



Le développement offshore Agile

Livre Blanc

Janvier 2006





Table des matières

1.	Introduction.....	4
2.	L'intérêt de l'Offshore en Informatique.....	5
2.1	Un rapide historique.....	5
2.2	Les arguments majeurs de l'offshore.....	6
2.2.1	Le gain économique.....	6
2.2.2	La qualité.....	6
2.2.3	Les gains complémentaires.....	7
2.3	Les géographies de l'offshore.....	7
2.4	Comment aborder la question de l'offshore ?.....	8
2.5	Les contraintes de l'offshore.....	9
3.	Introduction aux méthodologies Agiles.....	10
3.1	Les origines de l'Agilité.....	10
3.2	Vive le changement !.....	12
3.3	La revalorisation du développeur.....	14
3.4	Agilité et CMMI.....	15
4.	Agilité et Offshore.....	16
4.1	Une réponse aux contraintes de l'offshore.....	16
4.2	Visibilité et réduction des risques.....	16
4.3	Possibilité de travailler sur des projets complexes.....	17
4.4	Une bonne intégration des équipes offshore.....	17
5.	L'AOSD.....	19
5.1	Introduction.....	19
5.2	Organisation des équipes.....	19
5.2.1	Intégrer l'équipe distante au projet.....	19
5.2.2	Le nouveau rôle de Coordinateur.....	21
5.2.3	Répartition des rôles.....	24
5.3	Gestion de projet.....	26
5.4	Pratiques de développement.....	29
5.5	Infrastructure.....	31
5.6	Outils de communication.....	33
5.7	Pilotage.....	35
6.	Comment démarrer un projet offshore Agile ?.....	37
6.1	Les étapes en amont du projet.....	38
6.2	Le choix du premier projet offshore.....	39
6.3	Les différents types de prestataires offshore.....	40
6.4	Le facteur humain.....	42
6.5	Les modes contractuels.....	42
7.	Retour d'expérience : un projet offshore dans le monde bancaire.....	44
7.1	Contexte.....	44
7.2	Mission de Pivolis-KPIT.....	44
7.3	Les apports de Pivolis-KPIT.....	45
7.4	Déroulement du projet.....	46
7.4.1	Dates clés.....	46
7.4.2	Organisation des équipes.....	46
7.4.3	Outils de communication.....	50
7.4.4	Infrastructure technique.....	50
7.4.5	Le processus de livraison.....	52
7.5	Problèmes rencontrés et solutions apportées.....	52
7.5.1	Gestion du turnover.....	52
7.5.2	Difficultés culturelles et aspects sociaux.....	53
7.5.3	Apprentissage de la connaissance fonctionnelle.....	54



1. Introduction

Toujours en perpétuelle évolution, le marché des services informatiques connaît depuis plusieurs années un phénomène nouveau et qui semble inexorable : la montée en puissance de l'« offshore », c'est-à-dire l'utilisation de compétences techniques situées dans des pays étrangers. L'intérêt est de disposer, de manière souple, de ressources de qualité, à coût moindre – les économies pouvant atteindre 40 à 50%.

Cependant, la mise en œuvre de l'offshore au sein des directions informatiques pose un certain nombre de questions et demande de faire face à de nouveaux défis. Une complexité évidente réside dans la capacité à faire travailler conjointement différents acteurs éloignés géographiquement, et de cultures différentes. Cet état de fait conduit naturellement à exacerber les difficultés traditionnelles rencontrées par les méthodologies classiques, par exemple, le travail

en offshore sur un cahier des charges imprécis semble souvent risqué.

Parallèlement à ce phénomène, de nouvelles méthodologies de plus en plus populaires voient le jour : il s'agit des méthodes dites « Agiles ». Celles-ci offrent plusieurs avantages qui s'avèrent précieux dans le cadre de l'offshore, comme pouvoir travailler sur des spécifications non figées, ou encore faciliter grandement la phase d'intégration des différents modules développés. Appliquées à l'offshore et accompagnées d'outils et méthodes de travail collaboratifs, ces méthodologies se montrent extrêmement efficaces. Elles permettent de réduire et de maîtriser très significativement les risques fréquemment cités pour l'offshore.

La réflexion sur l'adéquation forte entre Agile et Offshore commence d'ailleurs à être partagée par de nombreux experts. Le cabinet d'analyse et de recherche

Forrester™ a publié en septembre 2004 un document intéressant sur ce sujet¹.

Partant des particularités du développement offshore, ce Livre Blanc se propose de décrire les méthodologies Agiles et l'intérêt de leur utilisation dans le cadre de projets offshore. Il met en exergue un ensemble de recommandations issues à la fois de la littérature et de notre expérience pour mettre en œuvre avec succès une stratégie offshore. Il décrit ainsi la méthodologie AOSD (Agile Offshore Software Development) mise au point par Pivolis et KPIT Cummins, et fruit de plus de 4 années d'expérience dans ce domaine au sein de grandes entreprises. Il conclut par le témoignage concret d'un projet important réalisé par Pivolis et KPIT Cummins au sein d'un de leurs clients français.

1: <http://www.forrester.com/Research/Document/Excerpt/0,7211,34978,00.html>

2. L'intérêt de l'Offshore en Informatique

2.1 Un rapide historique

Le phénomène « Offshore » n'est pas, à proprement parler, nouveau dans le secteur informatique, et ce, y compris en France. Sur le plan macroéconomique, il s'inscrit dans le vaste mouvement de globalisation de l'économie et des délocalisations/relocalisations qui l'accompagnent. Il a en fait été popularisé par les éditeurs de logiciels américains à la recherche de ressources de développement informatique à moindre coût ; puis l'ensemble des grandes sociétés informatiques (constructeurs, intégrateurs) leur a emboîté le pas. Un coup d'accélérateur a été donné lorsque la pénurie de ressources informatiques se fit criante au tournant des années 2000, lors du passage à l'an 2000 puis avec l'avènement de la nouvelle économie et de la multitude de projets informatiques que celle-ci déclencha.

Dès cette époque, l'Inde s'est imposée comme un acteur de choix dans ce domaine, du fait tout d'abord du très haut niveau de formation de ses ingénieurs, et également de sa capacité à mettre à disposition des ressources anglophones à faible coût. Ces premières expériences ont joué un rôle clé car elles ont permis de juger de la grande qualité des informaticiens indiens. Elles ont également incité les pouvoirs publics indiens à y voir un axe de développement économique majeur. De fait, des investissements importants ont été consentis pour faire de l'Inde le leader mondial de l'informatique offshore en encourageant la création d'entreprises dans ce domaine, et l'obtention de certifications telles CMM, tout en déployant une infrastructure de télécommunications de haut niveau. Sur ce dernier point, il faut naturellement souligner le rôle majeur que joua le développement d'Internet, facilitant les communications et les échanges informatiques entre pays.

Tous ces facteurs permirent à l'Inde de faire véritablement exploser cette activité, principalement en relation avec les Etats-Unis. Si dans un premier temps, ce furent les intégrateurs qui restèrent les grands donneurs d'ordre, les clients finaux - tels General Electric - suivirent et signèrent en direct des contrats avec des sociétés de services indiennes, trouvant ainsi le moyen de réduire sensiblement leur facture informatique.

Depuis, cette activité s'est généralisée, amenant de nombreux pays à se positionner sur ce créneau et de nombreuses grandes entreprises à faire appel à de telles ressources. Depuis la fin 2004, le phénomène a réellement pris son envol en France, où plusieurs entreprises n'ont pas hésité à créer elles mêmes leur centre de développement offshore, pendant que d'autres confiaient leurs projets à des sociétés spécialisées.

En France comme ailleurs, les impératifs de maîtrise des coûts, de qualité, de flexibilité, d'internationalisation et de globalisation des offres d'une part, la démocratisation des communications haut-débit et des outils collaboratifs d'autre part, permettent de penser que le développement offshore, devrait s'imposer durablement.

2.2 Les arguments majeurs de l'offshore

Les deux facteurs déterminants qui ont fait de l'offshore un mouvement de fond sont la réduction des coûts et l'amélioration de la qualité.

C'est en effet sur cette dualité particulièrement séduisante aux yeux des Directions Informatiques, mais également des Directions Générales et des Directions Financières, qu'ont joué les acteurs de l'offshore, notamment en Inde. Cela a intelligemment permis à ces prestataires de ne pas jouer uniquement sur l'argument « coût », sans que cela ne soit pour autant un simple argument marketing : la qualité des compétences et des réalisations est au rendez-vous dans la réalité. C'est en effet avant tout le retour d'expérience très positif de grandes sociétés comme General Electric aux Etats-Unis ou Essilor en France qui ont fait la crédibilité et le succès de l'offshore.

2.2.1. Le gain économique

Concrètement, la promesse avérée de l'offshore informatique consiste à permettre d'économiser entre 30% et 50% du coût d'un projet en le délocalisant.

Mais d'où provient donc cette différence si importante ?

Naturellement, la première explication vient des coûts salariaux : il existe un rapport de 1 à 3 ou plus entre le salaire d'un informaticien offshore et celui d'un informaticien français. La différence de coût du jour / homme entre le prix offshore et le prix d'une SSII française atteint ainsi aisément 50%.

Cela dit, il existe d'autres facteurs d'économie liés à la délocalisation, les plus cités étant : le prix des m², l'équipement du poste de travail, ... et plus généralement tous les frais liés à la présence d'employés dans les locaux de l'entreprise.

Enfin, et par souci de transparence, il ne faut pas oublier les surcoûts induits par l'offshore : l'ajout de ressources de coordination, les voyages et les coûts de communication (en fait réduits du fait de nouvelles technologies).

2.2.2. La qualité

Sur le plan de la qualité, les promesses sont également à la hauteur des espérances dès lors que l'on s'adresse à des entreprises offshore certifiées. Ces sociétés connaissent parfaitement les réticences bien naturelles de leurs clients à confier leurs projets à des entreprises et surtout à des informaticiens situés à plusieurs milliers de kilomètres de leur lieu de travail. C'est la raison pour laquelle elles ont investi fortement dans les aspects qualité et les certifications correspondantes. Rassurer est l'enjeu capital de toute leur démarche commerciale !

C'est notamment pour cette raison que l'Inde est devenu le premier pays au monde en nombre d'entreprises certifiées CMM niveau 5 (un des plus hauts niveaux de cette certification américaine réservée aux services informatiques. A titre d'exemple, en France, il est déjà difficile de trouver des SSII certifiées CMM niveau 2 !). Naturellement le label ISO 9001 est également très répandu parmi ces sociétés, notamment les SSII offshore des pays d'Europe de l'Est qui démarrent tout juste leurs certifications CMM.

Le résultat de ces efforts de certification se concrétise au quotidien par un ensemble de processus rigoureux dans la gestion d'un projet informatique qui, allié aux compétences des informaticiens offshore, produit un résultat de haut niveau. Il n'est pas rare de trouver des directeurs informatiques ou des chefs de projet jugeant la qualité des produits développés offshore supérieure à celle des sociétés locales. C'est bien la promesse des entreprises offshore et force est de constater qu'elle très souvent tenue.

2.2.3. Les gains complémentaires

Enfin, il existe d'autres éléments qui achèvent de convaincre les entreprises de confier une partie de leurs développements informatiques à des partenaires offshore.

Tout d'abord, la flexibilité, du fait de la capacité des SSII offshore à monter et descendre en charge rapidement; l'offshore apporte ainsi un gain fort en adaptabilité.

D'autres facteurs de gain sont tout simplement liés au fait d'externaliser les développements informatiques. Cela permet à l'entreprise de se concentrer sur son cœur de métier, et à ses informaticiens de développer leur expertise sur ces aspects métier en termes informatiques (plutôt que sur la programmation et les aspects techniques, sous-traités à l'extérieur).

2.3. Les géographies de l'offshore

La montée en puissance de la délocalisation informatique a créé un marché mondial très attirant pour les pays disposant d'un système d'éducation technique de haut niveau, et capables de mettre à disposition des ingénieurs et développeurs à des salaires inférieurs aux salaires « occidentaux ».

De fait, concentré au démarrage sur l'Inde, ce marché est devenu très dynamique et a vu de nombreuses nations se poser comme candidates au partage de cette nouvelle manne économique :

- **L'Inde** reste une destination incontournable du fait de la qualité et de l'expérience de ses prestataires et de ses informaticiens : nombre d'entre eux ont acquis les certifications CMM et ISO, et le panel de technologies couvertes est très large. Le rôle très moteur du gouvernement constitue également un argument réel pour sécuriser des investissements de long terme avec ce pays. En revanche se pose le problème de la langue qui cantonne souvent la demande française à des projets internationaux. Quant au décalage horaire, il faut rappeler qu'il n'est que de 4h en moyenne, et est souvent utilisé avantageusement pour faire travailler les équipes locales et offshore en horaires décalés.
- **La Roumanie** est devenue en quelques années une destination de choix pour la France, car on y trouve des sociétés de très bon niveau, certifiées ISO9001, avec le grand avantage de mettre à disposition des personnes francophones.
- D'autres **pays d'Europe de l'Est : Russie, Tchéquie, Pologne, Ukraine...** se sont également positionnés sur ce marché. Ils tablent en premier lieu sur la qualité de la formation de leurs ingénieurs et sur la taille significative de leur réservoir humain. Cependant, l'ancienneté, la taille et la pérennité des SSII, ainsi que les problèmes de langue restent en général des handicaps réels.
- D'autres pays émergent aujourd'hui comme des outsiders sur le marché français, car disposant d'informaticiens francophones dans un fuseau horaire souvent proche de la France : **le Maghreb** (Maroc en tête), mais également **l'île Maurice, Madagascar** depuis peu, **le Vietnam**... Pour chacun de ces pays, il est cependant essentiel de qualifier la compétence réelle des ressources, le niveau de certification ou à défaut les méthodologies utilisées, et la taille des sociétés pour être sûr qu'elles disposent de ressources en nombre suffisant.
- Au plan international, **la Chine** mène depuis peu une politique très volontariste dans ce domaine, mais la barrière de la langue risque d'être cruciale à court et moyen termes.

2.4. Comment aborder la question de l'offshore ?

Prendre la décision de déplacer une partie de ses développements informatiques en offshore n'est pas anodin. Un certain nombre d'aspects doivent être étudiés en amont pour valider cette stratégie et s'assurer du succès de sa mise en œuvre.

Il est important de comprendre que les gains de l'offshore s'accroissent avec le temps : la rentabilité est clairement plus visible dans la durée, au fur et à mesure que les projets s'enchaînent. Cela est principalement lié au phénomène classique de courbe d'apprentissage qui s'applique particulièrement à l'offshore. En effet, apprendre à faire collaborer ensemble des équipes distantes, à travailler le cas échéant au quotidien dans une langue étrangère et avec des personnes d'une culture différente, cela prend du temps et s'amortit d'autant mieux que l'on s'y engage pour le moyen ou le long terme.

Ce constat appelle plusieurs recommandations :

■ Impliquer la Direction Générale.

Pour que le mouvement vers l'offshore se fasse de la manière la plus aisée possible, la Direction Générale doit nécessairement être alignée avec la Direction Informatique ; elle doit comprendre les enjeux, y adhérer et adopter une direction claire évitant ainsi un phénomène de flottement au sein des équipes concernées.

■ S'informer sur les implications concrètes de l'offshore.

Cela passe notamment par la connaissance de la destination offshore choisie : la gestion des différences culturelles représente à elle seule une réelle contrainte qu'il faut absolument aborder dès l'amont du projet. Les caractéristiques et la culture du pays sont des éléments souvent pertinents du fait de leur impact sur le mode de travail au quotidien. Comprendre la culture très « orientale » d'un pays tel que l'Inde, permet de beaucoup mieux comprendre le mode de communication notamment verbale des collaborateurs. Son appréhension pourra apporter un gain en qualité et en productivité dès le démarrage du projet.

Du point de vue économique, il faut également disposer d'un panorama des acteurs offshore présents sur le marché, et plus particulièrement des différents « business models » existants et de leurs implications sur la mise en œuvre d'un projet offshore. Travailler avec une société étrangère ayant une simple représentation commerciale en France, avec une SSII française disposant de centres de développement à l'étranger ou encore avec une société-pivot, cela implique des modes de travail différents ; il est nécessaire de bien les appréhender.

■ Etre attentif à la méthodologie mise en place.

La mise en œuvre et la gestion d'un projet offshore implique naturellement un ensemble de contraintes spécifiques à prendre en compte. L'organisation du projet, les processus de travail entre équipes distantes et les outils collaboratifs utilisés sont autant de points essentiels sur lesquels le prestataire doit savoir apporter des réponses adaptées.

Pour toutes ces raisons, les entreprises qui jugent indispensable d'aborder l'offshore, le font de deux manières bien distinctes :

- Soit **de manière très structurée**, en s'appuyant sur une démarche en trois phases amont :
 - Une étude stratégique préalable (« Go/No go ») dont l'objet principal est de décider d'aller ou non vers l'offshore. Comme dit précédemment, cela est d'autant plus important que cette décision devrait engager l'entreprise sur le moyen voire le long terme.
 - Un appel d'offres, avec pour objectif de choisir son partenaire offshore.
 - Une étape de préparation, pour mettre en place tous les éléments nécessaires au bon déroulement et au succès du projet.

- Soit **de manière pragmatique**, en privilégiant l'expérience concrète. Ces entreprises identifieront ainsi un projet pilote et le lanceront dans la foulée avec des objectifs clairs. Sur un tel projet, il ne faudra pas en effet privilégier le gain économique mais bien le retour d'expérience sur les contraintes, les risques et les impacts internes qu'apporte l'offshore.

Nous reviendrons en détail sur ces éléments au sein du chapitre « Comment démarrer un projet offshore Agile » ?

2.5. Les contraintes de l'offshore

Les avantages cités ne doivent pas faire oublier qu'un projet offshore comporte des risques et des contraintes qu'il va falloir traiter. Voici une série de questions qu'il faut se poser et auxquelles ce Livre Blanc apporte des réponses :

- Comment réussir à communiquer facilement et efficacement ?
 - Deux contraintes doivent être surmontées : celle de la distance et celle de la langue et de la culture.

- Comment ne pas perdre la visibilité sur l'état du développement ?
 - Il faut éviter l'« effet tunnel²» qui peut facilement se manifester en raison de la distance.

- Comment éviter le problème de l'intégration du code développé en offshore ?
 - L'intégration est coûteuse et se contrôle très mal. Comment dans ce cas intégrer facilement le code développé à distance ?

- Comment contrôler la sécurité du projet en offshore ?

- Comment impliquer les équipes offshore dans le projet ?

- Comment prévenir les querelles entre « offshore » et « on-site » ?

- Comment éviter les dérapages budgétaires ? Quel mode contractuel choisir ?

² : « Effet de tunnel » ou « effet tunnel » : lors du développement d'un logiciel, les futurs utilisateurs expriment leurs besoins, puis voient six mois ou un an plus tard le logiciel terminé. Entre ces deux étapes, ils n'ont rien vu, ils n'ont été informés de rien, et bien évidemment, le résultat est souvent très loin de leurs désirs ou de leurs besoins. A ne pas confondre avec l'« effet tunnel » en physique quantique !

3.

Introduction aux méthodes Agiles

3.1 Les origines de l'Agilité

Depuis quelques années arrivent en France les méthodologies dites Agiles [Référence1]. Elles portent des noms évocateurs tels que eXtreme Programming (XP), Scrum, Adaptive Software Development (ASD), Crystal, Feature-Driven Development (FDD), etc. ; on y retrouve également certains modes de travail désormais répandus au sein des sphères Open Source reconnues (Apache, Codehaus, etc.).

L'histoire prend sa source dans les années 1990, mais c'est en 2001 qu'un groupe de « méthodologistes pratiquants » se rencontrent durant un atelier de travail et définissent ce qui caractérise leurs pratiques méthodologiques. Ils créent le mot « Agile Methodology » et en listent les concepts fondateurs [2, 3, 4] :

« Nous sommes en train de découvrir des meilleures façons de développer des logiciels en pratiquant et en aidant les autres à le faire. A travers notre travail nous avons appris à apprécier les valeurs suivantes :

- **Les individus et les interactions** plutôt que les processus et les outils
- **Un logiciel qui fonctionne** plutôt qu'une documentation complète
- **La collaboration avec le client** plutôt que de passer du temps à (re)négocier des contrats
- **Réagir au changement** plutôt que de suivre un plan tout tracé

Ceci veut dire, que même si les éléments à droite ci-dessus ont de la valeur, nous apprécions plus encore les éléments à gauche. » (Traduction de l'anglais).

Ces méthodologies Agiles se définissent par opposition aux méthodologies dites « traditionnelles » (Jim Highsmith les appelle « Méthodologies Monumentales » [5]), plus orientées vers la prévision et la planification rigoureuses dès les phases amont du développement. L'archétype de ce type de méthodologie est le modèle du cycle en V [6] (lui-même une amélioration du modèle « en cascade ») dans lequel les différentes phases du développement s'enchaînent les unes après les autres (analyse des besoins, spécifications et conception, développement, intégration, tests)³. A l'opposé les méthodologies Agiles sont itératives : l'ensemble du cycle de développement est réalisé pour chaque fonctionnalité à développer.

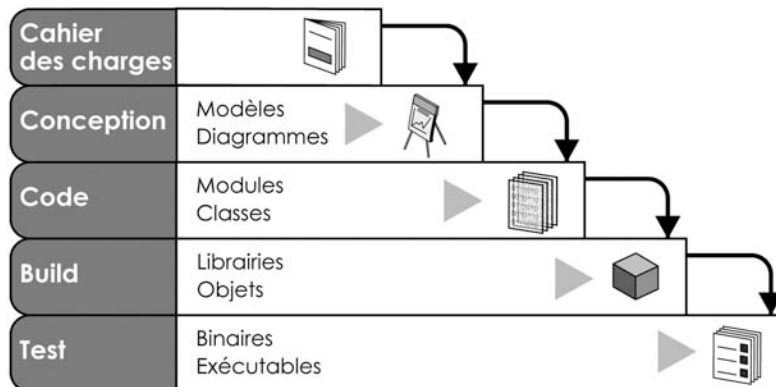
³: Il y a plusieurs variantes du modèle « en cascade » qui existent et certaines sont même itératives mais le concept qui est resté dans l'imaginaire collectif est l'enchaînement linéaire des phases. Lire « Iterative and Incremental Development: A Brief History » (<http://www2.umassd.edu/SWPI/xp/articles/r6047.pdf>) pour plus de détails.

Comme le montre la figure 1, les **trois différences fondamentales entre un cycle « en cascade » et un cycle itératif** sont :

- **Le périmètre et la profondeur d'analyse** : le cycle itératif travaille sur des fonctionnalités à développer dans le cadre d'un périmètre fonctionnel limité. Il s'attache à finir complètement une fonctionnalité avant de passer à la suivante. Un cycle « en cascade » s'intéresse à finir complètement une partie du cycle de développement avant de passer au suivant.
- **Les livrables** : le cycle itératif livre des fonctionnalités terminées (et utilisables) alors que le cycle « en cascade » livre des phases complètement terminées (mais non utilisables).
- **Les interactions** : le cycle itératif fait travailler les différentes équipes (conception, développement, intégration, tests) entre elles alors que le cycle « en cascade » les fait intervenir les unes après les autres.

CYCLE " EN CASCADE "

Périmètre large - Orienté Phase - Travail séquentiel



CYCLE ITERATIF

Travail en profondeur - Orienté fonctionnalités - Collaboration

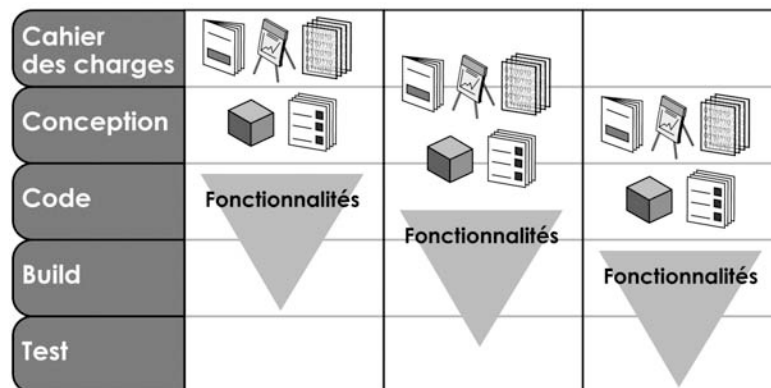


Figure 1 : Cycle « en cascade » vs cycle itératif*

*: Diagramme inspiré de l'article «The Agile Difference for SCM» par Brad Appleton, Robert Cowham et Steve Berczuk (<http://www.cmcrossroads.com/ubbthreads/showflat.php?Cat=&Number=34375>).

3.2. Vive le changement !

Les méthodologies Agiles partent du principe que pour aller d'un point de départ à un point d'arrivée, la meilleure stratégie n'est pas forcément la ligne droite. En effet, au sein d'un projet informatique le point d'arrivée est quasi systématiquement mouvant et donc difficilement prédictible. Si le critère de réussite est d'arriver à obtenir ce qui est attendu tout en minimisant les coûts, le chemin le plus court est alors le zigzag ! (voir figure 2)

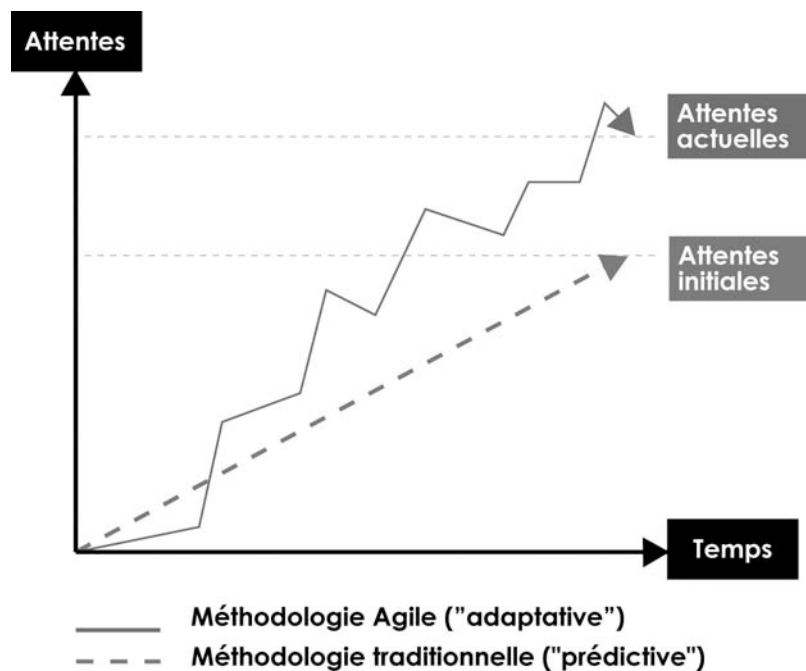


Figure 2 : Réponses des méthodologies au changement

Un apport majeur des méthodologies Agiles est de reconnaître que le changement est inéluctable pour diverses raisons dans un projet informatique :

- **Evolution des spécifications** : un changement des spécifications dû à une modification de l'environnement concurrentiel (le monde est de plus en plus changeant).
- **Mobilité des acteurs** : un changement au sein de l'équipe (les gens sont de plus en plus mobiles).
- **Evolution technologique** : les progrès technologiques s'emballent et les cycles de vie de certaines technologies sont de plus en plus courts.

La seule constante est que le changement arrive tout le temps ! Au lieu d'être réactives au changement, les méthodologies Agiles l'acceptent et proposent des méthodes pour s'en accommoder.

Les pratiques permettant d'accepter et de maîtriser le changement sont diverses :

- **Les itérations courtes**. Elles permettent d'avoir des retours fréquents provenant des clients/utilisateurs du système. Il est ainsi possible de corriger le tir si besoin est. Il faut noter que pour être d'une utilité maximale une itération doit couvrir un maximum d'étapes (conception, implémentation, test, intégration).

- **Une architecture/conception simple.** Il faut la garder simple et ne pas essayer de trop prévoir à l'avance des besoins hypothétiques futurs. Un bon principe est de ne concevoir que ce dont on a besoin pour la fonctionnalité en cours de développement. En effet, il n'est pas rare de constater que souvent les besoins qui arrivent au final ne sont pas ceux qui étaient prévus ou du moins pas exactement. Les méthodologies Agiles permettent donc de changer de direction plus rapidement le cas échéant.
- **Le « refactoring ».** Cette technique permet de modifier un élément de code tout en préservant sa fonctionnalité. Tout élément de code se détériore avec le temps (l'entropie du code est croissante). Le refactoring permet de lutter pour garder un système ouvert, flexible et capable de résister au changement (code bien découplé, avec des interfaces, etc). Sans lui, la « dette technique » [7] du projet ne fait que s'accroître et il est inéluctable qu'un jour il faille la payer d'une façon ou d'une autre...
- **L'automatisation des tests.** Les tests sont utiles en soi pour confirmer que l'application fonctionne. Dans le contexte du changement leur automatisation est importante. C'est à cette condition que les régressions peuvent être évitées. L'automatisation des tests est donc nécessaire et primordiale. C'est ce qui donne le courage de « refactorer » et d'accepter les changements requis. Sans tests automatisés, la prise en compte de changements utiles pour les clients/utilisateurs va être freinée et limitée du fait du risque inhérent.
- **L'intégration continue.** Afin d'être capable d'effectuer des itérations courtes, il est nécessaire d'automatiser et ainsi de pratiquer de manière continue l'intégration de tout le code développé, au sein même de ce code et avec les systèmes externes. Cette intégration continue passe par une construction automatique de l'application à partir des sources (build), ce qui permet entre autres d'exécuter les tests automatisés et ainsi de garantir que l'ensemble de l'application fonctionne à tout moment (et de corriger si ce n'est pas le cas !).

Toutes ces pratiques permettent ainsi de lisser le coût du changement comme le montre la figure 3 :

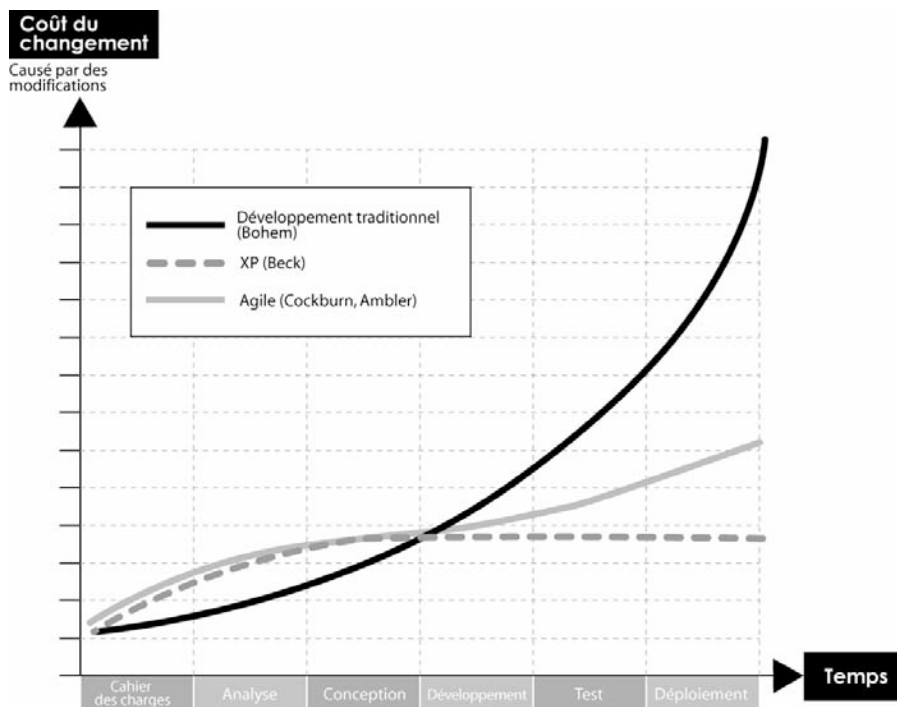


Figure 3 : Coût du changement par méthodologie*

Barry Boehm écrit en 1980 que le coût du changement croît d'un facteur dix à chaque phase [8]. Vingt ans plus tard, Kent Beck avance que cette exponentialité peut être aplatie sans sacrifier à la qualité [9]. Par la suite, Alistair Cockburn [10] et Scott Ambler [11] cassent le mythe selon lequel le coût du changement des méthodologies Agiles peut être complètement plate. Elle peut néanmoins être plus aplatie que celle des méthodologies traditionnelles et être quasiment linéaire après le point d'inflexion.

*: Diagramme inspiré de l'article «The Agile Difference for SCM» par Brad Appleton, Robert Cowham et Steve Berczuk (<http://www.cmcrossroads.com/ubbthreads/showflat.php?Cat=&Number=34375>).

3.3. La revalorisation du développeur

Un autre apport majeur des méthodologies Agiles est d'intégrer le développeur comme la clé de voûte d'un projet informatique. Il est l'un des acteurs principaux avec l'utilisateur et non pas une entité que l'on peut cloner, qui fait du travail répétitif et que l'on peut remplacer à discrétion. L'idée que le développeur devient moins indispensable face à des outils de plus en plus évolués vole en éclats. Un développeur n'est pas un codeur ! Un des grands principes des méthodologies Agiles est que le développeur doit être tout à la fois concepteur, implémenteur, testeur et intégrateur. L'intérêt pour le projet est évident : une adhésion forte des développeurs sur le projet engendre une meilleure productivité globale (ainsi qu'un turnover plus faible). D'autre part, l'équipe de développement connaissant mieux l'ensemble du projet, il est possible de réagir plus rapidement à tout changement requis ou autre événement externe non planifié.

Pour donner au développeur cette nouvelle dimension au sein du projet, les méthodologies Agiles préconisent notamment de :

- Favoriser le partage des connaissances par l'échange de personne à personne. Au lieu de multiplier la documentation (quasiment toujours obsolète), on va favoriser le « **pair-programming** » (travail à deux sur un poste de développement [12]) ou le « **mentoring** » (les développeurs seniors aident fréquemment les juniors). Cela permet de diffuser rapidement les connaissances à toute l'équipe. L'intérêt n'est pas tant d'améliorer la qualité du logiciel développé mais plutôt l'échange des connaissances. Comment savoir que telle portion de code existe déjà ? Comment savoir que telle pratique est conseillée plutôt que telle autre ?
- Utiliser **des outils de partage de connaissance « légers »** comme un wiki (site web éditable qui crée des liens semi-automatiquement entre pages) ou encore des tests unitaires montrant comment utiliser une API.
- Partager la connaissance de l'état du projet par des « **standup meetings** », littéralement des « réunions debout », quotidiens. Ce sont de courts meetings (10 minutes) qui se tiennent debout dans lesquels chacun va pouvoir partager tout ce qui peut être pertinent pour le projet (ce qui marche, ce qui ne marche pas, le but de la journée, etc.). L'intérêt d'être debout est de forcer le meeting à être court !
- **Intégrer le client / les utilisateurs dans l'équipe projet** (ou bien un représentant habilité à décider pour le client/utilisateurs). Cette personne va pouvoir définir, expliquer les spécifications, faire partager les connaissances métier, suivre les progrès de l'équipe et globalement faire émerger une connaissance métier partagée sur le projet.

Enfin, les méthodologies Agiles sont très pragmatiques et s'attachent à ce que chaque action ait une valeur directe pour le développement du produit logiciel. Elles font la chasse aux actions pour lesquelles un retour sur investissement ne peut être évalué rapidement.

3.4. Agilité et CMMI

Quand on parle d'offshore il n'est pas rare d'entendre parler de certification CMM (Capability Maturity Model) [13]. Par exemple toutes les SSII indiennes importantes sont certifiées CMM 4 ou 5 (les différents niveaux de certification, de 1 à 5 sont détaillés sur la figure 4). La dernière version du modèle s'appelle CMMI (Capability Maturity Model Integration).

Pour beaucoup CMMI ne rime pas avec Agilité. En fait, CMMI définit des processus à suivre en terme de traçabilité, répétabilité, efficacité, etc. CMMI dit ce qu'il **faut** faire mais ne dit pas **comment** le faire. Les méthodologies Agiles définissent des pratiques assez précises sur le comment faire ; elles sont un modèle d'implémentation. En ce sens, elles ne sont pas du tout incompatibles avec CMMI. Il est même possible d'effectuer la corrélation entre les pratiques Agiles et les 18 secteurs clés repartis sur les 5 niveaux de maturité du modèle CMMI (voir figure 4) [14, 15]. Par exemple, on s'aperçoit que XP répond assez bien à CMMI niveau 3 et même à certains secteurs clés des niveaux 4 et 5. En revanche, une différence principale est que CMMI s'attache à définir des processus qui doivent être appliqués à l'ensemble de la société alors que les méthodologies Agiles définissent des pratiques à utiliser au niveau du ou des projets informatiques. En fait, il est tout à fait possible de partir des pratiques Agiles et de les compléter pour atteindre le niveau CMMI que l'on souhaite. CMMI et Agile sont donc complémentaires.

Niveau CMMI	Objectifs principaux	Secteurs clés
5. Optimisé	Maîtrise du changement	Prévention des défauts
		Gestion des changements technologiques
		Gestion des changements du processus
4. Maîtrisé	Mesure et prévision	Gestion quantitative de processus
		Gestion de la qualité logicielle
3. Défini	Standardisation et institutionnalisation	Focalisation organisationnelle sur les processus
		Définition du processus de l'organisation
		Programme de formation
		Gestion logicielle intégrée
		Ingénierie de produits logiciels
		Coordination intergroupes
2. Reproductible	Structuration et discipline	Revue par les pairs
		Gestion des exigences
		Planification du projet
		Suivi et supervision du projet
		Gestion de la sous-traitance
		Assurance-qualité
1. Initial	Culte du héros	Gestion de configuration

Figure 4 : Les différents niveaux de certification CMMI et les secteurs clés associés

4. Agilité et offshore

4.1. Une réponse aux contraintes de l'offshore

Quel est l'intérêt des méthodologies Agiles pour le développement d'un projet en offshore ? Nous avons vu au chapitre 2 que l'offshore posait un certain nombre de contraintes. Les méthodologies Agiles vont notamment permettre d'y faire face et de répondre aux questions suivantes :

- Comment conserver la visibilité sur l'état du développement ? L'objectif étant de prévenir et d'éviter l'« effet tunnel » pouvant aisément survenir du fait de la distance.
- Comment optimiser l'intégration du code développé en offshore ? L'intégration est coûteuse et se contrôle mal, comment dans ce cas intégrer facilement le code développé à distance ?
- Comment favoriser l'implication des équipes offshore dans le projet ?
- Comment maîtriser les tensions entre les équipes offshore et on-site ?
- Comment prévoir et éviter les dérapages budgétaires ?

4.2. Visibilité et réduction des risques

Par définition même, le développement offshore se fait... à distance ! Or qui dit distance dit risque accru de divergence par rapport à l'objectif attendu. Il est donc impératif de trouver des pratiques qui réduisent ces divergences et l'« effet tunnel ». Les méthodologies Agiles sont de bons candidats car leur premier avantage est de fournir une visibilité à tous les niveaux :

- **Gestion de projet** (où en est-on ?) : les pratiques d'itérations courtes et d'intégration continue permettent cette transparence.
- **Réponse aux attentes** (cela correspond-il aux besoins des utilisateurs ?) : les itérations courtes et le fait de travailler fonctionnalité par fonctionnalité permettent de montrer rapidement aux utilisateurs des fonctionnalités qui marchent.
- **Management et productivité** (mon équipe est-elle motivée et adhère-t-elle au projet ?) : l'attention portée aux développeurs favorise cette motivation et favorise leur pleine participation au projet.

4.3. Possibilité de travailler sur des projets complexes

Plus précisément, les **itérations courtes** permettent d'avoir un retour rapide sur site du développement réalisé à distance. Cela permet aussi d'être capable de changer rapidement de direction en cas d'évolution des spécifications par exemple. Le **build continu** permet d'intégrer en continu le code développé par les équipes, qu'elles se trouvent offshore ou bien on-site. On peut même aller plus loin en intégrant des règles de vérification de qualité de code dans le build afin de s'assurer que les principes d'architecture et les meilleures pratiques sont respectés à tout moment. Encore plus intéressant, **l'exécution automatisée des tests unitaires et fonctionnels** permet de garantir (toujours en continu) que le code fonctionne en toutes circonstances et qu'il implémente effectivement la spécification fonctionnelle de la maîtrise d'ouvrage. Les tests fonctionnels automatisés agissent comme une spécification « exécutable ». En effet une spécification est toujours sujette à interprétation (cela est d'autant plus vrai dans une langue et une culture différentes) alors que des tests fonctionnels sont beaucoup plus précis et peuvent s'exécuter. On peut dire que les tests fonctionnels jouent le rôle de représentant en offshore des utilisateurs, comblant ainsi pour une large mesure le manque de présence locale continue des utilisateurs.

L'utilisation des méthodologies Agiles sur un projet offshore permet donc également de réaliser des projets complexes, aux spécifications mouvantes, qui concernent le cœur des systèmes d'information.

4.4. Une bonne intégration des équipes offshore

Un des points critiques d'un projet offshore est l'attention particulière portée au développeur. Dans le cadre de la gestion d'équipes offshore, les deux points suivants doivent impérativement être pris en compte.

■ **L'adhésion et le sentiment de participation au projet de l'équipe distante.**

Si la communication n'est pas parfaite, si les directions données et les informations circulent mal, l'adage « loin des yeux, loin du cœur » se matérialise rapidement. L'équipe distante en sera démotivée et aura du mal à s'identifier au projet.

■ **Le transfert des connaissances fonctionnelles.**

C'est un sujet difficile car par définition, la connaissance fonctionnelle se trouve quasiment tout le temps sur site. La transférer est un gage de qualité du code livré mais aussi de productivité accrue, car une équipe connaissant bien le domaine fonctionnel aura besoin de moins d'allers-retours avec les équipes sur site.

Le partage fort des connaissances ainsi que l'effort continu pour transformer le « codeur » en « développeur accompli » connaissant bien les aspects fonctionnels, participant à la définition des itérations et se souciant de la qualité du code livré en effectuant l'intégration du code permet sur le moyen et le long termes de construire une équipe offshore la plus autonome possible.

En pratique, il y a deux facteurs principaux de succès dans l'application d'une méthodologie Agile pour le développement offshore :

- **Disposer de coordinateurs offshore** permettant d'assurer cette communication essentielle,
- **Disposer des compétences** nécessaires pour identifier les outils indispensables à l'application des pratiques Agiles, et évangéliser les équipes sur le sujet.

La méthodologie AOSD préconisée par Pivolis offre des réponses à ces questions comme nous le verrons dans le chapitre suivant.

S'il ne fallait retenir que 3 points sur l'intérêt de l'Agilité pour le développement Offshore :



- l'Agilité diminue le risque projet dû à l'éloignement
- l'Agilité permet de réaliser des projets complexes en offshore
- l'Agilité permet d'augmenter la productivité en soignant la motivation des équipes offshore (combattant ainsi le syndrome « loin des yeux loin du cœur »)

5.

L'AOSD

5.1. Introduction

L'AOSD (Agile Offshore Software Development) est l'adaptation faite par Pivolis et KPIT des méthodologies Agiles au développement offshore. L'AOSD ajoute des pratiques nécessaires pour le travail à distance et définit concrètement comment appliquer les pratiques Agiles (quels outils, comment les utiliser).

L'AOSD associe de réels outils aux méthodologies Agiles. Ces outils étant bien évidemment spécifiques aux technologies utilisées, l'AOSD se décline par grande famille technologique : AOSD Java, AOSD .Net, AOSD SAP, AOSD WebMethods, etc. La plupart des outils utilisés proviennent du monde open source, qui, par définition a dû les concevoir et les créer afin de travailler en environnement hyper-distribué (i.e. chaque individu représentant un lieu physique de travail).

Cependant certaines technologies se prêtent mieux aux méthodologies Agiles que d'autres dans le sens où il existe déjà des outils appropriés. C'est le cas des développements à façon dans des technologies telles J2EE ou .Net. Mais ce qui est intéressant c'est de toujours garder à l'esprit les concepts Agiles qui peuvent, eux, s'appliquer quelle que soit la technologie (y compris sur des progiciels), de manière plus ou moins automatisée. Cela fixe un objectif d'amélioration permettant d'accroître la qualité et la productivité en continu, durant l'exécution du projet.

5.2. Organisation des équipes

5.2.1. Intégrer l'équipe distante au projet

L'organisation des équipes est cruciale car leur bien-être et leur efficacité en dépendent pour beaucoup. Afin d'assurer la réussite du projet, il est nécessaire de créer une organisation qui favorise la participation active de l'équipe offshore, et ceci, à tous les niveaux. L'écueil, bien souvent rencontré et à éviter absolument, est de considérer les équipes offshore comme de simples exécutants à qui il n'est pas nécessaire de fournir l'ensemble des informations projets (cela pourrait même les gêner dans leur travail car ils seraient « distraits » par ces informations, poseraient des questions, etc. !). Les conséquences en seraient dramatiques pour le projet :

- départ des membres les plus expérimentés car le spectre du projet ne serait pas assez large pour eux,
- équipes ne pouvant prendre des décisions de manière autonome et donc nécessitant de se référer très souvent au site central,
- équipes offshore vues comme faibles ou non-compétentes par les équipes on-site,
- etc...

Tous ces points vont tous dans le sens d'une productivité décroissante. Les quelques points de productivité difficilement obtenus en améliorant les outils et processus seraient complètement anéantis par le turnover et la mauvaise ambiance.

Le but est donc bien de travailler en permanence à mettre l'équipe offshore sur un pied d'égalité avec les équipes on-site. Egalité en termes d'information, de management et de communication. Mais attention, égalité ne veut pas dire appliquer aveuglement les pratiques qui fonctionnent en France. Une équipe indienne, par exemple, ne se gère pas comme une équipe française.

Le premier point est de ne pas faire de micro-management, c'est-à-dire qu'il ne faut pas essayer de diriger directement à distance les développeurs, testeurs, ... Il faut impérativement un chef de projet local (voir figure 5).



Règle 1 : Eviter le micro-management à distance

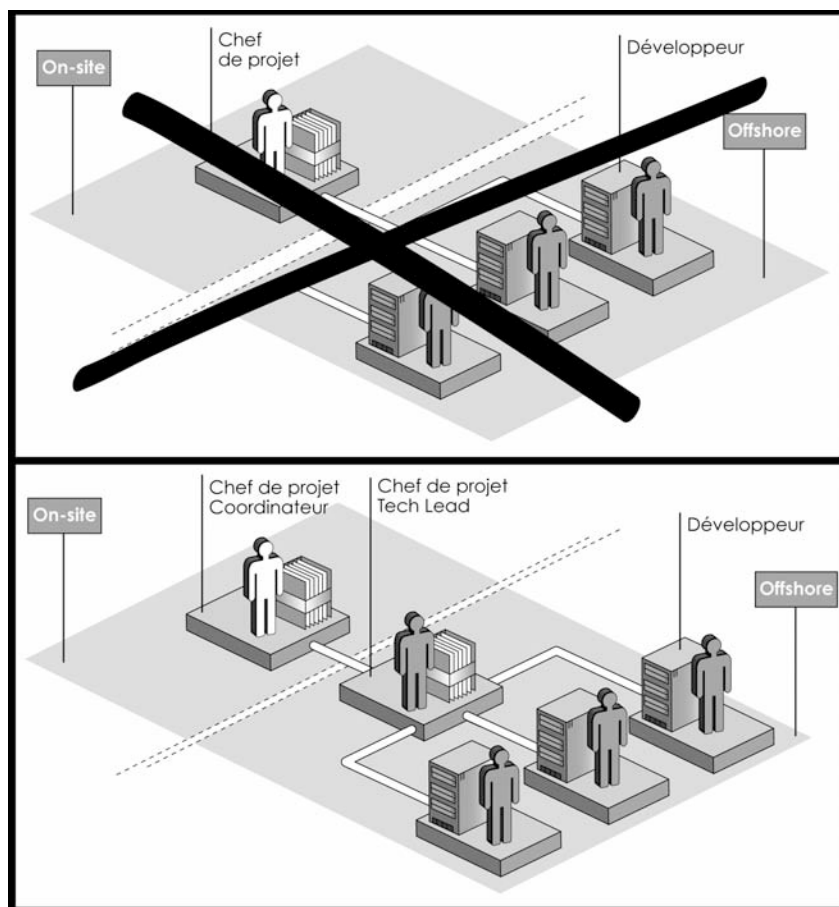


Figure 5 : Gestion à distance : éviter le micro-management

Le micro-management est néfaste car il est impossible d'appréhender à distance les problématiques locales : gestion des personnes, communications informelles à la « pause café », visibilité sur les événements locaux (vacances, fêtes, événements de la société). Le plus important est que l'équipe distante n'a en général pas le sentiment d'avoir une mission globale à exécuter. Au contraire, elle peut avoir l'impression que des tâches lui sont affectées au coup par coup sans pour autant bénéficier d'une vision constructrice qui lui permette de s'épanouir et progresser. De manière plus pragmatique, cela nécessite des communications extrêmement fréquentes qui coûtent en temps et en énergie (il est, encore aujourd'hui, plus difficile de communiquer à distance que de visu). Pour traiter ce point, la solution est de mettre en place un chef de projet ou « Tech Lead » local pour l'équipe offshore. La communication principale on-site / offshore passera par lui. Bien évidemment, il faut également permettre aux membres de l'équipe offshore de communiquer directement avec des personnes on-site, mais ce sera sous le contrôle de ce Tech Lead. Cette communication directe est nécessaire car elle évite le goulet d'étranglement et permet également :

- aux équipes offshore de communiquer avec le client, ce qui rendra les échanges plus efficaces (ex : précisions sur les spécifications, compréhension des besoins nécessaire à l'analyste/développeur, ...). De plus, cela est valorisant et gratifiant pour les membres de l'équipe et leur permet de développer des compétences internationales ainsi que de mieux connaître la culture française.
- aux équipes on-site de mieux comprendre les attentes des équipes offshore et de se faire leur propre avis sur les qualités et les compétences des individus.

5.2.2. Le nouveau rôle de Coordinateur

Une organisation typique fait intervenir 3 catégories de personnes (figures 6 et 7) :

- les chefs de projet et éventuellement les développeurs on-site,
- le(s) chef(s) de projet et développeurs offshore,
- le(s) coordinateur(s) offshore présents on-site.

On pourrait se demander à quoi servent ce(s) coordinateur(s) offshore (dirigés, s'il y en a plusieurs dans le cas de projets importants, par un « Responsable livraison offshore » qui n'est autre qu'un super-coordonateur). Les chefs de projet offshore ne pourraient ils pas en fait se retrouver directement sous la responsabilité du directeur de projet ? L'expérience nous montre que dans ce cas les deux écueils suivants sont souvent au rendez-vous :

- Le directeur de projet n'a en général pas suffisamment de temps à consacrer aux équipes distantes pour expliquer chaque décision, pour donner le contexte projet, pour répondre aux nombreuses questions et pour s'assurer que les questions posées entre équipes offshore et équipes on-site sont prises en compte dans des délais courts.
- Les attentes du directeur de projet et celles des chefs de projet offshore tendent à diverger au cours du temps. Cela cause rapidement des frustrations des deux côtés : mécontentement du côté on-site avec le sentiment que l'équipe offshore n'est pas productive ; sentiment d'injustice du côté offshore qui peut générer une baisse de productivité et un turnover important.

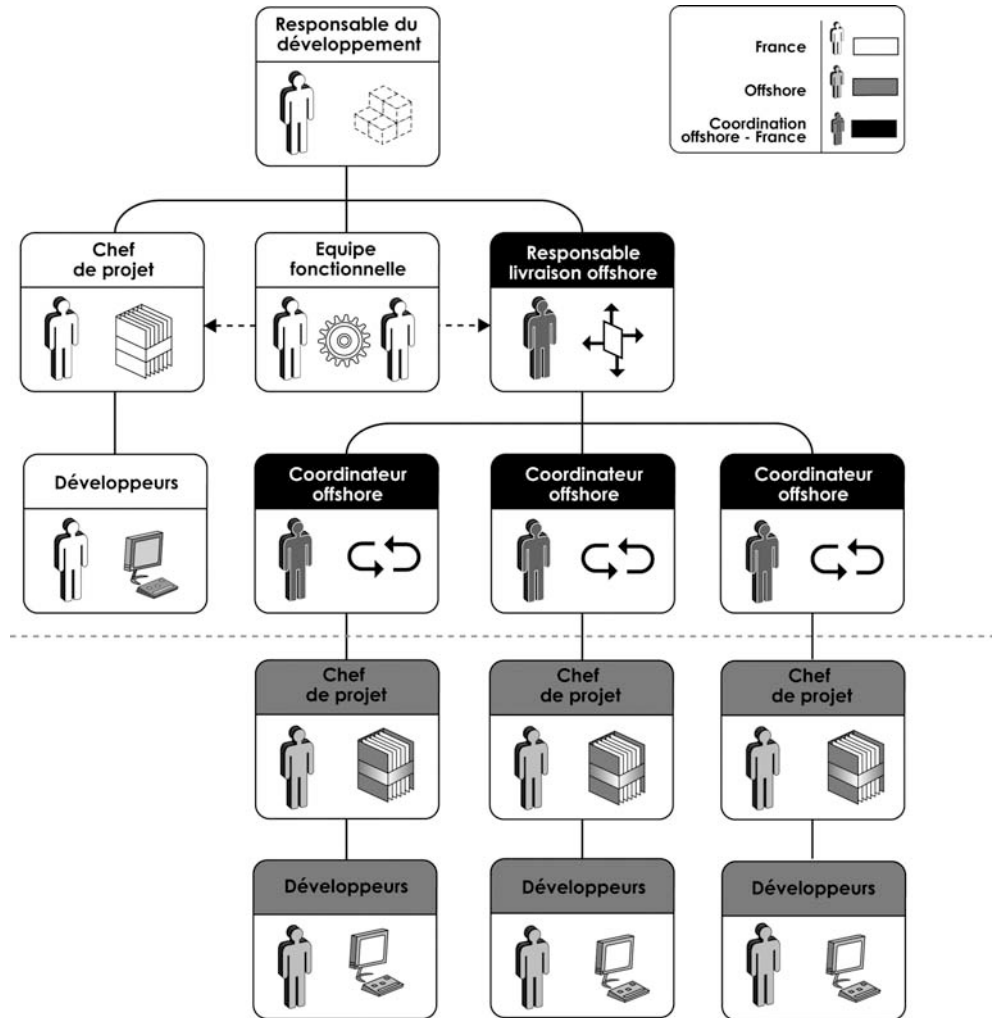


Figure 6 : Organisation on-site/offshore typique d'un projet de grande taille montrant le rôle clé des coordinateurs

Une telle organisation déclenche généralement la question suivante : combien faut-il de coordinateurs ? Il n'y a pas de réponse absolue, ce nombre est variable et dépend de plusieurs facteurs dont bien évidemment la complexité du domaine fonctionnel (d'autres facteurs sont la connaissance du domaine fonctionnel par les équipes, le temps passé sur le projet, etc). Notre expérience montre que le ratio coordinateur / nombre d'acteurs offshore varie entre 1 coordinateur pour 6 personnes offshore à 1 pour 10, ce qui est probablement une limite basse en deçà de laquelle il devient risqué de descendre.

La figure 7 montre une organisation on-site/offshore typique dans le cas d'un projet ayant moins de 10 personnes offshore.

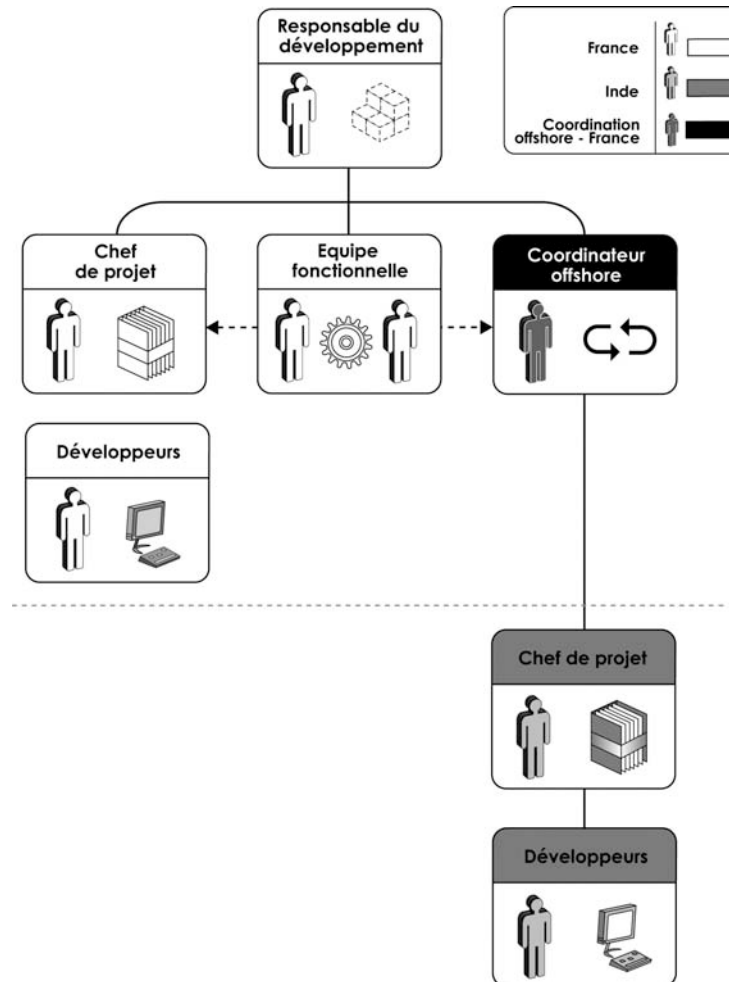


Figure 7: Organisation on-site/offshore typique pour un projet (moins de 10 personnes offshore)

Le rôle des coordinateurs offshore est donc de s'occuper exclusivement du lien entre équipes offshore et équipes on-site. Plus précisément :

- S'assurer que les questions posées par l'équipe offshore trouvent une réponse dans les meilleurs délais. En effet, une question est souvent bloquante et freine l'avancement du projet. Pour poursuivre le projet, il faut généralement passer à une autre tâche, ce qui s'avère difficile et coûteux. Une réactivité prompte est donc nécessaire et sans coordinateur cela fonctionne rarement. Le rôle du coordinateur est donc d'aller voir les personnes concernées on-site et de réussir à obtenir une réponse rapidement pour les équipes distantes.
- Fournir toutes les informations projet à l'équipe offshore. Comme elle ne peut participer à toutes les réunions on-site -réunions pendant lesquelles se prennent généralement les décisions- les personnes offshore manquent d'information sur le contexte projet et les raisons sous-jacentes aux décisions. C'est au coordinateur de combler ces lacunes.
- S'assurer que ce qu'attend le directeur de projet dans la prochaine livraison (en termes de contenu, mais aussi de qualité) correspond à ce que prévoit de réaliser l'équipe offshore. Il s'assure aussi que les dates de livraison correspondent.
- Réfléchir en permanence à l'amélioration du processus de développement afin qu'il soit optimisé pour l'offshore, et à celle des outils ou au moyen d'en trouver d'autres permettant de diminuer le facteur de

distance. Le coordinateur offshore est constamment concentré sur cette question : comment améliorer l'intégration de l'équipe offshore dans le projet. Il va dialoguer avec les différentes équipes on-site (équipe d'architecture technique, équipes fonctionnelles, équipes de tests et d'intégration, etc.) pour proposer des améliorations, pour expliquer les conséquences d'actions sur les équipes à distance et globalement s'assurer que toute action prend bien en compte la composante offshore.

- Jouer le rôle d'arbitre entre on-site et offshore. Le coordinateur offshore se doit d'être le plus impartial possible envers les deux côtés. Il va éclairer l'équipe on-site lorsqu'elle a une vision erronée d'un événement (qui s'est produit à distance) ou inversement expliquer à l'équipe offshore ce qu'elle ne saisit pas parfaitement sur certains sujets. Il est un harmonisateur.
- Garantir que les conférences téléphoniques soient régulières entre les différents sites géographiques.
- S'occuper des aspects logistiques (voyages, visas, ...) quand les acteurs se déplacent entre les sites et s'assurer qu'il y a suffisamment d'échanges et de déplacements (voir la section « communication » ci-dessous).

L'étude de cas à la fin de ce Livre Blanc donne des détails supplémentaires et concrets sur le rôle du coordinateur offshore.



Règle 2 : Avoir des coordinateurs offshore on-site

Ce rôle de coordinateur offshore est crucial. Il est illusoire de penser qu'il est possible de s'en passer même une fois aguerri au travail à distance. Il est capital car il contribue fortement à l'ambiance entre les équipes. Le coordinateur joue en effet le rôle de relais et de tampon, il est garant des temps de réponses sur les questions et il permet donc de limiter les situations d'inconfort liées à l'attente de ces réponses, il véhicule à bon escient les pressions mais aussi les succès et « victoires », or une bonne ambiance contribue largement à la réussite des projets.

Naturellement se pose maintenant la question de savoir ce que deviennent les chefs de projet on-site ? Certains continuent de diriger les équipes on-site. Cependant, ceux qui possèdent une équipe offshore subissent effectivement une mutation. Leur rôle évolue vers celui de coordinateur offshore. En effet, un certain nombre de tâches qu'ils assumaient auparavant (management des personnes) est maintenant effectué par le chef de projet offshore. Il faut donc gérer attentivement cette mutation et la prévoir dès le début car ce nouveau rôle est différent. D'autres évolutions sont bien évidemment possibles pour l'« ancien » chef de projet qui ne souhaiterait pas évoluer vers ce rôle de coordination : il peut se rapprocher du fonctionnel et travailler en assistance à maîtrise d'ouvrage, se rapprocher de la technique et rejoindre l'équipe d'architecture, devenir directeur de projet, rejoindre l'équipe de gouvernance offshore (voir plus bas), ...

5.2.3. Répartition des rôles

Le dernier point crucial au niveau de l'organisation des équipes est de disposer du maximum de types de profils en offshore. Il n'est pas rare d'observer des projets démarrer avec uniquement le profil de développeur en offshore. La raison est simple : avant d'investir davantage dans des profils et des rôles plus « pointus » sur le partenaire offshore, il est naturel de vouloir l'évaluer. Cependant, il est important d'élargir rapidement le spectre des profils. Dans le cas contraire, les personnes expérimentées vont rapidement quitter le projet car elles ne seront pas satisfaites d'un rôle de pur développement. De plus, sous l'angle de la productivité, une équipe qui n'est composée que de développeurs sera nécessairement amenée à beaucoup plus de séances de questions/réponses avec les équipes on-site du fait du manque de connaissances fonctionnelles poussées, de connaissances d'architecture, de maîtrise des stratégies de test, etc. Or nous avons vu que toute communication

à distance est difficile, coûteuse en énergie et potentiellement source d'incompréhension. Il est donc de l'intérêt du projet de mettre en place une équipe distante autonome le plus rapidement possible.

Autre aspect tout aussi qualitatif et ô combien important : la reconnaissance de l'équipe offshore par les équipes on-site. Pour qu'un projet se déroule correctement, la reconnaissance mutuelle des équipes doit être favorisée afin d'éviter les frictions naturelles qui risquent toujours d'arriver entre les deux côtés. Une stratégie pour cela est de composer des équipes homogènes des deux côtés afin de répartir les « forces » et d'éviter aux équipes on-site d'avoir un trop fort sentiment de « supériorité ».

Et maintenant la grande question : quels sont les rôles possibles pour l'offshore ? Tous les rôles sont-ils « offshorisables » ?

Notre expérience nous a montré que les rôles suivants se prêtent naturellement à l'« offshorisation » en complément bien évidemment des rôles de développeur et de chef de projet offshore :

- **Technical Lead / Expert technique** : il s'agit des acteurs qui représentent les équipes d'architecture ou les équipes techniques au sein des équipes de développement. Ce sont eux qui sont garants de la bonne application des meilleures pratiques, de l'architecture définie, etc. Ils sont aussi là afin d'œuvrer à l'amélioration continue de la qualité. En général, ce sont aussi eux qui sont les responsables du bon fonctionnement du *build continu*⁴ dans leur équipe. Ils agissent pro-activement pour assurer une bonne *intégration continue*⁵ et jouent le rôle de « pompier » quand le build échoue en coordonnant les efforts de réparation de ce dernier.
- **Concepteur** : il a pour tâche de faire le lien entre les spécifications fonctionnelles (ex. diagrammes UML de haut niveau) et le modèle d'implémentation. C'est un défricheur, il découvre les problèmes d'implémentation (faillite des spécifications, problèmes techniques). Ce rôle est essentiel à plus d'un titre car il permet à l'équipe offshore de s'approprier le modèle global. Dans certains projets, nous avons pu constater que la spécification détaillée était réalisée en amont, dans la phase de conception générale. C'est évidemment périlleux et très risqué car lors de l'implémentation, un niveau de détail trop fin donne des résultats souvent erronés et il est alors nécessaire d'effectuer une nouvelle conception détaillée. Par ailleurs, la connaissance fonctionnelle requise pour l'implémentation n'a pas été partagée et il est donc extrêmement difficile pour un développeur de l'implémenter en ayant le recul fonctionnel adéquat.
- **Testeur** : il a pour tâche d'interpréter les jeux de tests fonctionnels fournis par la maîtrise d'ouvrage et/ou les utilisateurs et de les exécuter de manière manuelle ou automatisée (grandement conseillé). Encore une fois, il s'agit d'un rôle parfaitement adapté à l'offshore car souvent le budget test est insuffisant. En bénéficiant des coûts de l'offshore, la qualité des livrables utilisateurs peut être largement améliorée par ce recentrage sur les tests. D'autre part, comme les testeurs sont en contact permanent avec les jeux de tests utilisateurs, ils acquièrent une connaissance fonctionnelle terrain qui se diffuse à l'équipe offshore (un développeur offshore peut ainsi demander à un testeur le test qui va valider son code et donc s'assurer le plus tôt possible que son code fonctionne correctement).
- **DBA/Infrastructure** : si l'équipe offshore est restreinte, ce rôle peut être joué par un des développeurs ayant des connaissances en bases de données. Selon le profil du DBA, ce rôle peut être étendu au rôle de gestionnaire des plateformes offshore. NB : Par plateforme nous entendons l'ensemble des machines et logiciels (SGBD, LDAP, serveur d'applications, ...) qui permettent de développer en offshore et d'assurer leur connectivité avec le site central (voir ci-dessous la section « Infrastructure »).

En revanche, il est préférable de conserver certains rôles stratégiques on-site comme par exemple les rôles en relation directe avec la maîtrise d'ouvrage (recette, direction de projet, etc.). Ceci est clé si l'entreprise souhaite

4 : Construction automatisée et fréquente (toutes les quelques heures) de l'application (voir sections « Gestion de projet » et « Pratiques de développement »).

5 : Synonyme de « build continu ». C'est ce qui permet de découvrir les problèmes d'intégration le plus tôt possible.

garder un contrôle sur son développement et être capable de réagir rapidement à toute demande métier. Nous recommandons également de conserver autant que possible le rôle de l'architecte on-site (à moins que 100% du développement ne se fasse à distance) afin de garder la maîtrise du système développé et parce que l'architecture devant être au service du fonctionnel, il est bon que les deux métiers soient les plus proches possibles.



Règle 3 : Avoir des équipes polyvalentes en offshore

Enfin, il faut savoir que la proportion d'acteurs on-site / offshore varie en fonction du cycle projet. Typiquement il sera judicieux de débiter avec un groupe –même réduit– d'acteurs offshore présents on-site afin de leur transférer la connaissance de base du projet. Puis la majorité sera en offshore pendant le reste du projet, avec une grande quantité d'échanges bilatéraux. A chaque itération majeure, il sera bon de faire venir une partie des analystes/concepteurs (souvent les chefs de projet offshore) on-site afin qu'ils rencontrent les acteurs fonctionnels et comprennent les cas d'utilisation (use cases) à développer. De même, il pourra être bon de faire venir des acteurs offshore pendant les grosses phases d'intégration afin de bénéficier d'allers-retours rapides pour la correction de bugs (à noter qu'une bonne intégration continue – voir ci-dessous – diminue grandement ce besoin). La règle des 70/30 est fréquemment citée, c'est-à-dire 70% offshore et 30% on-site (pouvant comprendre, outre les équipes client, des collaborateurs de la SSII offshore). Ce chiffre est une moyenne à ajuster selon le projet et dépend également de la phase du projet comme nous venons de le voir.

5.3. Gestion de projet

Une fois les équipes organisées, il convient de répartir les tâches à effectuer, gérer les livraisons et surveiller l'avancement.

La première règle cruciale est d'avoir des itérations les plus courtes possibles (de l'ordre de 2 semaines). Il s'agit déjà d'une bonne pratique pour du développement en un seul lieu géographique, cela devient vital lorsque l'on travaille à distance, et ce pour les raisons suivantes :

- Voir le résultat s'exécuter est rassurant, il s'agit d'un besoin naturel que de voir la preuve concrète du bon fonctionnement.
- Vérifier constamment que la communication s'est bien passée et que les équipes on-site et offshore sont en phase sur le travail à effectuer est le deuxième besoin flagrant constaté. En considérant des itérations de 2 semaines, les difficultés mettent, dans le pire des cas, un maximum de deux semaines à apparaître. Cela permet de corriger le tir rapidement.
- Les retours concrets sur les points d'achoppement auxquels font face les équipes de développement (base de données non synchronisées, questions en suspens, ...) constituent le troisième besoin. Les itérations courtes donnent la possibilité d'effectuer un retour arrière sur l'itération réalisée, d'en tirer les leçons et de répercuter les actions prises dans l'itération suivante.

La durée adéquate d'une itération dépend de la vélocité de l'équipe et de la taille du projet, donc de l'équipe. En général, nous avons observé qu'une durée d'une semaine est souvent difficile à tenir pour les équipes et qu'avec 3 semaines on perd un peu l'intérêt des itérations courtes citées ci-dessus. Un bon compromis est donc souvent 2 semaines. Nous parlons ici de rythme de croisière ; il est tout à fait possible de partir d'itérations beaucoup plus longues et au fur et à mesure d'en réduire la durée. Les bénéfices des itérations courtes restent

bien évidemment une fonction inversement proportionnelle de la durée de celles-ci. En 2005, les outils et pratiques nous permettent de descendre à des itérations de 2 semaines en travail à distance. Il est à parier que de meilleurs outils de communication et d'intégration continue permettront de réduire encore cette durée dans le futur.

La figure 8 montre les différents cycles qui existent :

- **Build continu** : il s'agit de la construction automatisée de l'application en développement. Nous y reviendrons dans la section suivante car c'est cette pratique même qui permet d'exécuter des itérations courtes sans essoufflement.
- **Itération** : ce sont les itérations dont nous avons parlé. Il est crucial de les gérer en « time-boxing », c'est-à-dire que les dates de livraison doivent être toujours fixes et seul le contenu doit pouvoir changer. Ceci a pour but d'éviter le syndrome du 80/20 (80% terminé, avec les 20% restant à faire qui sont les plus longs et difficiles). Le but pour l'équipe est donc d'implémenter complètement un maximum de « Use Cases » pendant une itération. On mesurera d'ailleurs ainsi la vélocité d'une équipe. Pour toute tâche dans une itération il est nécessaire de prouver à la fin de l'itération qu'elle fonctionne (souvent à l'aide de tests automatisés). Si un dérapage intervient, il pourra être décidé soit de déplacer une tâche dans l'itération suivante soit de couper une tâche en deux : c'est la gestion du changement.
- **Lot** : les itérations n'ont pas toujours un sens fonctionnel (il faut essayer de leur en donner un, mais cela n'est pas toujours possible). La notion de Lot représente un ensemble fonctionnel cohérent et utile pour les utilisateurs du projet. Sa durée est typiquement de 4 semaines (2 itérations), éventuellement plus. Cependant tout comme pour les itérations il est important de la garder la plus courte possible.

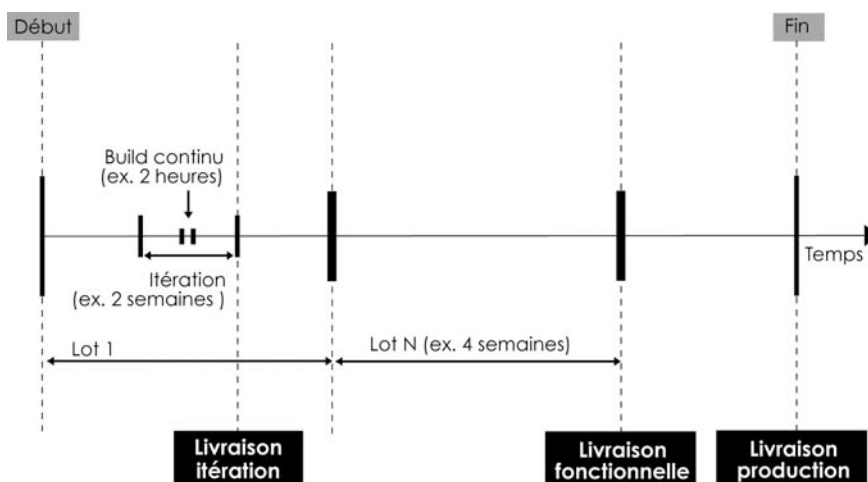


Figure 8 : Les différents cycles de livraisons



Règle 4 : Utiliser des itérations courtes et le « Time-Boxing »

Lors d'une itération de deux semaines, il est important de disposer de 2 points de contrôle, sous forme de conférence téléphonique avec les acteurs principaux (Chef de projet offshore, Chef de projet on-site, représentant

des utilisateurs, représentant architecture/qualité). Le premier point de contrôle se situe au milieu de l'itération et a pour objectif de faire le point sur l'avancement et le cas échéant de modifier le contenu de l'itération pour parvenir à livrer ce qui avait été prévu. Il sert également à initier la définition du contenu de l'itération suivante. Le deuxième point de contrôle se situe à la fin de l'itération et permet de finaliser le contenu de la nouvelle itération qui démarre immédiatement après. Il sert également à initialiser une « rétrospective », c'est-à-dire à capitaliser sur l'itération passée (défaut de fonctionnement, améliorations possibles, ...) et à réinjecter cette connaissance dans la nouvelle itération.

Une fois les itérations définies, il faut les rendre visibles pour tous et en suivre l'avancement. Il est crucial d'utiliser un outil adéquat permettant de travailler en mode itération et time-boxing. L'outil que Pivolis utilise fréquemment est JIRA (<http://www.atlassian.com>)⁶. La figure 9 montre la liste des tâches définies et leur statut pour l'itération en cours. JIRA est une application web et est donc accessible aisément à distance. Comme cet outil est manipulé par tous il doit être simple à utiliser. En complément, instaurer une discipline et un cadre d'utilisation de l'outil s'avérera nécessaire. Voici quelques règles simples à respecter :

- Toutes les tâches qui nécessitent une charge de travail d'une demi-journée ou plus doivent être listées. Cela permet d'éviter les désaccords sur le travail à réaliser. Cela arrive assez souvent en travail à distance pour cause de mauvaise compréhension dû à la langue, la culture ou la connaissance fonctionnelle. Si une tâche est effectuée sans avoir été listée, elle doit être rajoutée.
- A la fin de chaque tâche, la personne qui a terminé la tâche met à jour son statut. De même, si une tâche est bloquée, le point de blocage doit être commenté avant de passer à la tâche suivante. En d'autres termes, l'outil doit être une vue la plus à jour possible de l'état d'avancement en cours.

The screenshot shows the JIRA interface for the 'Cargo' project. The main section is titled 'Road Map' and shows a progress bar for version 0.6, indicating that 3 out of 9 issues have been resolved. Below this, a list of issues is displayed with their keys, status, and descriptions:

Issue Key	Status	Description
CARGO-124	UNRESOLVED	"Unable to unpack the application" error with WebLogic
CARGO-106	UNRESOLVED	Create an Orion Deployer
CARGO-125	UNRESOLVED	Does not support the cargo.[container].home property
CARGO-126	UNRESOLVED	Test on Resin 3.0.12
CARGO-116	UNRESOLVED	Add a Local Deployer for Tomcat
CARGO-117	UNRESOLVED	Add a Tomcat Deployer (using the manager)
CARGO-138	FIXED	did's are loaded from the web
CARGO-141	FIXED	EjbJarXml.getVendorDescriptor returns null for WebSphere
CARGO-140	FIXED	Maven cargo plugin references cargo-ant as a dependency

On the right side, there are sections for 'Reports', 'Preset Filters', 'Project Summary', and 'Open Issues'. The 'Project Summary' shows: Open (32, 23%), In Progress (21, 1%), Resolved (21, 1%), and Closed (105, 75%). The 'Open Issues' section is broken down by priority: Major (26, 76%) and Minor (8, 24%).

Figure 9 : Exemple de la liste des tâches à réaliser pour l'itération 0.6 du projet Cargo

⁶ : Il existe aussi d'autres outils comme par exemple ExtremePlanner (<http://www.extremepanner.com/>).



Règle 5 : Utiliser un outil collaboratif simple de gestion des itérations

Au-delà de la gestion des itérations, il est également nécessaire de bénéficier d'un planning général du projet. Les itérations n'apportent qu'une vue locale et de courte durée et il est toujours important de ne pas perdre de vue le projet dans son ensemble. La tenue du planning général est dévolue aux chefs de projet (le niveau de détail étant l'itération). Ils pourront utiliser pour cela des outils classiques comme Microsoft Project ou autre.

5.4. Pratiques de développement

Cinq bonnes pratiques fondamentales de développement sont à respecter en AOSD (pratiques XP pour la plupart) :

- **Build automatisé**
- **Intégration continue** (build continu central)
- **Tests automatisés**
- **Qualité active**
- **Refactoring**

L'**intégration continue** est l'élément clé qui favorise la réussite des autres pratiques. Le principe est celui décrit dans la figure 10 : le code développé est déposé dans un gestionnaire de sources en continu (plusieurs fois par jour) et un serveur de build prend le code du gestionnaire de sources et le construit (build). Dès qu'une erreur de build apparaît, un mail est envoyé aux équipes pour les avertir.

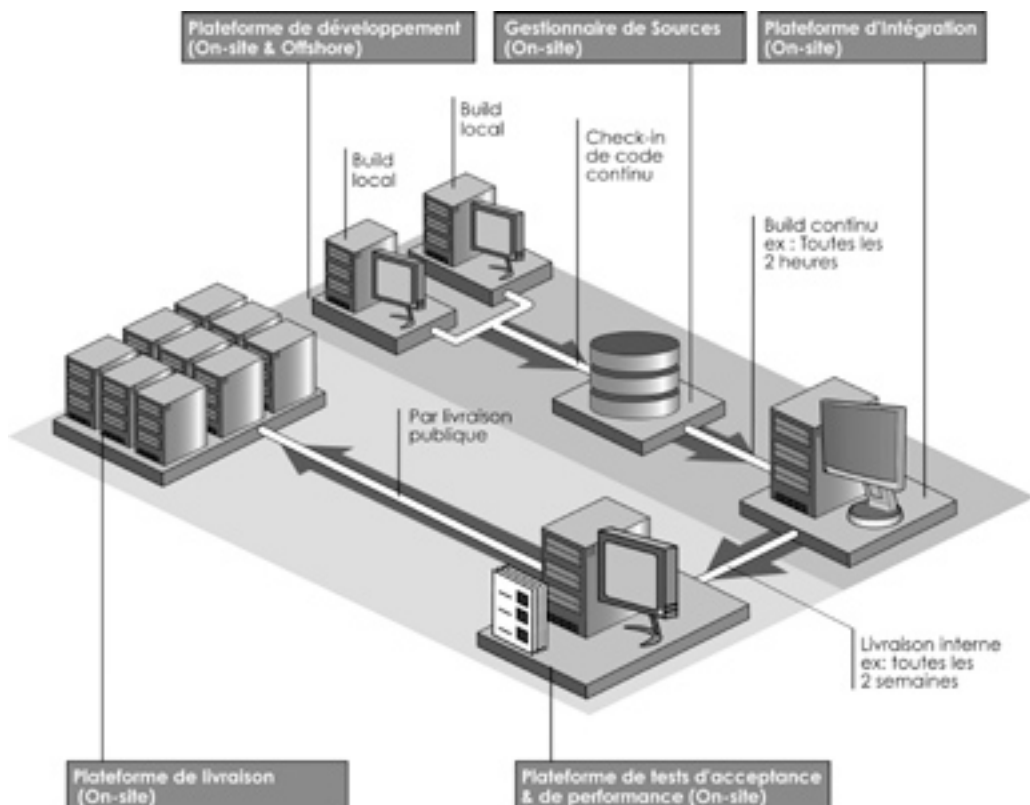


Figure 10 : Cycle de développement montrant les différentes plateformes

La qualité de l'intégration continue dépend donc directement de la qualité du build et des fonctions qui lui sont associées. Un build efficace doit pouvoir assurer automatiquement les fonctionnalités suivantes :

- Compilation et génération des exécutables
- Exécution des tests unitaires
- Packaging de l'application selon les environnements de déploiement
- Déploiement de l'application
- Exécution des tests fonctionnels
- Vérification de la qualité du code (couverture de tests, meilleures pratiques, convention de nommage, ...)

L'élément le plus critique dans le succès de l'intégration continue est probablement l'automatisation des tests fonctionnels. C'est souvent le chaînon manquant et pourtant c'est cette fonctionnalité qui garantit véritablement le bon fonctionnement de l'intégration et par conséquent celui de l'application. Bien évidemment, il est également nécessaire que le build continu fonctionne avec la fréquence la plus élevée possible (idéalement dès lors qu'une mise à jour est faite dans le gestionnaire de sources).



Règle 6 : Avoir un build continu fort est critère de succès

C'est l'intégration continue qui va permettre de livrer fréquemment et d'effectuer des itérations courtes. En effet, l'intégration continue est un exercice de livraison qui s'exécute à une fréquence de quelques heures et donc par cette pratique quotidienne, le jour de la livraison devient un non-événement ! Du point de vue organisationnel, le responsable du Build est le « Build Coach ». C'est lui qui va mettre en place le système de build automatisé (avec l'équipe de développement) et qui par la suite va en assurer la maintenance et l'évolution (un build continu se construit dans la durée).

Les tests automatisés trouvent tout leur intérêt en permettant d'effectuer une non-régression automatique de l'application. Cet aspect est indispensable pour permettre le refactoring qui deviendrait trop risqué autrement.



Règle 7 : Automatiser les tests unitaires et fonctionnels

Enfin, le build continu permet de vérifier automatiquement une partie des règles évaluant la qualité du code. Il s'agit de « Qualité Active » par opposition à la qualité « après coup » (telle que mesurée par des métriques) qui est passive.

Voici quelques exemples de règles de qualité que l'on peut appliquer dans le build et qui vont le faire échouer si l'une est violée :

- Taux insuffisant de couverture du code par les tests (unitaires et fonctionnels)
- Non-respect des règles de développement (mauvaise gestion des exceptions, nommage des classes/variables incorrectes, non-utilisation des frameworks recommandés, etc.)
- Découverte de duplication de code (copier/coller)

7 : Des exemples d'outils utilisés sont Checkstyle, PMD, FindBugs, Pattern Testing, Clover/Cobertura.

La qualité passive, via des métriques, est aussi nécessaire car elle permet de prendre du recul, de discerner les zones de qualité faible et ainsi de fixer des voies d'amélioration. La figure 11 montre un exemple de « tableau de bord qualité⁸ ».

Dashboard report												
Column legends												
Project	CS errors	CS warnings	Clover TPC	Clover LOC	Clover NCLOC	Simian TDL	JUnit Tests	JUnit Pass Rate	FindBugs Files with Violations	FindBugs Violations	All Tasks	SCM Changed Files
Core framework.	11	0	47%	2867	1733	0	13	100%	0	0	5	452
Monitoring console.	35	0	13%	2189	1393	0	5	100%	1	1	7	328
Sample.	2	0	-	-	-	0	0	-	0	0	0	428
WebApp pour le sample.	2	0	-	-	-	0	0	-	0	0	1	452
Total	50	0	N/A	5056	3126	0	18	N/A	1	1	13	1660

Figure 11 : Exemple de tableau de bord qualité

Chaque colonne représente un métrique. On y trouve par exemple :

- « CS errors » : nombre de violations des meilleures pratiques, mesuré par l'outil Checkstyle,
- « Clover TPC » : pourcentage de code couvert par les tests automatisés, mesuré par l'outil Clover,
- « Clover NCLOC » : nombre de lignes de code en excluant les commentaires,
- « All tasks » : nombre de « todo » disséminés dans les fichiers sources.

5.5. Infrastructure

Quelle infrastructure technique mettre en place pour faire de l'offshore en toute sécurité ? Aujourd'hui une solution de type VPN⁹ sur Internet apparaît comme suffisante (figure 12). Il s'agit de la solution la plus économiquement intéressante sachant que ces bandes passantes sont actuellement en mesure d'en assurer un bon fonctionnement.

Dans l'AOSD, nous préconisons au minimum 3 serveurs à rendre disponibles depuis l'offshore aussi bien que on-site :

- **Le gestionnaire de sources** : il s'agit du référentiel de code source des projets. Le gestionnaire de source versionne le code et permet de le partager entre les différentes équipes de développement.
- **Le serveur de build continu** : ce serveur a pour objectif de construire au fil de l'eau l'application à partir des sources, d'exécuter des tests, de déployer l'application, etc. Ce processus permet ainsi de pratiquer en continu et de manière automatisée les étapes d'une livraison. Des alertes mail sont également générées en cas d'échec ; enfin il génère des rapports de build dont il met le résultat à la disposition des équipes.

⁸ : Cet exemple est pris sur le projet open source JMonitoring (<http://jmonitoring.free.fr/>). Il est aussi possible d'avoir des historiques du tableau de bord qualité comme par exemple sur <http://jmonitoring.free.fr/dashboard-history-report.html>

⁹ : Définition de VPN extrait de Wikipedia.fr : « Un bon compromis consiste à utiliser Internet comme support de transmission en utilisant un protocole de « tunnelisation » (en anglais tunneling), c'est-à-dire encapsulant les données à transmettre de façon chiffrée. On parle alors de réseau privé virtuel (aussi appelé VPN, acronyme de Virtual Private Network) pour désigner le réseau ainsi artificiellement créé. Ce réseau est dit virtuel car il relie deux réseaux « physiques » (réseaux locaux) par une liaison non fiable (Internet), et privé car seuls les ordinateurs des réseaux locaux de part et d'autre du VPN peuvent « voir » les données. ». Voir l'article complet sur <http://fr.wikipedia.org/wiki/VPN>.

■ **L'intranet projet** : Ce serveur héberge notamment un wiki¹⁰ et l'outil de gestion des itérations que nous avons déjà abordé. Il peut également servir à publier les résultats du build, dont par exemple les tableaux de bord qualité (métriques) générés.

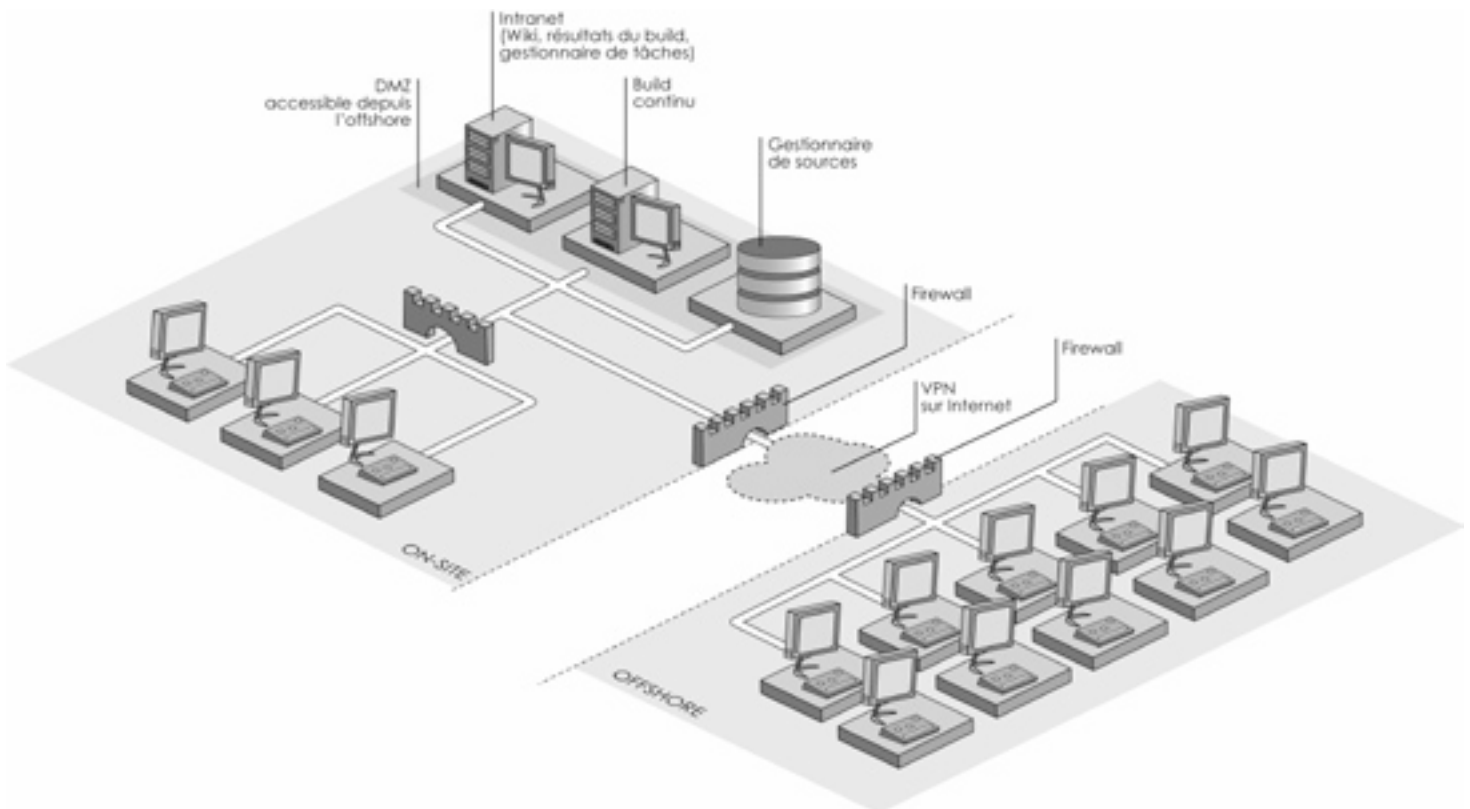


Figure 12 : Infrastructure VPN et serveurs partagés

Au delà de ces 3 serveurs, des serveurs d'intégration ou de recette doivent également être partagés afin que l'équipe distante puisse effectuer des tests plus avancés que ceux du build automatisé. Bien évidemment, il faut essayer de mettre le plus de tests possibles dans le build et prévoir cela dans la durée.

Enfin, point toujours critique, il va falloir définir une stratégie pour partager les données de référence/tests des bases de données utilisées. La meilleure stratégie que nous ayons trouvée à ce jour est d'inclure la gestion de la base de données comme un projet en tant que tel, donc de l'intégrer dans le gestionnaire de sources. Cela permet de contrôler les changements apportés aux schémas et données de référence et de disposer d'un mécanisme automatique pour reproduire à l'identique une configuration de base de données. Cela devient rapidement nécessaire dans une situation on-site/offshore puisque les développeurs des deux côtés doivent disposer autant que faire se peut des mêmes schémas et données.

D'autre part, on peut ainsi maîtriser l'état des bases utilisées pour exécuter automatiquement les tests fonctionnels. Il est toujours utile de rappeler qu'une gestion contrôlée des bases de données est primordiale dans tout projet de développement et encore plus dans un projet géographiquement distribué.

¹⁰: Définition wiki de Wikipedia.fr : « Un wiki est un site web dynamique dont tout visiteur peut modifier les pages à volonté. Il permet non seulement de communiquer et diffuser des informations rapidement (ce que faisait déjà Usenet), mais de structurer cette information pour permettre d'y naviguer commodément. Il réalise donc une synthèse des forums Usenet, des FAQ et du Web en une seule application intégrée (et hypertexte). ». Voir l'article complet sur <http://fr.wikipedia.org/wiki/Wiki>. Un bon exemple de wiki est Wikipedia (<http://fr.wikipedia.org/>).

Enfin, il faut également mentionner l'installation d'outils de communications, sujet de notre prochaine section.



Règle 8 : Rendre disponibles les outils et plateformes à distance

Tous ces outils sont simples à mettre en œuvre. Le plus gros frein à leur installation, ce sont souvent les cellules « sécurité groupe » des grandes entreprises qui, à juste titre, assurent leur devoir de précaution. Il est donc nécessaire d'anticiper le plus possible ces démarches pour ouvrir les portes de l'entreprise (ou du moins du projet) à un partenaire.

5.6. Outils de communication

Il existe pléthore d'outils de communication plus ou moins complémentaires et tous utiles ; l'art de la communication en offshore consiste à choisir l'outil adapté à une situation donnée. L'AOSD propose une classification des outils en fonction de la fréquence d'utilisation préconisée et du partage de la connaissance qu'ils peuvent apporter à l'ensemble de l'équipe (voir figure 13).

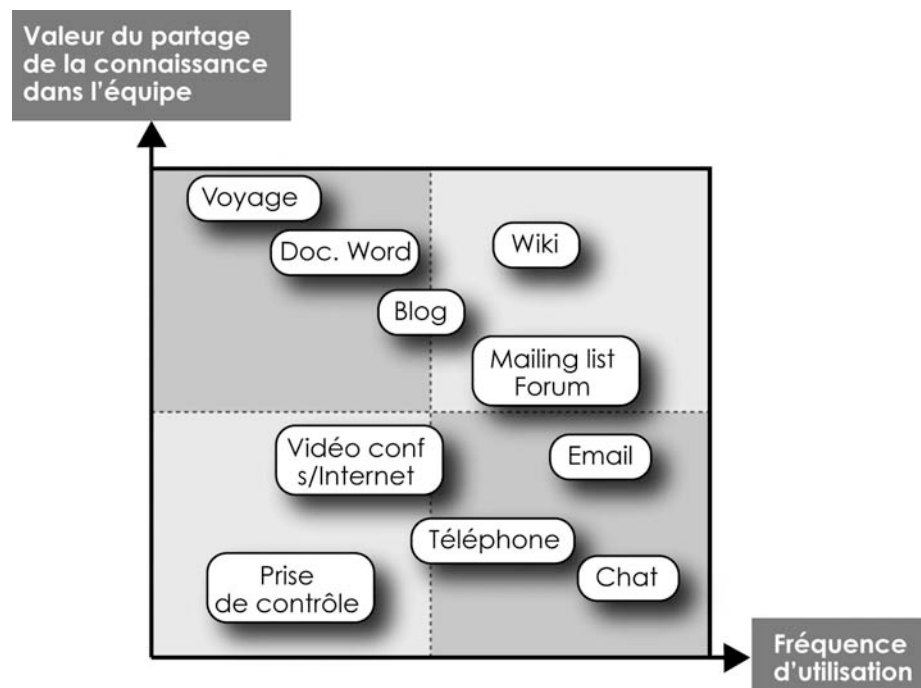


Figure 13 : Partage de la connaissance et fréquence d'utilisation des différents outils de communication utilisés

Les principaux outils utilisés dans l'AOSD sont ainsi :

- **La messagerie instantanée (« chat »)** : cet outil est installé sur tous les postes (équipes de développement, équipes fonctionnelles, équipes d'intégration, management, etc.). Il s'agit de l'outil le plus utilisé. A distance, le moindre problème peut être rapidement bloquant et il est primordial de pouvoir obtenir une réponse rapidement pour progresser. La messagerie instantanée est dans certains cas mieux adaptée que le téléphone car elle permet de vérifier la disponibilité de l'interlocuteur instantanément. De plus elle est moins intrusive que le téléphone qui force à répondre immédiatement et engendre ainsi des micro-coups néfastes pour le travail. La messagerie instantanée associée à des coordinateurs offshore situés on-site (et donc disponibles) constitue le couple gagnant. En revanche, le chat ne permet pas véritablement de partager la connaissance dans le groupe.
- **La « mailing-list »** : tous les projets open-source le savent : la mailing-list est un élément clé de cohésion et de partage de la connaissance. Il faut à tout prix éviter le cloisonnement de l'information. Il est ainsi souvent néfaste d'envoyer des mails uniquement à quelques personnes sans que les autres ne soient au courant. Cela crée inutilement des tensions. Le mieux est véritablement de toujours essayer d'avoir un minimum de mailing-lists. Lorsque le nombre de mails par liste devient trop important (à partir de 50 par jour), il est alors bon de scinder la liste en plusieurs. Le « spam » est aujourd'hui une véritable plaie, il est donc important de catégoriser l'information à envoyer. Par ailleurs il convient de différencier l'information nécessitant une réponse, pour lesquels le mail via une mailing-list est adéquat, de la simple diffusion d'information (métriques projets, différences sur les commits, nouveautés projets, ...) pour lesquels l'utilisation de fils RSS¹¹ est plus adaptée.
- **Le wiki** : c'est l'outil de gestion des connaissances « léger » par excellence. Il permet simplement à tous les membres du projet de rassembler de l'information sur tout type de sujet utile (trombinoscope, liste de tâches, meilleures pratiques de développement, définition de la plateforme utilisateur, description des procédures de build, de déploiement, etc.). Sa réussite tient dans sa simplicité d'utilisation. Il s'agit d'un site web éditable directement et qui tisse des liens quasi-automatiquement entre sujets. N'importe quel acteur du projet peut apporter sa contribution à un sujet existant et ainsi enrichir la connaissance globale du projet.



Le wiki c'est l'Excel du web !

- **Les voyages** : ils sont primordiaux, s'il y a bien une chose sur laquelle il ne faut pas rogner, ce sont bien eux. Il ne faut pas essayer de faire des économies sur les voyages, au contraire il est nécessaire de prévoir un budget dès le départ. Il est extrêmement difficile de travailler avec une personne que l'on n'a jamais rencontrée. Il faut donc organiser des rencontres pour les personnes qui vont travailler ensemble : leur vision de l'autre en sera complètement modifiée. Par ailleurs, pour une efficacité optimale, les voyages doivent se faire dans les deux sens.
- **La vidéoconférence et le téléphone** : nous pensons qu'aujourd'hui la vidéoconférence sur internet (à l'opposé de lignes louées) n'apporte pas un confort suffisant pour une utilisation professionnelle prolongée. Les personnes préfèrent en général se parler plutôt que d'utiliser la vidéo. En revanche, la téléphonie sur IP (VoIP) fonctionne en général suffisamment bien et est recommandée par rapport à l'utilisation du téléphone (coûts moindres, meilleure qualité d'écoute, possibilité de savoir si l'appelé est disponible à tout moment, conférence à plusieurs aisée, mains-libres, combinaison avec le chat pour transférer des URLs, images en temps réel, etc.). Bien évidemment la qualité obtenue en VoIP dépend fortement de la qualité des infrastructures réseaux des pays concernés (temps de latence par exemple).

¹¹ : Définition RSS de Wikipedia.fr : « Le RSS, acronyme de Really Simple Syndication (syndication vraiment simple), ou de Rich Site Summary (Résumé complet d'un site) est un format de syndication de contenu Web. C'est un dialecte de XML. Il existe sept formats différents de RSS, ce qui rend indispensable l'établissement d'une norme. ». Voir l'article complet sur http://fr.wikipedia.org/wiki/Really_simple_syndication

- **Les blogs** : le blog est plus connu comme « journal personnel ». Cependant c'est un formidable outil de « news » pour les projets distribués. Une des plaintes souvent évoquées en offshore est de ne pas être au courant des activités du projet en général et de ne pas avoir de vision de la cible et de l'objectif global. Le blog projet aide à combler cette lacune. Il est animé par un acteur à part entière et plusieurs membres du projet peuvent y contribuer. L'animateur a pour tâche d'encourager les contributeurs à écrire, pour récupérer les news projet. Initialement c'est lui qui écrit les entrées du blog. Il peut éventuellement y avoir plusieurs blogs : un blog projet général, un blog architecture, des blogs individuels, etc. A noter que certains wikis offrent aussi une fonction de blogging (appelés blikis [16], contraction de blog et wiki). Bien évidemment le gros intérêt des blogs est d'offrir (comme les wikis) un fil RSS afin que les news arrivent sur les postes de travail des acteurs du projet plutôt que d'obliger ceux-ci à aller les lire sur un intranet, ce qui ne fonctionne pas en pratique.



Règle 9 : Trouver des moyens pour toujours communiquer davantage

La communication est l'élément clé de réussite de tout projet, bien avant la technique. Une mauvaise communication entraînera à coup sûr des malaises, des préjugés et généralement une perte de productivité et de qualité. Il faut donc s'attacher à bien choisir des chefs de projets et des coordinateurs offshore qui doivent aimer la communication et être doués d'une forte empathie.

5.7. Pilotage

Il est souvent intéressant de mettre en place une cellule de « Gouvernance Offshore ». Cela signifie simplement avoir une ou plusieurs personnes qui s'occupent des aspects liés à l'offshore de manière neutre. Cette ou ces personnes peuvent être choisies parmi les employés de la société on-site, parmi les employés de la société offshore ou bien être trouvées auprès d'un acteur tiers offrant ces services (c'est le cas de Pivolis). Les points à valider pour qualifier de telles personnes sont : l'impartialité/neutralité et l'expérience de l'offshore.

Bien évidemment, ces personnes ne remplacent pas les chefs de projets/coordonateurs on-site et offshore qui sont, eux, opérationnels. En revanche, il est utile d'avoir une vision de plus haut niveau sur la façon dont le développement offshore fonctionne au sein du projet, les difficultés rencontrées, ce qu'il faut améliorer, ...

L'important est la neutralité, c'est-à-dire la capacité de l'équipe de Gouvernance Offshore à porter un œil critique extérieur aussi bien sur les équipes et l'organisation on-site que sur les équipes offshore.

D'autre part, cette équipe organise des Comités de Pilotage Offshore qui donnent une visibilité macroscopique aux acteurs du projet sur la partie offshore. Elle statue sur les axes d'amélioration à apporter pour améliorer le développement offshore. Les personnes présentes (physiquement ou via téléphone) à ces comités peuvent être :

- L'équipe « Gouvernance Offshore » bien évidemment,
- Le Directeur de Projet on-site
- Le Responsable des développements on-site
- Le Directeur de Projet offshore
- Le Responsable, au sein du prestataire offshore, de la relation avec le Client
- Les chefs de projet/coordonateurs on-site et offshore



Règle 10 : Avoir des acteurs dont la mission est l'amélioration de l'offshore

Enfin, la Gouvernance Offshore met en place, avec les chefs de projet, des indicateurs de santé du projet. Le tableau ci-dessous (figure 14) donne quelques exemples d'indicateurs.

Catégories	Indicateurs	Commentaires
Qualité du développement	Nombre de bugs sur plateforme d'intégration/recette par jour-homme de développement	Faire bien attention à séparer les bugs des demandes de modification, et généralement des bugs qui ne sont pas des bugs...
	Pourcentage de couverture de code par les tests	Une bonne valeur à atteindre est 70% ou plus. Surtout ne pas essayer d'atteindre 100%. Pré-requis : automatisation des tests
Vélocité de développement	Nombre de tâches/fonctionnalités par itération	La difficulté est de catégoriser les « use cases » par complexité (facile, moyen, difficile ou bien toute autre unité). Pré-requis : itérations courtes et de durée relativement fixe.
	Pourcentage de turnover	Permet de mesurer le bien-être dans le projet, qui est directement proportionnel en général à la productivité
Coût de l'intégration	Nombre de commits par jour (permet de juger de la dynamique du projet et de l'intégration continue)	Une bonne valeur est une valeur supérieure à 1 commit par développeur par jour. Variante : nombre de lignes de code/configuration committée par jour-homme
	Temps de correction d'un build qui casse (i.e. jusqu'au prochain build réussi)	Pour de petits projets une bonne valeur est inférieure à 1 heure ; pour des plus gros projets, inférieure à 2 heures. Pré-requis : build automatisé
	Temps nécessaire pour livrer une version sur la plateforme d'homologation à partir du moment où le feu vert est donné	Une bonne valeur à atteindre est inférieure à 4 heures de bout en bout (depuis la récupération du code source jusqu'à l'installation terminée sur la plateforme de livraison et tous les tests joués, avec génération de release note). Il faut s'entraîner à chaque itération afin que le jour de la vraie livraison ce temps soit le plus court possible et que la procédure (automatisée autant que possible) soit rodée.
Qualité des communications	Temps de réponse aux questions posées d'un côté à l'autre (offshore/on-site)	Une bonne valeur est inférieure à 1 heure pour, au moins, une réponse qui prend acte de la question et donne un délai de réponse final. Cela permet à l'équipe distante de pouvoir planifier et s'organiser.
Maintenance	Temps de réponse de bout en bout pour un bug remonté en production	

Figure 14 : Exemple d'indicateurs à surveiller

6. Comment démarrer un projet offshore Agile ?

Evitez de tout envoyer en bloc en offshore ! C'est le premier mot d'ordre à garder en tête avant d'aborder ce sujet.

L'offshore n'est pas une solution miracle. C'est un choix possible parmi l'ensemble des choix disponibles. Ce n'est pas nécessairement le bon. S'il paraît évident que dans le futur les sociétés vont se concentrer de plus en plus sur leur cœur de métier et donc que le développement informatique se fera quasiment certainement toujours auprès de sociétés spécialisées, cela ne veut pas dire pour autant qu'il faut tout envoyer en offshore d'un coup. Les questions suivantes doivent au préalable trouver leur réponse :

- Mes processus internes me permettent-ils de travailler à distance tout en gardant un même niveau de contrôle sur ce qui est développé ?
- Suis-je en mesure de conserver l'architecture des solutions développées, ce qui me permettra d'aligner mon système d'information avec ma stratégie business ?
- Ai-je choisi le bon partenaire et puis je en changer si besoin ? et surtout comment ?
- En remplaçant des développeurs internes, ne vais-je pas perdre un savoir important accumulé au fil des années et si oui, comment faire pour le transférer intelligemment sur la durée ?
- Ne vais-je pas créer des tensions internes inutiles en envoyant tout en offshore et n'y a-t-il pas une solution plus modérée et plus pérenne qui permette à mes collaborateurs d'évoluer vers de nouveaux postes ?

Nous préconisons une approche modérée, d'avantage fondée sur la collaboration (gage de succès des projets). Les méthodologies Agiles et la méthodologie AOSD permettent d'envoyer des parties de développement en offshore avec la possibilité à tout moment d'un retour arrière -si un problème important survenait- ou bien au contraire de progressivement accroître le nombre de personnes offshore, tout en gardant un écosystème maîtrisé. En effet, il faut que les processus internes et les acteurs s'adaptent à l'offshore en douceur avant qu'ils ne se sentent à l'aise et que des gains de productivité et de qualité puissent être constatés.

6.1. Les étapes en amont du projet

Comme cela a été évoqué, il est impératif de se lancer dans l'offshore en connaissance de cause. Si ce mode de fonctionnement est source de gains considérables, les écueils pour réussir une telle démarche sont réels. Nous avons déjà mentionné (cf § 2.4) les deux principales voies d'une stratégie offshore en informatique :

- **L'approche formelle**, qui consiste à mener une réflexion amont approfondie pour prendre en compte tous les risques, contraintes et opportunités liées à l'offshore. La décision¹² se fera ainsi en plusieurs étapes, ce qui pourra prendre plusieurs mois.
- **L'approche pragmatique**, qui consiste, une fois le panorama du marché offshore réalisé, à isoler un projet –ou un lot d'un projet important– sur lequel tester l'offshore de manière concrète. C'est ce test qui permettra de faire le « go / no go » puis, le cas échéant, de lancer une stratégie offshore de moyen-long terme. C'est ce que l'on appelle une approche « Test & Invest », grâce à laquelle on va « apprendre en marchant » avec un niveau de risque maîtrisé (du fait de la taille réduite des projets initiaux).

En réalité, les deux approches ne sont pas radicalement opposées. La seconde approche permet d'aborder l'ensemble des points essentiels à une démarche offshore. Cependant, à la différence de la première qui y travaille en amont, elle le fait sur le terrain en s'appuyant sur une expérience concrète. L'approche pragmatique nous paraît la plus intéressante car elle permet de toucher rapidement du doigt le vécu d'une expérience offshore, ce qui constitue à notre sens l'élément central d'une telle stratégie.

Quelle que soit la voie choisie, les éléments à aborder (chemin faisant ou en amont de la démarche) sont les suivants :

- **La compréhension du contexte offshore** : quelles sont les contraintes et les implications d'un travail et d'un pilotage à distance, notamment avec un pays dont la culture peut être sensiblement différente de celle de la France ? En pratique, l'entreprise pourra par exemple organiser un court atelier de présentation des enjeux et contraintes de l'offshore, des pays concernés. Une visite sur place de prestataires offshore se révélera rapidement un complément indispensable à cette démarche¹³.
- **Le choix du projet ou de l'expérience pilote** (cf ci-après).
- **Le choix du partenaire offshore**. Pour cela il faudra décider en premier lieu du pays cible. Cela dépendra principalement de 3 critères : la langue de travail du projet, la taille de l'équipe projet (qui, si elle est réduite, pourra amener à considérer des pays disposant d'effectifs moins importants que ceux des « grands » fournisseurs) ; le contexte du pays sur le plan politique (stabilité, maturité du système juridique, ...) et économique (croissance, inflation, politique offshore, taille du marché des services informatiques, ...) est également un facteur de choix (et de veto potentiel) important. Pour ce qui est du décalage horaire, l'expérience montre que – hormis les cas extrêmes– ce n'est pas un critère majeur ; le travail avec l'Inde – où, rappelons-le, le décalage n'est que de 4h en moyenne– montre que le travail en décalé présente en fait des avantages réels (l'exemple le plus courant étant la possibilité de faire préparer par les équipes on-site des listes de corrections de bugs, sur lesquelles les équipes indiennes travailleront pendant la nuit en France). Pour le choix du partenaire offshore, il est essentiel de bien qualifier les points suivants :
 - Compétences techniques
 - Certifications obtenues & méthodologies utilisées
 - Infrastructure de communication & outils collaboratifs utilisés
 - Références
 - Taille totale des effectifs (et turn-over constaté)
 - Prix

¹² : En fait il faudrait plutôt parler des décisions : « go / no go » dans un premier temps, puis choix du mode de coopération, choix du projet pilote, choix du type de prestataire (SSII étrangère, société-pivot, ...), choix du prestataire, ...

¹³ : Là encore, on peut différencier une approche très « amont » qui consiste à faire un voyage d'études pour visiter un certain nombre de prestataires génériques, ou une approche pragmatique qui consiste à visiter des prestataires présélectionnés dans le cadre d'un projet pilote.

- **Le plan de communication interne.** C'est un élément à ne pas négliger dès le lancement (même en test) d'un dispositif offshore. Il permettra de s'assurer du succès du projet sur un plan humain et social. Il faut souligner que l'absence d'adhésion (voire l'opposition) des équipes internes à un tel projet constitue un handicap majeur, parfois rédhibitoire.
- **Le cadre méthodologique.** Comme cela est expliqué et détaillé tout au long de ce livre blanc, l'organisation du projet, les processus et les outils collaboratifs doivent faire l'objet d'une réflexion particulière; ils doivent être définis et mis en place pour être opérationnels au lancement du projet.
- **Le cadre juridique du projet.** Les cadres contractuels dans lesquels s'inscrivent les projets offshore doivent nécessairement être précis sur des aspects propres à la divergence des droits. Les thèmes majeurs à préciser sont ceux relatifs à la propriété intellectuelle, à la propriété des sources, ... à la sécurité (confidentialité des informations et données échangées) et aux clauses de responsabilité (ressources humaines, licences logicielles, dommages,...), les engagements doivent être définis clairement (obligations de moyens et de résultat). Il convient également de s'assurer du respect des obligations fiscales et de la législation du travail (horaires, lieu, sécurité).

En outre, une attention particulière devra être portée à certains aspects administratifs et plus particulièrement à la gestion des visas pour le personnel offshore étant amené à venir sur des périodes relativement longues en France.

6.2. Le choix du premier projet offshore

Le choix de ce premier projet constitue un réel enjeu puisque la décision d'aller vers l'offshore, dont les implications internes peuvent être non négligeables, crée une pression supplémentaire pour réussir ce projet. Il est important de prendre conscience que tous les projets ne sont pas adaptés à l'offshore.

En premier lieu, nous préconisons de choisir un projet « réel » dont le résultat sera effectivement utilisé en interne (par opposition à un prototype, à vocation « jetable ») ; c'est en effet ce qui mobilisera réellement les ressources pour le réussir et ce qui convaincra ensuite les plus réticents.

Cela posé, on peut distinguer **deux stratégies projet** pour se lancer dans l'offshore :

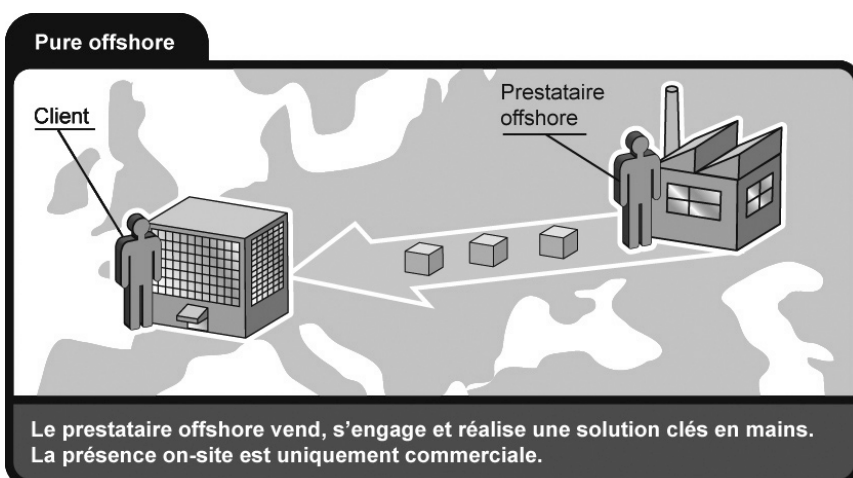
- **Le « pilote isolé »**, qui consiste à identifier un projet bien délimité de taille réduite pour faire un test. Il offre l'avantage évident de simplifier la mise en œuvre et de réduire les risques ; c'est un test même si il doit rester appliqué à un projet réel et concret. L'inconvénient étant évidemment sa portée limitée qui pourrait ne pas faire taire toutes les critiques ou doutes en cas de réussite.
- **Le projet « grandeur réelle »**, qui consiste à se lancer sur un projet de taille significative avec un réel impact business sur l'entreprise. Afin de ne pas prendre de risques excessifs dans ce premier projet offshore, il pourra être loti et seuls certains lots seront effectués en offshore, tout en se gardant la possibilité de monter en puissance sur le reste des lots par la suite. Il est possible de démarrer par exemple en y affectant 10-20% des ressources de développement dans les premiers temps, et d'augmenter ce ratio au fur et à mesure que le mode de fonctionnement offshore sera maîtrisé. L'avantage de cette approche étant naturellement un gain économique et une visibilité interne forts.

Quelle que soit la stratégie choisie, il conviendra donc dans un premier temps de faire une cartographie du système d'information et surtout du planning des projets, puis de leur appliquer des critères précis pour identifier le bon projet. Une partie de ces critères sera spécifique à l'entreprise ; nous pouvons toutefois en lister quelques-uns génériques, essentiellement pour limiter au maximum les risques liés à un premier projet :

- **La maîtrise fonctionnelle.** L'objet de ce livre blanc est de présenter l'intérêt des méthodologies Agiles dans le cadre de projets offshore ; un de ces intérêts est la capacité de ces méthodologies à prendre en compte des spécifications « mouvantes et évolutives ». Il n'en reste pas moins que pour réduire le risque sur un premier projet, une partie fonctionnelle bien maîtrisée sera préférable. C'est par exemple le cas des projets de portage, de migration ou de montée de version (upgrade) très fréquents dans le domaine offshore, mais cela peut également s'appliquer à un nouveau projet de développement.
- **La dépendance avec le SI.** En général, la multiplicité des interfaces d'un projet avec le reste du SI est facteur de complexité et donc de risque. Dans le cas de l'offshore, ce facteur là est souvent exacerbé car le travail à distance amène un certain nombre d'obstacles supplémentaires en termes de connexion aux différents sous-systèmes existants. Ce facteur est d'ailleurs souvent cité comme un critère général pour le choix de projets offshore ; c'est d'autant plus vrai dans le cadre d'un premier projet.
- **La taille.** Plus le projet-cible sera important, plus l'emphase sera mise sur les aspects méthodologiques et son approche sera structurée, ce qui est crucial dans un contexte offshore. Pour aller dans le même sens, il faut souligner que les gains économiques sont directement proportionnels à la taille des projets, notamment en phase d'apprentissage de l'offshore.
- **Les contraintes de délai.** Pour un premier projet, cet élément peut constituer un facteur de risque supplémentaire et il sera judicieux d'éviter de choisir un projet au planning trop serré.

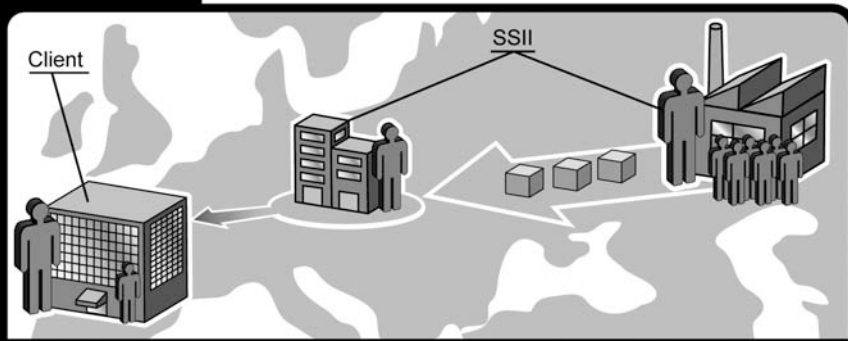
6.3. Les différents types de prestataires offshore

Le choix du bon partenaire offshore passe tout d'abord par la compréhension des différents business models qui co-existent sur le marché, et qui sont assez spécifiques. Nous en distinguons quatre principaux :



L'approche « **Pure offshore** » : le prestataire est une société dont le siège est situé dans un pays offshore. Dans ce cas, c'est elle qui s'engage directement auprès du client et réalise la très grande partie de la solution clés en mains à l'étranger. La présence locale de tels prestataires ne peut parfois être que commerciale (i.e. peu ou prou de direction de projet locale, ce qui peut constituer à notre sens un risque important qu'il sera important d'évaluer).

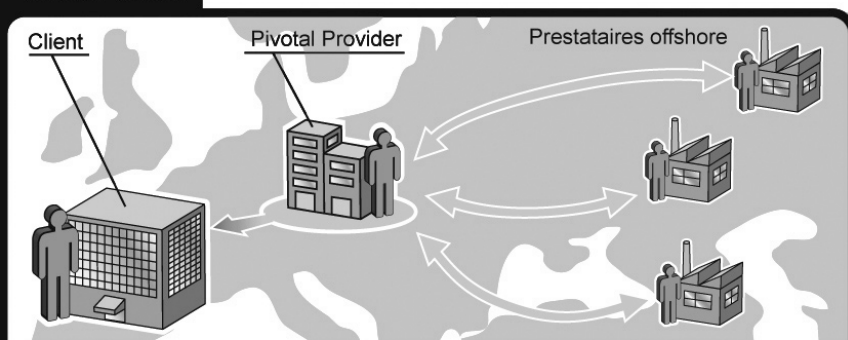
SSII



La SSII proche du client utilise ses propres ressources offshore pour faire baisser le coût de ses prestations. Le client n'est pas toujours averti qu'une partie de la prestation est réalisée offshore et seul le prestataire tire partie des économies réalisées.

L'approche « **SSII on-site/offshore** » : le prestataire est dans ce cas souvent une SSII française (ou internationale) qui utilise ses propres ressources offshore pour faire baisser le coût de ses prestations. Il est d'ailleurs arrivé que les clients ne sachent pas qu'une partie de la prestation était réalisée en offshore (dans ce cas seul le prestataire tire partie des économies réalisées !...). Dans tous les cas, la SSII, à la différence du premier modèle, utilisera ses propres ressources pour l'on-site. De fait, il faudra dans ce cas bien mesurer si le mix de ressources on-site/offshore est adapté, car l'on pourra constater dans ce modèle une dérive inflationniste des ressources on-site, induisant un gain final moins intéressant.

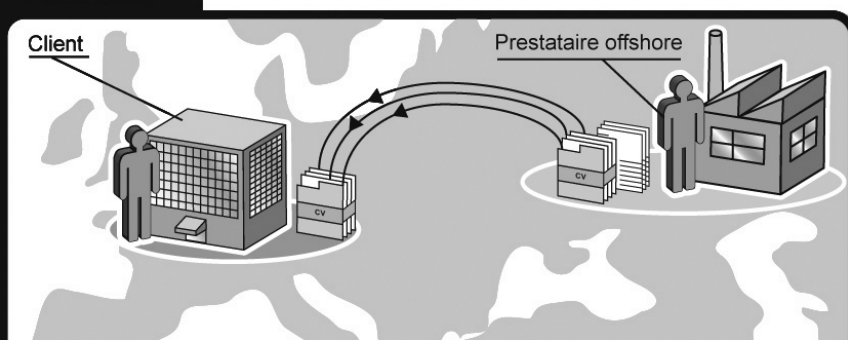
Pivotal Provider



La SSII noue des partenariats avec des prestataires offshore et joue l'intermédiaire auprès du client. Approche basée sur le concept on-site/offshore. Le pivotal provider s'engage alors sur le résultat final.

L'approche « **Pivotal Provider** » : le prestataire est une société française qui noue des partenariats avec des prestataires offshore. Ses ressources propres sont focalisées sur la direction de projet et la coordination offshore qu'elle prend en charge, ainsi que tout ce qui a trait à la gestion des aspects logistiques, juridiques et administratifs. Tout le travail de développement est effectué par le partenaire offshore sous sa direction. Le « pivotal provider » s'engage ainsi sur le résultat final, vis-à-vis de ses clients.

Ressources



le prestataire offshore met à disposition des CVs / ressources et essaye de les vendre au client final. La cible est le plus souvent des travaux à faible valeur ajoutée.

L'approche « **Ressources** » : le prestataire est une société offshore qui met à disposition de simples CVs de collaborateurs pour les vendre au client final. La cible est le plus souvent des travaux à faible valeur ajoutée.

6.4. Le facteur humain

Le choix d'une stratégie offshore n'est pas anodin et ne laisse certainement pas indifférents les collaborateurs au sein de l'entreprise. Il est clair que les résistances au changement seront multiples à cause des craintes que cela va générer au sein de l'entreprise. Certaines de ces craintes seront de nature rationnelle (« est-ce que je vais perdre mon job ? »), d'autres de nature plus émotionnelle (« moralement je ne peux pas adhérer ! »).

Il est de fait indispensable dans cette phase de démarrage de porter une attention particulière à la conduite du changement induit par cette nouveauté ; si ce point là n'est pas pensé dès l'amont, l'émotionnel pourra prendre le dessus et arriver rapidement à une situation de blocage ou simplement à des réticences telles qu'elles mineront le premier projet.

Il conviendra notamment de définir un plan de communication interne qui sera mené, à la fois vers le haut de la hiérarchie (la direction de l'entreprise, qui devra adhérer et donner l'impulsion) et vers les collaborateurs de la direction informatique. Ce plan sera très spécifique à l'entreprise, en fonction du contexte, des enjeux alloués à ce choix stratégique, ... Il sera judicieux d'utiliser différents outils tels que des ateliers présentant l'offshore, ses avantages, ses limites ainsi que le pays choisi ; ces ateliers permettront aussi (et surtout) de répondre à toutes les questions posées. Les avantages de l'offshore seront naturellement mis en évidence pour les informaticiens, notamment en terme d'élargissement du champ des compétences. Il est aussi recommandé autant que possible d'effectuer un ou plusieurs voyages ; c'est la meilleure manière de démystifier l'offshore.

6.5. Les modes contractuels

Il existe deux grands types de contrats de prestations de services informatiques, que l'on retrouve en offshore :

- **Le forfait.** En général cela veut dire à la fois un prix, un périmètre et un délai figés,
- **La régie.** La facturation s'effectue en fonction du temps passé et du nombre de personnes présentes sur le projet.

Aucune de ces deux solutions n'est parfaite telle quelle. Le forfait « traditionnel » n'est pas la meilleure solution car en informatique il est rare de prévoir précisément à la fois le temps qu'il va falloir pour développer un système et l'adéquation de ce système dans le temps par rapport à l'espérance qu'en ont ses utilisateurs. Il n'est donc pas rare de voir des facteurs 2 ou 3 dans les écarts entre chiffres réels et estimations [17]. Le forfait est donc souvent un « coup de poker » qui va pénaliser le client (il paie trop cher) ou le fournisseur (il perd de l'argent). En pratique c'est encore plus complexe que cela et beaucoup de fournisseurs acceptent des forfaits à prix bradés car ils savent que le coût réel sera plus élevé et jouent là-dessus pour aligner avenant sur avenant et ainsi récupérer leur mise. Le forfait n'est donc pas en général la meilleure solution pour le projet lui-même et ne va pas dans le sens d'une bonne collaboration dans la durée [18, 19, 20].

La régie présente aussi des risques : le fournisseur ne va-t-il pas s'endormir sur son travail, se laisser aller et avoir tendance à ne pas trop accélérer le développement afin d'avoir des prestations les plus longues possibles ?

La régie forfaitée nous semble la meilleure solution. Elle permet, grâce à un système de partage des gains, d'inciter la maîtrise d'œuvre et la maîtrise d'ouvrage à optimiser les charges et les délais du projet dans un véritable esprit de partenariat et de partage des responsabilités. Concrètement, il s'agit de fixer un budget pour une période donnée et de collaborer pour arriver à la meilleure liste de fonctionnalités possible pour une mise en production à cette date. Le but est de livrer un produit meilleur (c'est-à-dire plus adapté) que celui imaginé initialement car un apprentissage progressif va être effectué tout au long du projet. Ce sera aussi l'occasion de déterminer si le projet est plus compliqué qu'il n'y paraissait. Dans ce cas le planning pourra être modifié, et si le client n'accepte pas ce nouveau plan, il aura la possibilité d'annuler le projet en dernier recours. Bien que cela ne soit jamais une bonne chose, cela lui aura quand même permis de découvrir ces problèmes bien plus tôt que si le projet avait été réalisé en mode forfait « traditionnel »; en effet, ce dernier mode a tenté de suivre le plan initial coûte que coûte, en décourageant les changements et donc en empêchant souvent de se rendre compte que le projet est en danger. La régie forfaitée est le mode de contrat qui est nous paraît le plus adapté aux méthodologies Agiles et à l'offshore.

7. Retour d'expérience : un projet offshore dans le monde bancaire

7.1. Contexte

Le consortium Pivolis-KPIT intervient depuis Mars 2003 sur un projet qui se déroule actuellement au sein d'une grande banque internationale. Il s'agit d'un projet de refonte des systèmes d'information d'une ligne de métier de cet établissement. Le projet s'étale sur plusieurs années, incluant le déploiement du système développé sur plusieurs marchés. La taille de l'équipe de développement est de l'ordre de 100 personnes.

Le projet était déjà commencé depuis plus d'un an lorsque Pivolis et KPIT ont démarré leur intervention. Dans un cadre de contrôle des coûts, nous avons proposé de mettre en œuvre une solution de développement offshore collaboratif afin de pouvoir poursuivre le projet au sein du nouveau cadre budgétaire. Pivolis et KPIT ont apporté une solution globale : équipes on-site et offshore (en Inde), mise en œuvre de la méthodologie AOSD et pilotage du projet on-site avec le client.

7.2. Mission de Pivolis-KPIT

La mission globale de Pivolis-KPIT est de faire en sorte que la solution offshore fonctionne sur le projet. Nous intervenons de manière transversale aux équipes sur ce projet et mettons en contact directement les deux partenaires (offshore et on-site). Autrement dit, notre mission est celle d'un « accélérateur d'offshore », pour réussir à travailler en offshore dans les meilleures conditions possibles pour le client.

En pratique, 2 personnes en France interviennent depuis le démarrage du projet sur les aspects suivants :

■ Organisationnel et pilotage

- Apport d'une vision externe et dépassionnée quant à l'offshore.
- Organisation et animation des comités de pilotage offshore
- Organisation et participation aux réunions téléphoniques techniques et de management hebdomadaires avec toutes les équipes (équipes de développement + équipes d'intégration + équipes techniques) et suivi des actions décidées lors de ces appels
- Organisation et participation aux réunions de management avec les équipes on-site et offshore
- Facilitation des communications bilatérales
- Support projet des équipes offshore pour compenser les lacunes dans la relation on-site/offshore des équipes projets (et apport de solutions correctives)
- Sélection des membres de l'équipe offshore (recrutement)
- Création et amélioration des programmes de formation offshore

- Investigation et gestion des crises
- Accompagnement des membres de l'équipe on-site pendant les visites offshore, garantissant la bonne tenue des agendas et le succès des visites.

■ Infrastructure pour l'offshore

- Coordination et amélioration continue sur toutes les questions d'infrastructure (surveillance des performances, outils de paramétrages, etc.).

■ Méthodologie/Expertise

- Mise en place et surveillance d'indicateurs projets
- Définition du processus de développement Projet avec « l'offshore à l'esprit » et améliorations continues (introduction d'itérations courtes, utilisation de JIRA pour les planifications détaillées des itérations, définition de critères d'acceptation pour la livraison du code, utilisation d'un wiki, etc).
- Travail à l'intérieur des équipes pour mettre en œuvre le processus de développement collaboratif (AOSD)
- Suggestions qualitatives et mise en place d'améliorations (stratégie de test / améliorations orientées qualité des builds + mesures)
- Expertise sur le processus de build (Maven) ; mise en place et expertise sur l'intégration continue
- Expertise sur le développement Agile et collaboratif
- Expertise sur les tests (unitaires et de bout en bout).
- Architecture et implémentation d'une plateforme de non-régression de bout en bout.

■ Tâches administratives

- Administration des arrivées/départs sur le projet (cartes securld, mise à jour du wiki, enregistrement des différents comptes – mails, issue tracker, SCM, etc).
- Organisation des visites (identification des besoins, dates, agendas, participants) dans les deux sens (Paris ► Inde et Inde ► Paris). Le but étant d'avoir une visite réussie qui satisfasse les équipes on-site et offshore.
- Gestion des visas

Le rôle de Pivolis a évolué dans le temps : initialement concentré sur la mise en place de l'offshore (pratiques de développement collaboratif, outils collaboratifs, etc), il a évolué vers un double rôle d'amélioration continue de la qualité globale et de gouvernance offshore.

7.3. Les apports de Pivolis-KPIT

Les principaux apports de Pivolis-KPIT sur ce projet sont :

- La fourniture d'une solution globale (offshore, coordination et pilotage)
- Le rôle d' « accélérateur d'offshore »
- La sécurisation du projet
- L'expertise en méthodologies Agiles, via la mise en place de l'AOSD

7.4. Déroulement du projet

7.4.1. Dates clés

Les dates clés du projet sont :

- **Début février 2003 : Phase préalable.** Pivolis commence à travailler chez le client afin de :
 - définir concrètement comment l'intégration de l'offshore se fera (infrastructure, équipes, etc.),
 - commencer pro-activement à choisir les personnes offshore qui participeraient au projet (pour démarrer rapidement le cas échéant et préparer les visas)
 - expliquer l'offshore aux équipes internes et répondre à toutes les questions
 - finaliser les contrats de prestations (coûts des licences, modalités, etc.).
- **Début mars 2003 :** Décision d'utiliser l'offshore pour la montée en charge du projet (et choix de Pivolis et de KPIT comme prestataires)
- **Fin mars 2003 :** Démarrage du projet par un pilote (en partant du projet réel, mais sur un domaine et une durée réduits) avec 4 chefs de projet techniques on-site à Paris pendant les trois premières semaines, rejoints ensuite en offshore par 4 développeurs. On-site, les acteurs apprennent à maîtriser les outils utilisés sur le projet, rencontrent leurs interlocuteurs et commencent à appréhender le domaine fonctionnel. Ils implémentent on-site les premiers « use cases ». Le responsable du développement offshore vient aussi en France une semaine pour prendre connaissance du projet.
- **Fin avril 2003 :** Accompagnement du client en Inde pour rencontrer l'équipe, finaliser le choix des développeurs et continuer à fournir de l'information projet.
- **Mi-mai 2003 :** Fin du pilote.
- **Tous les 2 mois :** Définition d'un nouveau contrat récurrent de régie forfaitaire de 2 mois avec obligation de ressources/qualité et périmètre défini. L'avancement, la qualité et la productivité sont suivis finement par des itérations courtes de 2 semaines.
- **Fin 2003 :** L'équipe offshore a progressivement grossi jusqu'à plus de 60 développeurs (NB : la limite initiale avait été fixée à 20 développeurs).
- **Juillet 2004 :** Fin des développements de la première phase. Démarrage des développements de la phase suivante avec une équipe réduite (30 personnes en offshore).
- **Septembre 2004 :** En parallèle du projet principal, démarrage opportuniste d'un forfait de deux mois (600 jours/hommes) sur la refonte des écrans de l'application. Ce forfait démarre rapidement et bénéficie de l'infrastructure projet déjà en place.
- **Décembre 2004 :** Mise en production de la première phase. Le support est effectué par des personnes on-site et offshore.
- **2005 :** Le développement des phases suivantes continue.

7.4.2. Organisation des équipes

Les contributeurs du projet sont répartis sur deux implantations géographiques : en région parisienne et en Inde, à Pune. L'équipe en France est composée d'une moyenne de cinquante personnes et les effectifs de l'équipe en Inde ont atteint un maximum de soixante-dix personnes pendant certaines phases critiques, avec une moyenne d'une trentaine de personnes depuis le début. Les équipes on-site et offshore travaillent en mode collaboratif au quotidien, Pivolis-KPIT intervenant en tant que coordinateurs et responsables / Gouvernance Offshore entre ces deux équipes. Les équipes en France sont sous la responsabilité d'un responsable du développement qui encadre lui-même cinq chefs de projet fonctionnels suivant le découpage du projet en cinq volets fonctionnels.

L'équipe en Inde est dirigée par un responsable du développement offshore qui chapeaute plusieurs « chefs de projet offshore ». Le responsable en Inde est rattaché au responsable du développement en France. Tous les aspects de planning et de gestion des ressources sont gérés directement par les chefs de projet en Inde et en France.

La figure 15 montre l'organisation initiale avec ses 5 domaines fonctionnels. Nous avons choisi d'avoir des équipes mixtes on-site/offshore pour la plupart des domaines afin de pouvoir plus aisément partager les connaissances fonctionnelles entre les équipes on-site et offshore (figure 16). Par la suite (figure 17), chaque domaine a été séparé (pour être constitué soit uniquement de développeurs on-site soit uniquement de développeurs offshore). La raison principale est que les équipes sont maintenant complètement autonomes et qu'il est plus facile d'avoir une équipe regroupée en un seul lieu géographique. D'autre part, l'équipe offshore n'était initialement pas impliquée dans l'analyse fonctionnelle. Cela a rapidement évolué avec le voyage des chefs de projet indiens en France pour chaque nouveau pan fonctionnel à implémenter. Il est crucial que l'équipe d'implémentation participe à la conception métier (ou analyse fonctionnelle) sous peine d'avoir une productivité beaucoup plus faible. De même, les experts techniques en Inde sont les relais de l'équipe technique en France.

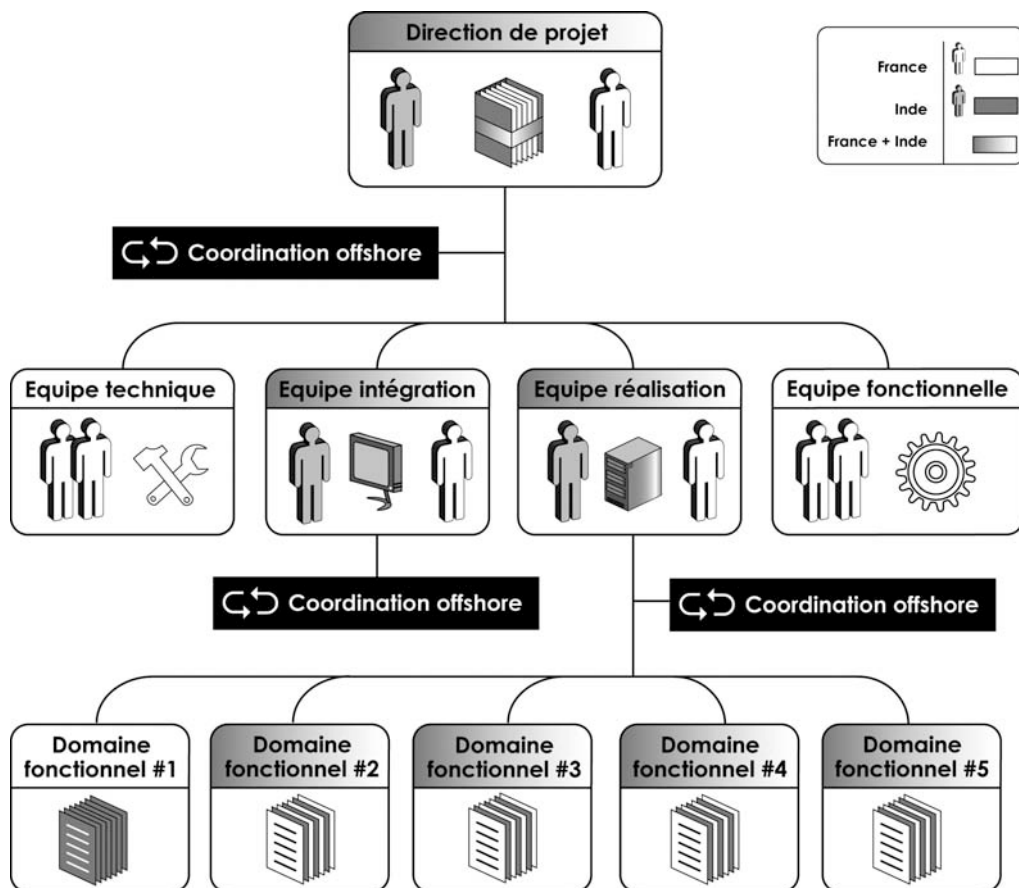


Figure 15 : Organisation générale des équipes, montrant la répartition des rôles on-site et offshore

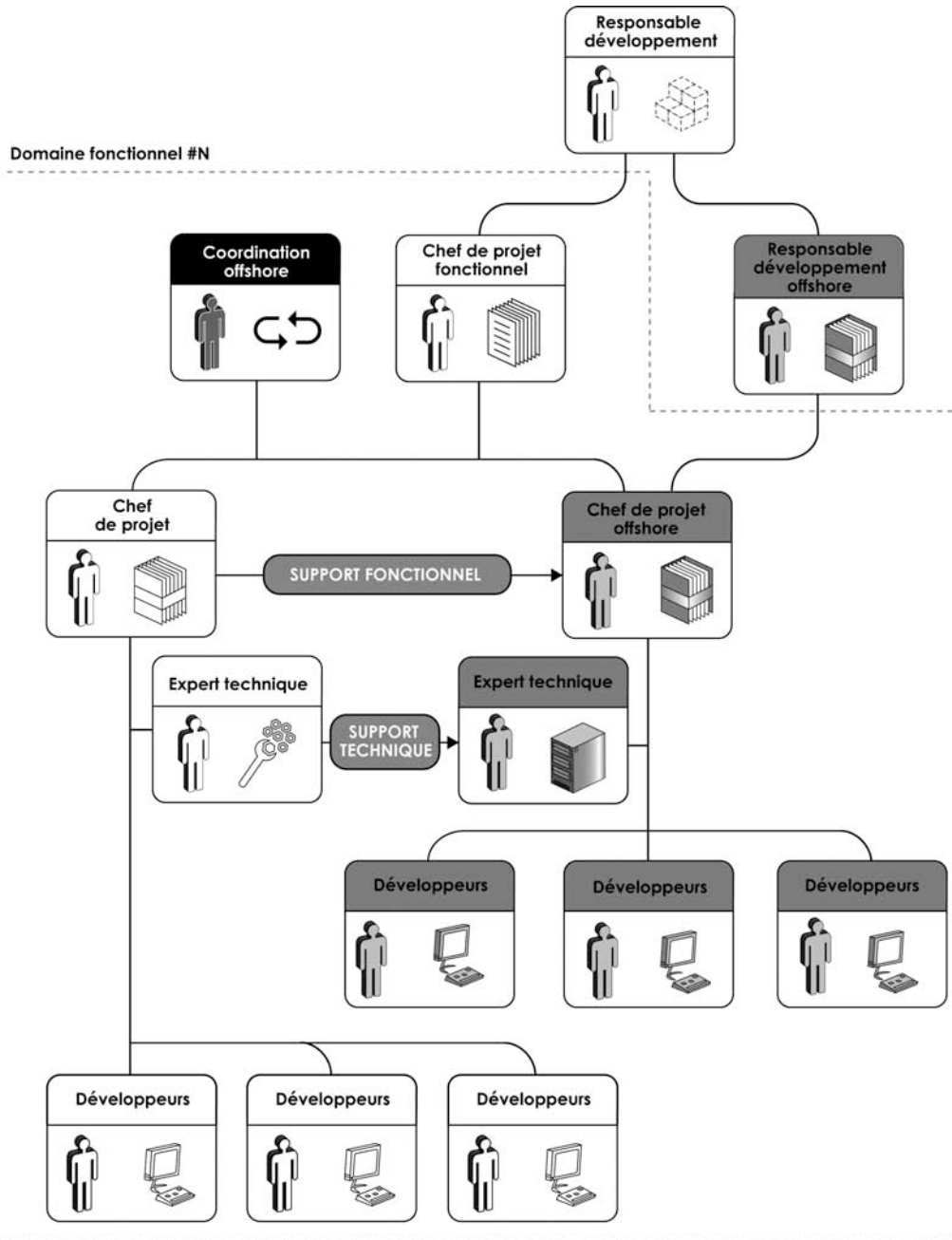


Figure 16 : Exemple d'organisation d'un domaine fonctionnel avec une équipe mixte on-site/offshore

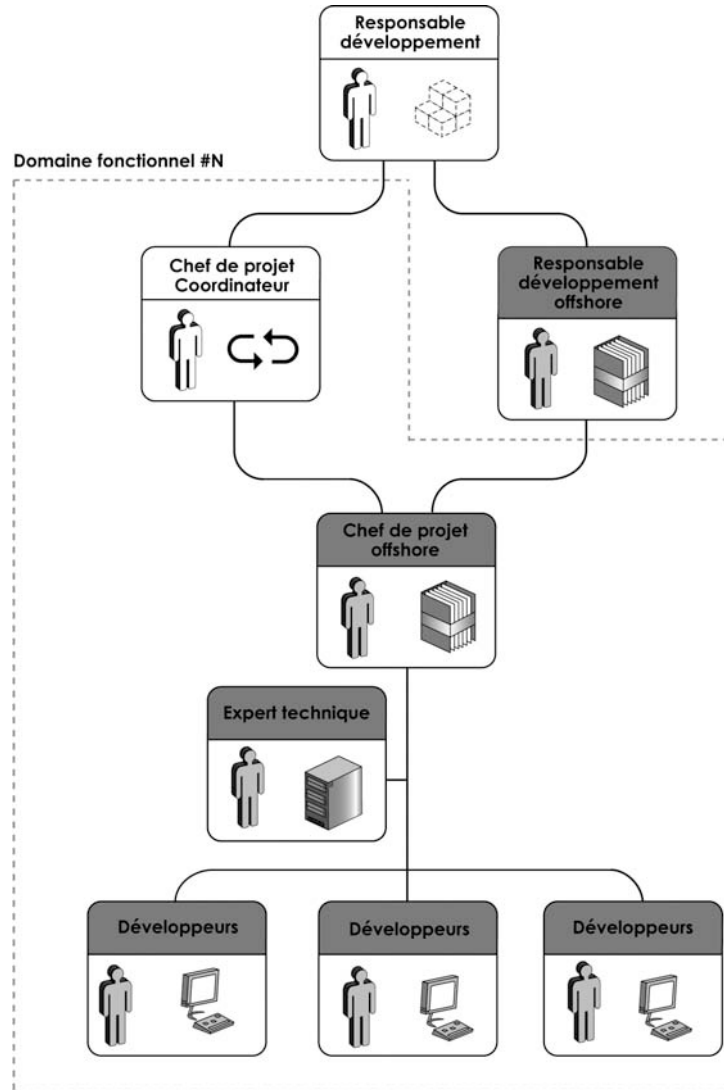


Figure 17 : Exemple d'organisation d'un domaine fonctionnel avec une équipe strictement offshore

7.4.3. Outils de communication

Puisque les équipes travaillent en mode collaboratif, il est indispensable d'assurer une communication directe et fluide, ainsi que l'accès le plus simple possible aux informations concernant le projet. Plusieurs outils de communication ont ainsi été mis en œuvre, suivant la méthodologie AOSD de Pivolis-KPIT.

La messagerie instantanée est l'outil quotidien, utilisée de manière systématique et fréquente pour communiquer entre les équipes on-site et offshore. Les équipes se sont immédiatement appropriées cet outil de communication. Naturellement, l'utilisation du téléphone est indispensable lorsqu'il y a des sujets urgents nécessitant une discussion approfondie, ou encore mettant en présence plusieurs interlocuteurs.

Au démarrage d'un projet, il ne faut pas sous-estimer le fait que les équipes françaises et indiennes seront réticentes à se parler au téléphone du fait de la barrière de la langue. D'expérience, ceci est à éviter à tout prix et il faut vraiment encourager tout le monde à se parler car le téléphone permet de créer un « contact humain » et une empathie qu'il est autrement difficile à établir. C'est un des rôles du coordinateur Pivolis-KPIT que de faciliter la communication, notamment au début, jusqu'à qu'il y ait une bonne compréhension entre les équipes on-site et offshore.

Le deuxième outil est le **wiki**. Son utilisation varie, depuis le trombinoscope avec photos des équipes, jusqu'à la présentation des meilleures pratiques de développement en passant par la documentation des frameworks techniques utilisés et la définition de l'agenda des réunions.

Le suivi du planning et la définition de contenu des livraisons sont assurés grâce à l'outil **JIRA**, un gestionnaire de tâches. Cela permet de disposer à tout moment d'une vision synthétique de l'état d'avancement et des retards éventuels.

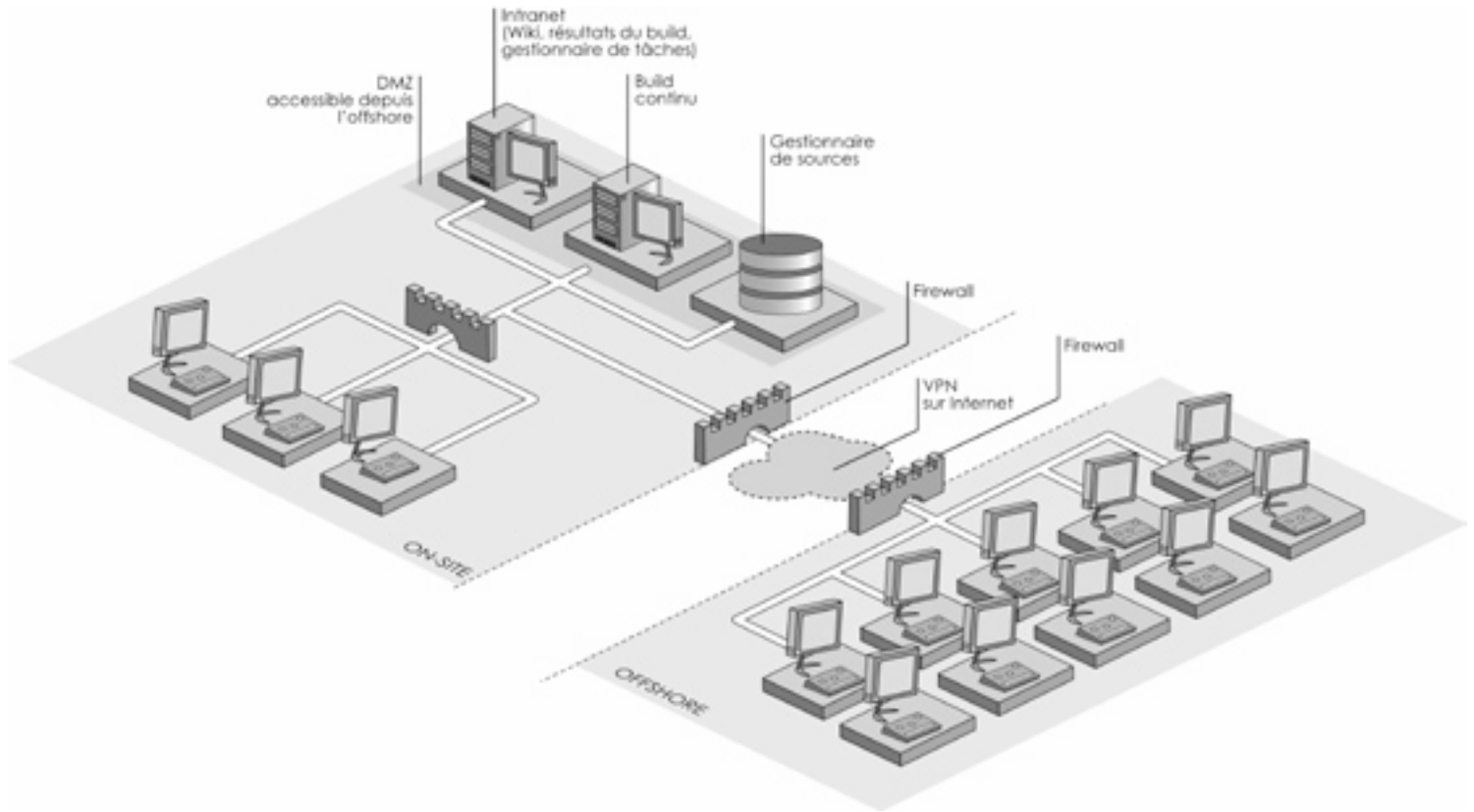
Enfin, le meilleur moyen (mais également le plus cher) pour améliorer la communication est d'organiser des **visites sur site** des deux côtés. L'expérience montre que ces voyages permettent d'établir d'excellentes bases pour la compréhension entre les équipes. En effet, après plusieurs mois de travail, parfois dans des conditions de pression importante, sans se connaître du fait de la distance, des tensions et des incompréhensions peuvent naître entre les personnes, qui sont clairement accrues par le fait de ne pas se connaître. Des voyages réguliers permettent d'éviter ce type de difficultés et contribuent au bon déroulement du projet. Ce projet a par exemple montré qu'un seul voyage en Inde d'un expert technique a complètement changé la perception de l'équipe indienne et de la personne française. De la même façon, une visite en France d'un chef de projet indien lui permet de mieux comprendre les attentes du management et comment les personnes on-site réagissent face aux diverses situations.

7.4.4. Infrastructure technique

Comme le développement se fait en Inde et en France, l'infrastructure technique a été bâtie pour faciliter la collaboration entre les équipes. Elle est également extrêmement sécurisée compte tenu de la nature et des enjeux du projet.

Comme le montre la figure ci-dessous*, il existe des sites informatiques sécurisés en Inde et en France. Une DMZ, localisée à Paris, permet à l'équipe offshore d'accéder à des ressources partagées avec l'équipe française (gestionnaire de sources, serveur de build et intranet). En termes de sécurité, nous avons mis en place un VPN sur Internet avec utilisation de cartes SecurId permettant d'assurer une authentification forte de chaque intervenant. L'équipe offshore possède des locaux entièrement dédiés à ce projet avec accès par badge, LAN séparé, imprimantes séparées, etc.

* : La méthodologie AOSD a été utilisée sur ce projet; il est donc normal de retrouver la même figure que la figure 12 présentée au chapitre sur l'AOSD.



Un outil de gestion de projet et d'intégration continue a également été mis en œuvre par Pivolis-KPIT et les équipes de client : il s'agit de **Maven**. Toutes les heures, Maven exécute les tâches suivantes :

- Extraction du code source du gestionnaire de sources
- Génération des exécutables à partir des sources
- Exécution des tests pour vérifier que le code fonctionne
- Exécution des tests de qualité (Checkstyle, Pattern Tests et tests unitaires JUnit)
- Génération de la documentation projet ainsi que des rapports de tests
- Si le processus de build s'exécute sans erreur, l'application est déployée
- Si le build échoue, un mail est envoyé aux personnes concernées pour qu'ils lancent une action corrective.

Ce système de build est central dans le processus, il permet d'intégrer l'application en continu, réduisant donc considérablement les risques et la durée de la phase d'intégration, en fin de développement.

Les vérifications automatisées de la qualité qui s'exécutent dans le build toutes les heures génèrent en outre la documentation projet, mise à disposition sur le site intranet. Par exemple nous avons un tableau de bord qui récapitule les informations sur le nombre d'erreurs Checkstyle, sur le pourcentage de couverture des tests unitaires par projet en utilisant l'outil Clover, et beaucoup d'autres métriques. Ce tableau de bord peut ensuite être utilisé pour décider où mettre l'effort de qualité lors des itérations suivantes.

7.4.5. Le processus de livraison

Sur ce projet, Pivolis-KPIT a aidé à mettre en place un cycle de livraisons courtes, toutes les deux semaines.

Avant le démarrage de chaque itération, il y a toujours une phase préliminaire. Pendant cette phase, l'équipe étudie le nouveau sujet, identifie les interdépendances et problèmes potentiels et planifie l'itération suivante. Toutes les tâches à effectuer dans cette itération sont saisies dans JIRA et assignées aux différentes personnes de l'équipe. Cela permet à la direction de projet d'avoir une bonne visibilité sur le travail et l'état d'avancement de l'équipe offshore. Une fois terminées, ces tâches sont clôturées par les développeurs puis, après vérification, par les chefs de projets.

Notre méthodologie d'intégration en continu avec des tests automatisés permet de faire des livraisons de qualité. En fait, si la qualité d'une livraison n'est pas jugée bonne, le processus de build échoue et la livraison peut être rejetée. En pratique, la livraison n'est rejetée que rarement.

Outre les tests techniques, une livraison fait également l'objet de vérification de la part d'équipes fonctionnelles et techniques. Les anomalies identifiées sont saisies dans JIRA et corrigées dans la livraison suivante.

A l'issue de chaque livraison (toutes les deux semaines), se déroule une phase parallèle de tests techniques et fonctionnels détaillés. C'est une équipe d'intégration qui les réalise. Au fur et à mesure que le projet progresse l'automatisation de ces tests s'accroît et ils s'exécutent de plus en plus dans le build continu automatisé. Une fois que l'application est jugée de bonne qualité, elle est passée en pré-recette pour des tests utilisateurs.

7.5. Problèmes rencontrés et solutions apportées

7.5.1. Gestion du turnover

Un des premiers problèmes auxquels le projet a été confronté a été le taux important de départs dans les équipes indiennes (après quelques mois), principalement des développeurs expérimentés. Les causes étaient multiples :

- Le marché est en pleine croissance en Inde et il y a donc une demande forte pour les bons développeurs. Des sociétés étrangères s'installent en Inde et perturbent parfois la donne en proposant des salaires largement au-dessus du marché.
- Nous n'avions pas de rôles suffisamment définis et valorisés pour les meilleurs développeurs ; il n'y avait donc pas de progression possible visible pour eux.
- Certaines équipes françaises étaient naturellement réticentes initialement et cette ambiance hésitante s'est fait ressentir côté offshore. De manière générale, l'offshore étant arrivé rapidement sur le projet, il a fallu un temps d'adaptation.
- Le processus de développement était chaotique au démarrage et n'était probablement pas assez bien défini pour éviter des frustrations de part et d'autre.
- Les spécifications détaillées avaient déjà été préparées par les équipes françaises et ne nécessitaient « que » leur transformation en code. Le problème était qu'une bonne partie de ces spécifications était soit

incorrecte soit obsolète et que les équipes indiennes n'avaient pas participé à leur conception. Il était donc difficile d'implémenter le code correspondant. De plus les développeurs expérimentés s'attendaient à faire de la conception détaillée.

- Il y a eu un manque de circulation d'informations projet venant de France, ce qui a frustré les équipes offshore qui voulaient s'intégrer au mieux dans le projet.
- Les critères de recrutement n'étaient pas optimaux et les ressources choisies n'avaient pas toutes les bons profils.

Afin de corriger ceci, nous avons :

- Réparti de manière plus équilibrée les rôles juniors/seniors afin de ne pas avoir trop de postes seniors,
- Amélioré les critères de sélection des membres de l'équipe indienne et introduit des programmes de formation (fonctionnel et technique),
- Introduit le nouveau rôle d'expert technique, pour encadrer des développeurs juniors,
- Fait participer les équipes indiennes de plus en plus à la conception détaillée et à l'analyse fonctionnelle ainsi qu'à la phase d'intégration,
- Introduit la notion de « remise de prix » (traduisant la reconnaissance des pairs et des hiérarchies on-site/offshore), et de primes financières pour les employés afin de les motiver à rester sur le projet,
- Accru les voyages dans les deux sens.

Ces effets ont été très positifs puisque aujourd'hui nous avons depuis plus d'un an un turnover comparable à celui de la France.

7.5.2. Difficultés culturelles et aspects sociaux

Une des plus grandes difficultés au démarrage était de prouver à l'équipe française que la sous-traitance offshore pouvait fonctionner et que la mise en œuvre de modes de travail collaboratifs permettait de communiquer de manière efficace. L'expérience encore faible de l'offshore en France crée nécessairement des attitudes de peur ou de rejet « a priori ». La mise en place de ces modes de travail, et l'échange interactif avec les équipes sur leurs questions et appréhensions a permis de réduire sensiblement ces difficultés.

L'autre difficulté résidait dans les problèmes liés à la distance et aux différences culturelles. Cela est vrai pour n'importe quel projet offshore, mais particulièrement pour un projet avec des équipes françaises et indiennes : non seulement il y a de grandes différences dans la façon de travailler, mais la barrière de la langue complique la communication. La langue officielle sur ce projet était l'anglais, mais l'anglais n'étant la langue maternelle ni des français ni des indiens, cela peut créer parfois incompréhension et confusion. Il existe aussi de grandes différences d'attitude sur des sujets comme le rapport à la hiérarchie. C'est sur ces aspects qu'une coordination on-site/offshore en continu sur le projet devient vite indispensable. Il a fallu par exemple développer l'esprit critique des équipes indiennes, peu habituées à dire ouvertement ce qu'elles pensent, souvent par peur d'être impolies et par peur de sortir de leur place dans la hiérarchie.

L'expérience montre en outre qu'il ne faut pas être chiche sur les coûts téléphoniques et de déplacement entre l'Inde et la France. Cela est absolument nécessaire. De petites réductions de coût sur ces aspects conduisent à terme à des difficultés de communication néfastes en termes de productivité.

7.5.3. Apprentissage de la connaissance fonctionnelle

L'apprentissage fonctionnel a été relativement long. Les équipes indiennes n'avaient pas de connaissances métier préalables dans le domaine concerné et la diffusion de cette connaissance à l'ensemble de l'équipe a pris du temps. Cela a parfois généré des frustrations pour les équipes on-site qui devaient souvent répéter plusieurs fois les mêmes informations.

Enfin sur un domaine riche fonctionnellement, comme cela était le cas sur ce projet, il est souhaitable que l'équipe offshore puisse être associée dès l'amont sur les aspects fonctionnels. Le travail offshore ayant démarré sur ce projet alors que toute la conception et le design avaient déjà été réalisés, l'équipe indienne n'a pas eu la connaissance fonctionnelle ni la vision globale du système. 70% du travail de développement ayant été fait offshore, cela a nécessité un réel effort de transfert de connaissances. Cela aurait pu être amplement simplifié – comme par la suite sur de nouvelles phases du projet – par l'implication des équipes offshore en nombre très limité dès les phases de conception.

L'équipe offshore est aujourd'hui complètement intégrée aux équipes françaises et fait partie du paysage quotidien et habituel du projet. Elle est devenue légitime.

Présentation de Pivolis-KPIT

Pivolis a été créée début 2003, avec pour vocation d'accompagner les grandes et moyennes entreprises françaises dans leur stratégie informatique offshore.

Pivolis est à l'origine une émanation de la société OCTO Technology – un cabinet de conseil en architecture informatique – qui, en 2001, a eu pour mission d'encadrer la réalisation en offshore d'un projet pour le compte d'un établissement financier au Royaume-Uni. C'est ce qui a permis aux futurs collaborateurs de Pivolis d'encadrer pendant 1,5 an des équipes indiennes allant jusqu'à 70 personnes. C'est également à cette occasion que Pivolis a défini et mis en œuvre la **méthodologie AOSD**, comprenant assez rapidement que les méthodologies Agiles présentaient des atouts incontestables pour guider le développement de projets offshore.

A sa création en 2003, Pivolis a choisi de se positionner comme une **société-pivot**. Ce modèle consiste à sélectionner des partenaires, par un processus rigoureux, au sein de pays « offshore » ; puis à proposer aux entreprises clientes le partenaire répondant le mieux aux besoins exprimés.

Après plus de **4 années d'expérience** dans l'informatique offshore, **Pivolis a annoncé fin 2005 qu'elle rejoignait le groupe KPIT Cummins, son principal partenaire indien.**

Pivolis-KPIT garde à son actif différents partenariats dans des pays autres que l'Inde notamment :

- **en Roumanie**, pays qui est apparu à ce jour le plus adapté et le plus mature pour travailler en langue française dans un contexte offshore, et jouissant d'une forte proximité tant géographique que culturelle avec la France.
- **au Maghreb**, qui propose depuis quelques années des infrastructures suffisamment matures pour prendre en charge certains projets, tout en étant parfaitement francophones.

Le modèle de **Pivolis-KPIT** consiste ainsi à prendre totalement en charge la responsabilité du projet auprès de ses clients en France. Ceci afin de présenter à ses clients une direction de projet française, capable de dialoguer avec les équipes locales pour assurer une coordination transparente avec le partenaire offshore, tout cela dans un cadre méthodologique parfaitement maîtrisé.

Avec à son actif l'encadrement de **plusieurs centaines d'années-homme de développement offshore en France**, Pivolis-KPIT se positionne aujourd'hui comme un des leaders indépendants dans ce domaine.



KPIT Cummins

KPIT Cummins Infosystems Ltd est l'une des plus importantes sociétés indiennes de conseil et de services informatiques, classée au 8ème rang d'après le Financial Dataquest Index. Après plus de 15 ans d'existence, elle comptait fin 2005 plus de 2000 collaborateurs, se positionnant ainsi à une taille idéale pour savoir à la fois répondre à une gamme de besoins étendue, tout en apportant à ses clients un service et une proximité tout à fait particuliers. Elle a enregistré un chiffre d'affaires de 56,8 M\$ en 2004 avec une croissance moyenne de 70% depuis trois ans. Fortement présente dans l'industrie, les technologies avancées et les services financiers, KPIT est cotée à la Bourse de Bombay. Au travers de ses filiales et bureaux, KPIT est présent dans 20 pays, notamment aux Etats-Unis, au Royaume-Uni, en France, en Allemagne, au Japon et au Moyen-Orient.

KPIT dispose de 7 centres de développement à Pune, en Inde ainsi qu'un 8ème à Bangalore. KPIT a été certifiée ISO9001 par KPMG en 1997, ainsi que SEI CMM 5 par QAI en 2003. KPIT a reçu l'agrément ISO 9001:2000 en mars 2004.

KPIT a développé de nombreuses compétences dans les technologies traditionnelles (client/serveur, mainframe, AS/400, ...) aussi bien que dans les nouvelles technologies (J2EE et .Net). En termes de solutions applicatives, KPIT est particulièrement présent dans les domaines de l'e-business, l'ERP, la Business Intelligence ou encore Lotus Notes. Depuis 2004, elle a également mis en place une équipe importante dans le domaine de l'informatique technique et de l'externalisation de R&D. KPIT fournit en outre du support 24h/24, 7j/7, 365j/an à de nombreux clients.

KPIT a établi de nombreux partenariats avec des éditeurs de logiciels, notamment SAP, IBM, Business Objects, Oracle, Intenia.

Références

- [1] **«The New Methodology»** par Martin Fowler en juillet 2004 (dernière mise à jour en avril 2003)
<http://martinfowler.com/articles/newMethodology.html>
- [2] **«Méthode agile»** sur Wikipedia.fr
http://fr.wikipedia.org/wiki/Méthode_agile
- [3] **«Manifesto for Agile Software Development»** en 2001
<http://agilemanifesto.org/>
- [4] **«Agile Alliance»**
<http://www.agilealliance.org/>
- [5] **«Agile Project Management, Adaptive Software Development»** par Jim Highsmith
<http://www.adaptivesd.com/>
- [6] **«Cycle en V»** sur Wikipedia.fr
http://fr.wikipedia.org/wiki/Cycle_en_V
- [7] **«Technical Debt»** par Martin Fowler en octobre 2003
<http://www.martinfowler.com/bliki/TechnicalDebt.html>
- [8] **«Software Engineering Economics»**
par Barry Boehm, publié par Prentice Hall PTR en octobre 1981
- [9] **«Extreme Programming Explained: Embrace Change»**
par Kent Beck, publié par Addison-Wesley en 2000
- [10] **«Reexamining the Cost of Change Curve»** par Alistair Cockburn, publié par XP Magazine en septembre 2000 - http://www.xprogramming.com/xpmag/cost_of_change.htm
- [11] **«Examining the Cost of Change Curve»** par Scott Ambler en 2004
<http://www.agilemodeling.com/essays/costOfChange.htm>
- [12] **«PairProgramming.com»**
<http://pairprogramming.com/>
- [13] **«CMMI Web site»** par Carnegie Mellon, Software Engineering Institute
<http://www.sei.cmu.edu/cmmi/>
- [14] **«Extreme Programming from a CMM Perspective»** par Mark C. Paulk, publié par «IEEE Software» en novembre/décembre 2001
<ftp://ftp.sei.cmu.edu/pub/documents/articles/pdf/xp-from-a-cmm-perspective.pdf>
- [15] **«Agile Development and the CMMI: Anti-Matter and Matter or Reconcilable Differences?»** par Steve Ornburn et David Kane en mai 2002
<http://www.stc-online.org/stc2002proceedings/SpkrPDFS/WedTracs/p752.pdf>
- [16] **«Bliki»** sur Wikipedia.fr
<http://en.wikipedia.org/wiki/Bliki>
- [17] **«Agility, Uncertainty, and Software Project Estimation»** par Todd Little
<http://www.macs.ece.mcgill.ca/~radu/304428W03/AgilityUncertaintyAndEstimation.pdf>
- [18] **«FixedPrice»** par Martin Fowler, publié sur son bliki en juillet 2003
<http://martinfowler.com/bliki/FixedPrice.html>
- [19] **«FixedScopeMirage»** par Martin Fowler, publié sur son bliki en septembre 2004
<http://martinfowler.com/bliki/FixedScopeMirage.html>
- [20] **«ScopeLimbering»** par Martin Fowler, publié sur son bliki en octobre 2004
<http://martinfowler.com/bliki/ScopeLimbering.html>

Autres références

- **«Iterative and Incremental Development: A Brief History»** par Craig Larman et Victor R. Basili, publié par «IEEE Computer Society» en Juin 2003 - <http://www2.umassd.edu/SWPI/xp/articles/r6047.pdf>
- **«The Agile Difference for SCM»** par Brad Appleton, Robert Cowham et Steve Berczuk, publié sur CM Crossroads en Octobre 2004 - <http://www.cmcrossroads.com/ubbthreads/showflat.php?Cat=&Number=34375>

Remerciements

Nous remercions chaleureusement les architectes d'OCTO Technology qui nous ont aidés à relire ce livre blanc, et tout particulièrement Laurent Henriet et Lionel Thouvenot.

Nous dédions ce livre aux membres de l'équipe projet avec laquelle nous avons travaillé lors de l'expérience décrite au chapitre 7.

Auteurs

Vincent Massol, assisté de l'équipe Pivolis-KPIT.

Vincent Massol peut être contacté à travers son blog (<http://www.massol.net>).



Dépôt légal : Janvier 2006
Imprimé pour Pivolis-KPIT chez
PORTAPRINT - 2, rue du Blanc Seau
59334 - Tourcoing CEDEX
N° de dossier : 200509.0225

