tant que développeur web je connais les failles les plus courantes concernant les sites web comme les CSRF, les injections SQL, les modifications Ajax etc … mais pour sécuriser un Micro-Framework de façon automatisée il faut choisir des algorithmes évolutifs compatibles avec la totalité du code existant et futur.

Et surtout la partie la plus importante dans les connaissances à acquérir est sans aucun doute la conception d'un Micro-Framework qui peut se traduire de la manière suivante :

Organiser tout le code de l'application de façon à structurer et automatiser son fonctionnement grâce à des modules et fonctionnalités tels que la réécriture d'url, le moteur de rendu de vues, la gestion des exceptions, la communication avec une base de données, la dynamisation et le design de l'interface client.

Pour le reste, n'ayant pas de développeur expérimenté dans ces domaines à l'échelle de mon service, ni d'exemple déjà réalisés en interne dans l'entreprise, de nombreuses recherches et de nombreux essais ont été nécessaires afin de comprendre, essayer et pratiquer ces nouvelles connaissances jusqu'à pouvoir les reproduire.

6.4 AMELIORATIONS A ENVISAGER:

Bien que ces axes soient les principaux piliers de l'avancement dans cette refonte de l'application, il reste d'autres améliorations possibles à envisager qui, certes moins importantes, pourraient faciliter l'avancée du projet.

La documentation de toutes les procédures et classes est à ajouter de façon simultanée à chaque refactoring ou développement d'une nouvelle partie du code. Pour que cette documentation soit prise en compte et intégrée totalement dans le projet il est également nécessaire de choisir une convention d'écriture afin de paramétrer l'IDE (Environnement de Développement Intégré) de façon précise. Ces types de paramétrages peuvent ensuite permettre d'automatiser par exemple certaines actions pour gagner du temps de développement.

Des conventions de codages peuvent également être mise en place, comme l'organisation dans chaque nouvelle classe créée des méthodes par leur utilité. Ainsi elles seront toutes regroupées pour être retrouvées plus facilement. Les premières catégories auxquelles j'ai pensé sont :

- **View**: toutes les méthodes qui retournent une vue à afficher dans le navigateur côté client.
- **Request**: Toutes les méthodes qui sont appelées par un lien ou un formulaire Html via les méthodes d'accès POST ou GET.
- **Ajax**: toutes les méthodes qui sont appelées par un script JavaScript via les méthodes d'accès POST ou GET
- **Other**: toutes les méthodes qui sont appelées par la classe ou par d'autres classes.

Toujours dans les conventions de développement, on pourrait envisager de nommer les méthodes d'une certaine façon lorsqu'elles retournent des vues, ou encore les variables lorsqu'elles contiennent des objets et non uniquement des valeurs ou des tableaux.

Parfois complexes à mettre en place mais incroyablement utiles dans certaines situation les patrons de conception, excepté celui du M.V.C qui sera pris en compte dès le début puisque indispensable pour réaliser le Micro-Framework, ne seront pas intégrés puisque pour le moment pas indispensables. Cependant ils sont à connaître et il est possible qu'ils deviennent des améliorations très utiles si de nouvelles décisions sont prises.

Beaucoup d'autres améliorations peuvent être envisagées non seulement dans la situation de refactoring d'une application mais aussi dans le cas de la création d'un Micro-Framework qui doit être orienté « métier ». Je garde donc à l'esprit qu'uniquement ce terme « Métier » laisse penser à de nombreuses adaptations possibles d'un code selon des cas bien précis du contexte et de l'entreprise qui ne seraient évidemment pas retrouvables ailleurs. Il est donc possible qu'à certains moments lors du développement de l'application et qu'avec plus d'expérience personnelle concernant les métiers de Thales je choisisse de développer des modules spéciaux, d'où l'utilité d'une formation sur les systèmes TUS réalisée en début d'apprentissage.

7 METHODES RENCONTREES LORS DE CAS SIMILAIRES

7.1 EXPERIENCES PERSONNELLES ET SOLUTIONS TROUVEES

Evidemment cette solution n'est sûrement pas proposée ici pour la première fois, d'autres développeurs sont forcément tombées sur des problèmes similaires et on cherchées des solutions.

- Expérience professionnelle

Méthode 1 : Interbat Services, redévelopper l'application et créer un framework privé

Ce qui me permet d'affirmer que d'autres entreprises ont connues cette situation de façon aussi certaine c'est la raison suivante : ma précédente entreprise en tant que développeur, Interbat Services, a rencontré une situation très similaire. Ayant plusieurs développeurs à disposition, pour eux, la meilleure solution et finalement celle qui a été sélectionnée consistait en la création en d'un nouveau site en parallèle de celui existant. Concrètement une majorité des développeurs étaient assignés à la création du nouveau site, les autres étaient assignés à l'entretien et au développement des fonctionnalités indispensables sur le site existant. De plus, de par la très forte personnalité métier de l'entreprise (dématérialisation des appels d'offres), il a également été décidé de créer pour le nouveau site un Framework privé et orienté « métier ».

C'est ainsi que j'ai effectué mes premiers pas en développement concernant un Framework privé.

- Recherches Internet

Dans la totalité de mes recherches internet, que ce soit sur les forums ou les sites spécialisés j'ai trouvé vraiment beaucoup moins de résultats complets et de qualité que ce que j'en espérais. En effet, beaucoup de développeurs et patrons d'entreprise choisissent de redévelopper entièrement l'application en pensant gagner du temps au détriment des coûts. Très peu de développeurs se donnent la volonté et le courage de tenter la transformation complète d'une application en réutilisant le code déjà disponible. Et encore moins, voire aucun à ma connaissance, n'a effectué une amélioration conséquente du code existant tout en continuant de développer les nouvelles fonctionnalités en essayant de respecter au mieux les objectifs fixés. Voici donc les méthodes et sources d'informations les plus pertinentes que j'ai trouvées.

Méthode 2 : Evoluer vers une architecture MVC en PHP

Sur le forum « developpez.com » il y a un tutoriel nommé « Évoluer vers une architecture MVC en PHP » réalisé par Baptiste Pesquet, enseignant informatique à l'éducation nationale. Aujourd'hui très bien détaillé, il ne l'était malheureusement pas autant il y a deux ans lorsque le projet a commencé, mais les notions principales du refactoring du code PHP vers un modèle orienté MVC y étaient déjà. J'ai beaucoup appris de ce tutoriel qui, à ma connaissance, est le seul aussi détaillé.

Ce tutoriel a également pour avantage de présenter des solutions permettant de créer par petits modules un Micro-Framework complet.

Méthode 3 : Conseil de la communauté, redévelopper complètement l'application et utiliser un Framework existant

La troisième méthode est celle préconisée par une très grande partie des développeurs rencontrés sur les forums ou sur les chats IRC : le redéveloppement complet de l'application. C'est la réponse unique que j'ai trouvé à chacune de mes recherches, alors je me suis évidemment remis en questions sur la faisabilité et l'intérêt de ma solution face à la leur. Mais il est indéniable que face à certains contextes d'entreprise il soit impossible d'attendre une nouvelle version qui contient les même fonctionnalités que l'existant et trop couteux pour sous-traiter ces développements. Surtout si l'application n'est ni stable ni en production et a besoin d'être achevée le plus tôt possible.

- Recherches dans des livres

Méthode 4 : Pro PHP Refactoring

Le livre « Pro PHP Refactoring » est un livre écrit par les italiens Francesco Trucchia et Jacopo Romei tous deux experts en programmation PHP. Ils ont également présentés des conférences concernant le refactoring du code PHP lors des phpDay de 2009 :

« Spaghetti code refactoring: come riscrivere codice con le pratiche del KISS, DRY, TDD utilizzando un approccio DDD con Design Pattern »

Ce livre m'a réellement aidé à comprendre la philosophie du refactoring et à prévoir ce qu'il ne fallait et ce que je n'allais pas faire lors de mes prochains développements, c'est à dire un code qui serait composé de ce qu'ils appellent des « Bad Smells ».

Il n'y a en effet pas qu'une seule façon de refactorer un code, mais il y a en tout cas des erreurs à corriger et à le plus commettre comme l'architecture des fichiers, la duplication du code, le fonctionnement procédural du code, la taille des méthodes d'une classe, les habitudes de développement etc … Ce livre m'a également amené à me poser les trois questions indispensables suivantes :

- Pourquoi refactorer ?
- Quand refactorer ?
- Quand ne pas refactorer ?

J'y ai également appris des best practices qui ont fortement influencées mon développement, bien que je les aie pas toutes pratiquées.

- **KISS: Keep It Simple, Stupid !**

Le principe KISS est un principe qui préconise la recherche de la simplicité dans la conception et le développement du code.

- **TDD: Test Driven Development**

Le développement piloté par test consiste en l'écriture de tests unitaires avant de développer le code qui sera validé par les tests.

- **DRY: Don't Repeat Yourself**

Philosophie de développement appliquée par tous les développeurs généralement de façon instinctive consistant à éviter la redondance du code dans le but de faciliter les évolutions, le débogage et la maintenance de l'application.

- **DDD: Domain Driven Design**

Domain Driver Design (domaindrivendesign.org) est un ensemble de lignes directrices qui permettent de focaliser un code complexe sur l'aspect métier pour lequel il est écrit. Il permet également une meilleure résistance au changement.

- **Design pattern**

Les patrons de conception sont une standardisation de développement via une unique solution face à un problème rencontré de façon récurrente par les développeurs.

Finalement, la méthode de cet ouvrage se rapproche très fortement de la méthode 2 présentée ci-dessus si ce n'est deux différences : la première est qu'il est beaucoup plus théorique, je peux donc m'en servir de façon complémentaire, la deuxième différence est qu'il est conseillé au final de préparer le code sous la forme du patron de conception MVC pour ensuite pouvoir le « verser » dans un Framework libre téléchargé et installé.

NB : Au cours de mes recherches concernant ce livre, j'ai également rencontré un article de Kim N. Lesmer « The Book: Pro PHP Refactoring » qui remettait en question une partie du contenu du livre et des conclusions qui en étaient tirées. Je n'ai donc pas considéré ce livre comme un tutoriel mais plutôt comme une source d'inspiration »

7.2 ANALYSE DE LEURS AVANTAGES ET INCONVENIENTS

L'analyse des avantages et inconvénients de ces méthodes doit être de mon point de vue selon deux axes importants permettant de mesurer avec précision l'efficacité d'une méthode d'un point de vue technique ou du projet.

En effet, chaque méthode peut présenter des avantages et inconvénients d'un point de vue technique : faciliter les développements ou intégration dans le système informatique spécifié par l'entreprise mais aussi du point du vue du projet comme l'économie de ressources financières ou de temps. J'ai choisi d'organiser ces avantages et inconvénients dans deux tableaux afin de permettre une meilleure comparaison visuelle **en Annexe 6 et 7**.

Croisons maintenant toutes ces informations dans des tableaux récapitulatifs adaptés à chaque méthode permettant ainsi de faire le point sur les avantages et inconvénients concrets qu'elles ont à nous apporter. De par la couleur rouge sont indiqués les points les points les plus bloquants pour le développement de l'application SSDPM en cours chez Thales, en vert sont indiqués les « meilleurs » avantages d'une méthode.

7.2.1 L'entreprise « Interbat Services »

Hypothèse I

Elle consiste à **redévelopper l'application totalement en créant un Framework privé** adapté aux ressources métiers de l'entreprise afin de résoudre de façon systématique certains problèmes qui auraient pu être rencontrés par la suite.

Avantages et inconvénients techniques

	Technique	
	Avantages	Inconvénients
Tout redévelopper	- **Possibilité de développement à plusieurs développeurs dès le premier jour** - Suppression totale du code « spaghetti » existant	- **Perte de l'existant**
Framework privé	- Le Framework créé est orienté métier - Choix du langage - Utilisation des dernières technologies (Design pattern, version des langages …) - **Choix des modules de Framework donc fonctionnement plus léger**	- « Do-it-Yourself » Limité à la connaissance des développeurs en matière de sécurité, structure et réalisations. - Pas de documentation technique existante - Pas de tutoriels - Pas de communauté d'entre aide

	- **Facilite l'intégration du projet dans le système informatique existant**	

Dans le cadre de l'application SSDPM, il n'y a, malgré l'inconvénient indéniable de la perte totale du code existant, il n'aucun problème technique au redéveloppement de l'application complet via un Framework créé de toutes pièces.

Avantages et inconvénients pour la gestion du projet

	Projet	
	Avantages	Inconvénients
Tout redévelopper	- Expérience acquise par tous les développeurs investis dans le projet	- Nécessite plusieurs mois de travail avant d'obtenir les premiers résultats. - **Stagnation de l'ancien projet** - Nécessite de nombreuses réunions de conception
Framework privé	- Maitrise à 100% des lignes de code par les développeurs - Utilisation de l'expérience des employés concernant les problématiques existantes dans le but de les prévoir à l'avance	- Temps d'acquisition des connaissances par un nouvel embauché relativement long - La conception et le développement ne peuvent se faire qu'à un nombre limité de développeurs

Ici, l'inconvénient du redéveloppement total de l'application principal est le temps. Se lancer dans la reprise depuis le départ d'un projet est toujours plus rapide car l'équipe a déjà maturé au niveau de l'expérience et connaît les erreurs à ne pas reproduire. Cependant il serait illusoire d'imaginer qu'une application développée en une année puisse l'être à nouveau en quelques semaines grâce à cette expérience. Nous pouvons estimer ce temps entre 50% et 75% du temps du projet initial ce qui donne toujours plusieurs mois de redéveloppement pendant que l'application, elle, n'évolue plus.

Et c'est ce facteur qui est ici décisif et bloquant dans cette solution pour l'application SSDPM. Il est absolument indispensable de commencer les développements de façon immédiate, nous l'avons vu précédemment le financement du projet étant dépendant de la réalisation des objectifs qui sont liés très étroitement aux fonctionnalités disponibles de l'application et donc au développement nous ne pouvons pas nous permettre d'attendre plusieurs mois la sortie d'une application équivalente.

7.2.2 « Evoluer vers une architecture MVC en PHP et créer un Micro-Framework »

Hypothèse II

Elle consiste en **l'amélioration progressive et conséquente du code de l'application dans le but de créer un framework métier** s'adaptant parfaitement à la structure des données de l'entreprise, ici Thales.

	Avantages et inconvénients techniques	
	Avantages	Inconvénients
Refactorer	- Maitrise totale du développeur de l'ancienne et de la nouvelle version de l'application - **Réutilisation du code existant**	- Effets de bords nombreux lors du refactoring - Suppression partielle du code « spaghetti » ou en tout cas progressive
Framework privé	- Le Framework créé est orienté métier - Choix du langage - Utilisation des dernières technologies (Design pattern, version des langages) - Choix des modules de Framework donc fonctionnement plus léger - Facilite l'intégration du projet dans le système informatique existant	- « Do-it-Yourself » Limité à la connaissance des développeurs en matière de sécurité, structure et réalisations. - Pas de documentation technique existante - Pas de tutoriels - Pas de communauté d'entre aide

Ici, de nombreux avantages sont intéressants comme la réutilisation du code existant ou le choix des modules du Framework pour un fonctionnement plus léger. Il y a également des inconvénients dont il faudra tenir compte lors de la comparaison finale des différentes méthodes tels que le « doi-it-yourself » qui limite les possibilités des développements aux connaissances techniques du développeur mais n'est pas bloquant.

	Avantages et inconvénients pour la gestion du projet	
	Avantages	Inconvénients
Refactorer	- **Avancement immédiat, le projet n'est pas bloqué car les nouvelles fonctionnalités peuvent être développées** - Réutilisation des fonctionnalités existantes - Amélioration continue de l'application	- Effets concrets obtenus après plusieurs mois ou années de travail - Possibilités réduite de développement à plusieurs car les effets de bords seront nombreux
Framework privé	- Le Framework créé est orienté métier	- « Do-it-Yourself » Limité à la connaissance des développeurs en matière de sécurité, structure

	- Choix du langage - Utilisation des dernières technologies (Design pattern, version des langages …) - **Choix des modules de Framework donc fonctionnement plus léger** - **Facilite l'intégration du projet dans le système informatique existant**	et réalisations. - Pas de documentation technique existante - Pas de tutoriels - Pas de communauté d'entre aide

Dans notre situation aucun point de cette méthode n'est bloquant. Elle est donc une situation tout à fait possible pour l'instant. Le meilleur avantage dans cette partie étant la possibilité d'avancement immédiate du projet. Le choix des modules du Framework et sa facilité d'intégration dans le système informatique de l'entreprise existant sont également des plus qui permettront de gagner du temps par la suite.

7.2.3 Le conseil de la communauté

Hypothèse III

Elle consiste à **redévelopper complètement l'application et à utiliser un Framework existant** est une des plus avantageuses :

	Avantages et inconvénients techniques	
	Avantages	Inconvénients
Tout redévelopper	- **Possibilité de développement à plusieurs développeurs dès le premier jour** - Suppression totale du code « spaghetti » existant	- Perte de l'existant - **Temps d'attente conséquent (mois voir années) pour débuter le développement de nouvelles fonctionnalités**
Framework open source	- **Utilisation des dernières technologies** (Design pattern, version des langages …) - Choix du langage - **Documentation existante** - **Communauté & entraide** - Développements solides et entraide pour tous les développeurs - Pas de "Do-it-Yourself"	- Pas orienté métier - **Règles « Fixes ». implicites. Le fonctionnement et l'architecture du Framework ne peut pas être modifié ou adapté au Système Informatique de l'entreprise** - Les failles et mises à jour du Framework dépendant d'une autre entité - Connaissances préalables obligatoires - Qualité - Pérennité

		- Modules intégrés pas forcément utile au projet

Cette méthode comporte des avantages supplémentaires de par l'utilisation d'un Framework Open Source. Il permet l'utilisation des dernières technologies, d'une documentation existante et disponible sur le site officiel mais également l'entre-aide via une communauté, des forums ou des chats IRC.

Cependant, deux inconvénients sont ici bloquants pour l'avancée de projet et risquent fortement d'empêcher l'utilisation de cette méthode. Tout d'abord le temps, qui est obligatoire pour redévelopper une nouvelle version de l'application et l'amener au même niveau de fonctionnement que l'ancienne version, ensuite, les règles fixes imposées par l'utilisation d'un Framework Open Source qui peuvent ne pas être adaptés à l'architecture réseau proposées par le système informatique de l'entreprise.

Il aurait été également envisageable ici de changer de langage de programmation pour du Python ou du Ruby.

	Avantages et inconvénients pour la gestion du projet	
	Avantages	Inconvénients
Tout redévelopper	- Expérience acquise par tous les développeurs investis dans le projet	- Nécessite plusieurs mois de travail avant d'obtenir les premiers résultats. - **Stagnation de l'ancien projet** - Nécessite de nombreuses réunions de conception
Framework open source	- Possibilité de développement à plusieurs développeurs dès le premier jour - **Embauche de développeurs opérationnels immédiatement car ils connaissent déjà le Framework**	- Maitrise partielle du Framework

Ici, l'avantage à prendre en compte est la possibilité d'embauche de développeurs opérationnels immédiatement. En effet le projet venant à grandir il est possible que d'autres développeurs s'y joignent dans les années à venir. Utiliser un Framework Open Source permet de recherche sur tous les sites d'embaucheur des développeurs qui maitrisent déjà la technologie et qui seront donc fonctionnels beaucoup plus rapidement que les autres. Ils connaissent également la communauté et peuvent même dans certains cas apporter leur expérience.

Evidemment en inconvénient, la stagnation du projet en attente du redéveloppement de l'application est un point très négatif.

7.2.4 Améliorer de façon progressive et conséquente le code pour le « verser » dans un framework

Hypothèse IV

Conseillée par les auteurs du livre Pro PHP Refactoring est cette méthode consiste en **l'amélioration progressive et** conséquente du code de l'application jusqu'à atteindre un niveau de compatibilité suffisant permettant de **« verser » le code refactoré dans un Framework Open Source.**

	Avantages et inconvénients techniques	
	Avantages	Inconvénients
Refactorer	- Maitrise totale du développeur de l'ancienne et de la nouvelle version de l'application - **Réutilisation du code existant**	- Effets de bords nombreux lors du refactoring - Suppression partielle du code « spaghetti » ou en tout cas progressive
Framework open source	- **Utilisation des dernières technologies** (Design pattern, version des langages …) - Choix du langage - **Documentation existante** - **Communauté & entraide** - Développements solides et entraide pour tous les développeurs - Pas de "Do-it-Yourself" - Qualité - Pérennité	- Pas orienté métier - Les failles et mises à jour du Framework dépendant d'une autre entité - Connaissances préalables obligatoires - Modules intégrés pas forcément utile au projet - **Le système informatique doit s'adapter à l'architecture du Framework ou il faudra modifier son fonctionnement pour qu'il s'adapte au système informatique**

Nous avons déjà vu précédemment les avantages du refactoring et de l'utilisation d'un Framework Open Source, cependant ils n'étaient pas combinés. De cette façon nous supprimons l'inconvénient du redéveloppement total de l'application mais il reste le problème d'adaptation du code à la structure réseau de l'entreprise.

	Avantages et inconvénients pour la gestion du projet	
	Avantages	Inconvénients
Refactorer	- **Avancement immédiat, le projet n'est pas bloqué car les nouvelles fonctionnalités peuvent être développées** - Réutilisation des fonctionnalités existantes - Amélioration continue de l'application	- Effets obtenus après plusieurs mois ou années de travail - Possibilités réduite de développement à plusieurs car les effets de bords seront nombreux
Framework open source	- Possibilité de développement à plusieurs développeurs dès le premier jour - Embauche de développeurs opérationnels immédiatement car ils connaissent déjà le Framework	- Maitrise partielle du Framework

Au niveau du projet, un refactoring pour verser le code dans un Framework Open Source permet de débuter les développements de façon immédiate pour ensuite profiter de tous ses avantages.

Conclusion de l'analyse

En conclusion de cette analyse des différentes méthodes proposées pour la récupération d'une application codée en procédural, nous pouvons déjà affirmer que les solutions contenants des points bloquants doivent être évitées ou contournées.

La méthode qui semble donc idéale pour ce projet est la deuxième « Evoluer vers une architecture MVC en PHP et créer un Micro-Framework » puisqu'elle ne contient aucun point bloquants. Elle n'est pas cependant celle qui présente le plus d'avantages qui sont les méthodes 3 et 4 avec 5 avantages chacune.

Alors, pourquoi ne pas considérer les méthodes 3 ou 4 et tenter d'éviter les points bloquants ?

La troisième a un inconvénient qui est absolument bloquant : la stagnation du projet en attente du redéveloppement de l'application, il n'est donc tout simplement pas possible de la mettre en place dans le cadre d'une application qui doit évoluer de plus en plus vite et dont on attend des résultats rapidement.

En revanche, la quatrième méthode a pour seul inconvénient bloquant de ne pas être adaptable à la structure de l'entreprise, ce qui est assez général et mérite d'être expérimenté. Il se peut par exemple qu'un Framework PHP fonctionne sur la bonne version du serveur web (IIS), avec la bonne version de langage PHP et ne rencontre pas de contraintes spécifiques à

l'utilisation pour qu'après refactoring je puisse « verser » le code de l'application dans ce Framework.

C'est après la mise en lumière de cette possibilité que j'ai décidé de tenter l'installation de façon autonome plusieurs Frameworks PHP qui comptent parmi les plus connus dans le but d'identifier les problèmes et tenter de leur trouver des solutions si besoin. De plus, si cette méthode n'est pas utilisée au final, cette recherche n'aura pas été une perte de temps puisqu'elle m'aura permis de comparer leur fonctionnement et de confirmer, choisir ou découvrir les modules qui seront nécessaire au développement d'un Micro-Framework.

- Zend Framework

Pas facile à appréhender il demande beaucoup de temps et d'investissement, il n'est pas conseillé de l'utiliser pour les petits projets.

- Yii

Son module de réécriture d'url est basé sur un fichier « htaccess » qu'il n'est pas possible d'utiliser via le serveur web fournit par le service informatique pour l'application SSDPM.

- Symfony

Symfony est un Framework très complet et donc lourd à l'utilisation, beaucoup de spécificités qui lui sont propres sont à utiliser pour par exemple créer simplement un formulaire, de plus il offre de mauvaises performances pour les petits projets. Il est donc difficilement adaptable à une structure MVC crée de toute pièce à partir du code natif de PHP.

Pour le moment ces Frameworks ne sont donc pas adaptés à la structure de l'entreprise ni aux spécificités métiers du code actuel qui pour l'instant ne peut être versé dans un Framework existant.

J'ai donc sélectionné suite aux nécessités temporelles et financières la méthode 2 « **Evoluer vers une architecture MVC en PHP et créer un Micro-Framework** » qui ne présente que des avantages dans ma situation. D'une part les développements des nouvelles fonctionnalités indispensables sont en rapport avec les objectifs fixés pour le projet, d'une autre part la fin établie du projet est d'assez bonne qualité pour permettre la réalisation d'une documentation technique, le travail à plusieurs développeurs, l'utilisation de modules d'url rewriting ou d'un moteur de rendue de vues.

De plus en fonction des connaissances des développeurs et du temps disponible sur le projet il sera possible de façon méthodique et bien plus facile de verser le code dans un Framework tel qui YII pour hériter instantanément de toutes ses fonctionnalités ce qui nous amène à un mélange entre les méthodes 2 et 4 et permet de conserver tous les avantages de la deuxième méthode pour le développement et le refactoring de l'application ainsi que tous les avantages de la dernière méthode concernant Frameworks pour la finalisation du projet.

A noter que cette prise de décision était personnelle et que l'activité d'améliorer le code de façon progressive n'était pas une demande spécifiée lors de mon attribution sur ce projet. En revanche il m'était clairement demandé de laisser un code commenté, clair et lisible et documenté à la fin de ma mission, ce qui n'aurait clairement put être le cas si je n'avais pas fait une analyse approfondie de ces différentes méthodes pour me permettre une organisation progressive et par paliers sur les deux prochaines années de travail.

8 DECISIONS CONCRETISEES

Après l'analyse des solutions existantes ci-dessus, nous savons maintenant que l'objectif fixé est de continuer le développement des nouvelles fonctionnalités du projet tout en refactorant le code existant progressivement pour atteindre une qualité, maintenabilité et réutilisation du code maximale à la fin du projet.

Il est maintenant nécessaire d'élaborer une planification d'avancement des priorités et développements ce qui amène à une prise de décisions appliquées concrètement dans le cas de ce projet.

Pour comprendre la mise en place de cette partie, nous commencerons dans un premier temps par expliciter l'analyse des besoins et faisabilités ainsi que les spécifications fonctionnelle et technique créées, puis dans un deuxième temps, nous nous concentrerons sur les différentes stratégies choisies et adoptées concernant la gestion du projet.
Ces grands axes d'avancement sont globalement ceux d'un cycle en « v » habituel mais cette fois-ci appliqués dans un contexte particulier car il reprend un projet déjà en cours.

Nous pourrons ainsi rendre compte de l'impact qu'ont eu les décisions prises dans le domaine technique sur le respect des objectifs et la gestion du projet ainsi que, inversement, l'impact qu'ont eu les décisions prises dans le domaine du management du projet sur la conception et le développement de l'application.

8.1 HISTORIQUE DES EVOLUTIONS ET DEVELOPPEMENTS

Tout d'abord, avant même de commencer la phase d'analyse des besoins et de faisabilité, j'ai choisi de consacrer les premiers jours à la recherche d'une solution de versionnage afin de

pouvoir suivre l'évolution de l'application face aux nombres de changements qui sont appelés à être effectués.

Le refactoring d'une application complète implique la forte modification du code et, un module juste développé peut finalement apparaitre, après réalisation, comme inutile ou mal adapté. Dans ce cas il est nécessaire de supprimer le travail effectué en revenant à une version précédente de l'application pour repartir d'une application stable et tenter une nouvelle solution plus adéquate.

Ainsi, j'ai immédiatement pensé à l'installation des logiciels de version tels que Subversion ou GIT qui sont les deux références à l'heure actuelle chez la majorité des développeurs. Mais lequel choisir ? Le premier est basé sur une gestion des sources via un dépôt situé sur un serveur distant, le second est basé sur une gestion des sources via un dépôt local synchronisé avec les autres dépôts des autres développeurs. Le fonctionnement est très différent mais l'utilisation reste la même à très peu de choses près. Alors, faire mon choix entre les deux logiciels devait venir d'un autre critère de sélection : l'intégration. Chez Thales, l'installation des logiciels n'est pas autorisée et une « demande de service » est nécessaire pour avoir l'autorisation attribuée à l'application souhaitée, c'est donc après cette demande que j'ai sélectionné et installé Subversion.

L'utilisation de cette application a eu des conséquences directes sur mon travail journalier qui se sont représentées par des mises à jour systématiques à la fin du développement de chaque fonctionnalité permettant ainsi, par la même occasion, de sauvegarder le travail effectué ainsi que d'ajouter des commentaires sur chaque évolution pour tracer l'historique de ces sauvegardes récurrentes.

Le déroulement du projet a également été simplifié grâce à cet historique puisqu'il est maintenant possible de donner des numéros de version à un ensemble de mises à jour qui sont détaillée de par leur titre et leurs informations supplémentaires.
Il devient aussi possible de travailler à plusieurs développeurs, les logiciels de versionnage rendent ce travail réellement beaucoup plus facile : Chaque développeur a sa propre version de développement et peu travailler sans générer de bugs à cause des effets de bords sur la version de ses collègues. Ensuite une fois qu'elle est stable, chaque développeur est libre de faire une mise à jour sur une version « TEST » qui est commune et permet de tester l'ensemble des fonctionnalités développées qui sont orientées métier par une personne qui n'a pas ces connaissances techniques. Cette phase de validation est déterminante pour un projet qui est appelé à grossir. En Annexe 8 un Schéma représentatif de la solution.

8.2 ANALYSE DES BESOINS ET FAISABILITE

Maintenant que l'application et toutes les modifications qui lui sont effectuées peuvent être sauvegardées, il est possible de rentrer concrètement dans le code afin d'y effectuer divers essais permettant ainsi d'analyser les besoins les plus élémentaires et la faisabilité des solutions techniques et immédiates trouvées.

Un framework est un ensemble cohérent de composants logiciels structurels. Ce sont ces composants logiciels qui apportent au développeur une rapidité et une efficacité de développement accrue. Les frameworks web les plus connus ont des modules similaires qui peuvent se regrouper sous un nom comme : un moteur de rendu de vues, la réécriture d'url, les objets de mappage aux bases de données relationnelles etc … mais ils ont également leurs propres composants qui ont une utilité particulière telle que la gestion de la sécurité pour éviter au développeur de créer des failles, comme notamment les injection SQL (insérer une requête SQL dans un champ normalement destiné à des informations sur le client par exemple) ou les failles de types CSRF (utiliser des fonctions disponibles uniquement par l'administrateur en temps normal via la récupération par des moyens illicites d'une adresse internet qui donne accès à cette fonction).

Mais quels sont donc les modules qui sont nécessaires, indispensables, pratiques ou utiles à prendre en compte et à intégrer lors du développement du micro-framework ? Tout d'abord il faut avoir une idée précise de ce que sera amenée à devenir l'application SSDPM, en résumant de façon concise, nous pourrions la décrire comme ceci :

Sonar Support Data Package Maker est **une application web** utilisée sur **le réseau Intranet** de Thales uniquement qui permet de **charger des données contenues dans un fichier au format CSV** ainsi qu'une arborescence de fichiers PDF et JPG en les stockant dans une **base de données** pour ensuite les modifier et mettre en forme, via l'utilisation de **serveurs FTP** (voir Annexe 9 Annexe 10), et finalement « générer » un package désigné de ces données sous la forme d'un dossier contenant principalement des fichiers aux format HTML, CSS, Javascript, PDF et JPG

En vert ci-dessus sont indiqués les indices qui vont me permettre de déterminer précisément quels sont les modules minimum permettant de créer un Framework adapté et métier. Il faut toutefois qu'ils soient compatibles avec les stratégies et management du projet. Reprenons les donc un à un.

Tout d'abord c'est **une application web**, elle doit donc en tant que tel avoir un code PHP du côté du serveur qui travaille avec une base de données et retourne du code au navigateur client sous forme HTML et JavaScript uniquement. Pour gérer ce fonctionnement il existe un patron de conception (ou Design pattern), le **M.V.C** « Modèle, Vue, Contrôleur », qui est un des patrons de conception les plus récurrents dans tous les Frameworks modernes. Associé

à ce module s'utilise généralement un composant de **réécriture d'url** (Url Rewriting) sous la forme d'un fichier « **htaccess** » ou d'un **script PHP** qui permet de réécrire les URL de l'application sous une forme humainement plus compréhensible.

Ensuite, elle utilise **le réseau Intranet** ce qui signifie qu'elle est par défaut sécurisée face aux attaquants internet via la politique et les règles de sécurité mises en place au niveau du Groupe. Son utilisation n'en sera donc que fonctionnelle, d'autant plus que les utilisateurs seront sélectionnés et confirmés par notre équipe. Le développement **de fonctionnalités liées à la sécurité** n'est donc dans un premier temps par une priorité. Ainsi, les modules de Frameworks dont le but est de sécuriser l'application ne sont pas nécessaires dans un premier temps.

Le chargement des données contenues dans un fichier CSV nécessite une analyse préalable du fichier par un script PHP pour le transformer en tableau utilisable simplement par l'application et stocker les données dans la base de données. Ici, PHP fournit nativement une fonction permettant de travailler sur ce système de fichier, « fgetcsv ». Il n'y a donc pas besoin d'**API spécialisée** supplémentaire ici.

La base de données est une partie commune à toutes les applications web, il y a donc évidemment dans les Frameworks des composants permettant de faciliter tous les opérations des développeurs. Ils sont appelés **ORM** : Object-Relationnal Mapping. Cette méthode de développement permet de créer l'illusion d'une base de données virtuelle sous forme d'objets de programmation, il en devient bien plus facile pour le développeur d'utiliser, modifier, supprimer et ajouter des données. J'ai pourtant décidé de ne pas sélectionner cette solution qui ne s'adaptait malheureusement absolument pas à l'existant. Dans le code de l'application se trouvait effectivement beaucoup de requêtes codées « en dur » c'est-à-dire qu'elles contenaient le code complet permettant d'ouvrir la connexion à la base de donnée, de « binder » les paramètres, d'exécuter la requête donnée et de fermer la connexion, mais il y avait également une importante partie du code SQL stockée directement en base de données sous forme de Package contenant des fonctions et procédures stockées.
Le code étant dispersé à plusieurs endroits, j'ai jugé prioritaire de rassembler toutes les requêtes en un seul endroit qui est le stockage dans la base de données sous forme de Package avant de penser à un ORM, tout en codant de façon à faciliter une intégration future.

Pour finir, la partie **serveurs FTP** et **SQL** me donne la puce à l'oreille que des **objets** seront utilisés pour gérer les connexions que ce soit au serveur FTP ou à la base de données. Là des composants supplémentaires ne sont pas nécessaires car ils seraient trop lourds et complexes à mettre en place face à la simplicité de création d'une classe qui permet les mêmes fonctionnalités.

Le Micro-Framework devra donc contenir comme composants indispensables au bon développement de l'application : le patron de conception M.V.C, la réécriture d'url et les objets orientés métier adaptés aux fonctionnalités attendues. La réécriture d'URL sera mise en place via un fichier de script PHP car, nous l'avons vu précédemment lors de l'analyse des possibilités existantes et du test des différents Frameworks utilisables, un fichier « htaccess » n'est pas utilisable dans ce contexte d'entreprise ne disposant tout simplement pas des droits de création et lecture pour les fichiers tels que celui-ci. Ces composants seront les piliers de l'application et tout développement devra s'y rattacher. En revanche, une API spécialisée dans la gestion des fichiers CSV n'est pas indispensable et alourdirait l'application, tout comme un module ORM ou des fonctionnalités liées à la sécurité qui seraient vus dans ce cas comme des possibilités supplémentaires d'améliorations mais pas utiles pour le minimum visé actuellement qui est le refactoring complet de l'application.

L'utilisation de ces modules permet donc au niveau technique un développement plus rapide de par des règles d'utilisation, mais aussi orienté métier qui permettra d'adapter l'application au contexte de l'entreprise et aux fonctionnalités spéciales à développer. **L'application devient modulable et adaptable.**

De par ces décisions du temps a été gagné. J'ai à la suite de la création de ces modules pris des habitudes de développements qui ont permis de ne plus concevoir chaque nouvelle partie de façon différente et distincte à l'inverse du code spaghetti qui laissait un très grand nombre de possibilités de développement pour chaque fonctionnalité souhaitée. De par ces composants mis en place, la création d'un formulaire par exemple est systématiquement composée de la création d'une vue, la création d'une méthode dans la classe correspondante – elle aussi créé si elle n'existe pas -, puis par la création d'une requête placée dans une procédure stockée en base de données. Presque toutes les évolutions souhaitées peuvent être développement selon ce schéma de développement ce qui fait gagner un temps conséquent et pour chaque fonctionnalité lors des développements pour leur :

- Evaluation en temps
- Organisation dans le temps
- Conception
- Réalisation
- Validation

Ainsi, nous avons dès maintenant un Micro-Framework structuré et fonctionnel selon le contexte métier de l'entreprise qui permet de gagner du temps à chaque ligne de code et de soulager indéniablement la réflexion précédent chaque nouvelle amélioration.

Suite à cette phase d'analyse, il est désormais possible de concevoir et rédiger avec précisions les spécifications techniques de l'application d'autant plus que les spécifications fonctionnelles

ont déjà été intégralement rédigées par mon tuteur, Marc-Henri, avant et pendant le début de mon alternance. Détaillées sur un ensemble de 160 pages, ces spécifications montraient clairement que mon tuteur savait exactement ce vers quoi il voulait se diriger, cependant et c'est une problématique récurrente dans de nombreuses équipes de développement, les prévisions effectuées permettant de répondre à l'ensemble de ces fonctionnalités sont très différentes avec un regard technique que ce soit dans la difficulté ou dans le temps de la réalisation. Avec la spécification fonctionnelle vient généralement la spécification technique. En faire l'ébauche avant de commencer à développer l'application permet de réaliser la phase de conception de l'application.

8.3 SPECIFICATIONS TECHNIQUES ET CONCEPTION ARCHITECTURALE

8.3.1 Qualité du code

La qualité du code est un concept relativement abstrait et souvent réalisé de façon inconsciente en fonction de l'expérience des développeurs, bien que ce soit l'un des facteurs les plus importants lors du développement d'une application. Cette qualité peut cependant être évaluée par différents critères.

- Dans un premier temps, **la lisibilité** du code est un facteur important d'autant plus lorsque l'application est appelée à être développée par plusieurs personnes ou transmise pour un support quelques années après son développement.

- **La portabilité** est un critère qui semble facultatif mais qui peut se révéler un gain de temps conséquent lors du développement de plusieurs applications ayant les mêmes bases. La possibilité de réutiliser un objet ou un ensemble de code destiné à accomplir facilement une fonctionnalité dans une nouvelle application implique une structure du code relativement bien définie et organisée.

- **L'optimisation des temps d'exécution** est également un critère de qualité d'autant plus pour les applications qui ont de longs et lourds traitements à effectuer. Attention, optimiser le temps d'exécution ne signifie pas obligatoirement un code raccourci.

- **Les normes**, dans le cadre de l'utilisation d'un Framework sont déjà définies et structurées, cependant lors de la création d'un Framework c'est une partie de la conception qu'il est important de garder à l'esprit et d'appliquer dès que possible, que ce soit dans le nom des méthodes ou structures de classes en fonction de leur résultat de retour ou dans le nom des variables. Ces choix de normes et mise en forme sont totalement arbitraires et dépendent de l'expérience ou des préférences du développeur qui les met en place.

- Finalement, **la gestion des erreurs** est ici le dernier critère de qualité du code. Elle implique une gestion avancée des erreurs rencontrées par le développeur pour les diviser en deux catégories: les erreurs connues dues à une mauvaise utilisation de l'application, elles doivent être prises en compte dans le code pour ne pas retourner un code PHP natif détaillant l'erreur mais un message à l'intention et compréhensible de l'utilisateur, puis les erreurs de développement qui elles doivent être récupérées, stockées et consultable via une interface spécialisée. Ainsi l'application est censée ne jamais « planter ».

Cette décision de me fixer et de m'imposer des critères pour la vérification de la qualité du code a réellement influencé mes développements et ma productivité au jour le jour dans le sens positif. L'optimisation générale de l'application au niveau technique s'est vue également considérablement améliorée.

C'est dans la partie conception du Micro-Framework qu'il a été indispensable d'imaginer et mettre en place ces critères pour permettre une intégration complète lors des développements futurs. Ainsi ces « best practices » ont finalement aidées en partie comme les modules présentés précédemment, la conception, la réalisation et la validation.

Cette optimisation constante du code est un avantage indéniable concernant la gestion du projet qui ne pourra que s'en retrouver amélioré puisqu'elles traduisent elles même une application d'une bonne qualité qui peut être transmise et comprise à d'autres développeurs.

8.3.1.1 Quel code supprimer ou refactorer ?

Après avoir vu les points précédents concernant la structure du Micro Framework et l'optimisation de la qualité du code, il devient assez simple de répondre à cette question. En effet, j'ai pris la décision de respecter l'affirmation suivante : **tout code qui ne correspond pas aux exigences de qualité fixées est un code qui, à terme, doit être modifié, redéveloppé ou supprimé.**

8.3.1.1.1 Code qui doit être modifié :

- Tout code dit répondant aux exigences "de qualité" fixées précédemment qui doit être adapté au Framework
- Tout code qui peut l'être, sera réutilisé à plusieurs endroits
- Tout code superflu

8.3.1.1.2 Code qui doit être supprimé :

- Tout code qui n'est plus utilisé
- Tout code qui se verra remplacé par un nouveau développement

Le code superflu étant considéré comme du code solutionné par des fonctions natives de PHP ou qui surcharge le nombre de ligne de code de façon inutile. Ce code peut généralement être refactoré via un algorithme basique.

Cette décision a apporté un inconvénient lors de l'avancée dans les développements, elle **a augmenté la durée en temps de chaque développement prévu**. Il a donc été très important d'en tenir compte dans les aspects managériaux et stratégiques du projet qui ont été impactés fortement au début du projet. Je précise « au début du projet » volontairement car le refactoring étant nécessaire sur l'ensemble de l'application au début du projet l'a été de moins au moins au fur et à mesure de l'amélioration de la qualité du code et de la mise en place des différents modules. Logiquement, plus le code qui doit être supprimé l'est et le code qui doit être modifié l'est également et plus les modules qui doivent mis en place le sont, plus la qualité du code augmente et moins il est nécessaire de prévoir du temps à rajouter lors du développement de chaque nouvelle tâches. Concrètement les premières tâches demandaient environ 50 à 70% de temps en plus à prendre en compte dans les prévisions pour être intégrées totalement de façon qualitative, cependant, ce chiffre est passé à 40%, puis 30% jusqu'à être réduit à pratiquement 0% à l'heure actuelle.

8.3.2 Conception détaillée

8.3.2.1 Comment organiser dans le temps le refactoring de l'application et au sein du développement des nouvelles fonctionnalités ?

Rappelons-nous le chemin directeur de ces évolutions : refactorer le code en vue de la future mise en place d'un Micro-Framework de façon progressive et simultanée aux développements de nouvelles fonctionnalités, l'objectif final étant l'utilisation de toutes les règles et facilités conçues jusqu'à présent.

Ainsi, toute la partie de conception étant bien élaborée, structurée et définie il reste à mettre en place un plan d'action à respecter au travers des prochains mois et années de développement. Il faut donc trouver une organisation concrète à suivre au jour le jour me permettant d'intégrer de façon progressive le refactoring complet de l'application au sein du management et de la gestion du projet afin de fluidifier le travail et d'en maitriser l'avancement.

Face à un code « spaghetti » j'ai donc élaboré un plan d'action réparti sur les deux années de mon apprentissage de façon à ce qu'il soit réalisable de façon transparente et progressive car, nous l'avons vu lors de la phase d'analyse, il n'est pas possible de bloquer plusieurs semaines ou mois de ce projet pour réaliser ce refactoring. Ce plan doit donc être techniquement efficace mais également stratégiquement compatible avec les objectifs du projet répartis dans le temps. En plus de trouver des solutions techniques il faut que celles-ci optimisent l'application et le travail de l'équipe complète.

Détaillons donc ce plan sous forme de points directeurs à élaborer dans un ordre chronologique :

La tâche directrice consiste à accomplir sur toute la durée du projet l'optimisation du code qui est effectuée à chaque découverte d'une partie de l'existant. C'est en effet la solution la plus stratégique ici. Pour refactorer le code qui doit l'être, j'attendrais une nouvelle fonctionnalité à développer qui va me faire découvrir le fonctionnement d'une partie du code existant ce qui me donnera non seulement une bonne raison de comprendre le code mais aussi d'en optimiser le fonctionnement afin de l'adapter et généraliser son fonctionnement.

Dans un premier temps ce refactoring comprend une séparation du code pour ensuite une organisation des fichiers et des chemins ce qui permettra la création des premiers objets de programmation. Une fois ces bases posées l'instauration d'un fichier de configuration, l'intégration d'un modèle M.V.C, la création d'un module de réécriture d'url, n'en sera que plus facile. Il restera finalement à ajouter des commentaires pour finaliser le design et la compréhension de l'application.

8.3.2.1.1 La séparation du code

L'existant est à ce niveau composé de scripts contenant un mélange des langages suivants : PHP, Javascript, CSS, HTML, SQL or le patron de conception M.V.C consiste à séparer ces parties au maximum. La meilleure stratégie à adopter ici est donc la séparation de ces parties à chaque fois que je découvre une page composée de cette façon.

- Le langage PHP

Pour ce faire, le code PHP doit être au maximum déplacé dans des fonctions. Ces fonctions doivent être regroupées dans des fichiers qui ne contiennent que des fonctions. Ces fichiers doivent être créés par thèmes ce qui permettra, lorsqu'il sera temps de transformer l'application en objet, de convertir ces nombreuses fonctions en méthodes de classes. Ainsi, par exemple, toutes les fonctions qui retournent des résultats concernant la partie « Administration » de l'application seront regroupées dans un même fichier. Nous placerons dans un premier temps ces fichiers dans un dossier nommé « Lib ».

- Le langage Javascript

Le code Javascript, qui permet de dynamiser le fonctionnement des pages est intégré dans le code HTML, ce qui est également à éviter pour les modèles M.V.C où ce code qui est visible du client car intégré dans la source retournée au navigateur doit être déplacé dans une partie « publique » généralement nommée « Asset ». Pour s'aligner sur le modèle M.V.C ici le plus logique est donc de placer ce code Javascript dans des fichiers nommés de façon similaire aux vues sur lesquelles ils agissent.

- Le langage CSS

Le résonnement concernant le code Cascading Style Sheet permettant de designer les vues utilisées par le client est exactement le même que celui appliqué au code Javascript.

- Le langage HTML

Tout comme les autres langages, le HTML dans le modèle M.V.C est séparé autant que possible du reste du code. Stockées dans un dossier nommé « views » sous forme de fichiers du même nom que les classes par lesquelles elles sont appelées, les vues contiennent le maximum de code HTML et le minimum de variable PHP possible qui sont les paramètres envoyées depuis l'objet appelant. Cette explication est certes très techniques mais nécessaire pour bien comprendre la suite du plan.

- Le Langage SQL

Cette fois-ci le langage SQL chargé d'exécuter des requêtes en bases de données fait exception à la règle. Certains frameworks utilisant des ORM suppriment définitivement ce langage lors du développement, mais ici nous avons vu précédemment qu'il est impossible d'utiliser cette fonctionnalité dans notre cas. Le code SQL est donc supprimé des fichiers et scripts PHP pour être stocké en base de données. Cette action permet notamment de gagner du temps de traitement lors de l'exécution des requêtes importantes.

8.3.2.1.2 Rassemblement de tous les chemins de l'application

Après la séparation du code, la prochaine étape est d'uniformiser les chemins au sein des fichiers pour une gestion plus dynamique des appels des fonctions, des vues, du style et du JavaScript. Cette partie est très technique, elle est d'autant plus améliorée que les fonctions PHP sont regroupées dans des fichiers du même thème. Le nombre d'imports entre les différentes classes et différents fichiers est ainsi limité.

8.3.2.1.3 Création des premiers Objets : SQL, FTP, Types

La création des premiers objets de programmation a validé pour la première fois dans l'application tout le travail précédent étalé sur plusieurs mois. Il est désormais possible de créer des objets et de les insérer dans ce contexte de l'application de façon fonctionnelle.

Les premiers créés sont ceux qui ont permis de simplifier le fonctionnement de l'application au maximum. Ils sont également des objets très orientés métiers qui permettent l'intégration de l'application dans son environnement tels que la connexion à la base de données, la connexion au serveur FTP et l'utilisation des « Types » (les Types sont des catégories déclinées pour chaque matériel composant chaque système vendu par Thales Underwater Systems : hardware, Software, Fonctionnel, Knowledge, Configuration, Opérationnel …).

L'intégration de ces objets de programmation est le pas décisif qui a permis la validation de tous le refactoring effectué précédemment ainsi que le point de bascule vers un Micro-Framework désormais de plus en plus proche et réalisable.

8.3.2.1.4 Création d'un fichier de configuration

Ce point bloquant relevé dans la partie d'analyse du contexte vue précédemment est désormais possible à mettre en place. La qualité du code et les objets de programmation autorisent dès maintenant l'utilisation de valeurs telles que des identifiants et mots de passes tirés d'un fichier de configuration chargé au lancement de l'application.

Ainsi cette amélioration, bien que rapidement mise en place a donné au projet la possibilité de réellement gagner en crédibilité face au Système Informatique de Thales qui demande pour toute mise en production une administration facile et réalisable par des Administrateurs réseaux sans compétences de développement d'une part pour être capable de résoudre un problème en absence du responsable de l'application et d'une autre part pour répondre aux exigences du groupe.

Désormais, si une migration de serveur FTP, IIS, ou SQL venait à être effectuée, la modification est réalisable simplement et rapidement.

8.3.2.2 Intégration du modèle MVC

Maintenant que les bases sont posées, place à l'optimisation !

Après plusieurs mois de développement et de refactoring lourd, il est enfin possible d'intégrer une logique « Modèle, Vue, Contrôleur » ce qui aura pour effet de d'améliorer en compréhension, en temps et en qualité tout prochain développement.

Une fois de plus la question se pose à nouveau : comment procéder à ces évolutions conséquentes sans impacter la durée des objectifs du projet fixés.

Après une évaluation rapide de la faisabilité immédiate des tâches telles que la mise en place de la réécriture d'url ou du moteur de rendu, j'ai décidé de **commencer par la restructuration complète de l'arborescence ce qui m'a imposé de commencer l'écriture et le développement du module de réécriture d'url.**

8.3.2.2.1 Mise en place de l'url rewriting

Ce module permet de très bons avantages techniques mais est également ressentie indirectement sur la gestion de l'application par les autres collaborateurs.

Tout d'abord dans le domaine technique, l'application s'est vue dotée d'une centralisation de la construction des chemins. La classe « Root » et le script « URL » gèrent maintenant à eux seuls :

- Les appels entre les classes
- Les liens permettant l'utilisation des serveurs FTP
- Les liens pointant vers des fichiers situés sur les serveurs IIS
- Les liens des packages « générés » par l'application
    - Sans plus d'explications, cette partie est difficilement compréhensible. L'application SSDPM est capable d'extraire une partie de ses données en formant un package qui est ensuite consultable sur n'importe quel ordinateur ou tablette disposant d'un navigateur web. Ainsi, les liens créés pour ce package doivent être « relatifs » et donc « absolus » à l'opposé de ceux utilisés pour l'application jusqu'à présent.

Cette centralisation de la construction des chemins a rendu possible l'instauration d'un algorithme et d'une logique de fonctionnement unique. Cette logique de fonctionnement a par la même occasion été conçue pour assouplir certaines directives imposées par le système informatique.

En effet, depuis le début de ce mémoire, j'indique les serveurs FTP contenant des données de façon évidente, mais ça ne l'était pas au début des développements pour la bonne raison qu'ils n'existaient pas, la gestion des fichiers JPG et PDF était effectuée localement et directement sur le serveur IIS de développement, ce qui chez Thales est interdit pour toute application en Production.

Il fallait donc trouver une solution pour gérer ces fichiers et permettre la mise en production de l'application selon les règles groupes définies. Après plusieurs tentatives, j'ai choisi de sélectionner le protocole FTP qui assurait une sécurisation et une utilisation relativement simple des données. De plus il est possible d'utiliser plusieurs FTP différents sécurisés à différents niveaux en fonctions du besoin de l'application. Je sais notamment qu'il est possible que dans un futur relativement proche l'application soit appelée à héberger des documents « Spécial France » qui est un des niveaux de sécurisation des données les plus importants chez Thales. De plus, savoir que les FTP peuvent être sécurisés par une variante du protocole SSL qui est un des protocoles les plus sécurisés du web est une très bonne option.

Une fois le fonctionnement de l'application unifié, dynamique et qualifié, il est possible et beaucoup plus facile de passer à la partie mise en place du patron de conception Modèle, Vue, Contrôleur.

8.3.2.2.2 Fonctionnement MVC

Le patron de conception modèle vue contrôleur est un modèle de programmation destiné à optimiser la structure et le développement des applications interactive telles que les applications web qu'elles soient utilisées via une connexion internet ou en intranet.

L'avantage principal de cette méthode de développement étant qu'elle est particulièrement adapté aux applications web.

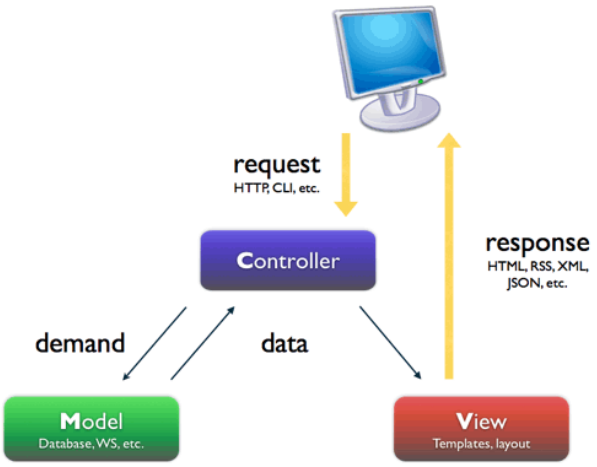


Figure 8 - Modèle M.V.C

Dans le schéma ci-dessus, la partie « View » représente l'interface utilisateur disponible via le navigateur web, la partie « Controller » représente les classes et script PHP utilisés côté serveur, tandis que la partie « Model » représente la base de données Oracle SQL.

8.3.2.2.2.1 Mise en place d'un patron de conception.

Ici, la mise en place d'une logique de fonctionnement fut relativement plus simple que ce que j'avais imaginé, et ce grâce à toute la préparation précédente du code.

En réutilisant l'image des Taxis utilisée précédemment pour comprendre et simplifier la problématique, nous pourrions voir les choses sous l'angle suivant : la mise en place du nouveau moteur du taxi a été simplifiée par l'organisation précédente des nouvelles pièces ajoutée sous le capot.

En effet, nous avons vu précédemment que les principaux objets sont maintenant fonctionnels, tout comme la réécriture d'url ainsi que les requêtes SQL qui ont maintenant disparues du code. Les fonctions sont triées par thèmes et réunies dans des fichiers spécifiques et le tout dans une arborescence qui ressemble de très prêt à celle d'une architecture MVC, il n'y avait donc qu'une « étincelle » de manquante et je l'appellerais la classe « Root ». En effet cette classe va faire le lien entre les différentes classes, les vues, le modèle, les contrôleurs et améliorer le fonctionnement de façon globale.

A partir de ce moment, la prévision des tâches futures à effectuer devient relativement claire :

- Créer la classe « Root »

La classe Root contient des méthodes qui permettent aux classes d'interagir de la façon M.V.C. Tout d'abord une méthode nommée « render » qui est le moteur de rendu de vues, il suffit de l'appeler pour afficher automatiquement la vue de la classe qui a fait appel à cette méthode, elle est donc contextuelle, puis les méthodes link, linkJs et linkCss qui permettent de construire dynamiquement les liens utilisé dans les vues pour chaque classe en fonction de son nom, ses héritages et du contexte également.

N'importe quelle classe héritant de la classe « Root » a donc accès à ces fonctionnalités

- Transformer progressivement tous les fichiers qui contiennent des fonctions du même thème en classe nommées héritant de la classe Root.

Tout nouveau développement est à partir de cette étape, considéré comme « stable », quant à l'ancien code il sera transformé à chaque fois que je passe sur un script, en classe orientée objet et intégrée dans le fonctionnement M.V.C.

Bien que ces développements soient effectués selon des standards de programmation connus des développeurs, il n'empêche que certains restent bien spécifiques à l'application et aux décisions orientées métier prises tout au long de la conception et de la création et deux actions sont possibles pour les clarifier au maximum, tout d'aborder conserver une logique de fonctionnement constante, puis commenter le code aux maximum.

8.3.2.2.2.2 Logique de fonctionnement

Dans une optique de clarification maximale, le nom des méthodes de classes a été uniformisé au maximum permettant ainsi de deviner leur fonction uniquement via leur nom,

8.3.2.2.2.3 Ajout des commentaires

Aussi, il est indispensable d'ajouter des commentaires au fur et à mesure des avancées, ce que j'ai effectué tout au long du développement des nouvelles fonctionnalités mais il reste une très grande partie du code existant qui n'était pas commenté et le fait qu'il soit ordonné ne suffit pas à permettre la compréhension facile d'un algorithme ou de spécificités.

De plus, ajouter des commentaires au code est également un critère de qualité dans le développement d'une application qui permet d'expliquer des choix et de mettre en lumière. Ainsi des mots clés tels que *@todo*, *@deprecated* ou *@refactoring* sont utilisés dans toute l'application. Tout comme le versionnage, c'est également un avantage majeur pour le travail en équipe et la stabilité du projet sur une longue durée.

8.4 TEST ET VALIDATION

Nous arrivons ici, dans la deuxième partie d'un « cycle en V » traditionnel, mais dans mon cas, l'équipe de développeurs étant limité à une seule personne, il n'y avait pas assez de ressources pour refactorer, développer, optimiser et tester de façon simultanée chaque nouveau développement.

J'ai alors décidé de mettre en place une logique de versionnage de l'application complète : chaque fois qu'une fonctionnalité importante sera développée je sauvegarderais la totalité des sources de l'application ainsi que la structure de la base de données sous la forme d'une archive au format Zip qui contiendra en plus un fichier Excel listant toutes les fonctionnalités existantes et pour chacune :

- La date du dernier test effectué
- La validation ou non du fonctionnement
- La liste des bugs connus

Il a également été très utile de mettre en place quelques mois plus tard une page de suivi des Exceptions retournées par l'application ainsi qu'un historique des actions effectuées. Ainsi, lorsqu'un bug est détecté, il devient d'autant plus facile de débuguer l'application en se référant à la dernière action effectuée, l'utilisateur qui l'a déclenchée et le détail de l'erreur.

Ainsi, environ toutes les 8 à 10 semaines, je sauvegarde les sources de l'application ainsi que valide et débugue toutes les fonctionnalités. Cependant, l'application devenant plus stable, cette étape devient de plus en plus rapide à effectuer pour deux raisons, premièrement parce que les évolutions techniques sont beaucoup moins importantes mais aussi parce que les effets de bords sont de plus en plus rares.

Le code orienté objet a cette capacité d'apporter au développeur la maîtrise d'un grand nombre d'erreurs et de la logique pour chaque développement. C'est donc grâce au paradigme de la programmation orientée objet que l'application est devenue stable et que la phase de test a pu être réduite.

Une fois de plus, étant le seul développeur attribué de façon fixe au projet, la phase de test avait besoin d'être réduite, le temps devant absolument être gagné sur tous les points possibles du projet pour permettre une bonne avancée dans les objectifs et cette phase de test pouvait être améliorée d'une part en aillant une meilleure expérience du code mais aussi d'une autre part en maîtrisant les erreurs, ce que les évolutions en qualité du code ont permis d'atteindre.

8.5 REUNIONS

Pendant plusieurs mois depuis le début de mon apprentissage, les réunions d'avancement ont été organisées par mon tuteur de façon hebdomadaire ce qui était relativement compliqué surtout pendant ma période de présence 2 jours par semaine en entreprise.

Les activités pendant ces réunions correspondaient à effectuer l'état des lieux concernant l'avancement effectué la semaine passée mais aussi en l'évaluation en temps de chaque tâche décrite dans la spécification fonctionnelle puis leur organisation dans les prochains jours selon leurs priorités.

Au fur et à mesure de l'avancée du projet, j'ai évidemment gagné en maîtrise sur le contexte et le développement de l'application SSDPM mais aussi sur le déroulement et la gestion du projet. De plus l'équipe a également compris mes motivations et m'a accordé sa confiance à propos des décisions techniques que je proposais, ce qui a déclenché pour ma deuxième année d'apprentissage un transfert de responsabilité concernant l'organisation des réunions.

Ainsi, je me suis vu préparer chaque réunion mais aussi les animer et rédiger les comptes rendus. Ma stratégie étant la suivante : une réunion doit porter principalement sur les sujets bloquants d'un projet qu'ils soient en cours ou à effectuer.

Donc **avant chaque réunion** j'envoie un mail de préparation contenant les points effectués la/les semaine(s) précédente(s), les points bloquants et les points à effectuer la/les semaine(s) suivante(s).

Pendant les réunions nous débattons dans un premier temps des points bloquants, puis de la priorité des points restants à effectuer sur le court ou long terme pour finalement s'il reste du temps débattre des suggestions possibles.

Pour finir, **après chaque réunion** j'envoie un message récapitulatif dans les 24h pour fixer les décisions abordées et choisies pendant la réunion.

L'ensemble de ces informations pour toutes les réunions est conservé dans le mail transmis avant et après chaque réunion ce qui permet de consulter facilement et rapidement l'historique d'avancée du projet tout en permettant la vérification qu'aucune information n'est redondante ou décision contradictoire. Cela permet également de consulter les dates des dernières réunions pour vérifier leur régularité si besoin est.

Evidemment ce mode de fonctionnement n'a pas été mis en place sans l'avis et le retour des toute l'équipe. J'ai également demandé l'avis de mon tuteur sur le fonctionnement qui serait le plus approprié. Changer ses habitudes n'est pas toujours la meilleure solution.

8.6 SUIVI DES TACHES

La zone de travail principale pendant les réunions a été dans un premier temps une feuille Excel listant toutes les fonctionnalités devant être développées du début de l'application jusqu'à la fin. Chaque fonctionnalité étant considérée comme une tâche et évaluée en temps (nombre de jours). Cependant, l'approche restait très fonctionnelle et peu technique ce qui permettait mal de retranscrire avec exactitude la difficulté d'une tâche :

Par exemple :

La fonctionnalité « créer un bouton pour envoyer plusieurs images au lieu d'une seule » peut facilement être estimée à 30 jours de développement ! Ce n'est pourtant qu'un bouton … oui mais ! Ce bouton implique une refonte de la base de données, de l'interface et du module d'envoi des images et il faut en plus compter le temps de refactoring associé.

Ayant déjà une expérience de travail en entreprise, j'ai proposé une approche plus technique des tâches via un logiciel disponible chez Thales : PMAT qui est un outil de suivi des actions qui permet d'associer à chaque tâche de façon rapide un commentaire pour répertorier un bug ou d'avancement. Il a également l'énorme avantage d'être accessible par plusieurs personnes en même temps, ce qui peut paraître évident, mais pourtant une feuille Excel ne l'est pas et des conflits d'accès étaient rencontrés régulièrement.

8.6.1 SCRUM

J'ai évidemment immédiatement pensé à la méthode agile SCRUM concernant la gestion de projet et l'avancement des tâches. Découvert en formation et appliqué dans de nombreuse entreprise c'est une méthode qui a fait ses preuves aussi bien sur des projets de petites tailles que des projets de grande envergure. Cependant lors du développement d'une application dans une équipe composée d'un seul développeur, est-il vraiment nécessaire de refondre tout le fonctionnement actuel et de procéder à la formation de toute l'équipe qui a son propre mode de fonctionnement depuis plusieurs années là où un simple outil de suivi et de gestion des tâches est suffisant ? Après une phase de recherche auprès des autres équipes de développement au sein de Thales, il s'est avéré que dans ce cas précis SCRUM n'était pas la méthode idéale. Elle n'est cependant pas oubliée et j'imagine la proposer à nouveau si l'équipe de développement venait à s'agrandir jusqu'à 3,4 ou plus développeurs.

De plus, SCRUM n'est pas utilisée pour de nombreuses raisons sur d'autres projets en cours de développement au sein de Thales Underwater Systems, la principale étant que des développeurs travaillent souvent sur plusieurs projets de façon parallèle.

9 UNE SOLUTION SPECIFIQUE ET PARTICULIERE

9.1 UNE CONCEPTION ADAPTEE A L'EXISTANT

L'ensemble de ces évolutions et de ces décisions techniques, managériales et stratégiques ont contribuées au caractère original de cette solution qui, si le contexte avait été différent, de ne serait-ce que d'une composante, aurait pût être totalement différente.

Prenons par exemple une situation où l'ancien stagiaire qui avait créé l'application serait resté sur ce projet. Ne connaissant pas les standards du web ou en tout cas ne les appliquant pas sur ce projet, le code spaghetti serait resté ce qu'il était et aurait évolué en conséquence. Les effets de bords auraient été beaucoup plus nombreux et les objectifs repoussés. Le projet aurait également pu souffrir d'un défaut de crédibilité. Le travail à plusieurs développeurs aurait été rendu beaucoup plus compliqué ce qui, avec l'embauche d'un prestataire comme c'est le cas aujourd'hui aurait considérablement augmenté le coût de développement de chaque nouvelle fonctionnalité.

J'ai également cherché une évaluation par les pairs en plaçant dans ma situation au cours d'une discussion des d'anciens collègues ou des collègues de travail et de formation afin de me remettre en question avant de commencer le développement. Beaucoup ont évoqué le courage d'une telle décision, la patience et auraient choisi une des deux options « facile » qui s'offrent à nous en tant que stagiaire ou apprenti : soit le redéveloppement complet de l'application si l'équipe de travail accorde un délai pour cette activité, mais dans ce cas-là l'intérêt du stage s'en trouverai limité puisque il n'y aurait pas de challenge, de conception ou de recherche de la satisfaction du client puisque les fonctionnalités à développer existent déjà, soit la poursuite des développements avec le code existant en essayant d'y toucher le moins possible mais c'est également problématique puisque le code existant était censé servir de base pour l'application et qu'en l'état il ne le permettait pas.

Imaginons maintenant que Thales Underwater Systems aie embauché un professionnel, prestataire ou intérimaire pour continuer le développement de l'application : Les développements auraient sans doute nécessité plus de temps qu'en situation normale et des coûts supplémentaires se seraient vite fait sentir.

Il y a également une partie de la communauté des développeurs qui évite le code « spaghetti » en entreprise. Il est arrivé que des développeurs refusent un emploi à cause de cette spécificité, de peur de devoir développer pendant plusieurs années du code de mauvaise qualité dont les erreurs et effets de bords auraient finalement dues être assumées et dont le travail ne saurait être mis en avant sur un C.V.

Pour appuyer mes propos avec un exemple concret, cette affirmation me rappelle l'affaire Toyota de Mars 2014 dont le non-respect des normes et l'écriture d'un code « spaghetti » par les développeurs avait coûté la vie à une automobiliste qui avait acheté le modèle « camry » et dont la commande de frein n'avait pas fonctionnée … dangereux ! (lien)

La solution de refactoring a également pour originalité d'avoir été effectuée de façon progressive, modérée et transparente. Aucune période n'a vraiment été réservée pour effectuer des améliorations conséquentes du code. Ainsi tous les objectifs se sont vus légèrement rallongés mais l'impact n'a pas été bloquant à quelque moment que ce soit.

La situation étant d'autant plus complexe que j'étais l'unique développeur à effectuer ces améliorations et que le travail n'a pu être partagé entre plusieurs personnes. La solution devait donc être parfaitement maîtrisée. Les bugs devaient aussi être résolus le plus rapidement possible car quelques mois après le début des développements des prestataires ont été embauchés pour utiliser l'application grâce aux premières fonctionnalités disponibles.

Il y a donc ici deux originalités dans ce contexte qui ont très fortement influencés la gestion du projet et qui ont reposés sur mes décisions :

- **Le choix du refactoring**

Refactorer une application est un tâche très gourmande en temps et qui se relève être une véritable difficulté pour les développeurs qui doivent repenser chaque partie du code déjà fonctionnel en un meilleur code. Refaire l'existant n'est généralement pas une activité passionnante ou gratifiante. J'ai pourtant choisi cette option tout d'abord pour l'utilité qu'elle pouvait apporter au projet mais aussi pour le défi qu'elle représentait qui m'intéressait et collait particulièrement bien avec mon historique professionnelle et scolaire.

- **Le choix de création d'un Micro Framework**

Créer un Micro-Framework est ce que l'on peut appeler dans le monde des développeurs du « Do-It-Yourself » (DIY) qui est, nous le verrons dans les critiques qui vont suivre, une pratique à n'exercer que dans des cas spécifiques. Ici justement ce projet de par son historique était une de ces originalités qui se prêtaient fortement à du développement DIY et non à la mise en place d'un Framework existant.

9.2 CRITIQUE

Bien que de nombreuses situations complexes soient survenues tout au long du projet, la solution choisie a je pense été une des plus rapide, stable, dynamique et sécurisante concernant les futures évolutions souhaitées.

La plus-value importante apportée par cette solution est l'amélioration constante des développements existants, chaque amélioration étant réalisée dans l'optique de faire gagner du temps lors des prochaines réalisations. Ce qui s'est avéré être une tâche particulièrement difficile au début, mais de moins en moins lourde et de plus en plus appréciable puisque de plus en plus de code était réutilisé dans les nouvelles fonctionnalités. Ainsi très rapidement, dès les 4 premiers mois de développements les bénéfices se sont fait sentir jusqu'à la finalisation du refactoring.

J'ai, depuis le début de ce mémoire et du projet mis en avant les nombreux avantages d'avoir choisi une telle approche mais il y a également des contraintes qui ont été soulevées de par la sélection même de ces méthodes de travail.

Tout d'abord, être le seul développeur d'un projet incite généralement à sélectionner une solution sûre déjà testées et qui a fait ses preuves. Un Framework open source récupérable sur Internet aurait évidemment été un très bon choix, toutefois j'ai entrepris de développer un Micro-Framework en DIY ce qui amené un inconvénient indéniable, la limite des réalisations à mes seules connaissances. Ainsi, la qualité du code, de développements, des réalisations, de la sécurité et des choix techniques, a donc été borné par mes connaissances acquises durant mes expériences professionnelles, ma période de formation et mes recherches préalables. N'ayant pas de pair capable de vérifier la technicité de mon travail via une « peer review », j'ai donc choisi de garder en tête cette information et d'assurer la plus grande clarté et simplicité possible de compréhension dans mon code pour des améliorations futures présumées qu'elles soient de l'ordre de la conception ou autre.

La seconde contrainte apportée par cette solution est l'orientation métier du code. Pour coller parfaitement à l'existant et rendre possible certaines améliorations ou conceptions spécifiques dans l'application SSDPM, certains objets et fonctionnement on dut être réalisés en accord avec les principes métiers de l'entreprise comme par exemple la création de liens internes au format « http » absolu pour les vues utilisée par les utilisateurs ou encore la création de ces mêmes liens de façon automatisée par la classe « Type » pour tout objet qui hérite de celle-ci.
Cette orientation métier très spécifique peu rendre l'apprentissage de l'application plus difficile que prévu par un non initié mais aussi l'intégration de modules open source plus complexe en fonction de leur besoins de fonctionnements et de leur niveau d'adaptation au niveau du code et de la base de données.

10 ANALYSE DE L'APPROCHE CHOISIE

10.1 RESULTATS OBTENUS

Choisir une approche aussi complexe que le refactoring d'une application pour la création d'un Micro-Framework en vue d'obtenir une application qui fonctionne sur des standards connus par la majorité des développeurs est nous l'avons vu, la meilleure de solutions qui était applicable à ce projet.

Anciennement très peu ou pas du tout maintenable, compréhensible et évolutive, l'application est maintenant totalement fonctionnelle et basée sur un Micro-Framework donc le fonctionnement est basé sur patron de conception « Modèle, Vue, Contrôleur ». Tous les modules ont été développés de la façon la plus générique possible. Des concepts puissants tels que les héritages et les importations de classes, la gestion des erreurs, la création automatique des chemins FTP nécessaires pour les objets qui en ont besoins sont maintenant également intégrés et permettent un développement plus souple dont les erreurs ne provoquent pratiquement plus aucun effet de bord qui, quand ils se produisent, peuvent être maîtrisés et résolus grâce à la clarté et au fonctionnement du code de façon vraiment rapide.

La documentation technique de l'application a également pu être effectuée et pratiquement divisée par 10 en nombre de pages face à ce qui aurait dû être fait en amont de cette solution. Maintenant que les logiques et les bases sont posées, il suffit de donner leurs règles et rares spécificités pour qu'un développeur de niveau ingénieur puisse comprendre et devenir efficace dans les jours qui suivent sa lecture.

Avec cette affirmation vient une expérience concrète qui s'est déroulée depuis le mois de juillet jusqu'à aujourd'hui, un développeur supplémentaire, David, a rejoint le projet de façon partielle pour récupérer la connaissance ainsi que pour aider dans l'avancée de certaines tâches considérées comme importantes et longues à réaliser. J'ai ainsi réalisé une ébauche de documentation technique dans un but de formation et, après quelques heures de découvertes David, et ce bien qu'il ne soit à l'origine pas un développeur web, est maintenant autonome quand-à l'utilisation du Framework. Il a d'ailleurs déjà développé un module supplémentaire qui a été présenté le 12 septembre. En tout, David a travaillé environ 15 heures sur ce projet, deux à trois jours sont donc nécessaires pour prendre ses marques et développer une fonctionnalité via le Micro-Framework que j'ai nommé **« Evolve »**

10.2 ANALYSE DU CHAMP D'APPLICATIONS DE LA SOLUTION ELABOREE

Le Micro-Framework « Evolve » permet maintenant des développements plus rapides avec une logique de travail récurrente et orientée métier, mais l'application de cette solution va s'étendre jusqu'à, peut-être, la création d'une nouvelle application web profitant ainsi des bases et des améliorations effectuées. Cette fois-ci je pourrais à nouveau me reposer les questions essentielles : « utiliser un Framework ? Développer un Framework ? » Si je n'avais pas déjà fait comme je l'ai démontré au travers de ce mémoire un travail d'analyse et de recherche approfondie. La demande de cette nouvelle application étant très similaire autant de par la programmation que par les fonctionnalités qui sont souhaitées, « Evolve » est le plus à même de répondre aux besoins techniques et fonctionnels ainsi qu'aux ressources humaines et financières qui sont liés à ce projet et limitées. En plus donc d'avoir amélioré la réussite du projet, « Evolve » va trouver une utilité supplémentaire et permettre la création de nouveaux projets au sein de Thales Underwater Systems.

10.3 MISE EN PERSPECTIVE

Bien que à l'heure actuelle la solution que j'ai élaborée soit auto suffisante et puisse être réutilisée je ne cesse de continuer à penser à des améliorations qui pourraient venir améliorer son fonctionnement et le temps gagné par les développeurs.

Dans un futur immédiat je pense fortement à la mise en place d'un code permettant d'utiliser la méthode, évoquée précédemment dans la partie Analyse et Recherches, Object-Relationnal Mapping. Cette méthode aurait dans d'autres circonstances contextuelles pu être implémentée au sein du Micro-Framework dès le début des développements. Malheureusement, le refactoring des requêtes SQL étant trop important à effectuer pour permettre le refactoring immédiat de toutes les requêtes, cette partie a dû être laissée en suspend. Maintenant que les requêtes sont toutes stockées en base de données et rassemblée à un endroit unique, il devient beaucoup plus envisageable d'implémenter cette technique de programmation.

Pour ce qui est des prochaines années à venir, car cette opération risque de demander beaucoup de temps de préparation, je pense qu'il serait réellement utile de préparer et « merger » le Micro-Framework « Evolve » dans un Framework tel que Yii. Cette action aurait un double intérêt majeur : premièrement elle permettrait de profiter d'une documentation technique enrichie, d'une communauté et de l'aide sur les développements voire même de recruter une personne déjà qualifiée, et, deuxièmement, elle comblerait tous les déficits qui ne sont à l'heure actuelle pas indispensables mais qui pourraient le devenir grâce à la présence de nouvelles possibilités de développement apportées par le Framework. Il est parfois nécessaire de créer le besoin pour découvrir de nouvelles perspectives.

11 LE REFACTORING, UNE VOLONTE D'AMELIORATION

11.1 EVALUATION DE CE TRAVAIL

Tout au long de ce cursus de 5 années, j'ai progressé tant sur le plan scolaire que professionnel et personnel.

Ce mémoire est pour moi une conclusion qui répond à une grande partie du travail que j'ai effectué dans un but de découverte, d'analyse, de recherche, de curiosité, d'utilité, de nécessité ou bien évidemment d'apprentissage. Il m'a permis de synthétiser, de formaliser, de rassembler et de réunir toutes les informations et réalisations que j'ai relevées à la manière d'un défi. Jusqu'à ma dernière année de stage, je ne savais pas que mon mémoire porterait sur le sujet du refactoring qui maintenant chez la communauté des développeurs est logiquement évité au maximum.

C'est en regardant la totalité de mon parcours que je me suis rendu compte qu'il fallait, dans un premier temps, formaliser une méthode concrète voir écrire un livre blanc sur mon travail récent qui pourrait peut-être aider d'autres personnes. Car au travers de mes recherches j'ai trouvé des méthodes qui détaillent de façon globale comment refactorer une application mais toutes sont très orientées technique et aucune ne présentait le cas dans une entreprise et les différentes solutions, avec leurs avantages et inconvénients, qui s'offraient aux développeurs dans cette situation. J'espère donc au travers de ce mémoire avoir pu expliciter mes idées dans la plus grande clarté possible de manière à ce que les idées principales qui en ressortent puissent resservir un jour peut-être que ce soit dans ma carrière ou ailleurs.

Cette volonté de refactoring d'application m'a d'un autre côté focalisé vers un monde de développement exclusivement web. Toutes les applications que j'ai développées dans un contexte professionnel sont écrites en PHP, langage qui est un très utilisé pour l'apprentissage par les débutants en programmation ce qui implique que beaucoup de développeurs avancés ou non peuvent ajouter cette « case » sur leur C.V. Cette réalité implique une baisse des salaires et des tests de technicité aux entretiens d'embauche relativement longs et difficiles à préparer puisqu'une grande partie de la communauté connaît ce langage.

Bien que la valeur ajoutée du langage ne soit pas impressionnante, je pense que la conception d'un Micro-Framework peut l'être plus facilement car il n'est pas courant que la situation se prête à la prise d'une telle décision et encore moins pour un apprenti qui n'a pas encore terminé ses études. Généralement un tel développement est complètement bloqué par le temps qui est nécessaire.

11.2 Des Acquis Diversifies

L'ensemble des expériences professionnelles ont gagnées en complexité de façon parallèle à mon apprentissage scolaire. Juste après le baccalauréat, j'ai commencé à apprendre la programmation orientée objet ainsi que le développement web et ai eu l'opportunité d'appliquer ces connaissances chez Monitoring Company. Par la suite j'ai découvert les Frameworks et les patrons de conception M.V.C que j'ai également pu mettre en application chez Interbat Services, expérience qui m'a apporté une plus-value : la découverte et l'amélioration d'un Framework Privé. C'est à ce moment que j'ai commencé à remettre en question l'utilisation d'un code dont la phase de conception avait déjà été achevée avec les Framework disponibles : *est-ce toujours la meilleure solution ?* Pour finir, le sujet d'apprentissage chez Thales Underwater Systems collait parfaitement avec mes volontés de réalisation scolaires, personnelles et professionnelles : concevoir un Framework de toutes pièces selon un besoin spécifique dans un contexte d'entreprise particulier.

« Evolve » est désormais un Micro-Framework qui est maintenant fonctionnel et qui permet de faire gagner du temps et des ressources au projet pour lequel il a été créé, c'est son avantage principal et indéniable. Cependant il est évident, et ce pour beaucoup de raisons telles que le temps, l'expérience, le nombre de développeurs et la fraîcheur de l'équipe, qu'il reste jeune et limité. Mais, comme nous l'avons remarqué au travers de ce mémoire, le concevoir m'a permis de progresser dans des domaines tels que la conception en programmation mais aussi dans l'organisation d'une activité sur une durée de plusieurs années avec une gestion du temps de façon détaillée et progressive.

Cette organisation a plus précisément été constituée de réunions que j'ai préparées, animées et résumées mais également de gestion des tâches qu'il a été nécessaire d'évaluer en temps et de prioriser, ce qui a été fait en groupe avec mes responsables. Ces responsabilités ont développées ma capacité de communication, de prise de décisions, de suivi de projet mais aussi de travail en équipe.

Travailler en équipe a aussi été un vecteur d'apprentissage vers de nouveaux acquis tels que la recherche d'informations qui dans une TPE - Très Petite Entreprise - sont à portée de mains mais qui dans un groupe comme Thales devient une tâche répétitive qui implique un mélange de recherche, de contacts et relations, répartis tout au long des réunions prévues plusieurs semaines à l'avance.

J'ai également récemment acquis une expérience en formation qui est d'autant plus intéressante qu'elle est basée sur mon travail, le Micro-Framework « Evolve ». J'ai présenté précédemment dans le mémoire David qui m'apporte maintenant des retours que ce soit sur la technicité et le code en lui-même, mais également sur ma façon de travailler, de m'organiser et de me répartir dans le temps.

11.3 Perspectives d'Evolutions Professionnelles

Ces compétences acquises tout au long de mon expérience professionnelle constituent maintenant l'essentiel de mes atouts à présenter lors d'entretiens de recrutements en plus de mon parcours scolaire atypique.

Tout d'abord, le refactoring du code pour obtenir une application de meilleure qualité est révélateur d'un état d'esprit constant et général qui montre la volonté de toujours se remettre en question et s'améliorer afin de maximiser la qualité du résultat souhaité.

Les analyses et recherches effectuées en amont des développements montrent une volonté de « faire bien du premier coup » pour ne pas avoir à revenir et arrière et effectuer deux ou plusieurs fois la même tâche suite à des erreurs survenues à cause d'un manque de réflexion précédent la réalisation. C'est pour moi une des qualités indispensables pour un ingénieur.

La conception d'un Micro-Framework est un défi sur le projet dont la prise de décision et la réalisation étaient difficile. J'ai, au travers de cette activité, pu mettre en valeur mes connaissances théoriques et les appliquer sur un cas concret. La finalisation et la réussite de fonctionnement constituent ainsi un point fort pour mon expérience professionnelle.

La réalisation de documentations techniques, d'exploitation, de mise en production est généralement un point évité par les développeurs, qui, très orientés techniques depuis le début de leurs études, n'aiment pas vraiment rédiger. J'ai cependant eu à les effectuer et finaliser ce qui m'a incité à analyser et remettre en question mon travail effectué

L'animation de réunions est un point que j'avais du mal à maîtriser au début de mon parcours scolaire et professionnel, mais avec la pratique et la prise en compte des conseils avisés de mes équipes successives, j'ai surmonté ce manque pour aujourd'hui être capable d'organiser, animer, résumer des réunions utiles et concises portant sur les objectifs importants et/ou bloquants d'un projet.

Un des points qui au final manque le plus dans ma carrière professionnelle est la pratique de la méthode agile SCRUM qui de nos jours est de plus en plus utilisée et requise dans les postes de développement informatique.

Ces développements avancés avec le langage PHP et activités réalisées au travers d'expériences professionnelles réalisées en parallèle de mes études, m'ont ouverts de nouvelles opportunités de postes qui sont habituellement réservées aux développeurs ayant une expérience de 5 ans ou plus dans le domaine et le langage tels que « Expert PHP » ou « Concepteur Développeur PHP ».

J'espère également de part ces expériences avoir adopté un état d'esprit et acquis un savoir-faire en gestion de projet suffisant pour me permettre la candidature à un poste marginal ce

qui de mon point de vue est généralement un poste bien plus intéressant de par les analyses, recherches et conceptions qui sont à effectuer, de plus ces postes ne se limitent en général pas à des développements techniques, ils demandent de la conception, de la réflexion, des recherches, de la communication etc …

12 CONCLUSION

Tout au long de mon cursus scolaire j'ai toujours été passionné d'innovations. La question « Comment faire mieux que ce que nous avons avec ce que nous avons ? » m'a toujours intriguée. Elle peut s'appliquer dans n'importe quel domaine que ce soit la finance ou les sciences par exemple etc .... Il semblerait que ce soit une question tout à fait naturelle puisque la nature est également basée sur ce schéma, l'évolution. Je me suis d'ailleurs inspiré de cette logique pour choisir le nom du Micro-Framework créé ces deux dernières années : « Evolve ».Je pense donc avoir recherché un sujet d'alternance qui bien qu'à une petite échelle de projet au sein d'une entreprise me propose cette problématique.

En finalité, ces développements auront permis la stabilisation du projet et la création de bases solides pour l'application SSDPM et réutilisables pour une autre application dans un contexte très similaire. Ce travail nous a montré qu'il est possible d'optimiser une application et son code malgré toutes les contraintes qui peuvent s'appliquer sur un projet et ce, sur une durée relativement courte. Bien que le projet soit naissant et les ressources financières et humaines soient légères, une telle opération a été rendue possible sous la forme de répartition des tâches concernant le refactoring de façon progressive laissant ainsi la possibilité de réalisation des objectifs fixés, le temps utilisé n'étant au final évalué qu'à 1h30 par jour. Si plusieurs développeurs avaient été assignés au projet, la contrainte des ressources humaines aurait été plus facile à gérer évidemment bien qu'il reste un facteur émotionnel du développeur qui effectuera toute la journée un travail qui est déjà fait et qui n'aurait, donc, pas eu de résultat visible pour obtenir une satisfaction personnelle, ce qui est tout de même important sur une période de deux années.

Cette problématique et sa réponse peuvent être caractérisés comme particuliers et surprenants de par leur contexte de mise en place et leur initialisation. En effet, l'application a été débutée par un apprenti qui a donné naissance à un code qui a impliqué cette problématique que tout développeur de niveau ingénieur aurait eu à se poser en reprenant le projet. Généralement les contextes professionnels évitent au maximum ces éventualités.

Ce mémoire aura soulevé plusieurs questions importantes qui doivent revenir dès que l'un des facteurs de la situation est modifié. Est-il obligatoire de redévelopper complètement l'application ? Est-il possible d'utiliser un Framework existant pour éviter le DIY ? Il y a-t-il des objectifs prioritaires à prendre en compte ou est-il possible de bloquer une certaine période pour optimiser le code de l'application ?

Ces questions nous ont menées des hypothèses qui s'entre mêlaient dans les possibilités des actions à effectuer et qui ont eu un impact important sur la gestion, la stabilité et l'avancement du projet. Ces quatre points principaux sont à la base de la réflexion menée : refactorer pour concevoir, refactorer pour convertir, redévelopper pour concevoir ou redévelopper pour utiliser

un Framework. Nous l'avons compris ces hypothèses sont toutes valables avec leurs avantages et inconvénients mais une seule est la plus adaptée en fonction du projet, le tout est de choisir la bonne au risque de perdre les investissements.

Aussi, à la problématique « **Comment optimiser une application codée selon un paradigme procédural vers une programmation orientée objet dans le contexte contraint d'un projet déjà amorcé en entreprise ?** » nous avons compris au travers de ce mémoire qu'il n'y a pas de réponse théorique mais bien plusieurs réponses concrètes qui peuvent s'appliquer selon le contexte de l'entreprise. Dans le cas de l'application SSDPM chez Thales Underwater Systems, après une phase d'analyse, il s'est avéré que la meilleure réponse est l'optimisation par le refactoring, l'amélioration de la qualité du code, la conception de classes et modules métiers, et la communication avec le système informatique dans une vision de construction, de clarté, de facilité et de gain de temps pour tous les développements à venir ce qui implique un impact positif conséquent sur la gestion du projet et l'évaluation des nouvelles fonctionnalités.

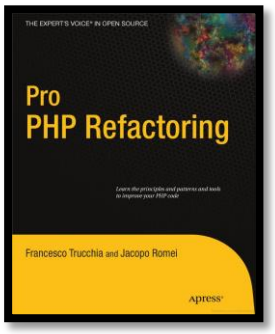
De par mes recherches et lectures concernant le refactoring des applications j'ai découvert évidemment le livre « Pro PHP Refactoring » et les livres blancs « Framework PHP Pour l'Entreprise » ainsi que « PHP en entreprise », mais aussi des cas similaires à celui que j'ai rencontré dans le langage « Cobol » qui se posent aujourd'hui auprès de nombreuses entreprises

Le site Silicon.fr démontre qu'environ 50% des entreprises maintiennent en production leur ancienne application dans ce langage et 45% ont débloquées un budget pour permettre le paiement d'un projet effectué en parallèle consistant en un redéveloppement total de l'application existante pour 2014-2015. C'est cette solution que j'ai cherché à éviter dans mon mémoire et ai essayé de faire ressortir tout au long de mes argumentations : cette décision demande des ressources humaines et financières que 55% des autres entreprises ne semblent pas pouvoir se permettre ce qui les pousse à faire appel à des prestataires pour le maintien de l'application. Les hypothèses posées dans ce mémoire, auraient-elles pu permettre une évolution progressive de ces entreprises et rendre leur migration vers une application optimisée plus transparente et économique ?

13 TABLE DES ILLUSTRATIONS

14 TABLES DES ANNEXES

15 BIBLIOGRAPHIE



Pro PHP Refactoring

Lien HTTP

Livres blancs

Framework PHP pour l'entreprise

Lien HTTP

PHP en entreprise

Lien HTTP

16 WEBOGRAPHIE

Developpez.com/…Specification-Formelle

Spécification Formelle PHP

http://php.developpez.com

Actualité PHP

bpesquet.developpez.com/tutoriels/…evoluer-architecture-mvc

Evoluer vers une architecture MVC en PHP

php.net/manual/...new-features

Nouvelles fonctionnalités PHP

http://php.net/manual/fr

Documentation PHP

wikipedia.org/…Histoire_des_langages_de_programmation

Histoire des langages de programmation

wikipedia.org/….Thales_Group

Thales Group

domaindrivendesign.org

domaindrivendesign.org

linuxfr.org/…/encore-un-exemple-de-code-spaghetti-toyota

Code Spaghetti, exemple Toyota

silicon.fr/…Cobol…

Application Cobol et Refactoring

http://fr.wikipedia.org/wiki/Principe_KISS

http://fr.wikipedia.org/wiki/Test_Driven_Development

http://fr.wikipedia.org/wiki/Ne_vous_r%C3%A9p%C3%A9tez_pas

http://fr.wikipedia.org/wiki/Conception_pilot%C3%A9e_par_le_domaine

http://fr.wikipedia.org/wiki/Patron_de_conception

Principes de Programmation

17 ANNEXES

Annexe 1 – Logiciels utilisés pour la gestion du projet

Plusieurs logiciels ont été indispensables pour le développement ainsi que la gestion du projet :

Toad : Développement de packages, fonctions, procédures, triggers, séquences SQL

NetBeans : Développement PHP de l'application

TortoiseSVN : Extension Shell pour Windows permettant d'intégrer les fonctionnalités de Subversion dans le menu « Clic droit » de l'exploreur.

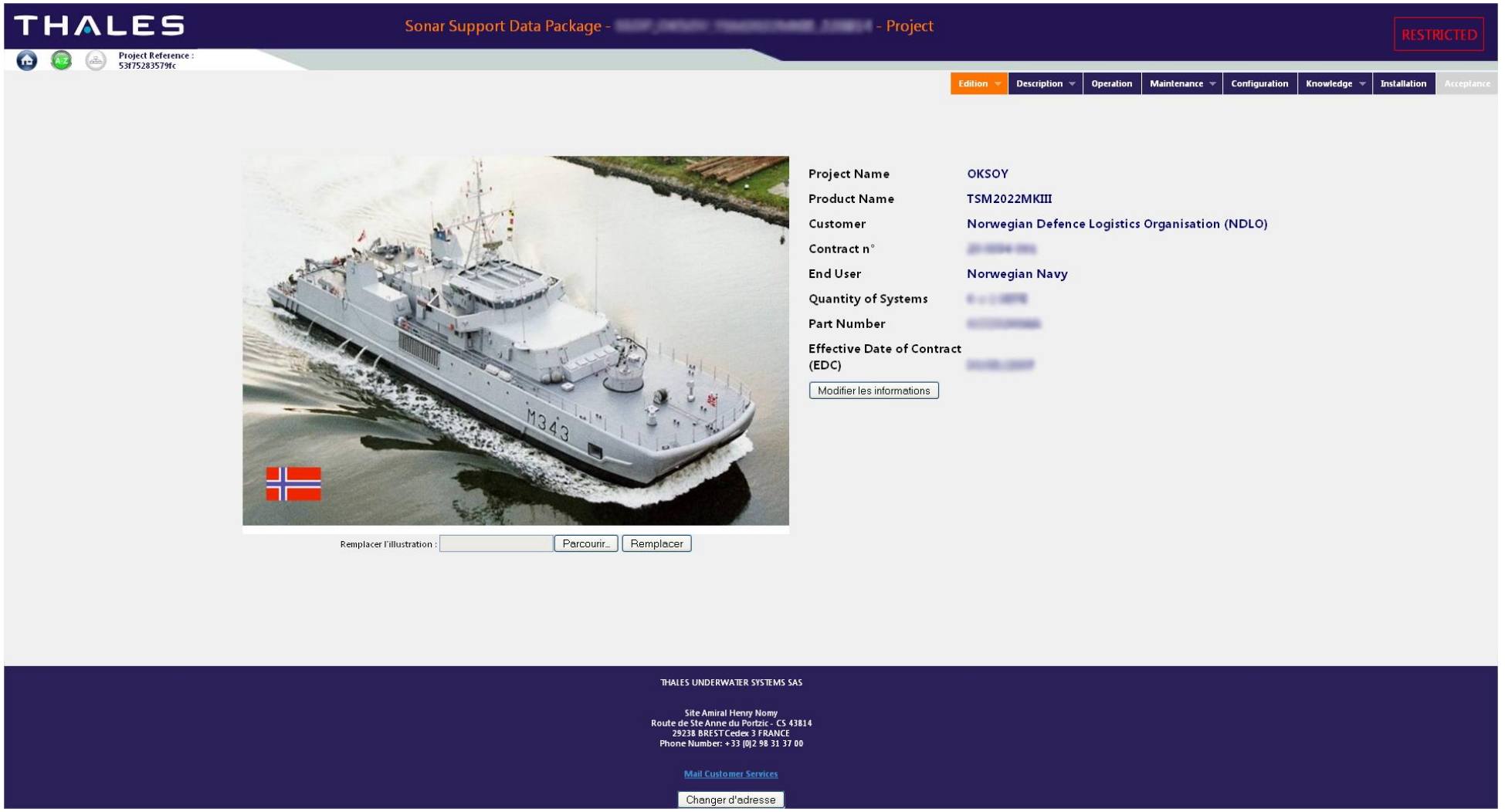
Subversion : Gestion de version permettant de garder un historique précis, complet et détaillé de toute modification effectuée dans le code.

PMAT : Suivi de tâches permettant de les organiser de façon chronologique

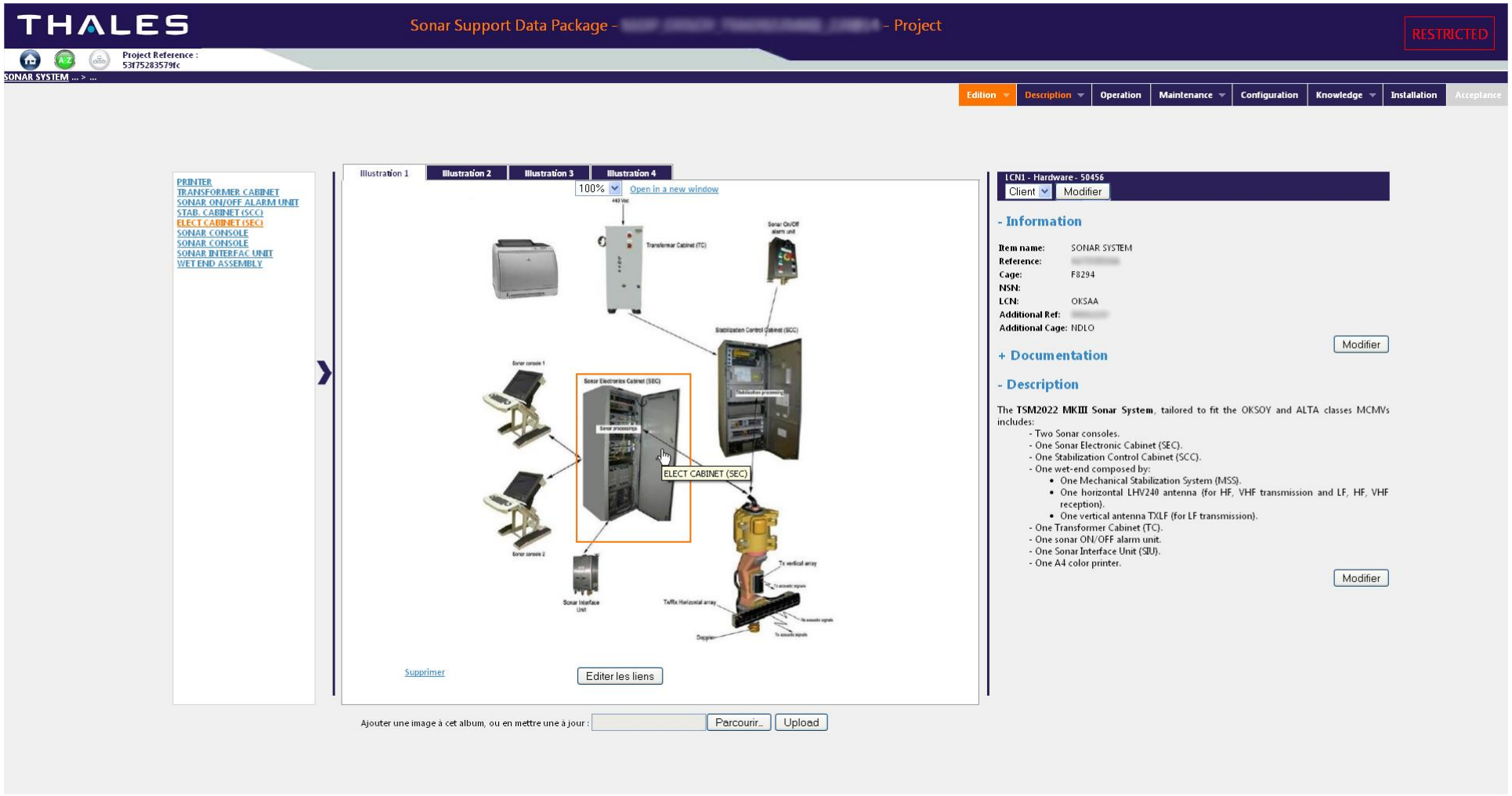
GIMP : Retouche, création d'images

Google Chrome - Internet Explorer - Mozilla Firefox

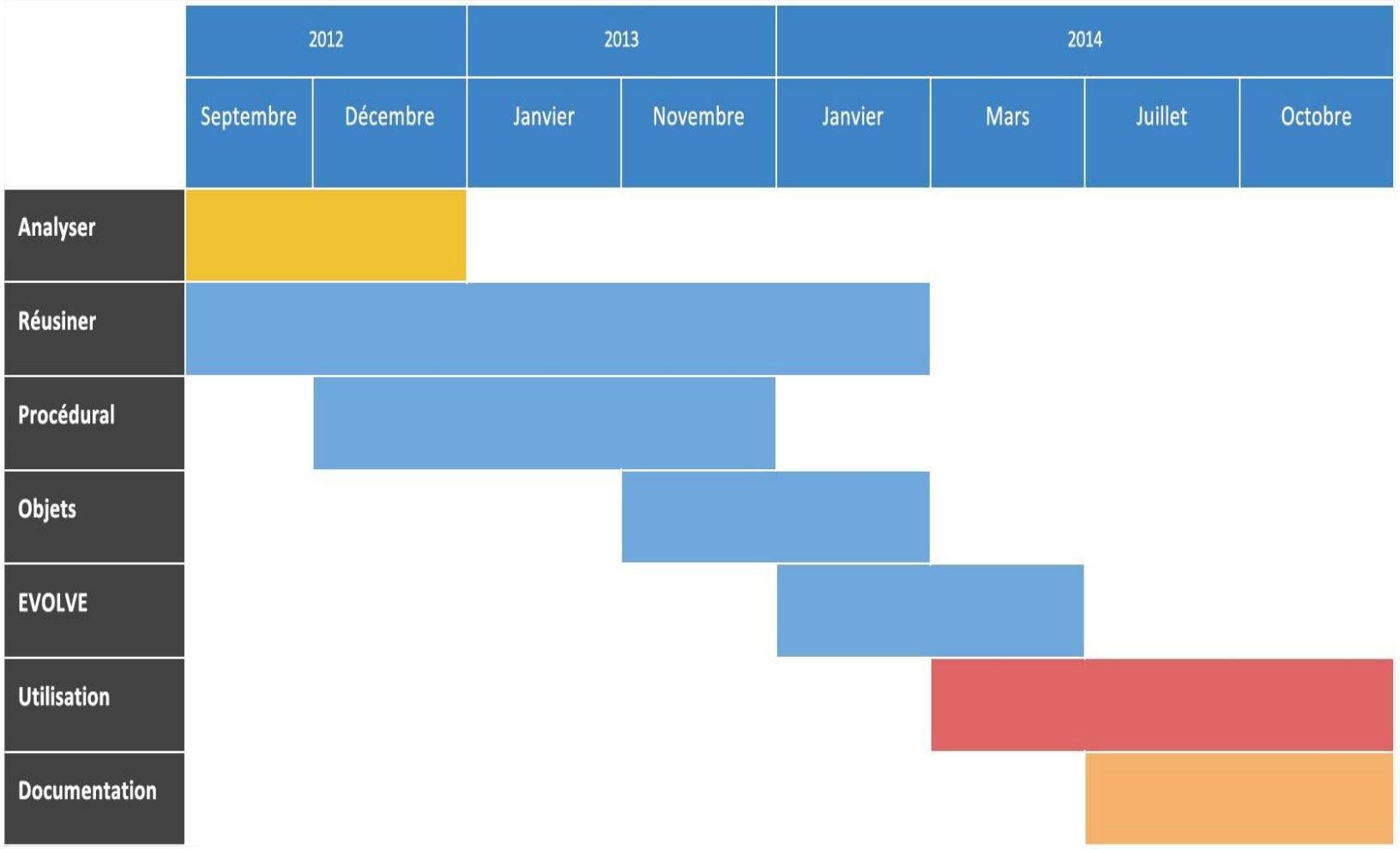
Annexe 2 – SSDP Page principale



Annexe 3 – SSDP 1er Niveau d'une arborescence

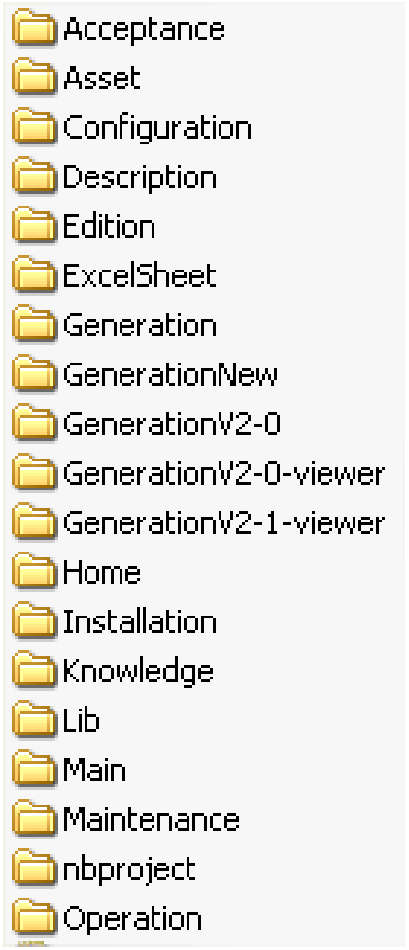


Annexe 4 – Diagramme de Gantt



Annexe 5 – Ancienne et Nouvelle Arborescence

Ancienne arborescence	Nouvelle Arborescence



- Trunk
  - Structure
    - Ajax
      - {Dossier}
        - Fichier(s)
      - *Shared*
    - Asset
      - {Dossier}
        - Css
        - Img
        - Js
      - *Shared*
    - Lib
      - Scripts PHP / HTML
    - Objects
      - {Dossier}
        - Fichier : classe
        - Fichier optionnel : interface
    - Request
      - Scripts PHP / HTML
    - View
      - {Dossier}
      - *Shared.*
    - *config.ini*
    - *config.php*
    - *connection.ini*
    - *url.php*
  - *index.php*
  - *phpinfo.php*

Annexe 6 – Avantages et Inconvénients Techniques

Avantages et inconvénients techniques

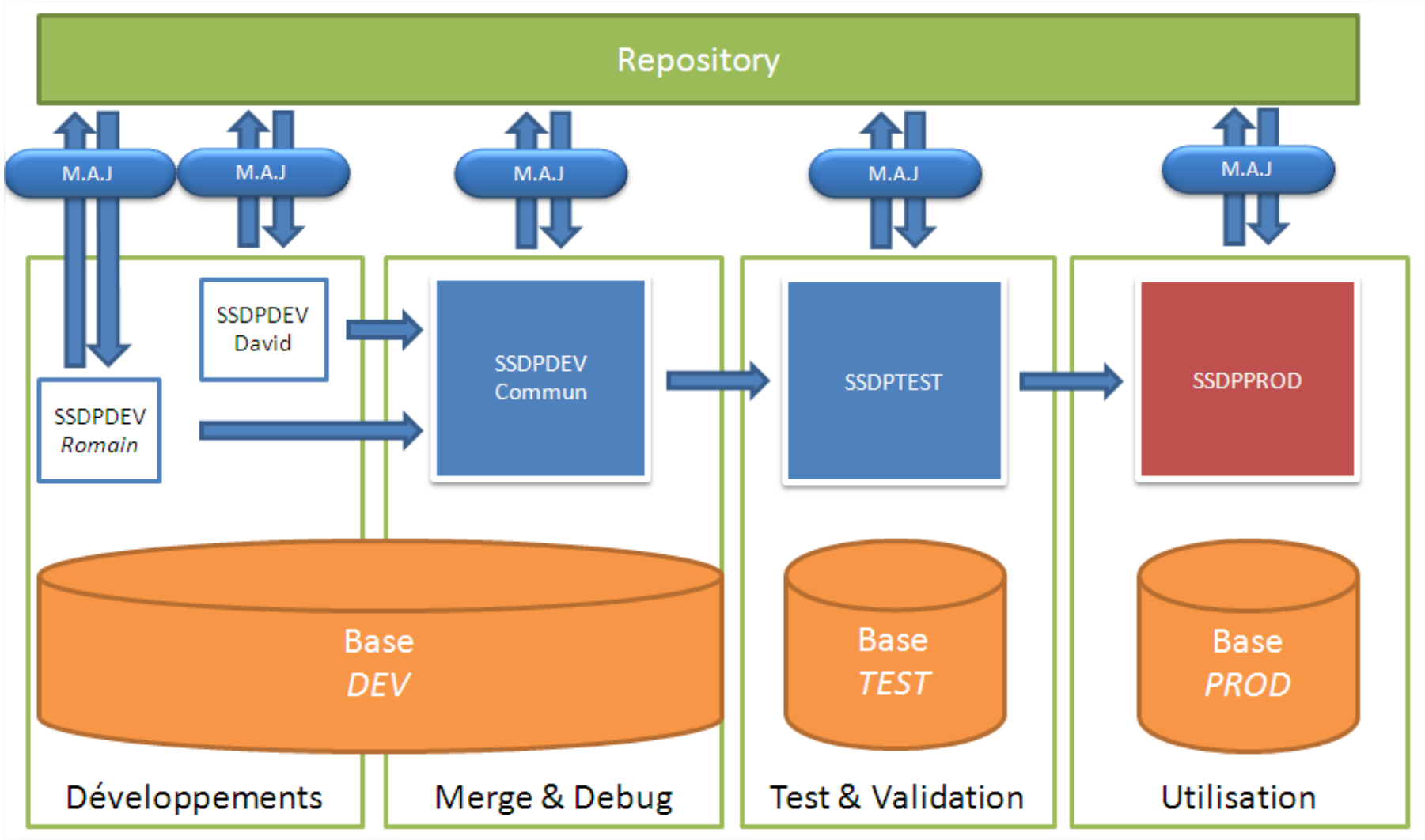
	Technique	
	Avantages	Inconvénients
Tout redévelopper	- **Possibilité de développement à plusieurs développeurs dès le premier jour** - Suppression totale du code « spaghetti » existant	- Perte de l'existant - Temps d'attente conséquent (mois voir années) pour débuter le développement de nouvelles fonctionnalités
Refactorer	- Maitrise totale du développeur de l'ancienne et de la nouvelle version de l'application - **Réutilisation du code existant**	- Effets de bords nombreux lors du refactoring - Suppression partielle du code « spaghetti » ou en tout cas progressive
Framework privé	- Le Framework créé est orienté métier - Choix du langage - Utilisation des dernières technologies (Design pattern, version des langages …) - **Choix des modules de Framework donc fonctionnement plus léger** - **Facilite l'intégration du projet dans le système informatique existant**	- « Do-it-Yourself » Limité à la connaissance des développeurs en matière de sécurité, structure et réalisations. - Pas de documentation technique existante - Pas de tutoriels - Pas de communauté d'entre aide
Framework open source	- **Utilisation des dernières technologies** (Design pattern, version des langages …) - Choix du langage - **Documentation existante** - **Communauté & entraide** - Développements solides et entraide pour tous les développeurs - Pas de "Do-it-Yourself"	- Pas orienté métier - **Règles « Fixes ». implicites. Le fonctionnement et l'architecture du Framework ne peut pas être modifié ou adapté au Système Informatique de l'entreprise** - Les failles et mises à jour du Framework dépendant d'une autre entité - Connaissances préalables obligatoires - Qualité - Pérennité - Modules intégrés pas forcément utile au projet

Annexe 7 – Avantages et Inconvénients pour la gestion du projet

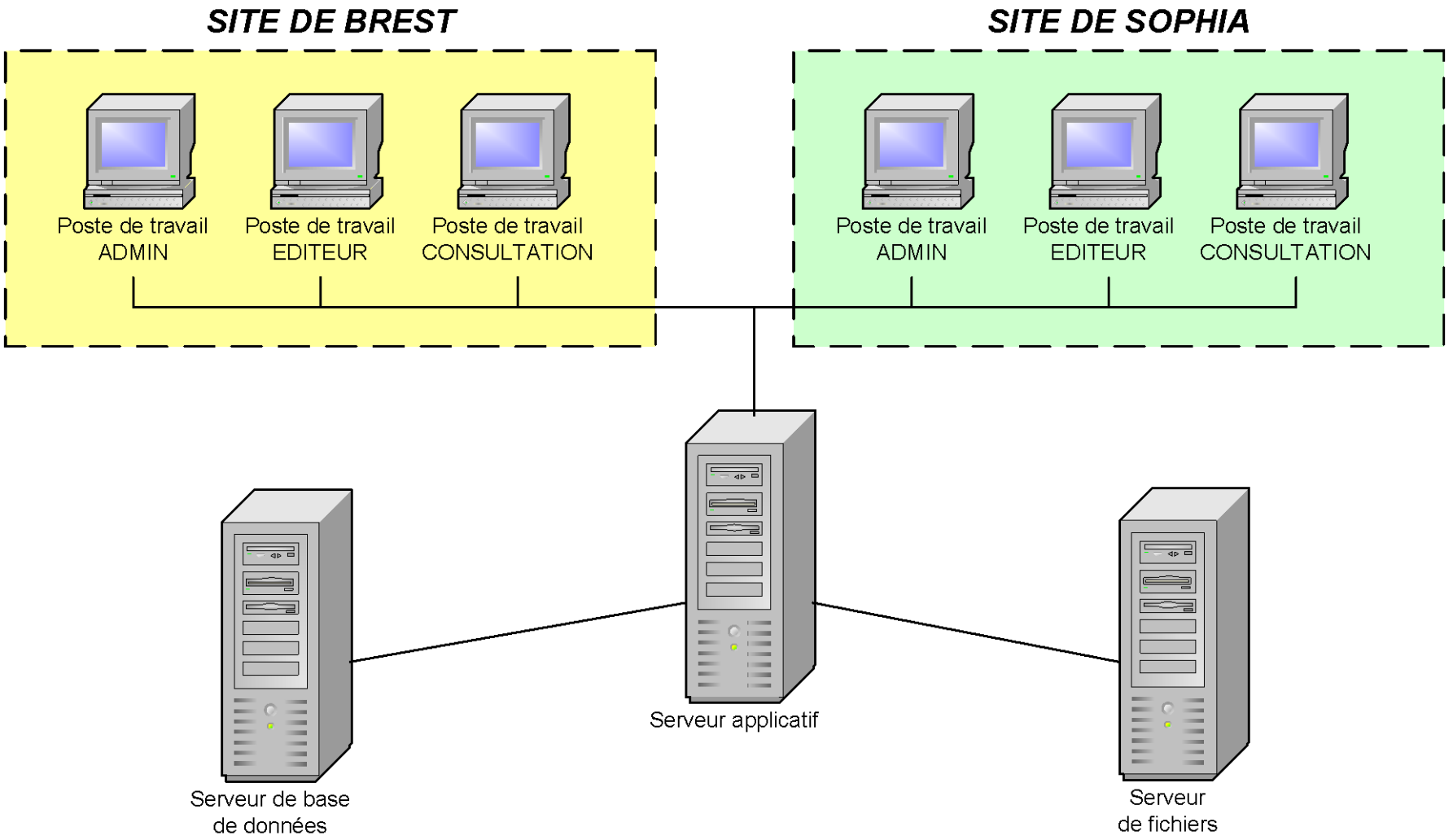
Avantages et inconvénients pour la gestion du projet

	Projet	
	Avantages	Inconvénients
Tout redévelopper	- Expérience acquise par tous les développeurs investis dans le projet	- Nécessite plusieurs mois de travail avant d'obtenir les premiers résultats. - **Stagnation de l'ancien projet** - Nécessite de nombreuses réunions de conception
Refactorer	- **Avancement immédiat, le projet n'est pas bloqué car les nouvelles fonctionnalités peuvent être développées** - Réutilisation des fonctionnalités existantes - Amélioration continue de l'application	- Effets concrets obtenus après plusieurs mois ou années de travail - Possibilités réduite de développement à plusieurs car les effets de bords seront nombreux
Framework privé	- Maitrise à 100% des lignes de code par les développeurs - Utilisation de l'expérience des employés concernant les problématiques existantes dans le but de les prévoir à l'avance	- Temps d'acquisition des connaissances par un nouvel embauché relativement long - La conception et le développement ne peuvent se faire qu'à un nombre limité de développeurs
Framework open source	- Possibilité de développement à plusieurs développeurs dès le premier jour - Embauche de développeurs opérationnels immédiatement car ils connaissent déjà le Framework	- Maitrise partielle du Framework

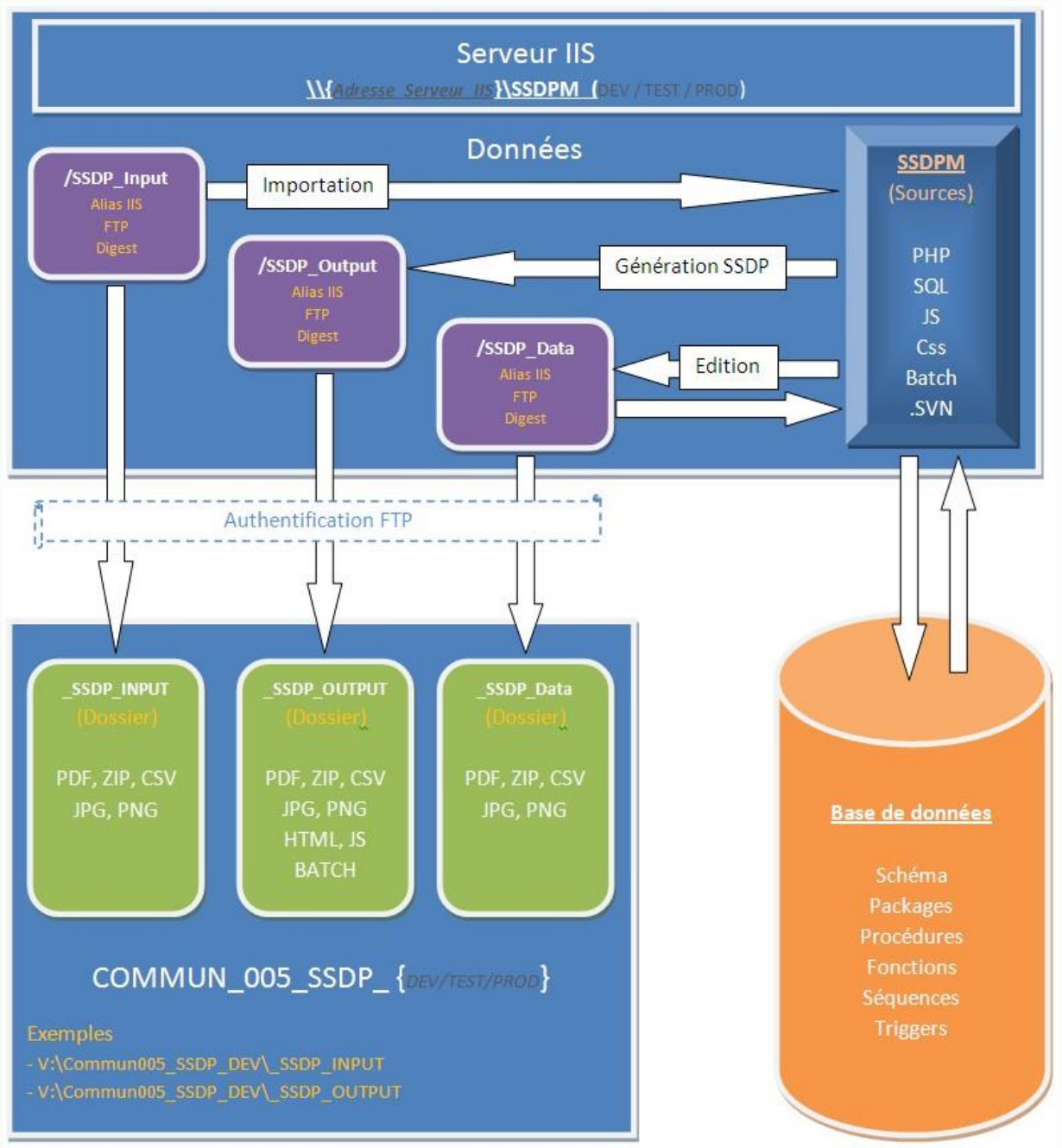
Annexe 8 – Versionning de l'application SSDPM



Annexe 9 – Schéma de fonctionnement Général de l'application SSDPM



Annexe 10 – Schéma de Fonctionnement de l'Application SSDPM



Annexe 11 – Fonctionnement des classes de l'application SSDPM