Amélioration de la réalisation de projets de développement logiciel par l'exécution de pratiques agiles

Jérôme Martin

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

Centre régional associé de Poitou-Charentes

MEMOIRE
présenté en vue d'obtenir
le DIPLOME D'INGENIEUR

Spécialité : Informatique

Option : Architecture et Ingénierie des Systèmes et des Logiciels

Proposé par

Jérôme MARTIN

Amélioration de la réalisation de projets de développement logiciel par l'exécution de pratiques agiles

soutenu le 13/12/2013

JURY

PRESIDENT :
- Stéphane NATKIN – Professeur CNAM Paris

MEMBRES :
- Marie-Christine LAFAYE – Responsable de la filière informatique CNAM Poitou-Charentes
- Annick LASSUS – Professeur agrégé en Economie – Gestion option Systèmes d'Information – IUT de la Rochelle
- Gilles LASPLACETTES – Secrétaire Général du CNDP
- Isabelle PARRA – Responsable de la Division des systèmes d'information du CNDP

Remerciements

Je remercie Madame Lafaye, responsable de la filière informatique du CNAM Poitou-Charentes pour ses relectures, ses conseils et son soutien dans la rédaction de ce mémoire. Je remercie également tous les relecteurs qui ont participé à l'amélioration de ce document.

Je remercie Monsieur Henri Kighelman, Directeur Adjoint du CNDP, qui m'a accompagné tout au long de l'exécution du projet CCR. Nos échanges ont été, pour moi, très enrichissants. Je remercie mes collègues du ministère, Alain Thillay et Jean-Michel Vite pour la confiance qu'ils m'ont accordée au cours du projet et pour les marques de reconnaissance qu'ils m'ont adressées.

Je remercie Madame Isabelle Parra, responsable de la DSI du CNDP pour m'avoir laissé l'autonomie nécessaire et pour m'avoir laissé expérimenter des pratiques agiles.

Bien sûr, j'adresse mes remerciements aux développeurs qui ont participé au projet. Nous avons vécu quelques moments stressants mais nous avons tenu le cap. Je n'oublie pas l'équipe de développeurs toute entière car elle fait preuve d'enthousiasme sur les pratiques au quotidien. L'agilité ne peut pas s'imposer, il faut la comprendre et avoir envie de changer sa manière de faire, sa manière d'être et les responsabilités qui vont avec. C'est parfois un vrai défi mais l'équipe le relève tous les jours.

Je ne peux pas citer ici tous mes collègues des services métiers mais je voudrais remercier tous ceux qui aujourd'hui expérimentent et partagent avec nous ces pratiques. Il y a de l'enthousiasme, de la curiosité et du plaisir retrouvé autours d'elles. C'est très encourageant pour tout le monde.

De manière générale, je suis reconnaissant, envers celles et ceux qui m'ont fait confiance et m'ont offert la chance d'évoluer dans mon métier. C'est pour cette raison que je remercie aussi les agilistes dans leur ensemble. L'agilité est passionnante à étudier et encore plus à pratiquer. Elle a, profondément changé ma vision du métier et elle m'a aidé à trouver une posture managériale adaptée dans laquelle je me sens efficace.

Enfin, je remercie ma compagne pour son soutien et pour avoir supporté mes très nombreuses indisponibilités au cours de ces six dernières années.

Abréviations

CCR	Catalogue Chèque Ressource
CMS	Content Management System
CMMI	Capability Maturity Model Integration
CNDP	Centre National de Documentation Pédagogique
CRDP	Centre Régional de Documentation Pédagogique
DGESCO	Direction Générale de l'Enseignement SCOlaire
DSI	Division des Systèmes d'Informations
GAP	Gestion des Activités et des Projets
GED	Gestion électronique de documents
IEEE	Institute of Electrical and Electronics Engineers
INRDP	Institut National de Recherche et de Documentation Pédagogique
MEN	Ministère de l'Éducation Nationale
MESR	Ministère de l'Enseignement Supérieur et de la Recherche
OFRATEME	Office Français des Techniques Modernes d'Éducation
PESI	Plan d'Évolution du Système d'Informations
PMI	Project Management Institute
PMBOK	Project Management Body of Knowledge
SCEREN	Service Culture Édition Ressources pour l'Éducation Nationale
TICE	Technologies d'Information et de Communication pour l'Enseignement

Glossaire

Back-office

En développement logiciel, on désigne par le terme Back-Office les éléments logiciels qui permettent de gérer des activités métiers dont le résultat est en général exploité et présenté sur des éléments logiciels en front-end.

Backlog

Le backlog est un référentiel d'exigences sur le produit. Il regroupe toutes les *stories* à implémenter dans un produit. On parle alors de *backlog* du produit. Lorsqu'il s'agit d'un sous ensemble de *stories* implémentées au cours d'une itération, on utilisera le terme *backlog* de *sprint*.

Big Design Up Front

Le Big Design est une approche de conception *top-down* ou approche descendante. L'écriture du code ne commence pas tant que la conception n'est pas documentée et détaillée.

Bottom-up

En ingénierie, l'approche *bottom-up*, également appelée approche ascendante, vise à construire un système en s'appuyant de manière itérative, sur des bases établies.

Buffer

En génie logiciel, un *buffer* est une zone mémoire tampon qui sert à stocker de l'information de manière temporaire en attente de traitement.

Bug tracker

Un *bug tracker* est un logiciel permettant de gérer les défauts d'un système ou d'un produit communiqués par les utilisateurs. Ce type de logiciel permet de configurer un flux de travail spécifique.

Commit

En informatique, un *commit* est une opération de validation. Dans les bases de données relationnelles, cette opération valide les ordres de mise à jour au sein d'une transaction. Dans les logiciels de gestion de versions le *commit* valide dans le dépôt un changement (une révision) intervenu sur le code.

Daily meeting

Le *daily meeting* est une réunion quotidienne ou toute l'équipe se réunit pour communiquer sur l'avancement des activités du projet. Les membres d'équipes s'expriment à tour de rôle sur ce qu'ils ont fait la veille et sur ce qu'ils vont faire dans la journée qui commence. Cette réunion porte également le nom de *daily stand-up* ou encore de *daily scrum*. Le terme français couramment employé est la « mêlée quotidienne ».

Framework

En génie logiciel, un *framework* est une organisation structurée des différentes composantes du code source. Il guide l'architecture logicielle en implémentant certains patrons de conception ou *design pattern,* par exemple le patron Modèle-Vue-Contrôleur. En méthodologie, il définit l'organisation d'un ensemble de pratiques, de rôles et de responsabilités .

Front-end

Il s'agit du terme utilisé pour désigner les éléments logiciels visibles par les utilisateurs finaux d'un service informatique. En développement web, le *front-end* désigne le site web visible par les internautes. Ce même site web peut être alimenté et géré grâce à un *back-office* visible uniquement par certains utilisateurs (administrateurs, gestionnaires de contenu, gestionnaires de communautés...).

Mockup

En ingénierie logicielle et plus particulièrement en programmation un *mock* est un objet utilisé pour prototyper un système. Un *mockup* est un élément de prototypage de l'interface utilisateur. Il reproduit l'interface de manière schématique avec un niveau très faible d'interaction.

Product owner

Le *product owner* est un rôle dans le *framework* Scrum. Il désigne le responsable du produit. C'est le représentant des utilisateurs. C'est lui qui porte la vision du produit auprès des développeurs. Il a pour responsabilité le développement du produit dans un processus piloté par la valeur.

Scrum master

Le *Scrum master* est le garant de la bonne exécution du processus Scrum. Ce n'est pas un chef de projet. C'est un facilitateur. Il lève les obstacles qui empêchent l'équipe d'avancer. Il aide le *product owner* dans ses tâches de définition du produit.

Sketching

Le *sketching* est l'activité qui consiste à concevoir des *mockups*. On parle également de *wireframing*.

Sprint

Dans le processus Scrum, un *sprint* est une itération dans laquelle toutes les activités de développement sont menées en parallèle. Il dure de 2 à 4 semaines. Il commence par une séance de planification et se termine par une revue et une rétrospective.

Sprint planning meeting

Dans le processus Scrum, un *sprint planning meeting* est une réunion d'estimation et de planification pour le *sprint* qui démarre. Il s'agit de déterminer le contenu du s*print,* autrement dit les activités qui devront être menées pour atteindre l'objectif du *sprint*.

Sprint review

Une *sprint review* ou revue de sprint est une réunion qui clôture un *sprint*. Elle permet aux développeurs de livrer le travail réalisé pendant le *sprint*. Une démonstration du produit est organisée à destination du *product owner.*

Sprint retrospective

La *sprint retrospective* ou plus simplement rétrospective est la deuxième réunion de clôture du *sprint*. C'est une séance de travail au cours de laquelle l'équipe s'interroge sur les pratiques exécutées en cours de *sprint*. Elle juge de l'opportunité d'ajouter, de supprimer ou d'améliorer ses pratiques. C'est l'incarnation du principe d'amélioration continue.

User story

La *user story* est un formalisme d'expression du besoin fonctionnel orienté utilisateur. L'implémentation des *user stories*, est la principale mesure d'avancement d'un projet agile de développement logiciel.

Table des matières

Introduction

Le Centre National de Documentation Pédagogique (CNDP) est un établissement public sous tutelle du ministère de l'Éducation Nationale. Il a entre autres missions, l'édition et la diffusion de ressources pédagogiques. Aujourd'hui, dans un environnement impacté par des évolutions technologiques fortes, contraint par des politiques de réduction des dépenses publiques, et évoluant dans un contexte concurrentiel important, l'établissement vit une mutation importante de son histoire. Le CNDP se repositionne stratégiquement sur ses missions conformément aux enjeux présentés par sa nouvelle direction générale. L'établissement redéfinit ses modes de fonctionnement. Les processus sont revisités et l'organisation s'aligne petit à petit.

Par ailleurs, le PESI (Plan d'Evolution du Système d'Information) qui comporte plusieurs projets d'envergure, comme le projet GAP (Gestion des Activités et des Projets), participe à cette mutation. C'est en partie dans le cadre de ce plan que l'établissement s'interroge, redéfinit ses pratiques et fait évoluer son organisation. Dans le cadre du projet GAP, l'établissement s'est doté de l'outil de gestion de projets Sciforma et a mené un travail introspectif sur ses pratiques.

En 2011, l'équipe de développeurs a eu l'occasion d'expérimenter des pratiques agiles dans le cadre du projet « Catalogue Chèque Ressources (CCR) » commandé par notre tutelle. Cette expérimentation a permis d'obtenir un premier retour d'expérience riche d'enseignements. Par conséquent, étendre le déploiement de pratiques agiles à l'ensemble des projets pourrait permettre à l'établissement de les exécuter à l'avenir avec plus d'efficacité, tout en garantissant le niveau de qualité requis. Parallèlement au projet GAP, et à cette première expérience de l'agilité, deux autres problématiques ont également amené les développeurs à s'interroger sur les pratiques exécutées. La première concerne le suivi des demandes de développement au travers d'un outil dédié. La deuxième concerne l'incorporation des intégrateurs web au sein de l'équipe de développeurs. Aujourd'hui, le déploiement de pratiques agiles est engagé et se poursuit au sein de la DSI. Mais il dépasse aussi le périmètre de cette entité en influençant les processus partagés avec d'autres services, et en particulier les pratiques liées à l'expression du besoin.

Dans ce mémoire, je tenterai de démontrer les atouts des pratiques agiles et des outils associés dans l'amélioration de l'exécution de nos projets mettant en œuvre des activités de développement logiciel.

Dans une première partie, nous abordons le contexte dans lequel j'évolue depuis le printemps 2009. Je présente l'établissement et ses missions, la Division des Systèmes d'Information et les technologies qu'elle met en œuvre. Je fais également un état des lieux du processus pour les projets numériques du CNDP.

Dans une deuxième partie, je présente les différents modèles de cycle de vie ainsi que l'agilité et les méthodes agiles les plus utilisées dont sont issues les pratiques déployées au CNDP.

Je présente ensuite dans une troisième partie, la première expérience des développeurs dans l'exécution de pratiques agiles. Cette première expérience s'est déroulée dans le cadre du projet « Catalogue Chèque Ressources ». J'explique quelles étaient les pratiques exécutées et

quels enseignements nous en avons retenu.

Dans une quatrième partie, je présente les pratiques que nous exécutons depuis l'été 2012 ainsi que les indicateurs que nous avons choisis pour mesurer la qualité d'exécution de nos activités sur les projets qui nous sont confiés. Je présente également les outils qui viennent en support de ces activités et l'utilisation que nous en faisons.

Enfin, dans une cinquième partie, je présente une projection possible sur l'organisation ainsi que des pistes d'évolutions de notre processus projet par l'adoption et l'institutionnalisation de pratiques.

1. Le contexte

Cette première partie présente le contexte dans lequel j'ai effectué le travail décrit dans ce mémoire. Il est important de comprendre quel est l'environnement des équipes qui interviennent sur la réalisation des projets, et sous quelles contraintes structurelles, organisationnelles et managériales elles exécutent leurs missions. Plus globalement, le CNDP exerce ses missions sous l'influence de contraintes externes et internes majeures. De plus, son environnement et son marché sont concurrentiels et en évolution. L'établissement lui-même est actuellement en pleine réorganisation. L'équipe de développeurs de la Division des Systèmes d'information intervient dans ce contexte sur la réalisation des projets numériques qui requiert des compétences en génie logiciel. Elle met en œuvre des technologies diverses sur une typologie de projets hétérogènes.

1.1. Présentation de l'établissement et de son environnement

Le Centre National de Documentation Pédagogique (CNDP) est un établissement public à caractère administratif sous tutelle du ministère de l'Éducation Nationale.

Le CNDP est éditeur multisupports (audiovisuel, web, imprimé) de ressources pédagogiques à destination du monde de l'enseignement. C'est également un acteur majeur de l'ingénierie éducative et documentaire sur les domaines des TICE, des arts, de la culture. Il apporte, sur ces domaines, son expertise et développe les actions de conseil et de formation à destination des différents acteurs de l'Éducation Nationale.

Dans ces domaines, le CNDP est également opérateur et prestataire de services pour le ministère de l'Éducation Nationale (MEN) et pour le ministère de l'Enseignement Supérieur et de la Recherche (MESR).

Le CNDP est la tête du réseau Scéren. Ce réseau a pour mission de produire et diffuser des ressources pédagogiques destinées à renforcer l'action des enseignants et des établissements en faveur de la réussite des élèves.

1.1.1. Historique de l'établissement

1932 : Création du Centre National de Documentation Pédagogique.

1950 : Officialisation du nom de « Centre National de Documentation Pédagogique » (CNDP)

1955 : Début de création des Centres Régionaux (CRDP) et départementaux (CDDP).

1970 : L'établissement est scindé en deux. D'un côté l'OFRATEME qui a un rôle d'édition de ressources pédagogiques et de l'autre l'INRDP qui a un rôle de documentation, de recherche pédagogique et de tutelle des centres régionaux et départementaux.

1976 : Un décret modifie la répartition des rôles entre les deux établissements ainsi que leur nom : l'OFRATEME redevient CNDP et reprend la mission de documentation et donc la tutelle des CRDP et des CDDP. L'INRDP devient l'Institut National de Recherche

Pédagogique (INRP).

1986 : Début de l'informatisation des fonds documentaires des CRDP qui apportent, en outre, une assistance technique à l'informatisation des Centres de Documentation et d'Information des établissements d'enseignement (CDI), accompagnent l'équipement et le développement des TICE dans les établissements, et mettent au point divers thesauri (dont Motbis, à partir de 1987). Les CRDP se consacrent à la formation initiale des documentalistes de CDI, à la formation continue des enseignants, à l'animation pédagogique, à l'aide aux établissements en matière de projet éducatifs et d'informatique et à l'élaboration de produits pédagogiques d'intérêt régional.

1992 : Les CRDP acquièrent leur autonomie

2002 : Le décret du 19 avril précise les missions du CNDP et des CRDP. Cet ensemble est désigné sous le terme « Scéren » : Service Culture Édition Ressources pour l'Éducation Nationale.

2003 : L'arrêté du 27 juin entérine la délocalisation du CNDP, dont le siège social est désormais implanté à Chasseneuil-du-Poitou, dans la Vienne.

1.1.2. Missions et périmètre

D'après le décret du 19 avril 2002 le CNDP exerce les missions suivantes.

- Édition et production de ressources pédagogiques, dans tous les domaines de l'éducation.
- Veille et développement des usages en France et à l'étranger.
- Promotion et développement des technologies de l'information et de la communication
- Promotion et développement de l'éducation artistique et de l'action culturelle.
- Animation des centres de documentation et d'information au sein des établissements d'enseignement.
- Formation des enseignants
- Coordination des centres régionaux de documentation pédagogique

Dans l'exercice de ses missions, l'établissement peut :

- Concevoir, distribuer et vendre des produits ou des services liés à ses activités.
- Assurer des prestations d'ingénierie et de conseil.
- Acquérir ou exploiter des droits de propriété intellectuelle.
- Attribuer des subventions conventionnées aux organismes qui collaborent à l'exercice de ses missions.
- Coopérer avec les organismes étrangers et internationaux compétents en matière de documentation pédagogique.
- Participer à des groupements d'intérêt public, à des groupements d'intérêt économique, y compris européens.

- Prendre des participations ou créer des filiales.
- Etre chargé de la production et de la diffusion des publications administratives et juridique du ministère de l'Éducation Nationale.

Le CNDP exerce ses missions pour les acteurs de l'enseignement du premier et second degré mais aussi pour les acteurs du milieu universitaire.

1.1.3. Les contraintes

Les activités du Centre sont menées sous des contraintes budgétaires et structurelles importantes. La composition de la carte budgétaire du CNDP est en évolution et n'échappe pas à la MAP (Modernisation de l'Action Publique). Cette contrainte budgétaire forte pousse l'établissement à compresser les effectifs puisqu'il lui faut rendre des postes au Ministère. Aussi pousse-t-elle peu à peu l'établissement à se repositionner stratégiquement sur son métier et ses missions, à trouver des partenariats mais aussi à mener un travail introspectif sur ses processus de production et sur l'efficacité d'exécution des projets. Les chiffres du tableau ci-dessous (figure 1) illustrent la contrainte budgétaire de l'établissement.

Année	Subventions de l'état (en K€)	Carte budgetaire (en K€)	Dépenses de fonctionnement (en K€)
2008	27551	20038	40079
2009	27064	20830	39730
2010	26016	16427	34083
2011	24993	16915	NC

Figure 1 : Tableau de présentation du budget du CNDP

En plus de cette contrainte budgétaire, l'établissement a dû trouver son rythme de croisière depuis 2009. L'année 2009 a marqué la fin de la délocalisation du CNDP sur le site de Chasseneuil-du-Poitou (voir figure 2). L'exécution des missions du CNDP a été impactée tout au long de cette délocalisation entre 2004 et 2009. Ces dix dernières années le CNDP a connu quatre directeurs généraux, certains services ont été restructurés et certaines missions abandonnées.

1.1.4. Le réseau Scéren

Le CNDP pilote le réseau SCEREN qui est composé de 31 CRDP (Centres Régionaux de Documentation Pédagogique), 85 CDDP (Centres Départementaux de Documentation Pédagogique) et 70 CLDP (Centres Locaux de Documentation Pédagogique) répartis sur tout le territoire.

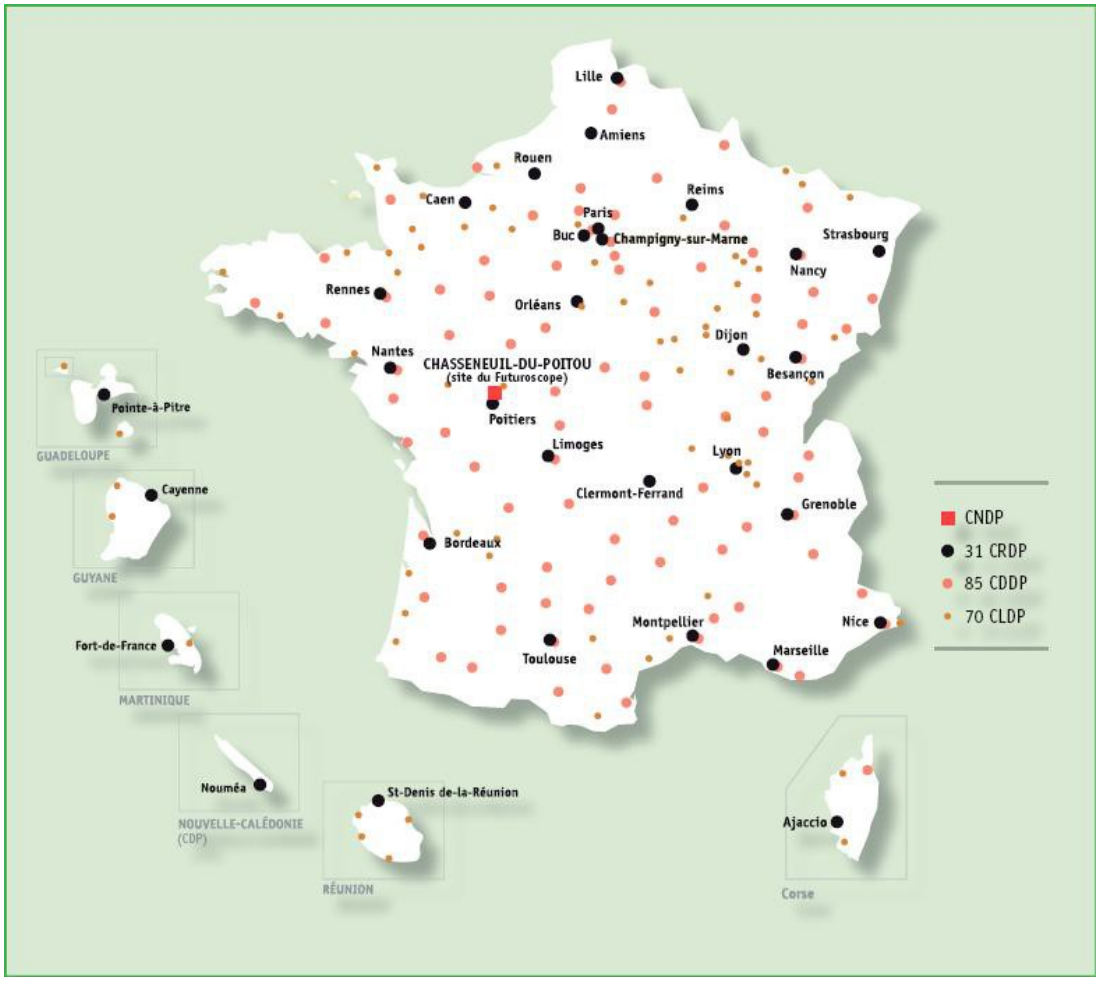


Figure 2 : Carte des établissements du réseau Scéren

1.1.5. La tutelle

Le CNDP est un établissement sous tutelle du ministère de l'Éducation Nationale. Plus précisément, l'établissement exerce ses missions sous l'autorité de la Direction Générale de l'Enseignement SCOlaire (DGESCO) du ministère. Si la direction de l'établissement peut être amenée, en fonction des dossiers, à travailler en collaboration avec le cabinet du ministre ou son secrétariat général, c'est pour la sous-direction des programmes, de la formation et du développement numérique de la DGESCO que les personnels du CNDP exercent leurs missions sur les projets qui leurs sont confiés.

Le CNDP en fonction des dossiers intervient soit en tant qu'opérateur d'un service, soit en tant qu'hébergeur, soit dans la conception et la réalisation d'un produit ou service. Il est parfois sollicité sur ces trois dimensions pour un même projet. Les domaines de compétences mobilisées sont l'édition et l'ingénierie documentaire. Les projets menés pour la DGESCO étant principalement numériques, les compétences informatiques sont donc également mobilisées.

1.1.6. Le marché de l'édition de ressources pédagogiques

Le marché de l'édition scolaire est un marché très concurrentiel dont aujourd'hui quelques éditeurs se partagent la plus grosse part. Il représente environ 9 % de l'édition en France et se chiffrait, en 2009, à environ 240 millions d'euros. L'édition papier est très largement majoritaire et même si elle subit de plein fouet la concurrence d'Internet, la part du numérique est aujourd'hui négligeable sur le marché. Le numérique est considéré comme un domaine en prospective. Malgré tout, aujourd'hui, les ressources papiers sont en général proposées avec du contenu numérique soit sur cd-rom soit en ligne sur Internet. Le marché est structuré de la manière suivante : les manuels scolaires pour les élèves sont édités par le secteur privé tout comme les manuels pour enseignants. Mais l'édition publique peut également éditer des ressources. Au travers du réseau Scéren, elle assure donc sa mission de service public en éditant des ressources pédagogiques pour des domaines confidentiels qui ne seraient pas rentables pour le privé. Cet équilibre historique [ARC01] est bouleversé depuis une quinzaine d'années. L'avènement d'internet remet en question l'entente entre les éditeurs et les enseignants. Ces derniers peuvent de plus en plus aisément se passer des manuels pour préparer leurs cours. Ils peuvent produire, échanger, trouver, diffuser de la ressource eux-mêmes, le tout dans un mode collaboratif grandement facilité par les outils de communication. Concernant les manuels scolaires, les éditeurs privés peinent à trouver des modèles économiques viables. Les manuels numériques des éditeurs privés ne s'imposent pas encore puisqu'ils ne représentent actuellement que 0,6% du marché. De plus, il est possible de trouver des manuels gratuits sur Internet pour l'enseignant et pour l'élève. Ces manuels sont souvent produits par des enseignants.

Les manuels scolaires du collège sont financés par les départements. Les régions financent les manuels pour le lycée. Ceux du primaire sont financés par les mairies. L'enseignant a un rôle de prescripteur. Le CNDP, sans abandonner l'édition papier, se positionne de plus en plus sur le numérique. Il produit des ressources mais travaille également sur les aspects de la valorisation et de la diffusion au travers des plates-formes d'hébergement.

1.1.7. Les 8 objectifs de la nouvelle direction

En novembre 2011, le CNDP a accueilli un nouveau directeur général. Depuis, ce dernier a eu l'occasion à plusieurs reprises de communiquer sa stratégie pour les prochaines années.

Cette stratégie s'articule autour de huit objectifs clés :

- Redéfinir la chaîne éditoriale et la production des contenus tout en facilitant leur référencement et leur identification.
- Développer la convergence des outils et l'interopérabilité.
- Redéfinir la politique "réseau" du Scéren en identifiant les atouts de ses composantes.
- Développer les usages du numérique.
- Équilibrer l'environnement de l'établissement en développant des partenariats avec le secteur privé et une relation de qualité avec la tutelle ministérielle autour des projets porteurs des politiques éducatives.
- Développer de nouveaux modèles économiques basés sur les partenariats.

- Identifier, développer et manager des communautés éducatives.
- Adapter le management de l'établissement au mode projet et favoriser la créativité et l'innovation.

Ces objectifs permettent d'appréhender l'importance de l'amélioration des processus et des pratiques de gestion de projet et de génie logiciel. L'enjeu plus général pour l'établissement est d'obtenir, d'une part la reconnaissance de son expertise sur des domaines objectivés, et d'autre part, d'obtenir la reconnaissance de sa capacité à mener à bien des missions de service public cadrées par des contraintes de qualité, de délais et de coûts. Pour atteindre ces objectifs, le CNDP tente d'acculturer, davantage encore, les personnels à la gestion de projet au travers de son programme de formation continue. Enfin, les objectifs cités ci-dessus sont portés par un projet d'envergure qui concerne le réseau Scéren tout entier. Le nom de ce projet est Scéren2017.

1.1.8. L'organigramme du CNDP

Le CNDP est composé d'une direction générale, d'un secrétariat général et de services métiers et supports. Les directions et départements métiers sont sous la responsabilité de la direction générale alors que les fonctions supports sont sous la responsabilité du secrétariat général. L'organigramme présenté ci dessous (figure 3), est celui qui était en vigueur au début de la rédaction de ce mémoire. Dans le cadre du projet Sceren2017, l'établissement a vécu une réorganisation de ses services métiers au cours de l'année 2013. Celle-ci visait entre autre à fusionner les directions de la production et de l'édition. Cette nouvelle organisation n'est pas présentée puisqu'elle ne caractérise pas la période de référence (2010 à 2012) sur laquelle portent les observations contenues dans ce mémoire. L'organigramme de la nouvelle organisation est fourni en annexe 1. En dehors de cette restructuration, les intégrateurs web de la direction de la production ont rejoint l'équipe de développeurs de la DSI à la fin du mois de décembre 2012. Sur les projets numériques, la direction de la production qui, historiquement, intervenait sur le développement logiciel, intervient aujourd'hui uniquement en amont sur la conception graphique et en aval sur les contenus.

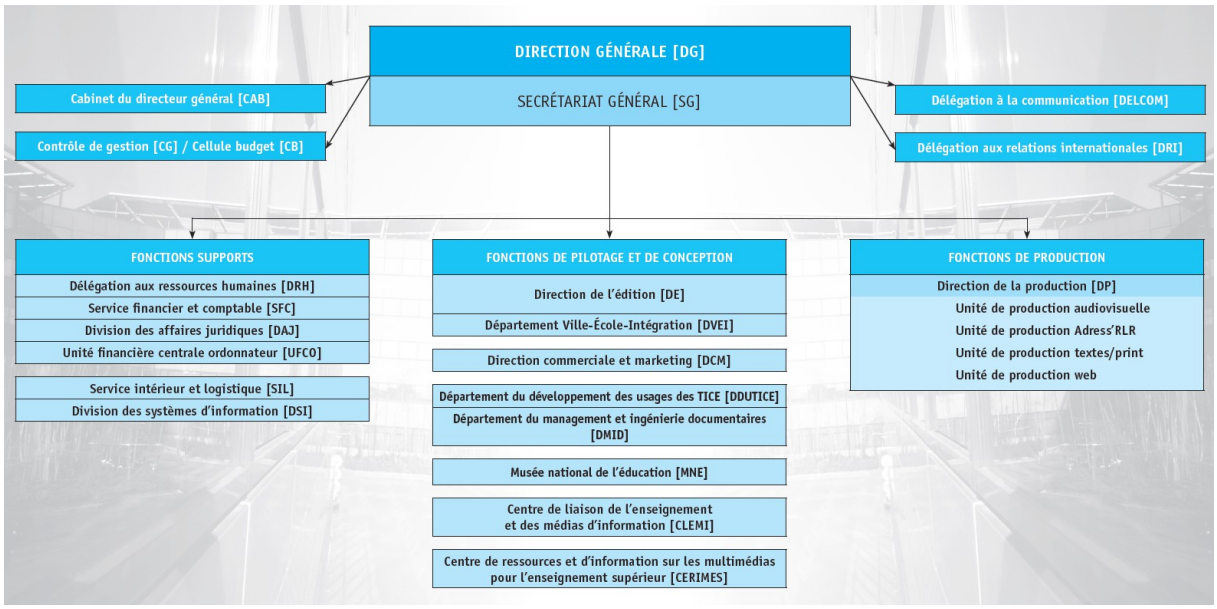


Figure 3 : Organigramme du CNDP

Depuis l'automne 2012, une cellule fonctionnelle a été créée ainsi qu'une cellule de chef de projets. Ces deux cellules sont, pour le moment, sous la responsabilité du secrétariat général. La première prend en charge le soutien fonctionnel sur les applications de gestion et les plates-formes développées dans le cadre des projets dits « structurants ». La seconde prend en charge la conduite de ces projets. Ce type de projets concentre tous les domaines d'expertise de l'établissement. Ils sont structurants pour l'établissement car au travers des compétences qu'ils mobilisent, ils favorisent l'adoption d'un mode de coopération transverse. Les projets plus classiques sont conduits par des chefs de projets répartis dans les différents départements et services.

1.2. La gestion de projet au CNDP

Le CNDP mène environ 60% de son activité au travers de projets divers portés par les différents départements. Les projets sont directement liés aux missions attribuées à l'établissement par le décret du 19 avril 2002 et décrites au paragraphe 1.1.2.. On retrouve au travers de ces projets, l'implication de l'établissement sur la production et l'édition de ressources à destination de la communauté éducative ainsi que le travail de veille et de promotion des technologies d'information et de communication.

Remarque : La période de référence utilisée pour la rédaction de ce mémoire porte sur les années 2010 à 2012. Cependant, j'ai choisi dans cette partie dédiée à la gestion de projet au CNDP, de présenter les changements qui s'opèrent depuis l'automne 2012. J'aborde ces changements au travers de la présentation des processus et des instances. Ailleurs le nom des départements et services est celui de l'organigramme en vigueur sur la période de référence.

1.2.1. Les types de projet

Le CNPD mène des projets de nature et d'envergure assez hétérogènes. Ces projets peuvent être classés dans les groupes suivants.

- Les projets de l'édition.

  Ils sont portés et pilotés par la direction de l'édition. Ces projets visent à produire et diffuser des ressources à destination de la communauté éducative ou de l'administration. Ces ressources peuvent être de différentes natures. Il peut s'agir de livres, de manuels, de brochures, de fascicules, de revues (TDC, l'École Numérique), d'applications mobiles, d'applications ou de sites web, d'applications logicielles de type desktop, de services de contenu en ligne. Ces ressources peuvent être des commandes du ministère de l'Éducation Nationale ou d'un autre ministère (culture, agriculture etc.) ou bien simplement être le résultat d'une idée interne à la direction de l'édition.

- Les projets de la tutelle.

  Ce sont les projets de plus grosse envergure. Ils sont souvent l'expression directe de décisions et d'orientations prises par la tutelle. Il s'agit cette fois principalement de

projets numériques (École Numérique Rurale, Catalogue Chèque Ressource, P@irformance, Réseau Professionnel des Enseignants, etc.). Ces projets font l'objet d'un suivi particulier de la part de la Direction Générale. Il s'agit en général de produire une plate-forme logicielle à portée nationale.

- Les projets d'étude

Il s'agit surtout de projets de recherche et développement ou d'études sur les usages pédagogiques de ressources. Le CNDP participe, le plus souvent en tant que partenaire, mais il peut aussi être l'initiateur du projet (projet Edutablette86). Ces projets sont généralement complexes par le nombre d'acteurs engagés à l'échelle locale, nationale ou internationale. Par exemple, le projet Itec rassemble des membres de l'European Schoolnet autour de la création de scénarios pédagogiques et des usages associés dans des environnements numériques (tableau blanc interactif, tablettes etc.). Il peut s'agir également de travaux de recherche et développement sur les problématiques documentaires. Dans le cadre du projet Vocabnomen, des travaux ont été menés en collaboration avec la tutelle sur la structuration, la gestion et l'implémentation des vocabulaires contrôlés et des nomenclatures de l'Éducation Nationale. Ces travaux sont en général effectués en collaboration avec le département d'ingénierie documentaire du CNDP.

De par la nature du support (imprimé, dvd.) et de l'objectif visé, certains projets ne nécessitent pas de compétences techniques informatiques. C'est le cas de certains projets d'édition et de projets d'étude. Les développeurs de la DSI ne sont donc pas impliqués sur ces projets. Par contre, ce n'est pas le cas des projets réalisés pour la tutelle qui requièrent systématiquement en exécution des compétences techniques spécifiques, d'abord sur la conception, le développement et ensuite sur le maintien en conditions opérationnelles. En plus des compétences techniques requises, leur pilotage requiert des bases méthodologiques pour travailler efficacement avec les équipes en charge de la réalisation.

1.2.2. Les services impliqués

Tous les services peuvent être porteurs de projets dès lors que ces projets font partie d'un des trois groupes précédemment définis. Les projets d'édition sont tout de même menés par la direction concernée. Les projets sont exécutés par une équipe pluridisciplinaire (commercial, juridique, documentaire, technique, etc.) répartie dans différents services. Leur conduite est confiée à un chef de projet appartenant au service émetteur du projet. L'organisation mise en place sur un projet reste simple et dépend du groupe auquel il appartient.

- Les projets d'édition.

Ils sont menés par la direction de l'édition. Celle-ci possède un groupe de personnels dont l'activité principale est le pilotage de projets. Lorsque le projet est lancé, la direction de l'édition désigne un pilote parmi ce groupe. Le pilote du projet identifie les compétences dont il a besoin et s'adresse aux services concernés pour constituer l'équipe. La réalisation est ensuite pilotée sans autre intervention hiérarchique. Lorsque le projet est terminé et que le produit est lancé, c'est la Direction de la Valorisation qui en assure le suivi.

- Les autres projets.

La direction générale désigne un pilote parmi sa cellule de chefs de projets (cette cellule a été créée en 2012), ou dans les directions et départements. Le choix du pilote est fonction de la nature du projet. Par exemple, pour un projet d'expérimentation sur le référencement de ressources pédagogiques, le département d'ingénierie documentaire est sollicité pour identifier un chef de projet.

Remarque : Sur la période 2010 à 2012, le pilotage de la réalisation du projet était différent. Le chef de projet n'avait pas la possibilité de coordonner l'équipe projet. Les membres de l'équipe projet ne participaient pas systématiquement à toutes les réunions autour du projet et recevaient alors les directives de la part de leur manager qui, lui-même, les avait reçues du chef de projet (figure 4).

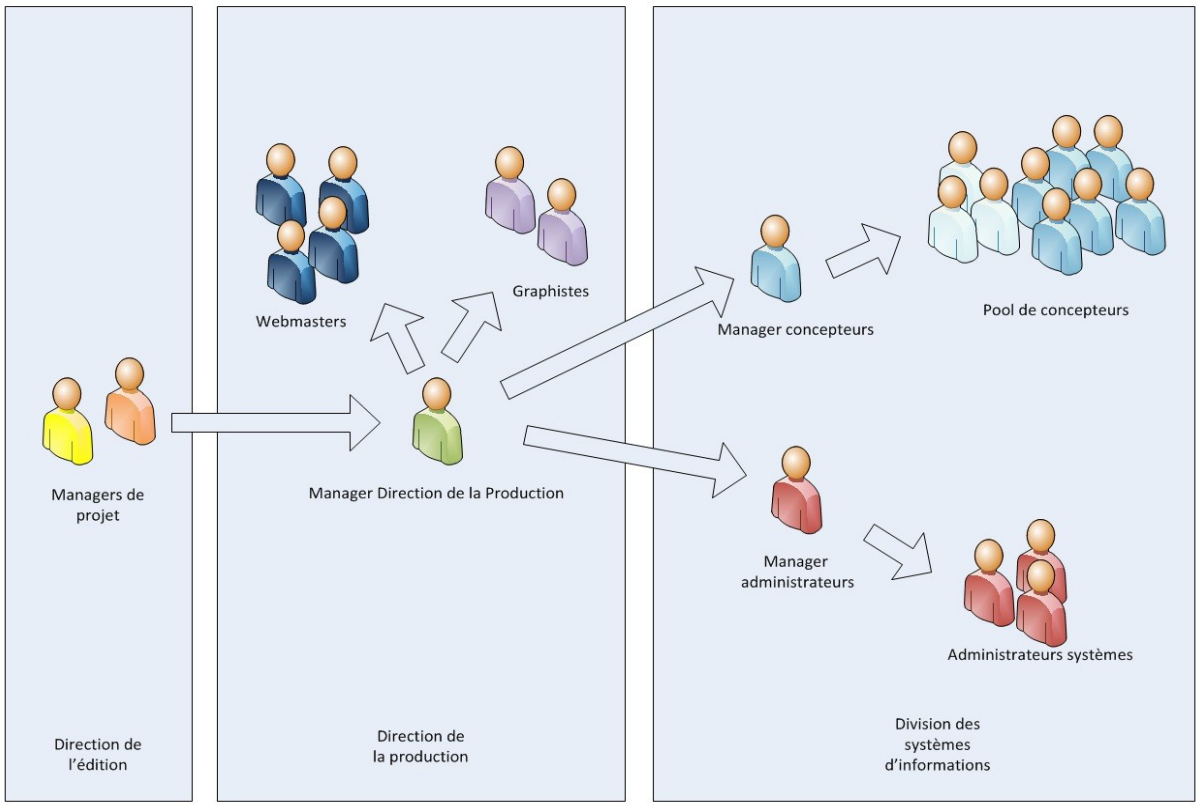


Figure 4 : Représentation du pilotage de la réalisation des projets web.

1.2.3. Les instances

L'établissement assure la gouvernance des projets au travers de deux instances, le comité de programmation et le comité de programmation éditoriale.

- Le comité de programmation.

Le comité de programmation est une instance de pilotage et de gouvernance des projets menés au sein de l'établissement. Il se réunit toutes les semaines. Il est composé des chefs de services, de coordinateurs dédiés dans chacun de ces services et de représentants de la

direction générale qui assurent l'animation du comité. Le comité assure un pilotage et un suivi des projets grâce à un phasage en trois étapes au minimum. Un projet est d'abord présenté au stade de l'idée. Il obtient en séance l'autorisation d'exécution. Il peut débuter officiellement par une réunion de lancement avec l'équipe au complet. Les phases suivantes concernent le suivi de la réalisation du projet. Il s'agit ici de faire remonter de l'information auprès des décideurs, d'émettre des avis, d'orienter ou réorienter le périmètre d'un projet et de veiller à la cohérence stratégique au travers des projets exécutés.

- Le comité de programmation éditoriale.

Il fonctionne sur le même principe que le comité de programmation mais il a une portée nationale puisqu'il s'adresse au réseau. Il a pour objectif de veiller à la cohérence des projets d'édition.

Remarque : Sur la période de référence, ces deux instances n'existaient pas. Les projets en dehors du champ de l'édition ne faisaient l'objet d'aucun suivi formel en séance comme celui réalisé par le comité de programmation. Les projets d'édition étaient suivis par le comité éditorial (pour les projets du CNDP) et l'observatoire de l'édition (pour tous les projets du réseau Scéren).

1.2.4. Approche méthodologique

Le mode de gouvernance impose une approche phasée. De ce fait, tous les projets sont jalonnés par des étapes de contrôle, ou tout du moins d'observation, exécutées par les instances. Pour autant ce phasage reste léger et permet de rencontrer des pratiques de gestion très diverses d'un projet à l'autre. Pour les projets qui font intervenir des compétences techniques spécifiques comme le développement logiciel, l'approche est plutôt hybride et très dépendante du projet et des acteurs en présence.

Pour les projets exécutés en interne, il est parfois possible de déployer des pratiques agiles autour de la cotation de projet ou du recueil des besoins, alors que d'autres projets seront exécutés dans un mode plus classique. L'approche est qualifiée d'hybride car si le développement du produit logiciel est en général itératif et incrémental on ne peut pas réellement le considérer comme agile (voir partie 2).

Pour les projets exécutés par des prestataires, l'établissement s'adapte en général à la méthode qu'ils proposent. Ainsi, le projet de refonte du RLR (Recueil des Lois et Règlements de l'Éducation Nationale), exécuté en 2009 par un prestataire de service, a été réalisé avec une approche en cascade (voir partie 2.). De même, le projet Primtice a lui aussi été exécuté dans le même mode. En revanche, le projet de refonte de la plate-forme de formation en ligne P@irformance, est exécuté à l'aide du processus itératif du prestataire, inspiré de la méthode Scrum.

L'approche méthodologique de la gestion de projet du CNDP est peu formalisée. Elle repose d'abord sur le phasage induit par le mode de gouvernance. Elle repose ensuite sur un panel de compétences de base plus que sur une appropriation de pratiques formelles exécutées en gestion de projet. En dehors du suivi organisé par les instances et en dehors de la réunion officielle de lancement du projet, l'approche méthodologique repose sur les compétences

suivantes :

- L'animation de réunion.
- La rédaction de compte rendu avec relevé de décisions.
- L'expression du besoin sous forme de cahier des charges.

1.2.5. Le processus projet

La réorganisation de l'établissement initiée depuis 2012 a impacté le processus projet au travers du nouveau mode de gouvernance installé. Ce dernier s'applique à tous les projets du CNDP quel que soit leur type. Le processus présenté (voir figure 5) est légèrement adapté aux projets sur lesquels interviennent les développeurs. Il s'agit des projets numériques précédemment décrits.

Ce processus n'est pas aujourd'hui complètement stable car il n'intègre pas encore les activités de gouvernance du comité de programmation éditorial (voir paragraphe 1.2.3.). Ce comité est en cours de constitution et n'est pas encore complètement opérationnel.

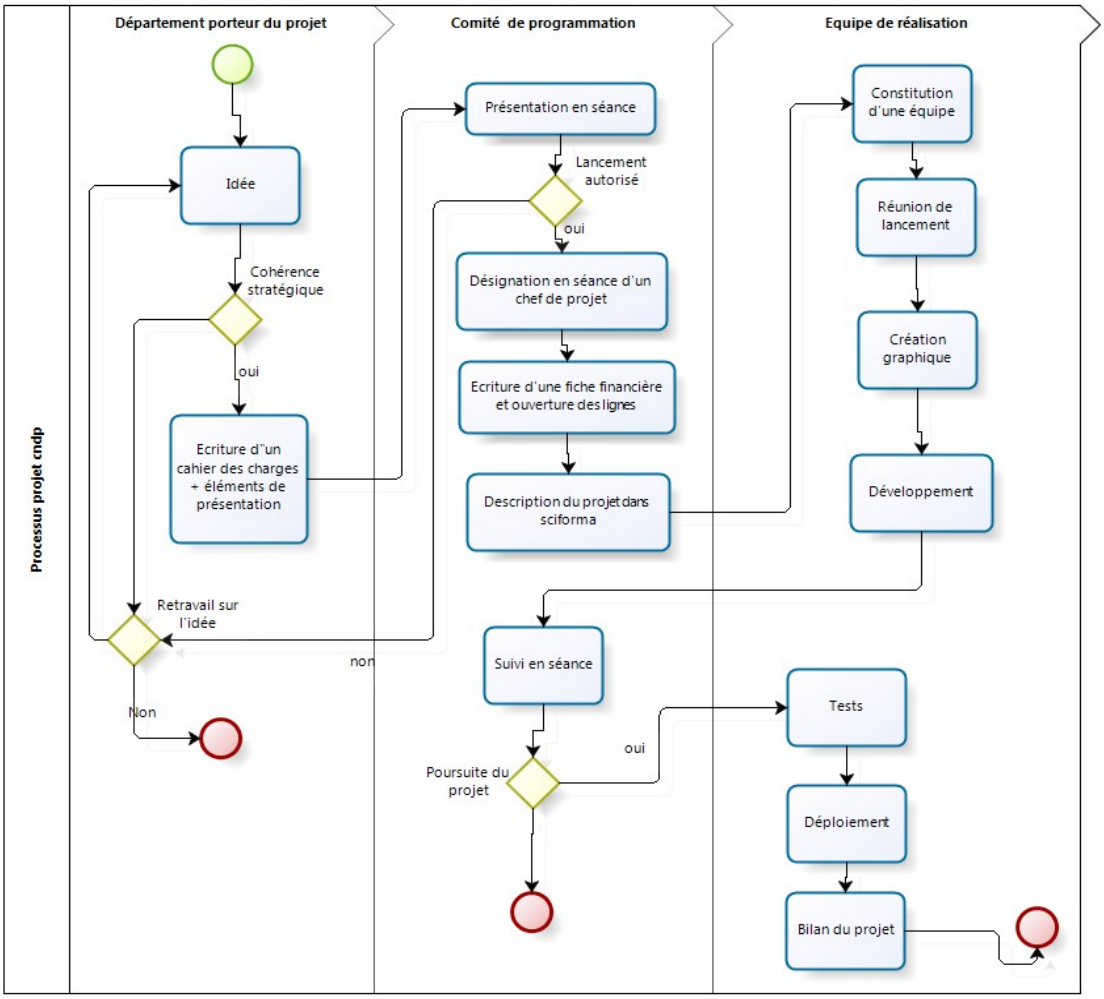


Figure 5 : Processus projet en vigueur au CNDP en 2013

A la base des projets, il y a une idée formulée par un auteur ou par un chargé de

mission. Cette idée est ensuite étudiée dans le service émetteur par des chargés de mission qui essaient d'évaluer la pertinence du projet et la cohérence avec la politique éditoriale de l'établissement. Lorsque le projet est recevable, un cahier des charges est écrit. Ce document est ensuite utilisé pour communiquer avec les instances, notamment avec le comité de programmation.

La présentation du projet en comité de programmation permet de désigner un chef de projet. Les lignes financières sont ouvertes en comptabilité et le projet est décrit dans notre logiciel de gestion. A ce stade, il n'est pas encore planifié par le chef de projet. La présentation est une étape qui permet éventuellement de modifier son périmètre ou de redéfinir les moyens alloués. En fin de séance, le lancement est autorisé ou bien reporté. Il appartient alors au chef de projet désigné de constituer une équipe.

Pour créer son équipe, le chef de projet prend contact avec les responsables des services concernés. Ces derniers désignent des ressources pour les périodes souhaitées. Un ajustement de l'intervention de ces personnels est effectué avec le chef de projet en fonction des projets en cours. Lorsque l'équipe est complète et que la date de début des travaux approche, une réunion de lancement est organisée. Elle marque le début officiel du projet. Dans le cas d'un projet numérique, les travaux de conception graphique sont les premiers lancés. Les travaux de développement viennent ensuite.

Pendant le déroulement du projet, le chef de projet peut demander la présentation de l'avancement en comité de programmation. Celui-ci peut alors être amené à allouer des ressources supplémentaires. Il veille aussi à l'alignement de l'exécution sur le périmètre initialement défini. Pour le suivi, le nombre de passages devant le comité de programmation est fonction de l'envergure du projet. Les projets les plus modestes ne passent qu'une seule fois devant le comité. En fin de projet, des tests du produit sont organisés avant qu'il ne soit livré. Un bilan est normalement effectué en fin de projet.

Ce processus est en vigueur depuis l'hiver 2012. Les activités de gouvernance sont adaptées au fil des semaines grâce à l'expérience acquise. Concernant les activités de développement spécifique aux projets numériques, les chefs de projets les gèrent chacun à sa manière. Chaque chef de projet a son approche et ses pratiques. Généralement, nous travaillons dans un mode itératif et incrémental.

Certains projets numériques ne passent pas par le comité de programmation, soit parce que les demandes sont de trop faible envergure pour justifier toutes les activités de gestion associées, soit parce qu'il s'agit de demandes du ministère gérées en dehors des instances de gouvernance de l'établissement. C'est par exemple le cas du projet de plate-forme de formation en ligne P@irformance, ou du projet de Réseau Professionnel des Enseignants.

Dans tout le processus, les développeurs interviennent à deux niveaux. Lorsque l'idée est validée et considérée comme cohérente par rapport à la stratégie, les développeurs peuvent, dans un premier temps, être sollicités pour une étude de faisabilité technique, mais ce n'est pas systématique. En fonction de l'envergure du projet, des ateliers de travail sont organisés et permettent d'ajuster le futur périmètre du projet. Ce travail permet aussi de faire une première estimation du temps de développement. Ensuite, les développeurs interviennent lorsque le projet est officiellement lancé. Entre temps, les différentes étapes sont gérées par le chef de projet et les gestionnaires du comité de programmation.

1.2.6. Le projet GAP et les processus projet

Au début de l'année 2010, le CNDP a lancé le projet PESI (Plan d'Évolution des Systèmes d'Information) pour la refonte de son système d'information et pour apporter des solutions à des problématiques variées. Ces dernières sont traitées au sein de sous-projets. Le PESI a lui-même été intégré dans le projet Scéren2017 porté par la Direction Générale. C'est ainsi qu'en 2011, le projet GAP (Gestion des Activités et des Projets) a été lancé pour répondre aux problèmes rencontrés en gestion de projet. Ces problématiques étaient les suivantes :

- La planification projet n'était pas partagée et n'était pas visible par les décideurs.
- Les informations étaient difficiles à consolider par le contrôle de gestion.
- Il était difficile de juger du niveau de fiabilité des informations issues de la gestion de projet et de la gestion des activités des services.

Ces problématiques engendraient de la complexité dans le pilotage des projets de l'établissement qui pourtant mène une grande part de son activité dans ce mode (58% du fonctionnement selon le contrôleur de gestion).

Le but du projet GAP était d'outiller la gestion des activités et des projets. Les processus ont à cette occasion dû être retravaillés pour être modélisés et paramétrés ensuite dans le futur logiciel. Après plusieurs mois de travail, l'établissement s'est donc doté d'un outil de planification et de gestion des activités. Sciforma est l'outil retenu. Les processus projet ont été révisés et le paramétrage de l'outil adapté en conséquence.

A la rédaction de ce mémoire, le projet GAP est en phase pilote. Les projets sont désormais saisis dans l'outil par les gestionnaires du comité de programmation et par les chefs de projet. Petit à petit, ces derniers réalisent la planification des projets au sein de l'outil. Le projet GAP est un projet critique pour l'établissement.

Lorsque le projet GAP a débuté, un comité de suivi était régulièrement instancié pour émettre des propositions et pour aider l'assistance à maîtrise d'ouvrage dans l'organisation des ateliers de travail avec les futurs utilisateurs. Tout au long du projet, des ateliers ont donc été organisés avec les utilisateurs pour recueillir les besoins, puis par la suite pour valider avec eux les processus modélisés. Alors que la modélisation des processus était effectuée et validée service après service, les développeurs de la DSI vivaient, à la même période leur première expérience agile (voir partie 3.) sur le projet « Catalogue Chèque Ressources » mené pour la tutelle. Aujourd'hui, l'outil est utilisé pour la gestion des activités. Les temps saisis sur les feuilles d'heures sont ventilés sur des codes d'activité projets.

Dans le cadre du projet GAP, le paradigme méthodologique de la conduite de projet qui a influencé les réflexions et la modélisation était celui du cycle en cascade. Les pratiques issues des méthodes agiles, notamment celles expérimentées à la même époque par les développeurs, n'ont pas été examinées. Nous n'avions pas encore le retour d'expérience nécessaire puisque nous étions en pleine expérimentation de ces pratiques agiles. Il reste aujourd'hui à adapter le paramétrage de l'outil à la nouvelle organisation de l'établissement. Sciforma est utilisé pour stocker les informations de description des projets. La prochaine étape est la planification et le partage transverse de ces informations.

1.3. La division des Systèmes d'information

La division des systèmes d'information est un service support placé sous la responsabilité du secrétariat général. Ce positionnement peut parfois poser des problèmes sur certains projets. La DSI n'est pas seulement une fonction support, dans certains cas elle est directement productrice de valeurs sur le cœur de métier du CNDP. C'est notamment le cas sur les projets d'édition web et de production de ressources pédagogiques.

1.3.1. L'organigramme

La Division des systèmes d'information comporte trois équipes visibles sur l'organigramme présenté en figure 6.

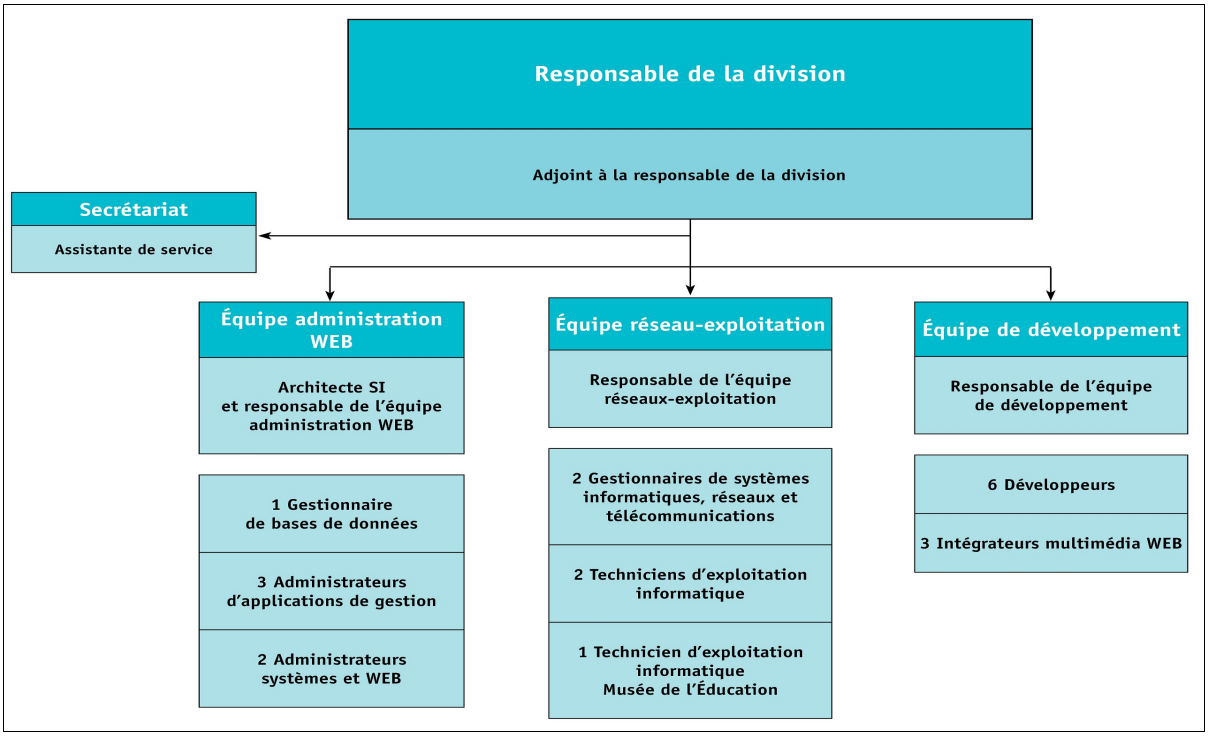


Figure 6 : Organigramme de la DSI du CNDP

- Une équipe infrastructure de cinq personnes qui assure la sécurité du système, la gestion du réseau, des équipements, et la maintenance de tous les matériels déployés : imprimantes, ordinateurs, serveurs, vidéo-projecteurs, flotte de téléphones et smartphones, téléphonie...

- Une équipe système web et applications de sept personnes. Cette équipe assure l'administration système linux. Elle est de plus orientée sur l'administration et l'architecture web et travaille en étroite collaboration avec les développeurs. Elle gère également les applications de gestion (logiciels de gestion de ressources humaines, gestion commerciale, comptabilité...) pour l'ensemble du réseau Scéren.

- Une équipe de développeurs de dix personnes qui intervient sur des projets de

développement logiciel. L'équipe est présentée au paragraphe 1.3.2.

1.3.2. L'équipe de développeurs

L'équipe de développeurs est constituée de dix personnes en incluant son manager. Parmi ces dix personnes, trois sont des intégrateurs web venus de la direction de la production au courant de l'automne 2012. Ces compétences permettent à l'équipe d'être autonome sur l'intégration HTML et sur le design web à base de feuille de style CSS.

Les développeurs sont globalement opérationnels sur plusieurs technologies mais une répartition des compétences s'est tout de même installée au fil du temps. Trois développeurs interviennent principalement sur les développements JAVA. Ils sont également compétents pour intervenir sur des développements PHP. Trois autres interviennent sur des développements PHP et sur des développements d'applications mobiles conventionnelles. L'équipe comporte trois seniors dont deux sur le domaine de compétences PHP et applications mobiles. Cette répartition nous permet d'assurer la réalisation de tous les projets de développements qui nous sont confiés.

Cependant la DSI avec l'accord de la direction générale a choisi d'anticiper sur l'augmentation de la charge de travail en publiant un accord cadre. Cet accord cadre, procédure de marchés publics, doit nous permettre de mettre en concurrence des professionnels du développement logiciel, dans le cadre d'une commande de prestation de développement au forfait ou en régie. A terme, et en fonction de notre charge de travail, l'équipe de développeurs pourrait être renforcée par des prestataires.

Entre 2010 et 2012, la capacité de prise en charge des projets était de 3262 j/h. L'équipe était composée de 4 développeurs en 2010. Deux recrutements en 2011, ont porté ce nombre à 6 augmentant de ce fait la capacité. Ces recrutements ont permis de développer et d'asseoir les compétences Java au sein de l'équipe.

1.3.3. Les technologies

Les technologies mises en œuvre par l'équipe de développement sur les développements web sont en grande majorité des technologies open source. L'annexe 2 présente les données brutes sur les projets entre 2010 et 2012 .

Les développements réalisés en environnement PHP représentent environ 50% de la charge de l'équipe entre 2010 et 2012 (voir figure 7). Ces développements comprennent les sites et plates-formes développés avec les CMS (Content Management System) typo3 et wordpress ainsi que les développements réalisés sous la plate-forme moodle. Enfin une partie de ces développements PHP concernent des projets « from scratch ». Dans ce dernier cas, le *framework* symfony est parfois utilisé.

Les projets réalisés en environnement PHP/Typo3 sont majoritairement des projets d'édition. Il s'agit le plus souvent de produire un site web pour publier du contenu pédagogique. Certaines applications et certains back-office de site web ont été développés en PHP. C'est le cas de la plate-forme Sialle ou bien du site de l'agence des usages. Aujourd'hui, ce type de développement est davantage réalisé en JAVA pour permettre à ces compétences de

se développer et de se maintenir dans l'équipe.

Les développements réalisés en JAVA représentent environ 40% de la charge de travail de l'équipe entre 2010 et 2012. La technologie JAVA est utilisée pour construire des applications web documentaires (projet pertiweb, Mondeca, gestion des entrepôts OAI ...) ou des plate-formes web de diffusion de ressources (projet Catalogue Chèque Ressources). Les flux de données de la gestion commerciale ainsi que la gestion des abonnements aux ressources en ligne sont développés en JAVA. Les frameworks utilisés sont les suivants :

- Struts 1 et 2
- Spring
- Hibernate
- Google Web Toolkit (GWT)

L'équipe de développeurs intervient en maintenance corrective sur des développements réalisés en technologies Microsoft. Il s'agit d'applicatifs documentaires développés en asp 3.0, de ressources logicielles développées en visual basic 6 .0 (logiciel Anagène) et de certaines parties applicatives du site web du CNDP (base des diplômes). Cette maintenance représente 7% de la charge consommée en mode projet entre 2010 et 2012.

Enfin sur cette même période, l'équipe a développé une première application mobile sous IOS. Le développement d'application mobile doit se poursuivre notamment sur les systèmes Android. Certains projets d'édition nécessitent des compétences en HTML5 et CSS3. Ces compétences sont mises en œuvre pour développer des ressources pédagogiques plus interactives que les sites web conventionnels.

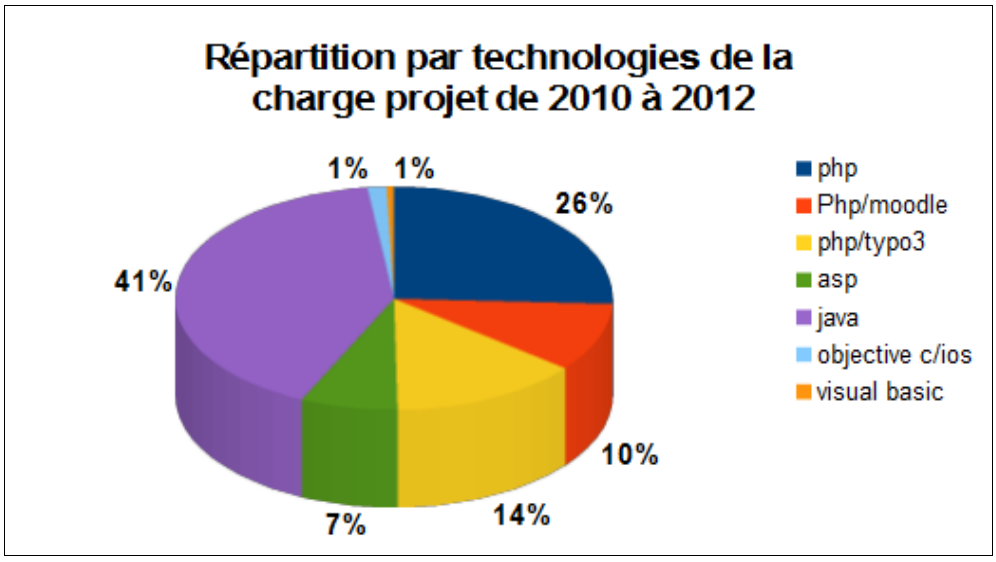


Figure 7 : Diagramme de répartition de la charge projet par technologies de 2010 à 2012

1.3.4. Les types de projets exécutés par les développeurs

Les développeurs interviennent sur des projets numériques (figure 8) situés dans le périmètre de chacune des catégories présentées dans le paragraphe 1.2.1.

Dans le cadre des **projets éditoriaux**, les développeurs sont sollicités pour créer des sites web, des applications mobiles et des applications pédagogiques déconnectées. Les sites web sont dans la grande majorité des cas développés sur la base de CMS. Il s'agit de travaux de paramétrage et de développement d'extensions spécifiques. Ces sites web sont ensuite identifiés comme ressources pédagogiques de par la nature des contenus publiés (espace « Mag film » du site cndp.fr, espace « emilangues »). Il peut s'agir de produits en ligne soumis à abonnement et donc connectés au back office de gestion commerciale (produit « engrenages et manivelles », « Images de France »). Plus simplement, il peut s'agir d'une ressource complémentaire qui est développée en accompagnement d'un produit physique (projet « expériences fondamentales »). Les applications mobiles peuvent aussi être un complément à une offre existante ou être un produit unique à par entière. Enfin, certaines ressources pédagogiques sont des applications logicielles autonomes et déconnectées, utilisables en classe par l'enseignant avec ses élèves (application Anagène en Sciences et Vie de la Terre, application educameta utilisée pour préparer le CAPES de documentation).

Les **projets de la tutelle** s'adressent le plus souvent au champ de la diffusion. Il peut s'agir de développer une plate-forme de diffusion de ressources pédagogiques sélectionnées par le ministère (Cas des plate-formes « Ecole Numérique Rurale » et « Catalogue Chèque ressources »). Il peut également s'agir de sites traitant d'un sujet particulier. Par exemple le site Internet Responsable traite du droit autour des usages d'internet. La plupart de ces projets nécessitent des travaux de développements de back office, de descriptions et d'implémentation de moteurs de recherche. Le CNDP mène également des projets d'envergure pour la tutelle, autour des problématiques de formation en ligne des enseignants (800 000 utilisateurs) et de réseaux sociaux professionnels (100 000 utilisateurs) entre autres. Les compétences des développeurs sont sollicitées sur ces projets dans des rôles d'analyse et de conseils.

Enfin, **les projets d'étude** sur lesquels interviennent les développeurs portent principalement sur l'informatique documentaire. Les développeurs du CNDP créent des services et des applications logicielles autours des normes de description documentaire (lomfr, scolomfr …), des vocabulaires contrôlés, des nomenclatures et des thésaurus. Ces travaux peuvent être effectués en collaboration avec la tutelle. C'est par exemple le cas du projet vocabnomen portant sur l'uniformisation des vocabulaires et des nomenclatures à l'échelle nationale.

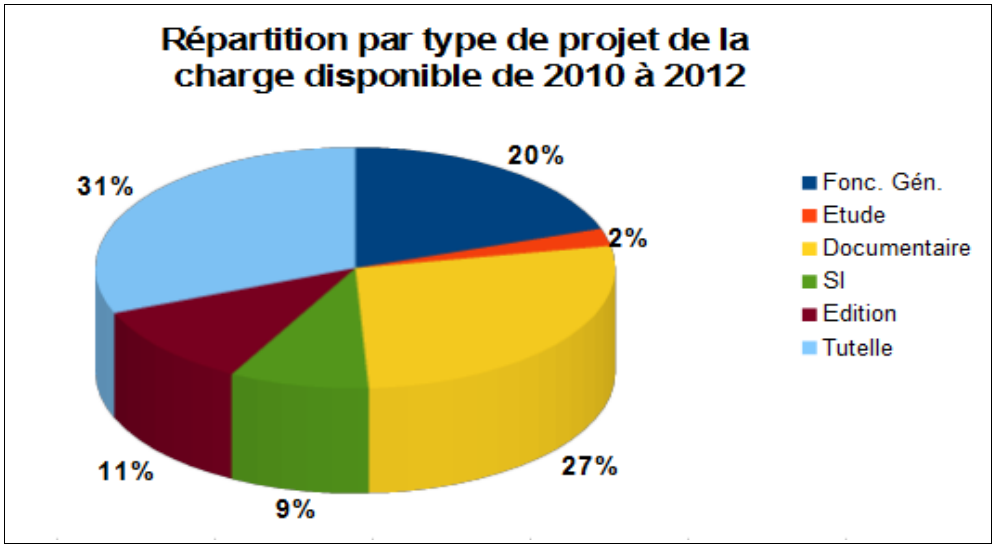


Figure 8 : Diagramme de répartition de la charge par type de projet entre 2010 et 2012

En dehors de ces trois catégories de projets, les développeurs assurent la maintenance des applicatifs métiers. Il peut s'agir des flux de données de la gestion commerciale, des outils métiers de description de ressources (ingénierie documentaire) ou bien d'applications spécifiques de gestion (application convention, application de gestion du document unique…).

1.3.5. Cartographie des travaux de développements

La cartographie suivante (figure 9) représente les 50 éléments du système d'information sur lesquels les développeurs ont réalisés des travaux de développements entre 2010 et 2012 soit en maintenance évolutive et corrective, soit dans le cadre d'un nouveau projet. Trois métriques sont représentées sur une échelle de 1 à 3. Les éléments sont répartis dans trois groupes. Un premier rassemble tous les projets exécutés pour le MEN (Ministère de l'Éducation Nationale). Le deuxième rassemble les projets Front End du CNDP qui sont à dominante éditoriale mais intègrent parfois des problématiques commerciales et-ou documentaire. Le troisième concerne les travaux en Back office. Ici le domaine qui domine est le documentaire et-ou le commercial.

La première métrique est la valeur métier de l'élément. Il s'agit de représenter les impacts d'un retrait de ce produit pour le métier (utilisateurs finaux du produit) en coût de fonctionnement. Ces impacts peuvent être évalués en calculant les coûts d'une solution alternative. La métrique est représentée par la surface des éléments. Ici, les éléments de faible surface causent un impact négligeable. Par exemple, un retrait des éléments « Musagora » ou « Planète chinois » aurait un très faible impact pour les utilisateurs finaux (enseignants) et ne remettrait pas en question leur mission. En revanche, le retrait de l'élément « Concours des 10 mots » ou bien de « Média Scéren » serait plus problématique pour les utilisateurs. Dans le cas de l'élément « concours des 10 mots », il les obligerait à mettre en place un traitement humain. Dans le cas de l'élément « Média Scéren », le retrait signifie la perte d'un marché.

La deuxième métrique représentée est le coût de l'élément. Combien coûterait un nouveau produit ? Ce coût intègre la construction (build) et l'exploitation (run). Il est représenté par la couleur des éléments. Plus l'élément est foncé, plus le coût est élevé. Les coûts évoluent en fonction du niveau de complexité de la construction et de l'exploitation. La construction intègre également les efforts de pilotage. Par exemple, les portails disciplinaires ont un coût important par le nombre d'intervenants sollicités pendant la construction et l'exploitation au contraire de l'élément « Média Scéren ». L'élément « P@irformance », est représenté par une couleur foncée et une enveloppe épaisse à cause de son architecture. Elle nécessite de multiples compétences réparties géographiquement, ce qui génère des coûts supplémentaires.

Enfin, la troisième métrique représentée est la dette technique et fonctionnelle. Plus la croûte d'un élément est épaisse et plus la dette accumulée est importante. Cette dette représente l'effort à produire pour ramener cet élément du système dans des normes de qualité et de productivité optimale. Dans la cartographie suivante, lorsque la dette technique est la plus épaisse, elle représente une dette telle que la réécriture de l'élément s'impose. Pour certains éléments en cours de développement (SI documentaire) elle représente plutôt un effort très important à produire pour aboutir à une version stable et opérationnelle de l'élément. Une dette technique moyenne signifie que l'élément mérite que certaines fonctions

soient corrigées ou améliorées. Enfin une dette technique faible signifie que le produit ne requiert pas d'attention particulière. Il est fiable et opérationnel.

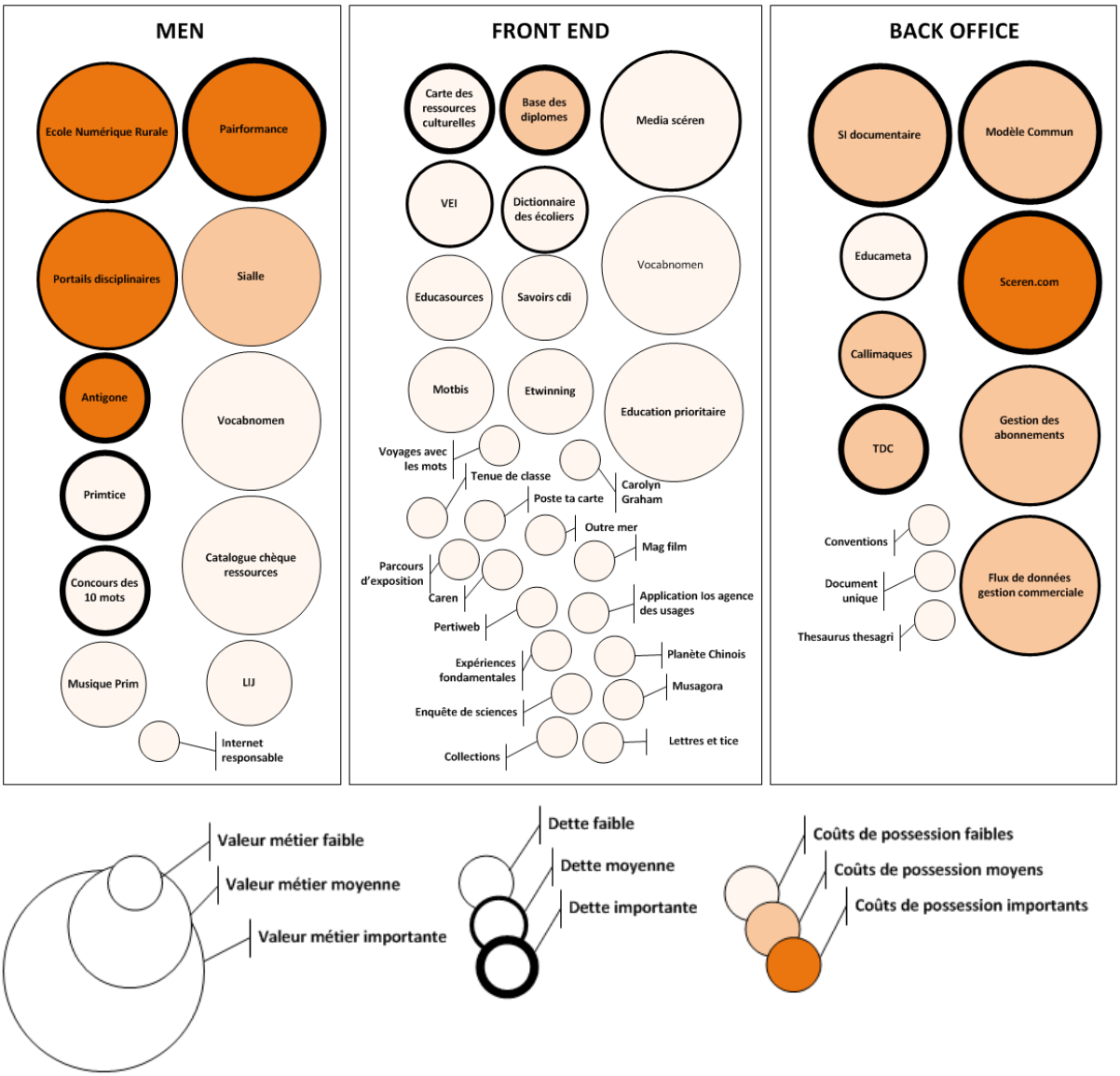


Figure 9 : Cartographie des travaux de développement exécutés entre 2010 et 2012.

1.3.6. Volumétrie

La mesure de l'activité des développeurs est effectuée au travers des outils de suivi de demandes. Depuis 2009, l'équipe a utilisée trois outils de suivi de demandes. Elle a d'abord travaillé avec Mantis, puis avec GLPI. En 2010, JIRA a été expérimenté sur de petits projets de développement et uniquement au sein de l'équipe, avant d'être utilisé sur le projet « Catalogue Chèque Ressources » en 2011. Il s'est révélé être le meilleur outil pour gérer l'activité des développeurs grâce à son orientation projet et sa typologie de demande.

Remarque : J'ai analysé la répartition des demandes par projets dans les outils. Cette analyse s'est révélée difficile car les demandes dans GLPI ne sont pas stockées par projet. J'ai donc dû effectuer des regroupements à l'aide de requêtes SQL directement sur la base de données. De

plus, les demandes de développements créées dans GLPI ne sont pas catégorisées. Il est donc impossible de faire la différence entre les défauts, les développements de nouvelles fonctionnalités ou les améliorations.

Entre 2010 et 2012, 2808 demandes ont été traitées par l'équipe (figure 10). L'utilisation de JIRA s'est imposée peu à peu depuis la fin de l'été 2012. Il y a donc eu une période de recouvrement entre les outils. De plus, le rappel régulier des consignes auprès des développeurs et des chefs de projets participe à l'élévation du nombre de demandes. Les consignes étant que chaque ligne de code produite et poussée dans les dépôts doit normalement être tracée par rapport à une demande. Aucune activité de développement ne doit normalement être exécutée sans qu'une demande ne soit initiée. L'équipe a aujourd'hui complètement basculé la gestion de ses activités sous JIRA, mais des demandes sont toujours adressées par GLPI lorsque les rapporteurs ne possèdent pas de compte. Dans JIRA, les rapporteurs de demandes de développement sont au nombre de 60.

	MANTIS	GLPI	JIRA	***total***
2010	14	331	123	468
2011	0	622	217	839
2012	0	1156	345	1501
total	14	2109	685	**2808**

Figure 10 : Evolution du nombre de demandes traitées entre 2010 et 2012

Sur le premier semestre 2013, 731 demandes ont été créés dans JIRA pour 55 demandes dans GLPI. Les demandes JIRA se répartissent équitablement entre les défauts, les améliorations et nouvelles fonctions, et les tâches (voir figure 11).

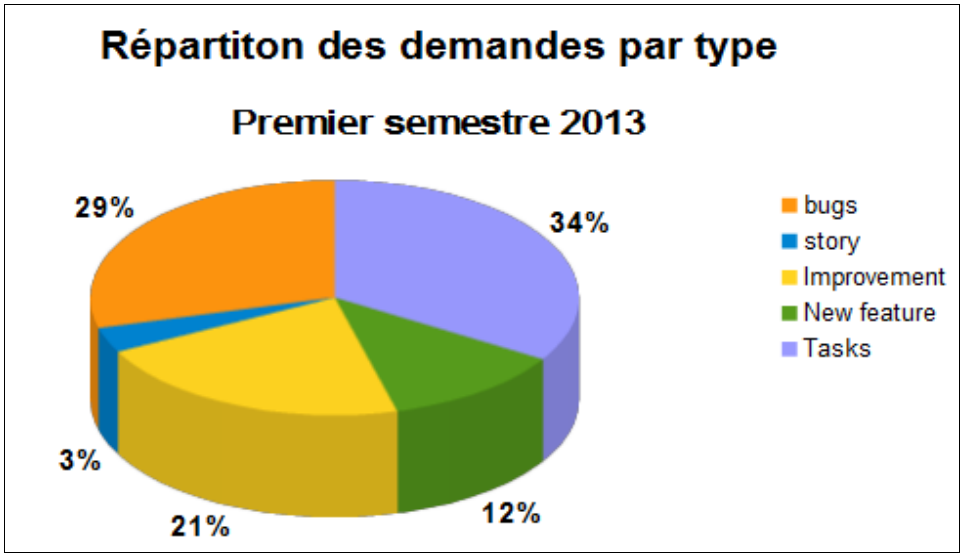


Figure 11 : Répartition des demandes par type

Le nombre important de tâches est dû à une mauvaise catégorisation des demandes d'amélioration ou de nouvelles fonctionnalités de la part des rapporteurs. De plus, les demandes de développement créées dans JIRA sur cette période sont encore majoritairement des demandes effectuées en dehors du projet et donc traitées dans le flux.

2. Gestion de projet, approches traditionnelles et agiles du développement logiciel.

2.1. Généralités

Depuis les années 90, les méthodes agiles se développent et gagnent en notoriété dans la sphère du développement logiciel et dans les métiers de l'informatique en général. Leur pénétration dans les organisations semble s'accélérer au cours de ces cinq dernières années. Les valeurs de l'agilité font aussi écho en dehors de la sphère du génie logiciel. On parle désormais d'organisation agile ou de management agile [BAR10][BAR12][PEZ10]. Les mouvements, les méthodes et frameworks partagent des valeurs communes et des principes sous-jacents qui du point de vue des agilistes font défaut aux méthodes traditionnelles. Pourtant, les outils méthodologiques et normatifs de cette approche qualifiée de traditionnelle peuvent être pertinents et nécessaires dans des environnements complexes et critiques [PRI10]. Au delà de l'exécution itérative de projet, quelles sont les caractéristiques d'une exécution de projet dite « agile » ? Quelles sont ces méthodes agiles ? Comment fonctionnent-elles ?

2.2. Quelques chiffres sur la gestion de projet

La figure 12 montre la répartition de projets menés avec succès, de projets en échec et de projets en succès partagés. Ces chiffres datent de 2010 et ont été publiés par le standish group. 37% des projets des sociétés interrogées auraient été menés avec succès. Ce chiffre est d'après le standish group en augmentation par rapport aux années précédentes. Il était de 28% en 2000 et seulement 16% en 1994.

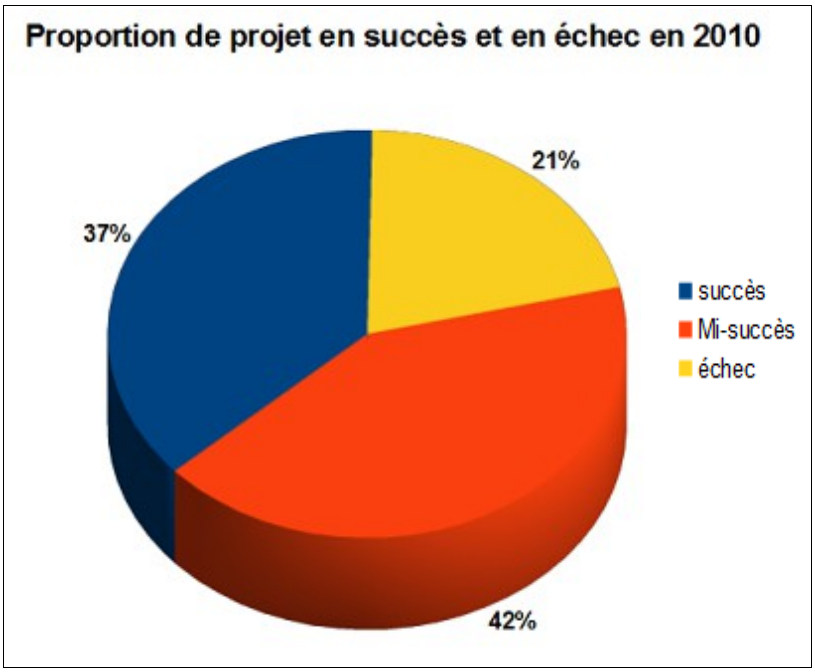


Figure 12 : Statistiques 2010 du standish group concernant le réussite des projets.

La figure 13 présente des chiffres de 2011 sur la répartition du succès entre les méthodes agiles et les méthodes traditionnelles. 42 % des projets exécutés à l'aide de méthodes agiles seraient menés avec succès contre seulement 14% des projets pour les méthodes traditionnelles. Les méthodes agiles contribueraient donc à la réussite des projets.

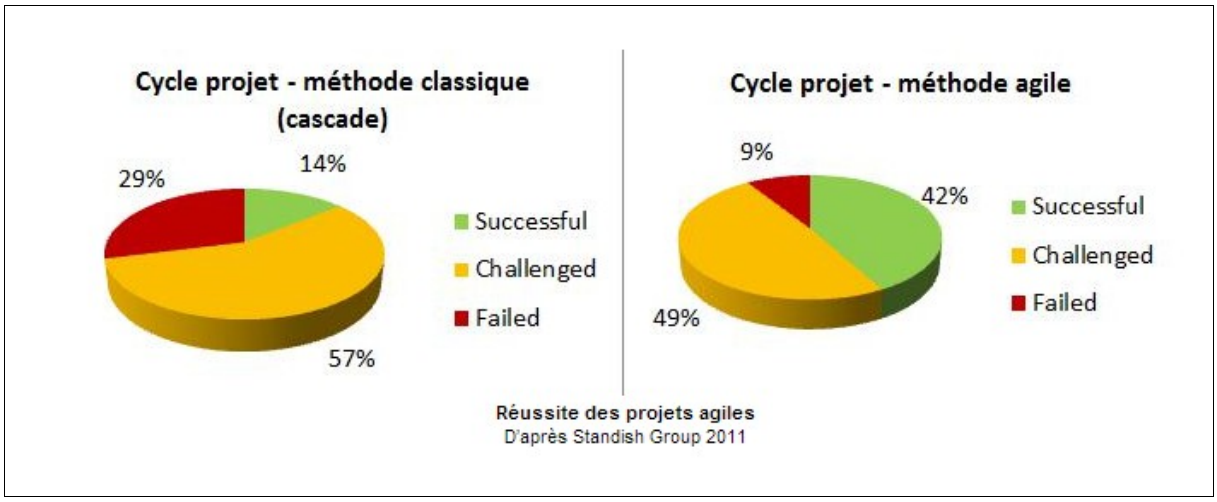


Figure 13 : Répartition dus succès entre méthodes classiques et méthodes agiles (source standish group 2011)

2.3. Repères historiques

Avant d'aborder les différences fondamentales entre les approches traditionnelles et agiles du développement logiciel, il est utile de rappeler les dates clés de l'ingénierie et de la gestion de projet. Ceci afin de comprendre comment les valeurs de l'agilité se sont développées en capitalisant sur les connaissances acquises en ingénierie.

1880 : Naissance du taylorisme. Taylor met au point une méthode d'organisation du travail basée sur l'analyse scientifique du travail. Il n'y a qu'un seule bonne manière de faire, c'est « the one best way », basée sur une analyse fine des tâches exécutées en production et sur une rémunération objectivée. Le travail est spécialisé.

1910 : Henry Laurence Gantt met au point le diagramme Gantt qui est depuis très utilisé en gestion de projet.

1950 : William Edwards Deming présente au patronat japonais ses théories d'amélioration de la production qu'il a développées pendant la guerre. Il développe au Japon le concept d'amélioration continue ou PDCA de Walter A. Shewhart que l'on appelle aussi aujourd'hui la roue de Deming.

1960 : Adoption par Toyota des principes et méthodes de l'ingénieur Taiichi Ōno. Les méthodes JAT (juste à temps) sont peu à peu mises en œuvre dans les entreprises japonaises. Contrairement au taylorisme, les ouvriers doivent être polyvalents. C'est la naissance de la pensée Lean.

1970 : Crise du logiciel qui met en avant la mauvaise qualité des logiciels produits depuis l'arrivée des circuits intégrés, l'absence de maîtrise du développement et de la gestion des projets.

1986 : Barry Boehm met au point la méthode en spirale.

1991 : La méthode RAD est formalisée par James Martin après plusieurs années de travaux sur le formalisme.

1995 : Ken Schwaber et Jeff Sutherland formalisent le framework Scrum et le présentent à la conférence nord américaine OOPSLA.

1997 : Grady Booch, Ivar Jacobson et James Rumbaugh mettent en commun leur travaux sur la modélisation et la méthode et donnent naissance au processus unifié et à UML.

1999 : Kent Beck publie son ouvrage sur le framework de l'eXtreme Programming qu'il a mis au point avec Ward Cunningham.

2001 : Plusieurs ingénieurs américains de renom, dont Kent Beck, Ken Schwaber, Jeff Sutherland, Ward Cunningham élaborent et signent le manifeste agile connu sous le nom de « Agile Manifesto ».

2005 : Déclaration d'interdépendance par David Anderson, Alistair Cockburn, Mike Cohn, Jim Highsmith

2009 : Publication du manifeste pour l'artisanat logiciel.

2010 : Naissance du mouvement DevOps. Les valeurs de l'agilité dépassent la sphère du développement logiciel et sont désormais portées par les opérationnels.

2.4. La gestion de projet décrite par le pmbok

La gestion de projet est aujourd'hui un domaine de compétence bien documenté et normé. Le guide de bonnes pratiques pmbok (Project Management Boby Of Knowledge) [PMB08], maintenu par le PMI (Project Management Institute), fait référence dans la conduite de projet au côté de Prince2, des normes ISO (21500, 10006...) ou encore des référentiels et modèles de maturité spécifiques à l'IT comme le CMMI. Le guide pmbok est élaboré par un collectif de professionnels de la gestion de projet qui travaillent dans des organisations publiques ou privées à travers le monde et dans des secteurs d'activités variés. Le guide traite donc de la gestion de projet sans distinction de domaine. C'est également le cas des normes ISO ou de Prince2. Ces outils de référence intègrent au fil du temps les remarques, les recommandations et l'expérience des professionnels de la gestion de projet.

2.4.1. Présentation du guide pmbok

Le pmbok n'est pas une méthode. C'est un guide qui décrit des processus qui peuvent

être exécutés sur les projets. Ils peuvent d'ailleurs être exécutés dans une approche traditionnelle ou agile du développement logiciel. Le pmbok n'est pas vraiment prescripteur sur le sujet. Il appartient aux organisations de choisir les processus et le niveau de rigueur qui leur conviennent en corrélation avec leurs valeurs et principes, qu'ils soient agiles ou non. La description des processus comporte : les données d'entrée du processus, les outils et techniques et les données de sortie. Enfin, on retrouve sensiblement le même contenu et le même champ lexical dans le pmbok, les normes ISO (21500, 10006...), et les modèles de maturité plus spécifiques à l'IT.

Le pmbok (5th edition) comporte 47 processus (figure 14). Ils sont regroupés dans 5 groupes et dans 10 domaines de connaissances.

Les 10 domaines de connaissances sont les suivants :

- Processus de gestion de l'intégration du projet
- Processus de gestion du contenu du projet
- Processus de gestion des délais du projet
- Processus de gestion des coûts du projet
- Processus de gestion de la qualité du projet
- Processus de gestion des ressources humaines du projet
- Processus de gestion des communications du projet
- processus de gestion des risques du projet
- Processus de gestion des approvisionnements
- Processus de gestion des parties prenantes

les 5 groupes sont les suivants :

- Groupe de processus de démarrage
- Groupe de processus de planification
- Groupe de processus d'exécution
- Groupe de processus de surveillance et de maîtrise
- Groupe de processus de clôture

Remarque : La numérotation des processus (figure 14) est conforme à leur présentation dans le pmbok (5eme édition).

Domaines de connaissance	Groupes de processus				
	Groupe de processus de démarrage	**Groupe de processus de planification**	**Groupe de processus d'exécution**	**Groupe de processus de surveillance et de maîtrise**	**Groupe de processus de clôture**
4. Management de l'intégration du projet	4.1 Élaborer la charte du projet	4.2 Élaborer le plan de management du projet	4.3 Diriger et piloter l'exécution du projet	4.4 Surveiller et maîtriser le travail du projet 4.5 Mettre en œuvre la maîtrise intégrée des modifications	4.6 Clore le projet ou la phase
5. Management du contenu du projet		5.1 Recueillir les exigences 5.2 Définir le contenu		5.4 Vérifier le contenu 5.5 Maîtriser le contenu	

		5.3 Créer la SDP			
6. Management des délais du projet		6.1 Définir les activités 6.2 Organiser les activités en séquence 6.3 Estimer les ressources nécessaires aux activités 6.4 Estimer la durée des activités 6.5 Élaborer l'échéancier		6.6 Maîtriser l'échéancier	
7. Management des coûts du projet		7.1 estimer les coûts 7.2 Déterminer le budget		7.3 Maîtriser les coûts	
8. Management de la qualité du projet		8.1 Planifier la qualité	8.2 Mettre en œuvre l'assurance qualité	8.3 Mettre en œuvre le contrôle qualité	
9. Management des ressources humaines du projet		9.1 Élaborer le plan des ressources humaines	9.2 Constituer l'équipe projet 9.3 Développer l'équipe de projet 9.4 Diriger l'équipe de projet		
10. Management des communications du projet	10.1 identifier les parties prenantes	10.2 Planifier les communications	10.3 diffuser les informations 10.4 Gérer les attentes des parties prenantes	10.5 Rendre compte de la performance	
11. Management des risques du projet		11.1 Planifier le management des risques 11.2 Identifier les risques 11.3 Mettre en œuvre l'analyse qualitative des risques 11.4 Mettre en œuvre l'analyse quantitative des risques 11.5 Planifier les réponses aux risques		11.6 Surveiller et maîtriser les risques	
12. Management des approvisionnements du projet		12.1 Planifier les approvisionnements	12.2 procéder aux approvisionnements	12.3 Gérer les approvisionnements	12.4 Clore les approvisionnements
13. Management des parties prenantes		13.1 Planifier la gestion du contenu 13.2 Planifier la gestion des plannings Management 13.3 Planifier la gestion des coûts 13.4 Planifier la gestion des parties prenantes		13.5 Contrôler la gestion des parties prenantes	

Figure 14 : Tableau des processus du pmbok

2.4.2. Éléments du pmbok

Le pmbok rappelle certaines notions de base de la gestion de projet. Parmi elles ont retrouve les notions de portefeuille, de programme, de cycle de vie du projet ainsi que les notions présentées ci-dessous. Le pmbok présente également les différentes cultures organisationnelles et rappelle que la disponibilité des ressources peut être affectée par ces

différences. Le chef de projet doit comprendre ces différentes « normes » organisationnelles et identifier les personnes en capacité de prendre les décisions. Il existe 3 types de structures (figure 15) organisationnelles dans lesquelles l'autorité et la responsabilité sont réparties différemment entre les chefs de projets et les responsables fonctionnels.

	Fonctionnelle	**Matricielle**	**Par projets**
Autorité du Chef de projet	Peu ou aucune	Modérée	Forte à quasi totale
Disponibilité des ressources	Peu ou aucune	Faible à modérée	Forte à quasi totale
Responsable du budget du projet	Responsable fonctionnel	Mixte	Chef de projet
Engagement du chef de projet	Temps partiel	Plein temps	Plein temps

Figure 15 : Tableau des structures organisationnelles

- Le projet - Un projet est un effort temporaire exercé dans le but de créer un produit, un service ou un résultat unique. La nature temporaire des projets implique un commencement et une fin déterminés.

- Le rôle du chef de projet - Le chef de projet est la personne, désignée par la l'organisation réalisatrice, qui est chargée d'atteindre les objectifs du projet.

- La gestion de projet - La gestion de projet est l'application de connaissances, de compétences, d'outils et de techniques aux activités d'un projet afin d'en satisfaire les exigences.

- Le programme – Un programme est un groupe de projets apparentés servant des objectifs communs et une stratégie commune. Les projets du programme sont interdépendants.

- Le portefeuille – Lorsque l'on souhaite regrouper les projets en fonction d'un ou plusieurs critères (client, technologie, fournisseur etc.), on effectue ce regroupement au sein d'un portefeuille de projets. Les projets ne sont pas forcément liés.

Par ailleurs le pmbok insiste sur la géométrie variable de la gestion de projet.

Les processus de management de projet s'appliquent globalement et dans tous les groupes d'industries. « Bonne pratique » signifie qu'il existe un large consensus sur le fait que l'application des processus de management de projet améliore les chances de succès de projets très divers. Ceci ne signifie pas que la connaissance, les compétences et les processus décrits doivent être appliqués de manière uniforme à tous les projets. Il incombe toujours au chef de projet, en collaboration avec l'équipe de projet, de déterminer quels processus sont appropriés pour un projet donné, et quel niveau de rigueur est approprié pour chaque processus.

Remarque : Le projet CCR n'a pas été conduit en utilisant les recommandations du pmbok, recommandations que j'ai utilisées pour le présenter (cf 3.2. à 3.6.).

2.4.3. Spécificité du développement logiciel

Dans le cas du développement logiciel, l'approche retenue (agile ou traditionnelle) influencera grandement la manière dont les activités de gestion de projet et de génie logiciel seront menées et enchaînées. C'est le cas de l'expression du besoin et de sa gestion tout au long du projet ainsi que de l'estimation et de la planification. Les méthodes et *frameworks* agiles proposent un certains nombre de pratiques qui répondent aux processus identifiés dans le guide de gestion de projet pmbok. Cependant, les processus présentés dans ce même guide ne trouvent pas toujours de pratiques correspondantes dans les méthodes agiles.

Remarque : Il existe des points d'accroche reconnus entre le modèle CMMI et l'agilité. Aujourd'hui les modèles de maturité s'adaptent et reconnaissent les pratiques agiles. De la même manière, la maturité est une notion qui apparaît en agilité notamment au sein du framework Scrum.

2.5. L'approche traditionnelle

L'approche dite « traditionnelle » du développement logiciel désigne les méthodes dont le pilotage est effectué par les plans (plan driven). Cette approche est entre autre caractérisée par un cycle de vie où les activités de développement dûment documentées et contrôlées s'enchaînent de l'expression du besoin jusqu'à la livraison du produit.

2.5.1. Le modèle de cycle en cascade

Le modèle en cascade (*waterfall*) a été présenté en 1970 par le Dr Winston W. Royce [ROY70]. Dans ce modèle, les étapes du développement sont exécutées séquentiellement. Le cycle est constitué d'un ensemble de phases qui s'enchaînent de l'étude de faisabilité jusqu'à la livraison du produit. Chaque phase doit être documentée et contrôlée. Les produits d'activités des phases en amont alimentent les processus des phases en aval (figure 16). Dans ce modèle, le travail itératif est possible mais normalement limité entre phases voisines pour maintenir la portée du changement dans des limites gérables. L'objectif était de limiter les coûts [BOE88] qui pourraient être engendrés par du travail itératif sur des phases plus en amont du cycle. Ce modèle incorpore également une phase de prototypage qui est menée en parallèle de l'analyse des besoins et de la conception. Royce conseille dans le cas d'un logiciel développé pour la première fois de s'arranger pour livrer la deuxième version, si le logiciel comporte une certaine criticité. C'est ce que Royce appelle le « *build it twice* » [ROY70]. Le but est donc de développer une version pilote qui doit aider le chef de projet à s'affranchir des jugements humains qui sont, dans le domaine de la conception logicielle, trop optimistes selon Royce. Pour un effort de 30 mois, la version pilote du système doit être obtenue au bout de 10 mois soit 1/3 de la charge globale. Cette proportion tombe à un quart pour un effort de 12 mois.

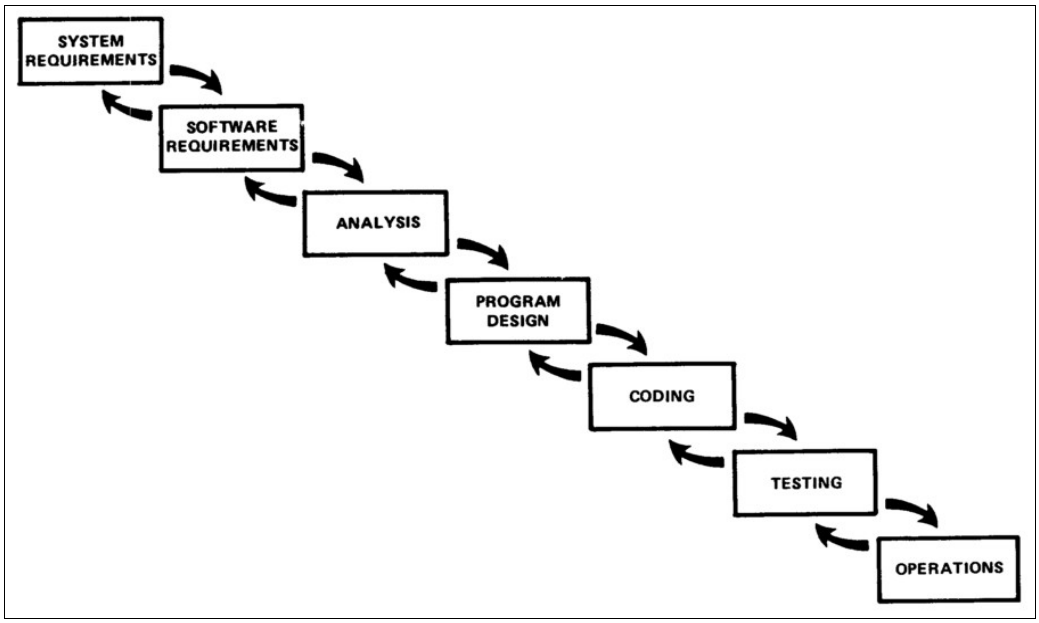


Figure 16 : Cycle en cascade de W. Royce

Enfin Boehm rappelle que ce modèle devait permettre de sortir du modèle « *code-and-fix* » qui ne permettait plus de piloter les projets de développement logiciel dans de bonnes conditions. En outre, le modèle en cascade a posé les fondements qui ont permis aux normes de se développer sous l'influence des commandes de l'administration nord américaine [BOE88].

2.5.2. Le modèle de cycle en V et en W

Ces modèles sont dérivés du modèle de cycle en cascade. Le modèle en V [DER84] fait correspondre à chaque phase du développement représentée dans la partie gauche, une activité de spécification de test spécifique dans la partie droite (figure 17). Cette version du cycle en V est dite consolidée [BOS04].

La première phase du V correspond à l'expression et à l'analyse des besoins contractualisée par un cahier des charges. Ensuite, la conception est réalisée en réponse au cahier des charges. Ces activités de conception engendrent des spécifications détaillées. Lorsque ces spécifications sont produites, la conception technique peut être réalisée. A l'issue de cette phase un cahier d'architecture est produit. Ensuite, la conception détaillée précisera l'architecture logicielle du système (structures de données, algorithmes, prototypage des fonctions et procédures, protocoles de communication) pour permettre d'aborder la dernière phase du V qui correspond aux activités de codage. Ces dernières comprennent l'écriture du code. En remontant dans le V, chaque activité de test correspondante est réalisée conformément aux spécifications de test réalisées dans la partie descendante. Les tests unitaires permettent de vérifier la conception détaillée. Les tests d'intégration permettent de vérifier la conception générale et la bonne intégration du système avec des éléments externes. A ce niveau la vérification est effectuée. Elle consiste avant tout à s'assurer que le produit fonctionne correctement. Ensuite, la validation permet de s'assurer que les spécifications fonctionnelles sont bien implémentées et que le produit rend bien le service attendu. Enfin la

recette est la dernière vérification avant la livraison.

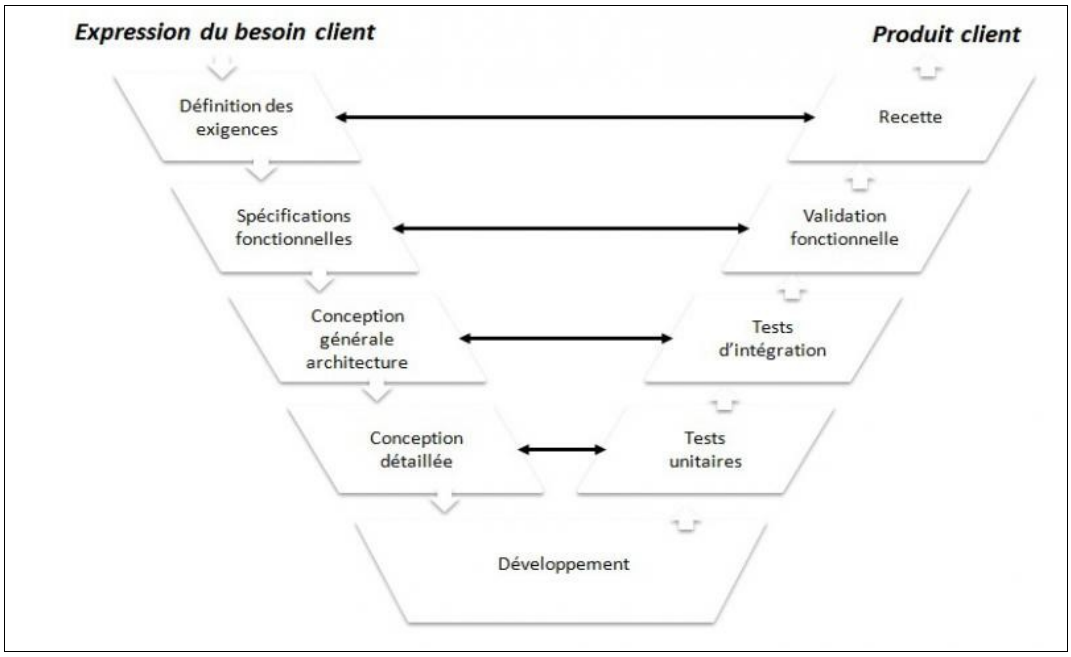


Figure 17 : Cycle en V
(Source : http://www.creative-ingenierie.fr/fr/notre-offre)

Le Cycle en W (figure 18) de Paul Herzlich est une représentation plus complète du modèle en V. Il tente de mieux représenter l'importance des tests en superposant deux V. Le premier V représente le développement du produit (partie gauche) et les phases d'intégration (partie droite). Le second V représente les activités de revues et de tests associées aux étapes de développement et d'intégration du premier V.

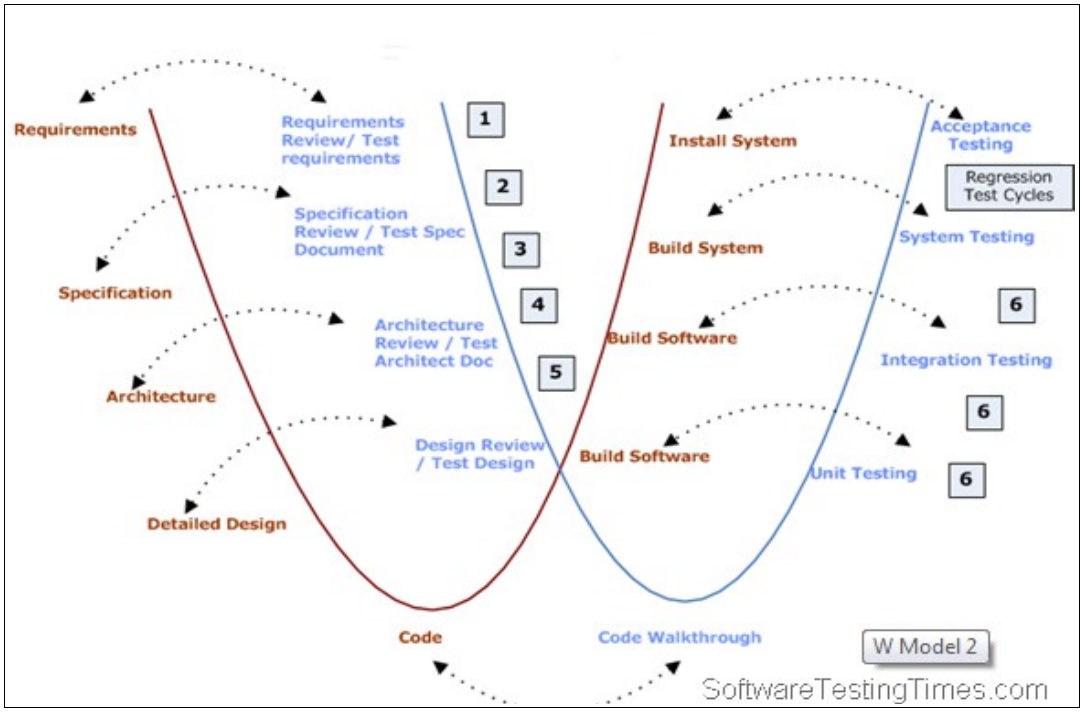


Figure 18 : Cycle en W de Paul Herzlich
(source : http://www.softwaretestingtimes.com/2010/04/v-model-to-w-model-w-model-in-sdlc.html)

2.5.3. Le modèle de cycle en spirale

Le modèle en spirale a été présenté en 1988 par Barry Boehm [BOE88] dans la revue « Computer » de l'IEEE. Ce modèle était proposé pour résoudre les différents problèmes des modèles classiques. Ce modèle de cycle en spirale est dit « *risk driven* », autrement dit piloté par les risques alors que le modèle en cascade et ses variantes sont dits « *document driven* » ou bien pilotés par les documents.

L'axe vertical (figure 19) représente les coûts cumulés alors que l'axe horizontal représente la progression au travers des différentes étapes du cycle de vie. Chaque cycle comporte les même étapes.

Un cycle commence par une identification des objectifs, des contraintes (coûts, qualité, délais) et des alternatives (achat de composants, réutilisation etc.) au développement pour le cycle en cours. Cette partie du cycle est représentée par le quart supérieur-gauche de la spirale.

La deuxième partie du cycle (quart supérieur-droit) consiste à évaluer les risques et les zones d'incertitudes à partir de l'identification menée dans la première partie. La levée des risques est entre autre effectuée grâce à du prototypage, du *benchmarking*, de la modélisation, des ateliers de travail avec les utilisateurs.

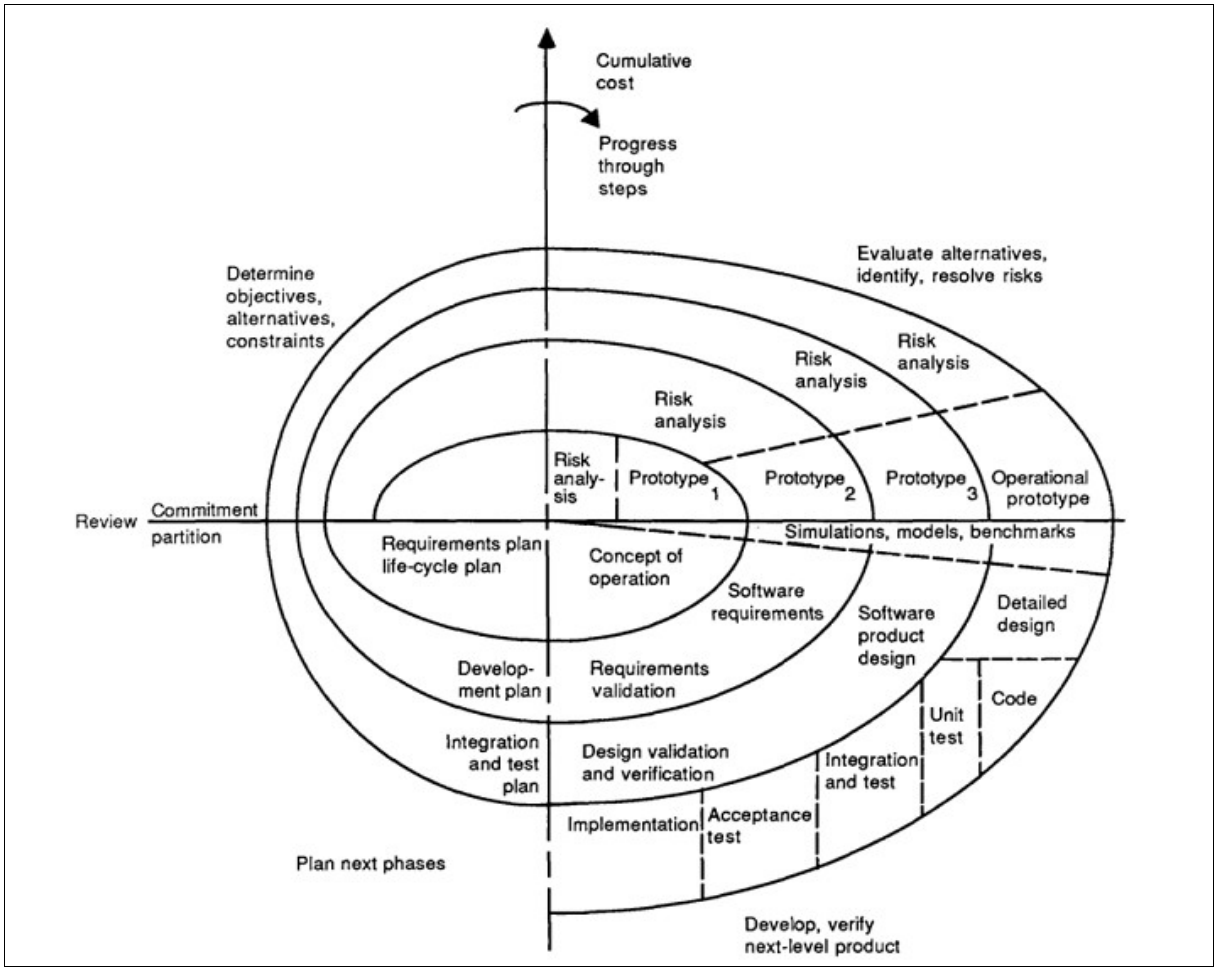


Figure 19 : Cycle en spirale de Boehm.

A la troisième étape, si les risques dominent sur le développement, l'effort est porté sur les travaux d'analyse et de spécification sinon le cycle suit les étapes du cycle en cascade tout en permettant le développement incrémental. Cette étape du cycle comporte une validation des produits d'activités.

Enfin la dernière étape consiste à définir les plans du cycle suivant. Elle se termine par une revue, effectuée par les parties prenantes, sur l'ensemble des travaux exécutés sur le cycle. Elle permet en outre de maintenir le niveau d'engagement des parties prenantes.

2.5.4. Le processus unifié

L'approche traditionnelle permet également d'instancier un cycle de développement itératif pour permettre de conserver l'alignement du produit sur les besoins du métier tout au long du projet. Le processus unifié (*Unified Process*) mis au point par Grady Booch, Ivar Jacobson et James Rumbaugh, embarque toutes les étapes de développement au sein d'une itération. Le processus unifié utilise largement le langage UML. Il est en outre conforme aux normes ISO comme la 12207 [PRI10]. Il existe plusieurs déclinaisons de l'UP mais les plus répandues sont certainement le RUP de la société Rational (IBM a racheté cette société en 2002) et le 2TUP (T*wo Track Unified Process*) ou cycle en Y de la société Valtech Computing.

La figure 20 présente les quatre phases du processus unifié.

- L'inception - Cette phase correspond au développement de la vision du produit. Il s'agit de faire l'analyse des fonctions principales du système et de délimiter le périmètre fonctionnel du projet.

- L'élaboration - Ici on se concentre sur l'architecture en levant les risques liés aux fonctions critiques du système. A l'issue de cette phase, les risques doivent être suffisamment limités et les besoins suffisamment stables pour lancer la phase de construction. La phase d'élaboration doit aussi permettre d'établir les plans de coûts et de délais. Un ou plusieurs prototypes peuvent être produits.

- La construction - Pendant la construction, l'accent est mis sur la gestion des ressources, des coûts et des délais. A l'issue de la phase de construction une version béta du produit doit être disponible.

- La transition - C'est au cours de cette phase que le produit est livré aux utilisateurs. Cela signifie que les fonctions clés du système sont disponibles et que le niveau de qualité est suffisamment élevé pour que le produit puisse être placé en production.

Chacune de ces phases recouvre l'ensemble des activités du processus. Par ailleurs elles comportent plusieurs itérations au cours desquelles, la conduite de ces activités évolue. Dans la phase d'inception, l'accent est mis sur les activités de modélisation et de recueil du besoin. Alors qu'en passant à la phase d'élaboration l'accent est davantage mis sur l'analyse et la conception.

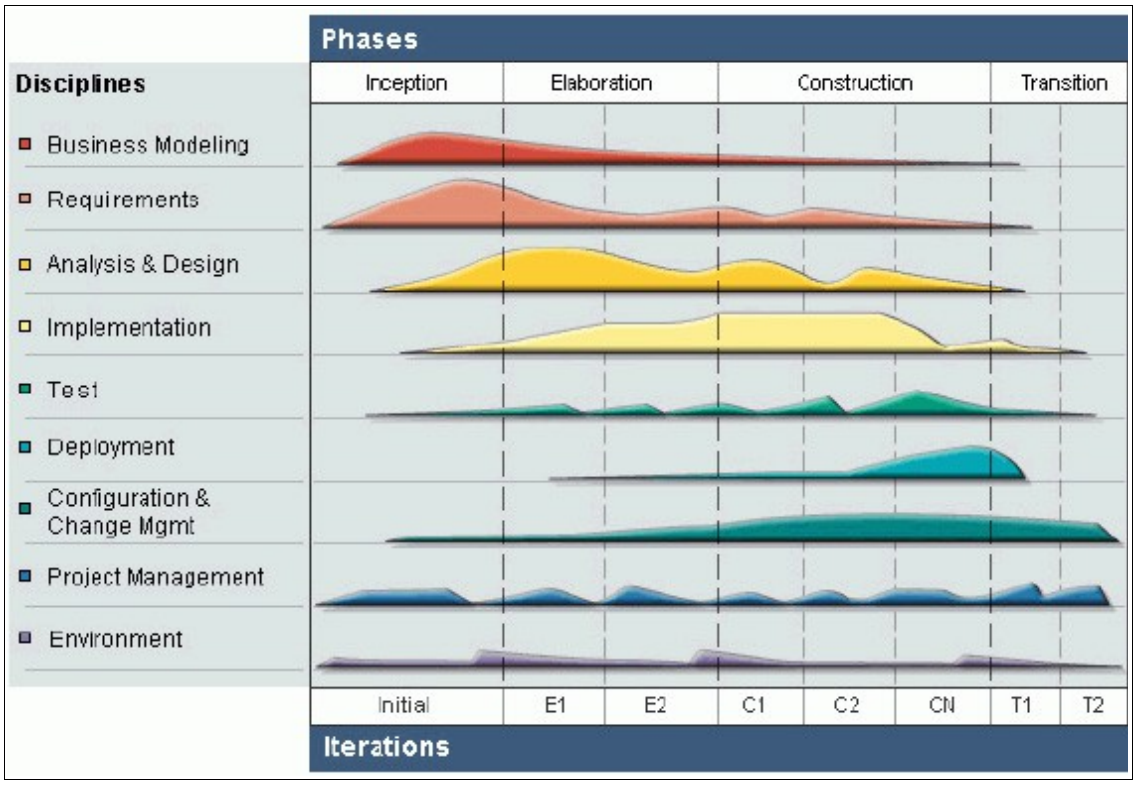


Figure 20 : Le processus unifié et ses différentes phases
(source http://www.ibm.com/developerworks/library/ws-soa-term2/)

Pour autant, le processus unifié est sujet à controverse dans la communauté agile. Il n'est pas considéré en tant que tel comme un *framework* agile dans la mesure où il reste très phasé et que chaque groupe d'itérations est dominé par un type d'activité [BOS04][ROT10]. De plus il contient, un nombre élevé d'artefacts qui rendent nécessaires un travail d'adaptation du processus au projet pour le rendre véritablement agile [KNI10]. Sur quels critères peut on qualifier d'agile, une méthode ou un processus ?

2.5.5. Critique de l'approche traditionnelle

Ces modèles sont aujourd'hui assez critiqués par les agilistes pour leur manque de souplesse et d'adaptabilité tout au long du projet. Ils souffrent de contraintes qui peuvent mettre le projet en situation d'échec.

Dans une approche traditionnelle, la gestion de projet s'exécute autour de deux grands rôles. La maîtrise d'ouvrage que l'on désigne par le terme MOA et la maîtrise d'œuvre que l'on désigne par MOE. La MOA est responsable de l'ouvrage, c'est le client, le commanditaire. La MOE réalise l'ouvrage conformément aux besoins exprimés par la maîtrise d'ouvrage. Dans le cadre de projets informatiques, on rencontre parfois un chef de projet MOA et un chef de projet MOE, chacun représentant l'un des deux rôles sur le projet.

La maîtrise d'ouvrage élabore un cahier des charges dans lequel elle spécifie le besoin. Sur la base de ce cahier des charges, la maîtrise d'œuvre conçoit le produit, le développe, et vérifie le fonctionnement du logiciel. La maîtrise d'œuvre livre le produit à la maîtrise

d'ouvrage qui peut alors effectuer la validation. Autrement dit, la maîtrise d'ouvrage valide que le produit rend bien le service attendu conformément aux besoins exprimés.

Dans l'approche traditionnelle, chaque phase s'exécute une fois et suppose que la solution retenue en amont est à la bonne. Lorsque les plans sont établis (on parle de pilotage par les plans), le chef de projet s'attache à les respecter [ROT10]. Dans cette approche, les besoins sont figés et contractualisés. La MOA forte de cette contractualisation ne s'implique pas toujours dans le projet comme elle le devrait ce qui conduit la MOE à travailler davantage sur la base de documents projets plus que sur la communication avec le client. Une des conséquences de cette mauvaise collaboration entre MOA et MOE est l'effet tunnel qu'elle engendre. L'effet tunnel est caractéristique de l'approche traditionnelle. Le client qui ne s'implique pas dans le développement du produit et laisse la maîtrise d'œuvre avancer fini par découvrir tardivement que le produit n'est pas aligné sur les exigences. Dans cette situation, c'est le suivi des plans qui prime.

Un autre problème de l'approche dénoncé par les agilistes est l'importance donnée à la documentation du projet en amont du développement. Comme le projet est piloté par les plans et que le développement doit refléter le contenu de cette documentation, il faut que celle-ci soit la plus complète possible. Mais chaque phase de l'approche nécessite aussi sa propre documentation pour que la phase suivante puisse être menée. Le temps passé sur la documentation est autant de temps pris sur le développement. Hors, d'après les agilistes, le bon alignement du produit sur les exigences ne peut être validé que sur une version opérable. Ce qui signifie aussi que l'écriture de code doit intervenir plus tôt dans le cycle et que l'ingénierie du test doit garantir le niveau de qualité requis, notamment dans une approche du développement piloté par les tests.

Les agilistes mettent en avant la complexité du développement logiciel. Il s'agit d'une activité difficile avec différents niveaux d'abstraction. Il est difficile de tout prévoir au travers des plans. Le *feedback* continu du client est essentiel dans le cycle de vie du développement du produit [ROT10].

2.6. L'approche agile

Depuis la fin des années 90, une autre approche du développement logiciel se répand peu à peu dans les organisations. Elle impacte les processus exécutés en gestion de projet et bouleverse la manière de concevoir et de produire du logiciel. Elle bouleverse aussi l'organisation en redéfinissant des rôles et des champs de responsabilités. Le management traditionnel est lui aussi impacté par cette approche « agile ». Aujourd'hui, les agilistes témoignent de la pénétration de plus en plus forte et rapide de l'agilité dans les organisations, depuis ces cinq dernières années.

L'approche agile désigne les méthodes agiles utilisées sur les projets de développement logiciel. Cependant, il s'agit plus de *frameworks* ou d'ensembles de pratiques. L'agilité désigne les valeurs que ces outils méthodologiques respectent.

Dans le développement logiciel, l'approche agile désigne un mode de fabrication par itération et par incrémentation [ROT10]. Chaque itération permet d'obtenir une partie du

produit potentiellement livrable [BEC04][SUT12]. D'itération en itération, le produit s'enrichit et est incrémenté de nouvelles fonctionnalités. Plutôt que de travailler sur une longue durée sur un gros ensemble qui reste longtemps inopérant, on travaille de manière rythmée sur de petits sous-ensembles que l'on rend rapidement opérables [KNI10] et dont on récupérera un vrai retour.

Les méthodes, les *frameworks*, les courants de pensées respectueux de l'agilité se sont développés depuis les années 80. On retrouve les méthodes DSDM, Crystal, Rad, Scrum, XP, FDD, ASD, Kanban, Scrumban mais aussi le "*lean software development*", le "*craftmanship software development"* et le mouvement devops. Au total, ce sont aujourd'hui plus de 60 pratiques (source institut agile) qui sont identifiées au travers de ces méthodes, *frameworks* et courants de pensées. Le *framework* Scrum mis au point par Jeff Sutherland et Ken Schwaber reste la méthode agile la plus pratiquée et la plus connue avec les pratiques XP (voir figure 21). Ces dernières sont généralement associées à Scrum et répondent pour certaines à des problématiques de génie logiciel.

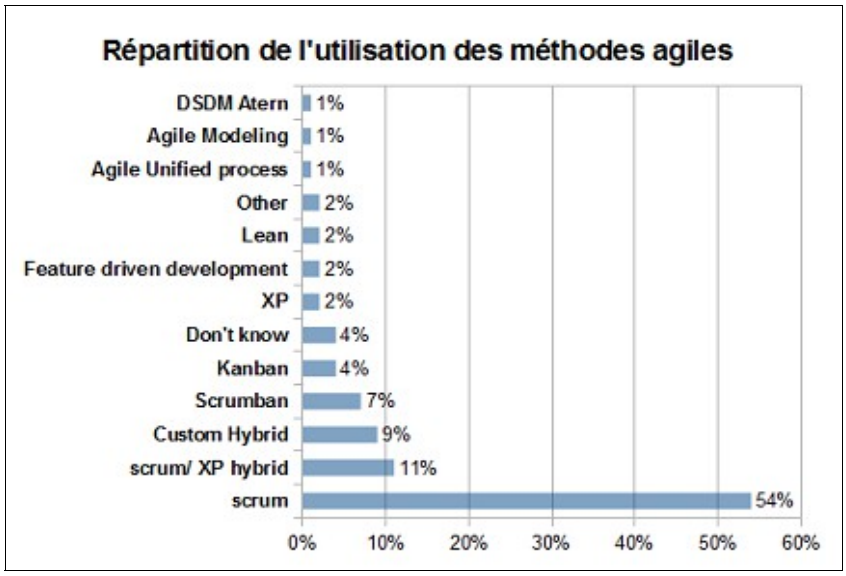


Figure 21: Répartition de l'utilisation des méthodes agiles (Source state of agile, survey 2012)

L'approche agile s'est formalisée en 2001, lorsque dix sept ingénieurs de renom en génie logiciel ce sont réunis autour de valeurs communes pour écrire leur manifeste agile du développement logiciel. Ces méthodes et bonnes pratiques sont leur réponse à la crise du logiciel et au taux d'échec important des projets.

Leur manifeste est le suivant [AGM01] :

« Nous découvrons comment mieux développer des logiciels par la pratiques et en aidant les autres à le faire.
Ces expériences nous ont amenés à valoriser :

- *Les individus et leurs interactions plus que les processus et les outils.*
- *Des logiciels opérationnels plus qu'une documentation exhaustive.*
- *La collaboration avec les clients plus que la négociation contractuelle.*

- *L'adaptation au changement plus que le suivi d'un plan.*

Nous reconnaissons la valeur des seconds éléments, mais privilégions les premiers. »

Ces 4 valeurs sont déclinées en 12 principes sous-jacents.

- *Notre plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée.*

- *Accueillez positivement les changements de besoins, même tard dans le projet. Les processus agiles exploitent le changement pour donner un avantage compétitif au client.*

- *Livrez fréquemment un logiciel opérationnel avec des cycles de quelques semaines à quelques mois et une préférence pour les plus courts.*

- *Les utilisateurs ou leurs représentants et les développeurs doivent travailler ensemble quotidiennement tout au long du projet.*

- *Réalisez les projets avec des personnes motivées. Fournissez-leur l'environnement et le soutien dont ils ont besoin et faites-leur confiance pour atteindre les objectifs fixés.*

- *La méthode la plus simple et la plus efficace pour transmettre de l'information à l'équipe de développement et à l'intérieur de celle-ci est le dialogue en face à face.*

- *Un logiciel opérationnel est la principale mesure d'avancement.*

- *Les processus agiles encouragent un rythme de développement soutenable. Ensemble, les commanditaires, les développeurs et les utilisateurs devraient être capables de maintenir indéfiniment un rythme constant.*

- *Une attention continue à l'excellence technique et à une bonne conception renforce l'Agilité.*

- *La simplicité – c'est-à-dire l'art de minimiser la quantité de travail inutile – est essentielle.*

- *Les meilleures architectures, spécifications et conceptions émergent d'équipes auto-organisées.*

- *À intervalles réguliers, l'équipe réfléchit aux moyens de devenir plus efficace, puis règle et modifie son comportement en conséquence.*

Ces quatre valeurs et douze principes sont les fondamentaux de l'agilité. Deux autres manifestes on été élaborés en 2005 et 2009. Il s'agit de la déclaration d'interdépendance [DOI05] et du manifeste pour l'artisanat logiciel [MSC09]. On retrouve dans ces deux manifestes les valeurs et principes sous-jacents initiaux.

2.7. Scrum

Scrum est un *framework* d'aide au développement de produit. Scrum a été inventé par Jeff Sutherland et Ken Schwaber au début des années 90 et formalisé en 1995 pour être présenté à la conférence OOPSLA de la même année. Scrum signifie mêlée en anglais. Aussi, il est souvent représenté par une mêlée de rugbymen. La métaphore, met l'accent sur l'importance du travail de groupe.

Le but du *framework* est de construire un produit opérable de manière itérative et incrémentale. Le produit est construit et évolue au fil des itérations par ajout de fonctionnalités. Pour effectuer cette construction l'équipe s'engage sur des cycles courts dont le périmètre, c'est-à-dire le nombre de fonctions à implémenter, est borné. A intervalles réguliers, l'équipe s'interroge sur ses pratiques et essaie de s'améliorer. Elle livre potentiellement un produit avant de recommencer une nouvelle itération jusqu'à la livraison finale du produit.

Remarques :
1. Le champ lexical de l'agilité et de Scrum en particulier est très marqué par la langue anglaise. Au sein du CNDP, je suis amené à traduire ce champ lexical en français, ou tout du moins, à utiliser les termes anglais avec parcimonie, pour communiquer plus facilement avec mes collègues. Cependant, j'ai volontairement choisi de ne pas faire cette traduction ici. Les termes anglais utilisés font référence dans les milieux initiés et permettront au lecteur de s'y retrouver aussi bien dans les contenus anglo-saxons que dans les contenus francophones traitant du sujet.

2. La définition Larousse pour le terme processus est la suivante, *« Enchaînement ordonné de faits ou de phénomènes, répondant à un certain schéma et aboutissant à quelque chose »*, aussi j'utiliserai indifféremment le terme méthode, *framework* ou processus pour présenter Scrum.

Scrum est un processus empirique [SUT11]. Le postulat est que la connaissance ne s'acquiert que par l'expérience et que la prise de décision n'est pas possible sans connaissance. Cette approche empirique permet d'améliorer la prédictibilité et la gestion des risques. Scrum repose sur trois valeurs essentielles : la transparence, l'inspection et l'adaptation. La transparence signifie que tout le monde doit partager la même vision du produit et la manière de le construire. L'inspection signifie le contrôle des éléments produits par le processus. Elle ne doit engendrer aucun impact sur la manière de produire. Enfin l'adaptation signifie l'ajustement du processus dès lors que ce dernier n'est plus aligné sur le but.

L'équipe Scrum est transversale. Elle rassemble toutes les compétences requises pour la construction du produit. Elle est également auto-organisée, ce qui signifie qu'elle décide elle-même de la meilleure manière de construire le produit. Le but recherché est la flexibilité, la créativité et la productivité. Scrum est un processus discipliné, itératif et incrémental qui vise à favoriser le feedback des utilisateurs.

Enfin, Scrum est un *framework* qui est une sorte de conteneur pour d'autres pratiques. Les règles du *framework* définissent l'articulation des trois composants que sont, les rôles, les cérémonies et les artefacts. Scrum est un *framework* léger, simple à comprendre, mais difficile à maîtriser [SUT11].

2.7.1. Les rôles

Le product owner : personne qui représente les utilisateurs du produit. Il porte la vision et la stratégie auprès des développeurs. Il représente le métier. Le *product owner* n'est pas un chef de projet même si certaines de ses activités s'approchent de ce rôle. Il a la responsabilité du produit et doit tout au long du projet faire émerger les besoins et prioriser leur implémentation. Il porte également la responsabilité de la validation du produit. Il participe aux tests ou tout du moins les coordonne. Il organise également les revues de prototype avec les utilisateurs et les différentes parties prenantes. Il veille aussi à leur implication sur le projet. Il a pour principale responsabilité de maximiser la valeur du produit grâce au travail de l'équipe.

Le Membre d'équipe : désigne le développeur, l'intégrateur, l'ingénieur qualité, le graphiste et toute personne qui participe à la construction du produit. En général, on considère que la taille maximale d'une équipe est d'environ 7 à 8 membres. Chaque membre de l'équipe est responsable de la construction du produit quel que soit son domaine de compétences. Scrum ne reconnaît qu'un seul libellé celui de membre d'équipe.

Le Scrum master : Tout comme le *product owner*, le *Scrum master* n'est pas chef de projet. C'est le garant du processus Scrum. Il s'assure que Scrum est bien compris par l'ensemble des intervenants. Il s'assure également de l'adhésion aux principes, aux règles et processus Scrum dans son ensemble. Il anime les cérémonies comme les rétrospectives ou le *daily meeting*. C'est un facilitateur pour l'équipe. Il participe à lever les obstacles qui empêchent l'équipe d'avancer. Il participe également à maximiser la valeur produite par l'équipe en optimisant les interactions de celle-ci avec l'extérieur. Le *Scrum master* aide le *product owner* en communiquant sur la vision du produit. Il aide l'équipe et intervient en tant que *coach* pour l'aider à s'organiser. Il intervient également en dehors de l'équipe pour aider l'organisation. Dans ce cas, il peut être amené à expliquer le fonctionnement de Scrum. Il peut aussi aider l'organisation à adopter Scrum plus largement.

2.7.2. Les cérémonies

Les cérémonies permettent de respecter les trois valeurs de Scrum que sont la transparence, l'inspection et l'adaptation, chaque cérémonie Scrum étant une occasion d'inspecter les pratiques et de les adapter.

Dans Scrum, l'itération est désignée par le terme *sprint*. Ces itérations sont des séquences de travail de deux à quatre semaines. L'exécution des *sprints* est rythmée par les cérémonies. Les *sprints* ont pour finalité de livrer un incrément utilisable et potentiellement livrable. Pendant un *sprint*, un changement ne peut être pris en compte s'il a une incidence sur le but à atteindre initialement présenté à l'équipe par le *product owner*. De plus la composition de l'équipe doit rester constante.

Sprint planning meeting : réunion qui marque le début d'une itération. Au cours de cette cérémonie, les exigences sont présentées aux membres d'équipe par le *product owner*. Une cotation de ces exigences est effectuée par les développeurs. Elles sont aussi priorisées par le *product owner*. A l'issue de la cérémonie, l'équipe doit avoir compris ce qui doit être réalisé. Elle s'auto-organise pour cela et dispose d'un *backlog* à jour (*sprint backlog*) pour le *sprint* qui

débute. De plus, elle doit être en mesure d'expliquer au *product owner* et au *Scrum master* comment elle va effectuer le travail.

Daily meeting : réunion quotidienne de l'équipe au cours de laquelle, chaque membre prend la parole et s'exprime sur ce qu'il a fait la veille, sur ce qu'il va faire le jour de la réunion et les problèmes éventuels qu'il rencontre. Les problèmes ne sont pas traités au cours du *daily meeting*. La réunion dure environ un quart d'heure. C'est une réunion rythmée d'échanges autour des travaux en cours et restant à exécuter. A l'issue de cette réunion, l'équipe replanifie le travail restant à réaliser sur le *sprint* en cours. Elle doit être en mesure d'expliquer au *product owner* et au *Scrum master* comment elle va effectuer le travail restant.

Sprint review : cérémonie de revue au cours de laquelle l'équipe présente au product owner à la fin de l'itération le travail accompli sur le produit. Elle effectue une démonstration des fonctionnalités implémentées. Le but de cette revue est d'obtenir un retour de la part du *product owner* et des parties prenantes. Au cours de cette cérémonie, le *product owner* peut être amené à faire évoluer le contenu du *sprint* suivant et à changer les priorités des exigences exprimées. La revue permet d'aligner le travail de l'équipe sur le produit. Le *product owner* sait à l'issue de la revue ce qui est implémenté dans l'incrément. L'équipe explique les difficultés qu'elle a rencontrées au cours du *sprint*.

Sprint retrospective : séance de travail qui vise l'amélioration des pratiques de l'équipe. Au cours de cette cérémonie, l'équipe s'interroge sur sa manière de travailler et de construire le produit. Elle peut être amenée à abandonner certaines pratiques, à en adopter d'autres ou plus simplement à les améliorer. L'objectif visé est d'améliorer la livraison de valeur. Cette cérémonie se déroule entre la démonstration (*sprint review*) et le *daily meeting* qui marque le début du *sprint* suivant.

Remarque : Pendant la cérémonie de *planning meeting*, la cotation est habituellement effectuée à l'aide de la pratique du *planning poker* (voir paragraphe 2.9.3.). Cette pratique n'est pas spécifique à Scrum et est courante dans les processus agiles. Par ailleurs, elle n'est pas décrite dans le guide Scrum de Jeff Sutherland.

2.7.3. Les artefacts

Dans Scrum les artefacts participent à la transparence sur la création de la valeur.

Product backlog : référentiel dans lequel sont portées les exigences du produit. On parle également de « *sprint backlog* » pour désigner les exigences qui doivent être implémentées durant l'itération. Le *product owner* est responsable du *backlog* et doit le tenir à jour. Les exigences au sein du *backlog* sont triées (par valeur, par priorité..,) pour donner aux développeurs l'ordre d'implémentation dans le produit. Si le *product owner* est responsable de l'expression des exigences, les développeurs sont responsables de l'estimation.

Sprint backlog : correspond aux éléments du *backlog* qui seront implémentés au cours de l'itération. Le contenu du *sprint backlog* ne peut pas changer pendant le *sprint*. Seule l'équipe est autorisée à en modifier le contenu. La progression de l'équipe au cours d'un *sprint* est mesurée au travers du *backlog*. Plus précisément, c'est en comptabilisant les éléments restants du *backlog* à implémenter que l'on mesure la progression.

Remarque : Le *product backlog* est généralement constitué de *user stories*. Ce formalisme (voir paragraphe 2.9.) n'est pas spécifique à Scrum et est courant dans les processus agiles. Par ailleurs, il n'est pas décrit dans le guide Scrum de Jeff Sutherland.

La figure 22 montre l'articulation des composants du *framework* entre eux et comment ils participent à la production d'un logiciel opérationnel («*working software* »).

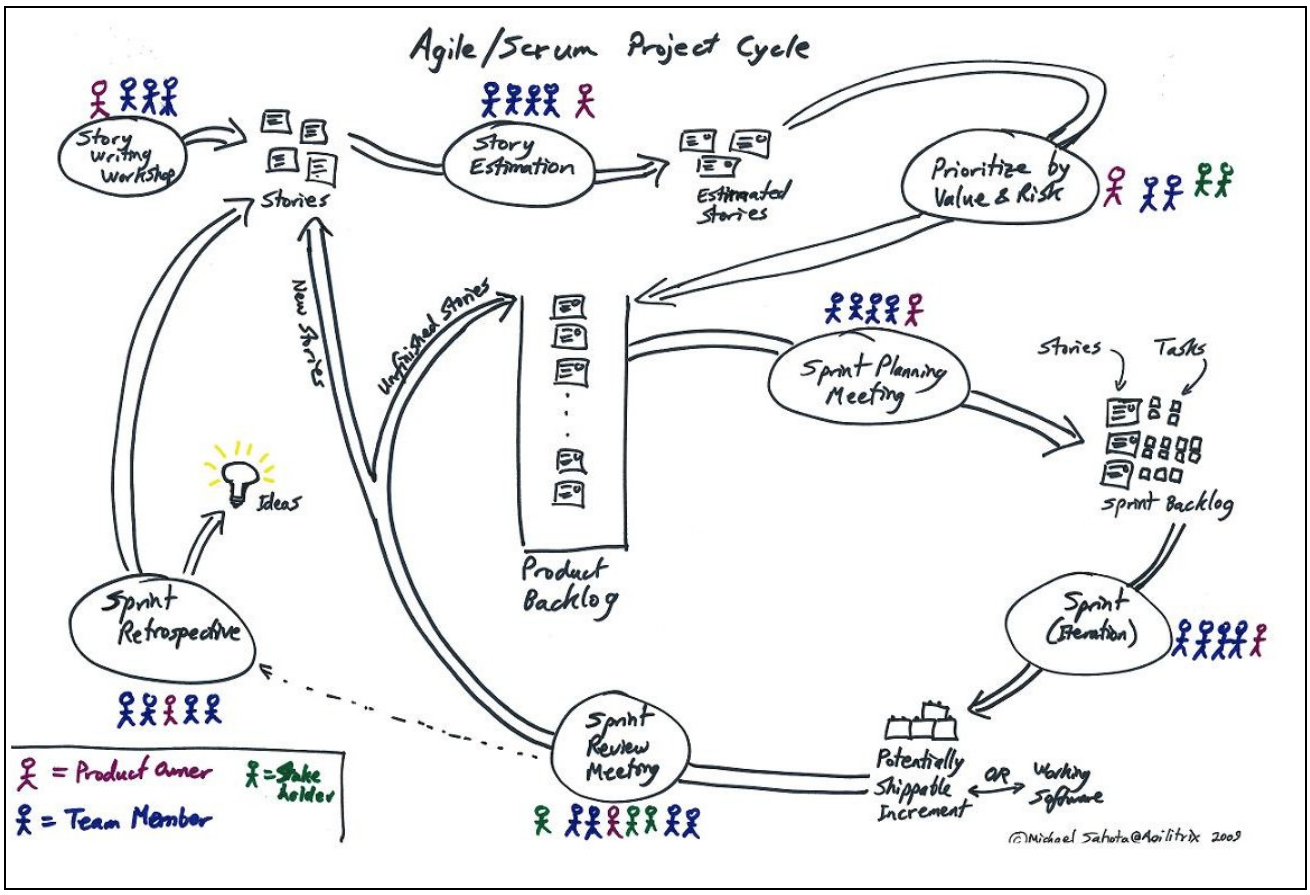


Figure 22 : Schéma du *framework* SCRUM (Thanks to Michael Sahota – agilitrix.com – Cclicense)

2.7.4. *Definition of done*

La définition du « fini » ou « *definition of done* » (DoD) [BOD11][ROT10] est l'ensemble des critères selon lesquels l'équipe considère que sa mission est remplie. Ces critères d'appréciation portent sur des activités qui génèrent la valeur et la qualité du produit. La définition du « fini » concerne chacun des éléments du *backlog*, un incrément, une version du produit.

Exemple de dod pour un élément de *backlog* :

- Test d'acceptation automatisé.
- Code revu.
- Fonctionnalité disponible sur le serveur de validation.

2.7.5. Time boxing

Le *time boxing* ou boîte temporelle est une pratique de gestion du temps. Elle permet de fixer une durée limite pour effectuer une tâche. Dans le cas de Scrum, certaines activités sont exécutées sur ce principe. Par exemple, le *daily meeting* ne devrait pas dépasser quinze minutes. Le *sprint planning meeting* ne devrait pas dépasser huit heures pour un *sprint* de quatre semaines effectué par une équipe Scrum standard (5 à 7 personnes).

2.8. eXtreme Programming (XP)

L'eXtreme Programming a été formalisé à la fin des années 90 par Kent Beck [ROT10]. Ward Cunnigham et Ron Jeffries ont participé à la naissance de l'eXtreme Programming en travaillant avec Kent Beck. Ce dernier a décrit l'eXtreme Programming à la fois comme une méthode, une philosophie du développement logiciel, et un vecteur du changement social au sein de l'organisation [BEC04][BOS04].

C'est une méthode légère qui concerne surtout la manière de développer du logiciel même si elle s'adresse aussi aux métiers. L'eXtreme Programming permet de s'adapter rapidement aux changements. Les cycles de développement sont courts (idéalement une semaine) et le *feedback* est rapide, concret et continu. L'eXtreme Programming encourage l'excellence technique par l'amélioration continue.

L'eXtreme Programming est présenté sur trois axes, les valeurs, les pratiques et les principes qui font le pont entre les deux. Les valeurs sont universelles mais elles sont trop abstraites pour servir de guide, d'autant plus qu'elles peuvent concerner d'autres domaines que le génie logiciel. Les valeurs donnent du sens aux pratiques de l'eXtreme Programming au travers des principes. Ces derniers permettent d'exécuter de nouvelles pratiques tout en étant respectueux des valeurs.

2.8.1. Les valeurs

- la communication - La communication entre les membres de l'équipe et avec le client permet de rester aligné sur les objectifs du développement. La communication permet de résoudre les problèmes de manière plus efficace en partageant l'information sans retenue. Elle donne du sens et permet de maintenir le niveau d'engagement. Elle permet et favorise également la transparence.

- la simplicité - La complexité qui engendre les gaspillages doit être évitée. En génie logiciel, les développeurs éviteront de produire du code et des architectures de code compliquées dans l'espoir qu'ils serviront plus tard pour de futurs et éventuels besoins. La simplicité prescrit de se limiter à la satisfaction des besoins immédiats.

- le *feedback* - Le retour permet de s'adapter. Plus tôt, l'équipe a connaissance d'un changement, plus tôt elle peut s'adapter et modifier le produit. Le changement est inévitable car il est impossible d'avoir une idée précise dès le départ. Le *feedback* est favorisé par des livraisons fréquentes et une implication du client.

- le courage - Le courage est nécessaire pour dépasser ses craintes, ses peurs et ses faiblesses. Il en faut pour accepter le changement, pour communiquer avec les autres membres de l'équipe et avec le client, pour exposer la situation même lorsqu'elle n'est pas bonne. Il en faut également pour communiquer sur des solutions jugées risquées ou compliquées.

- le respect - Le respect des membres de l'équipe, les uns vis à vis des autres, le respect du client, le respect du produit sont des gages de réussite d'un projet XP.

2.8.2. les principes

Les quatorze principes qui font le pont entre les valeurs et les pratiques XP sont les suivants :

- Humanité - Le développement logiciel ne doit pas aller à l'encontre des besoins personnels des membres de l'équipe. Ces mêmes besoins et ceux de l'équipe font l'objet d'un compromis. Le rythme de travail doit être soutenable.

- Economie - La priorité est la création de valeur la plus haute pour le métier. La conception incrémentale et itérative permet de générer rapidement des revenus en permettant de retarder les dépenses moins prioritaires.

- Bénéfices mutuels - Les activités de développement logiciel doivent s'effectuer dans un esprit de bénéfices mutuels. Les activités menées aujourd'hui doivent servir tous les intervenants du projet (le métier et les développeurs) dans une logique « gagnant-gagnant ».

- Similarité - Certaines activités ou solutions mises en œuvre dans un contexte particulier peuvent l'être dans un autre, y compris à une échelle différente.

- Amélioration - L'amélioration est toujours possible et ne doit pas être reportée sous prétexte d'attendre la perfection.

- Diversité - La diversité des solutions, des avis, des comportements, des compétences n'est pas un frein mais une opportunité à la création de valeur.

- Réflexion - La réflexion est nécessaire et alimente l'amélioration mais elle vient après l'action.

- Flux - La valeur produite par les activités de développement est livrée rapidement et régulièrement. Toutes les activités de développement sont menées en simultané pour permettre cette livraison de valeur au plus tôt.

- Opportunité - Les problèmes et les difficultés rencontrés sont des opportunités d'amélioration, d'apprentissage et de changement.

- Redondance - La redondance dans les pratiques ne doit pas être assimilée à du gaspillage. Elle permet parfois de résoudre un problème par une succession

d'interventions.

- Échec - L'échec n'est pas une perte de temps. Il est préférable de risquer l'échec en essayant quelque chose plutôt que de ne rien faire

- Qualité - La qualité est non négociable sur les projets et n'est pas une variable d'ajustement.

- Petits pas - De petites avancées certaines sont préférables à de grands changements qui risquent de rencontrer l'échec.

- Responsabilité - Avec la responsabilité vient l'autorité qui créée des distorsions de communication dans l'équipe. Seul l'individu peut décider s'il est responsable ou pas.

2.8.3. Les pratiques

Les pratiques sont avant tout situationnelles. Les équipes choisissent de les exécuter en fonction du contexte mais comme ces pratiques sont interdépendantes, elles dégageront tout leur intérêt si elle sont exécutées en cohérence et non de manière isolées. Les pratiques sont réparties en deux groupes (figure 23), les pratiques primaires et les corollaires.

Les **pratiques primaires** sont les suivantes :

- S’asseoir ensemble - Réunir l'équipe dans un même lieu améliore la communication et la collaboration sur le projet.

- L'équipe unifiée - Rassembler les compétences nécessaires au projet au sein d'une même équipe et développer la cohésion et le sentiment d'appartenance.

- L'espace de travail informatif - L'espace de travail de l'équipe est organisé de manière à favoriser la collaboration et le bien être mais il est aussi le support d'informations sur la santé du projet. Les informations sont affichées dans l'espace et visibles de tous.

- Travail sans tensions - Tenter de maintenir un rythme insoutenable est contre productif pour soi et pour l'équipe. Les membres de l'équipe sont encouragés à faire attention à leur santé et à leur rythme.

- Programmation en binôme - La programmation en binômes favorise la qualité du code, le partage des connaissances et la montée en compétence. Le binôme dispose d'un poste de travail. Il n'y a donc qu'un seul codeur mais les rôles sont alternés.

- Histoires - Les fonctionnalités que le système doit intégrer sont exprimées sur des cartes et affichées à la vue de tous. Les contraintes sont également exprimées sur ces cartes, notamment l'effort de développement et les critères d'acceptation liés à la fonction.

- Cycle hebdomadaire - Le cycle de développement commence par une séance de

planification. Les tests sont ensuite écrits et le cycle se poursuit par l'écriture du code. Lorsqu'il est terminé, les scénarios utilisateurs (*stories*) sont implémentés dans le système.

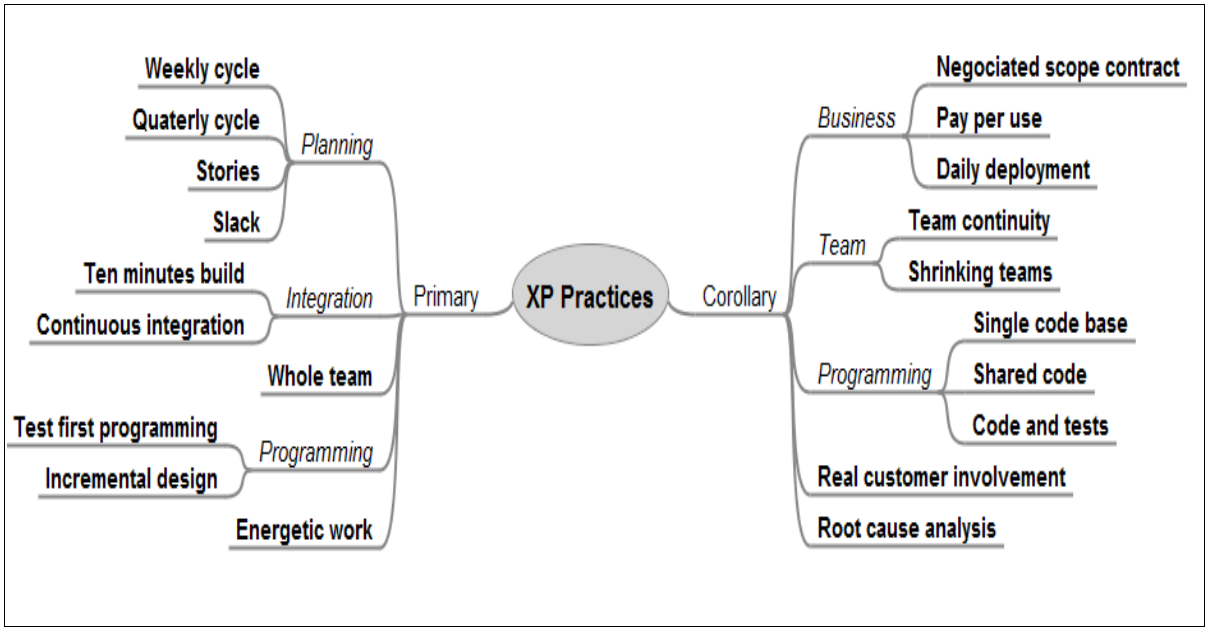


Figure 23: Schéma des pratiques de l'eXtreme Programming

- Cycle trimestriel - Le cycle trimestriel est propice à la planification d'activités qui ne peuvent être réalisées sur des cycles courts d'une semaine. C'est le cas d'opérations de maintenance ou d'amélioration.

- Tampon - Dans le cycle, il est conseillé de se garder une marge de manœuvre. Il peut s'agir de *stories* moins critiques qui sortiront du périmètre du cycle si l'équipe est en retard. Il peut s'agir aussi de plages de temps réservées à autre chose.

- Construction en dix minutes - La construction du logiciel doit être automatisée. Elle doit inclure le lancement de tous les tests et être effectuée en dix minutes pour pouvoir être lancée aussi souvent que nécessaire et donner ainsi le *feedback* attendu.

- L'intégration continue - Les changements sont intégrés rapidement (plusieurs fois par jour) pour limiter les problèmes d'intégration.

- Développement des tests - Les tests sont écrits avant le code. Le développeur commence par écrire des tests qui échoueront tant que le code qu'ils doivent tester n'est pas écrit. Cette pratique permet d'obtenir un code plus concis et plus propre. Le développeur se concentre sur le code utile.

- Conception incrémentale - La conception est l'objet d'une attention permanente tout au long de la vie du produit. Cette attention permet de conserver l'alignement du produit sur les besoins exprimés par le client.

Les **pratiques corollaires** sont les suivantes :

- Implication du client - La pratique vise à réduire les gaspillages de temps et l'inertie, en mettant en contact les vrais porteurs du besoin (et non un représentant trop éloigné) avec ceux qui vont le satisfaire.

- Déploiement incrémental - Il est préférable de déployer un système par lot et le plus tôt possible au lieu d'attendre et de le faire au travers d'un scénario de type « *big bang* ».

- Continuité d'équipe - La pratique vise à maintenir une continuité dans la constitution des équipes en laissant les gens qui se connaissent et se font confiance, ensemble. Au travers de cette pratique, l'organisation reconnaît la valeur de l'équipe et des échanges qu'elle génère.

- Compression d'équipe - Cette pratique permet de réduire la taille d'une équipe et d'éliminer les gaspillages.

- Analyse de la cause racine - L'exécution de cette pratique a pour objectif d'éliminer la cause d'un problème. L'objectif est également de s'assurer que le problème ne se reproduira pas de nouveau.

- Code partagé - Une fois que la responsabilité collective du code est assumée par l'équipe, n'importe qui doit pouvoir intervenir sur n'importe quelle partie du code à tout moment.

- Code et tests - Le code et les tests sont les deux artefacts à maintenir en permanence. Ils contribuent directement à la création de valeur pour le client. Les artefacts qui ne contribuent pas à cette création de valeur sont considérés comme des sources de gaspillage.

- Base de code unique - Il faut veiller à n'avoir qu'une seule base de code et éviter les versions multiples qui sont des sources de gaspillage. Le développement dans les branches ne devrait pas excéder quelques heures.

- Déploiement journalier - L'écart entre la version en développement et la version en production doit être le plus faible possible pour limiter les risques et augmenter le *feedback*. Pour cela il faut déployer au moins une fois par jour. Tout l'environnement de construction et de déploiement doit être automatisé et permettre des retours en arrière en cas de problème.

- Périmètre négocié - Le périmètre du projet est négocié dans le but de favoriser le *feedback* et la collaboration entre le client et le fournisseur. Plusieurs contrats sont signés tout au long du projet pour conserver un alignement optimal de la réalisation sur les besoins.

- Payer à l'usage - La monétisation de l'utilisation du produit conduit à une amélioration du produit grâce aux informations générées. Dans ce modèle, le *feedback* est plus

complet que dans un modèle de *release* classique où le client paye pour des licences.

2.8.4. Les rôles XP

Les rôles en eXtreme Programming ne sont pas figés. Les membres de l'équipe peuvent intervenir sur des activités qui ne font pas initialement partie de leur périmètre, de leurs responsabilités. L'alignement de l'autorité et de la responsabilité est un principe important de l'eXtreme Programming [BEC04]. Un individu n'impose pas à d'autres une manière de faire sans en partager les conséquences. Enfin, le savoir être d'un membre d'équipe XP est une aptitude essentielle au regard de l'importance de la communication dans certaines pratiques (pair programming, stand up etc.)

- Testeurs - Le rôle des testeurs est de définir et d'écrire les tests du système avant que le code soit implémenté. Les testeurs travaillent en collaboration avec les développeurs en apportant leur savoir faire sur ces activités. Ils aident également le client à définir des tests cohérents et pertinents sur les fonctions attendues.

- Concepteurs - Les concepteurs travaillent avec le client et l'aident à clarifier les besoins. Ils participent à l'écriture des *stories*. De manière générale, ils aident les développeurs à comprendre l'environnement utilisateur.

- Architectes - Les architectes participent aux tâches de développement comme les développeurs. Ils n'imposent pas l'architecture du système aux développeurs car celle-ci fait l'objet d'un consensus au sein de l'équipe. Les architectes doivent conserver une vision d'ensemble du système.

- Manager de projet - Le chef de projet est un facilitateur. Il facilite la cohésion et la confiance au sein de l'équipe en travaillant sur la communication. Il gère la communication avec les clients et les décideurs ainsi que la communication au sein de l'équipe. Il compile les informations produites sur le projet et les met en forme. Il facilite également les activités de planification.

- Responsable de produit - Il doit être capable d'expliquer les orientations fonctionnelles du produit. Il créé les *stories* pour les développeurs et doit pourvoir les prioriser et expliquer leur valeur pour le métier. Il facilite lui aussi les relations entre les utilisateurs et les développeurs.

- Utilisateurs - Les utilisateurs participent à la définition du système. Pour cela, ils aident le responsable du produit en apportant la connaissance du métier.

- Programmeurs : Les programmeurs estiment la charge de travail des *storie*s et les implémentent dans le système. Ils écrivent eux aussi les tests et participent à l'automatisation de la construction du système.

2.8.5. Découpage et gestion d'un projet XP

Comme Scrum, l'eXtreme Programming est une méthode itérative et incrémentale. Un

des fondements en agilité est le découpage du projet en grains plus ou moins fins en fonction du niveau de prédictibilité que l'on souhaite obtenir. XP est une approche empirique dite aussi approche « *bottom-up* ». On part de l'expérience acquise pour améliorer sa connaissance et donc sa prise de décision.

Le cycle de développement se décompose en trois étapes, l'exploration, l'engagement et le pilotage [BOS04][BEC04].

- L'exploration (*exploration*) : Cette première étape correspond à la communication du besoin. Sa durée varie en fonction de la maturité de l'équipe et de l'envergure du projet. Elle sera plus longue au début du projet et plus rapide par la suite. Le responsable du produit écrit les scénarios utilisateurs (*user stories*) et les travaille avec l'équipe. Les scénarios sont ensuite présentés aux développeurs en atelier pour être estimés. Le travail d'estimation peut conduire à l'écriture, à l'éclatement ou à la fusion de certains scénarios. Le produit est découpé pour être estimé (figure 24). Les éléments issus du découpage sont regroupés dans des composants (*component*).

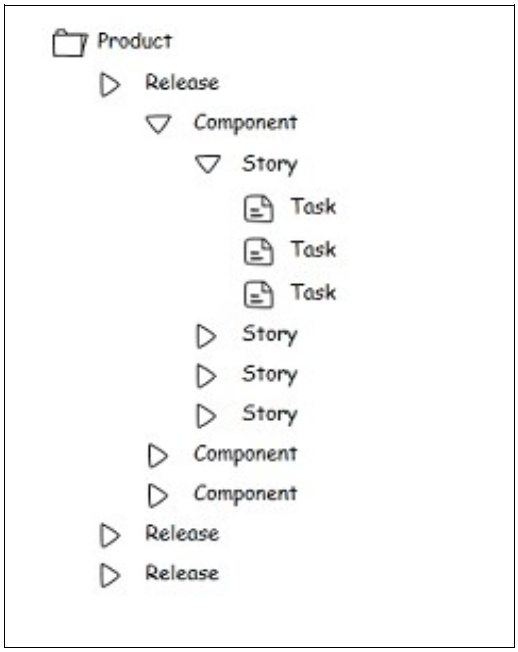


Figure 24 : Découpage d'un produit logiciel

- Un projet XP comporte plusieurs livraisons (release). Les développeurs estiment l'effort d'implémentation du scénario dans le produit. Cet effort est exprimé en points ou en jours idéaux. Le point d'effort est une valeur abstraite alors que la durée idéale correspond à une durée d'exécution sans perturbations (réunions, autres projets etc.) Les éléments sont estimés les uns par rapport aux autres en prenant des éléments étalon qui serviront de référence.

- L'engagement (*commitment)* : Les scénarios sont triés et priorisés en fonction de l'importance qu'ils représentent pour le métier. L'équipe détermine sa vélocité pour pouvoir s'engager sur un nombre de scénarios choisis avec le responsable du produit. Pour cela elle se base sur les itérations précédentes. Ces scénarios sont ensuite placés

dans le plan de livraison (plan de *release*) établi en exploration. A l'issue de l'étape d'engagement l'équipe sait ce qu'elle doit implémenter dans le produit.

- Le pilotage (*steering*) : Cette étape correspond au développement du produit. L'implémentation des scénarios est pilotée et surveillée au travers des scénarios de tests écrits et joués avec succès au fur et à mesure de l'implémentation. Le pilotage et le cycle se terminent par la livraison du produit.

Ces trois étapes décrites sont exécutées de manière itérative tout au long du projet. Un projet XP comporte plusieurs livraisons (release) qui, elles-mêmes, sont réalisées en une à plusieurs itérations. Les trois étapes peuvent être conduites à ces deux niveaux de découpage du projet (figure 25). En fonction de l'envergure du projet, les étapes d'exploration et d'engagement seront menées en une seule étape. De la même manière, les deux niveaux de gestion (release et itération) peuvent être réunis pour n'avoir qu'un seul niveau [BOS04].

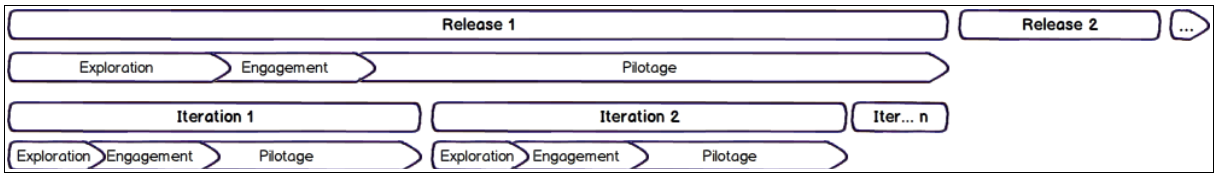


Figure 25 : Exemple de découpage d'un projet XP

2.9. *user story* - formalisme

2.9.1. Généralités sur le formalisme

Une *user story* est un scénario utilisateur issu d'un découpage fonctionnel du système ou du logiciel à développer [BEC04][COH04][ROT10]. Les *user stories* sont produites par le *product owner* (rôle Scrum) ou le *product manager* (rôle XP). Elles sont le support de communication entre les développeurs et le *product owner.* Le processus de développement est piloté par les *stories* (*story driven*) lorsque la mesure de l'avancement du développement est effectuée sur la base des *stories* implémentées.

Les *user stories* présentent 3 caractéristiques :

- **Carte** - La *user story* est matérialisée par une carte. Cette carte contient le texte de la *story* ainsi que des informations relatives à la priorité et à l'effort nécessaire pour son implémentation.

- **Conversation** - Initialement, le texte de la *story* ne contient pas tous les détails pour encourager la communication. Toutefois les détails les plus importants peuvent être portés sur la *story* pour permettre sa bonne implémentation. Une *story* peut donc être accompagnée de *mockups*, de diagrammes UML, de cas d'utilisation et de tout élément qui a du sens pour l'équipe et qui permettra l'implémentation de la *story* dans de bonnes conditions.

- **Confirmation -** Il s'agit ici pour le responsable de produit de définir les critères sur lesquels il jugera que la *story* est bien implémentée. Les développeurs utilisent ces critères pour coder les tests unitaires et les tests fonctionnels.

2.9.2. Présentation du formalisme

Le formalisme de la *user story* permet de se placer à un niveau fonctionnel suffisamment fin pour permettre d'exprimer toutes les fonctions attendues dans le système. C'est la description d'une fonctionnalité qui a de la valeur pour l'utilisateur final ou pour l'acheteur. Le formalisme ci-dessous est préconisé par les agilistes mais la clause exprimant la valeur métier peut ne pas être toujours présente.

Une *user story* s'exprime de la manière suivante :

```
En tant que <rôle>, je veux <exigence fonctionnelle>
        pour <valeur/justification métier>.
```

Exemple :

Carte :

```
En tant que client je peux voir des informations relatives aux produits dans le résultat d'une recherche pour faciliter ma sélection.
```

Conversation :

```
Pierre dit qu'il faut au moins afficher la description du produit, le prix et sa disponibilité.
```

Confirmation :

```
- Doit fonctionner avec une description vide.
- Doit fonctionner avec une description très longue.
```

Une bonne *user story* possède les 6 attributs suivants.

- ***I**ndependent* – Une *story* est, dans la mesure du possible, indépendante pour éviter les problèmes de planification et d'estimation.

- ***N**egotiable* – Elle est négociable. La communication doit restée ouverte. C'est pour cette raison que seuls les détails les plus critiques sont notés. L'expérience permet de savoir quel est le niveau de détail optimal.

- ***V**aluable* – Une *story* doit avoir de la valeur pour l'utilisateur final ou pour l'acheteur du produit.

- ***E**stimable* – Il doit être possible d'estimer l'effort à fournir pour l'implémentation. Les difficultés d'estimation liées au manque de connaissances du domaine ou au manque de connaissances techniques des développeurs devront être levées.

- ***S**mall* – La taille de la *story* doit être appropriée pour permettre l'estimation. Une *story* trop importante est en fait une *epic* (épopée) qu'il faudra scinder en plusieurs *stories*. Une *story* doit pouvoir être implémentée au cours d'une itération.

- ***T**estable* – Une *story* doit être testable à partir des critères d'acceptation.

2.9.3. Estimation des *stories*

Pour pouvoir piloter un processus de développement par les *stories* (*story driven process*) il faut que les développeurs procèdent à l'estimation des *stories* produites par le responsable du produit. Cette estimation est généralement effectuée à l'aide de la pratique du *planning poker*. Il s'agit d'une pratique collective qui vise à estimer l'effort de développement pour implémenter la *story*. Cette estimation porte en général sur le besoin fonctionnel exprimé dans la *story* et concerne toutes les activités liées à l'implémentation (interface homme-machine, couche métier, couche données, écriture des tests unitaires, documentation, discussion en cours de développement).

L'estimation est donc effectuée par les développeurs. Le responsable produit présente les *stories* les unes après les autres et les développeurs effectuent l'estimation pour chacune d'elles. Une fois que la présentation de la *story* est effectuée, chaque développeur s'exprime sur l'effort que représente la *story*. Si il y a des différences de cotation, une discussion s'engage et un nouveau tour de table est réalisé afin d'obtenir une convergence. L'estimation est ensuite portée sur la *story* avant de passer à la suivante.

Le point d'effort est exprimé par une valeur abstraite ou par une durée idéale (durée sans perturbations). Pour faciliter l'estimation la séance commence par l'estimation de *stories* qui servent d'étalons pour les autres estimations. Lorsque la séance est terminée, les estimations sont contrôlées les unes par rapport aux autres en se focalisant sur d'éventuelles dépendances et en regroupant les *stories* de même taille.

2.9.4. Planification des *stories*

La planification des *stories* est effectuée conformément à l'esprit des *frameworks* agiles (Scrum, XP...). Autrement dit, une fois que la *story* est estimée son implémentation est planifiée dans un plan d'itération ou de *release*. Pour cela, le responsable produit doit travailler sur la priorisation des *stories* entre elles. Il y a deux approches. Soit la priorité est donnée aux *stories* les plus risquées, soit elle est donnée aux *stories* qui ont le plus de valeur pour l'utilisateur. Cette dernière approche est préférée par les agilistes mais il est de la responsabilité du responsable produit de faire ce choix en prenant en compte les remarques des développeurs. Les opérations de factorisation et de règlement de dettes techniques peuvent s'avérer coûteuses lorsque les *stories* présentant un risque ont trop longtemps été repoussées.

Les itérations ont généralement une durée comprise entre 1 et 4 semaines. L'équipe doit déterminer sa vélocité, c'est à dire sa capacité à implémenter des *stories* ou bien encore sa capacité d'effort. Si les points d'effort d'une *story* expriment une durée idéale alors la vélocité exprime une durée idéale totale pour l'itération (somme des points d'effort). La vélocité peut aider à choisir une durée d'itération. Cette durée d'itération doit rester constante par contre la vélocité évolue au cours du projet. On estime qu'il faut environ 2 à 3 itérations pour la stabiliser. Lorsque la vélocité est déterminée et que la durée de l'itération est fixée, le responsable produit choisit les *stories* qu'il veut voir implémenter à la fin de l'itération.

2.10. Lean software development

Le *lean software development* est une discipline issue des principes de la démarche de *lean manufacturing* très connue dans le milieu industriel. Cette démarche est la conceptualisation par les chercheurs américains du système de production de Toyota (Toyota Production System). Le *lean software development* comporte des points d'accroche avec les *frameworks* et méthodes agiles. Certaines valeurs et certains concepts sont partagés. Qu'il s'agisse du *lean thinking,* du *lean office,* du *lean management* [PEZ10], du *lean software development* [POP03][KNI11] ou du *lean manufacturing,* tous s'inspirent des concepts originels du Toyota Production System conçu par l'ingénieur japonais Taiichi Ono dans les années 50-60.

Les 5 principes de la pensée *lean* sont les suivants :

- La valeur et les clients - Le principe est d'identifier ses clients et les activités qui ont de la valeur pour eux. Tout le reste est susceptible d'être supprimé. Dans une organisation, la valeur n'est produite que par une petite fraction de l'effort.

- La chaîne de valeur - Une fois que les clients sont identifiés, il faut identifier et cartographier l'enchaînement des activités qui génèrent de bout en bout la valeur. C'est la chaîne de valeur.

- Flux et gaspillages - Il s'agit ici d'éliminer tous les gaspillages qui entravent le flux de la chaîne de valeur.

- Flux tiré - La production fonctionne selon le principe du flux tiré. Toutes les activités exécutées dans la chaîné de valeur sont la réponse à une demande en aval.

- Amélioration - Le principe d'amélioration continue est le lien entre tous les principes et vise à éliminer par améliorations successives toutes les sources de gaspillages et à optimiser la création de valeur.

Les 7 principes du *lean software development* sont les suivants [POP03] :

- Eliminer les gaspillages - Tout ce qui n'apporte pas de valeur à l'utilisateur est du gaspillage. L'idéal dans la pensée *lean* est de livrer à l'utilisateur exactement ce qu'il veut et rien de plus, le plus rapidement possible. Les trois types de gaspillage sont les muda, muri, mura. Les muda sont tous les gaspillages qui n'apportent pas de valeur. Les muri représentent les gaspillages par excès (surcharge, sur-qualité.) alors que les mura représentent la variabilité, les ruptures de flux.

- Apprendre - Le développement logiciel est vu comme une activité qui nécessite un apprentissage accru pour être amélioré. Cet apprentissage est rendu possible par un cycle court dans lequel s'enchaînent les activités de développement exploratoire et de mesure rapide des résultats..

- Repousser la prise de décision - Dans un environnement changeant, il est conseillé de repousser la prise de décision jusqu'à ce que les conditions soient les plus favorables.

Face à l'incertitude, il faut plutôt adopter une démarche qui permettra de s'adapter au changement plutôt que de se fixer sur des choix trop tôt.

- Livrer le plus rapidement possible - En développement, le cycle d'exploration, de découverte et de livraison est critique pour le *feedback*. En réduisant le cycle, on améliore la connaissance. Réduire la chaîne de valeur autant que possible participe à une livraison plus rapide et à l'élimination des gaspillages.

- Donner de l'autonomie aux équipes - Parce que le rythme est soutenu et que la prise de décision est reportée, une autorité centrale ne peut coordonner les activités de l'équipe. Les membres de cette équipe sont les mieux placés pour prendre les décisions sur les tâches qui les concernent, personne ne peut prendre de meilleures décisions qu'eux. Par ailleurs les tâches à accomplir sont organisées au travers de plusieurs techniques, mécanismes et pratiques (*daily meeting*), représentation graphique, intégration continue, activités de tests...).

- Intégrité du produit - L'intégrité fait référence à la qualité du produit perçue par le client et la qualité de son architecture. Du côté du client, le produit doit être intuitif et répondre exactement à son besoin. Du côté de l'architecture, la maintenabilité, la cohérence et l'adaptabilité constituent l'intégrité conceptuelle.

- Vue d'ensemble - Un système n'est pas la somme de ces composantes mais de ses interactions. L'optimum global n'est pas la somme des optimums locaux. Il faut veiller à conserver une appréciation globale du développement et éviter les angles de vue influencés par la spécialisation.

2.11. Kanban

Kanban fait partie de l'approche *lean,* c'est un outil visuel de gestion de la chaîne de valeur. Les activités du processus sont cartographiées et gérées dans un tableau visible par l'ensemble de l'équipe (figure 26). Cet outil est largement utilisé par les équipes agiles. Dans le domaine du développement logiciel, il permet de cartographier le processus de développement et de faire apparaître les goulots d'étranglement, les ruptures de flux et les gaspillages (muda, muri, mura).

L'application de Kanban commence par le découpage du produit (ce découpage est une pratique importante des méthodes agiles) à réaliser en éléments de taille homogène dans la mesure du possible. Chaque élément est tiré dans la chaîne de valeur sur les activités du processus [KNI10][KNI11][LAD08]. La gestion des éléments dans la chaîne de valeur doit respecter deux principes.

- La limite de l'en-cours (*work in progress*) - Limiter l'en-cours sur les activités du processus permet d'améliorer la visibilité d'éventuels goulots d'étranglements. Un en-cours limité est plus motivant pour les équipes et donne un meilleur *feedback*.

.

- La gestion en flux tiré (*pull flow*) - Ce sont les activités en aval du processus qui tirent le flux de valeur jusqu'à la fin du processus. La limitation de l'en-cours facilite le flux

tiré en donnant plus de visibilité sur les capacités. Le flux tiré signifie que rien ne devrait être produit sur les activités en amont sans que celles en aval ne le demandent.

Grâce au tableau Kanban, on cherche à optimiser le temps de livraison (*lead time*), c'est-à-dire le temps nécessaire à un élément pour traverser la chaîne de valeur, en éliminant les gaspillages et les goulots d'étranglement. Il peut être utile de créer dans le système des *buffers* entre deux activités mais dans l'esprit du *lean* leur usage doit être limité.

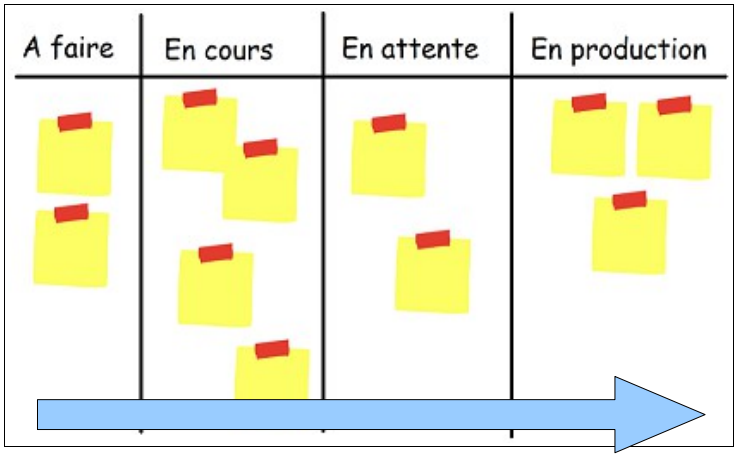


Figure 26: Tableau Kanban représentant le flux de travail.

2.12. Les indicateurs

L'approche agile est empirique. La connaissance vient de la pratique. Les décisions prises pour piloter un projet ou un produit s'appuient directement sur cette connaissance. Les indicateurs sont de deux types. Le premier type regroupe les indicateurs de pilotage du projet. Le second regroupe des indicateurs de qualité du produit. Quel que soit leur type, ils seront à choisir en fonction de la maturité de l'organisation et des équipes et de la complexité des projets.

L'**exécution agile des projets** peut être mesurée à l'aide des indicateurs suivants :

- Valeur acquise - La performance du projet est mesurée au travers de la valeur acquise. Le produit à fabriquer est décomposé en fonctions qui feront l'objet d'une cotation. Cette cotation peut être exprimée en durée, en coût ou en points d'effort. Au cours du projet, l'implémentation des fonctions permet de mesurer la performance grâce à la valeur acquise. C'est une information partagée par tous et habituellement représentée par des graphiques de type *burndown* ou *burnup (*figure 27).

- La vélocité - La vélocité de l'équipe est sa capacité à acquérir de la valeur au cours d'une itération. L'équipe s'engage au cours d'une itération, à implémenter un nombre d'éléments dont la somme de leur valeur correspond à la vélocité. La vélocité est ajustée au cours du projet grâce à l'expérience retirée des itérations exécutées.

- Durée d'engagement - Il s'agit de la durée entre le moment où l'on commence à investir des moyens sur une idée et le moment où cette idée commence à générer de la

valeur.

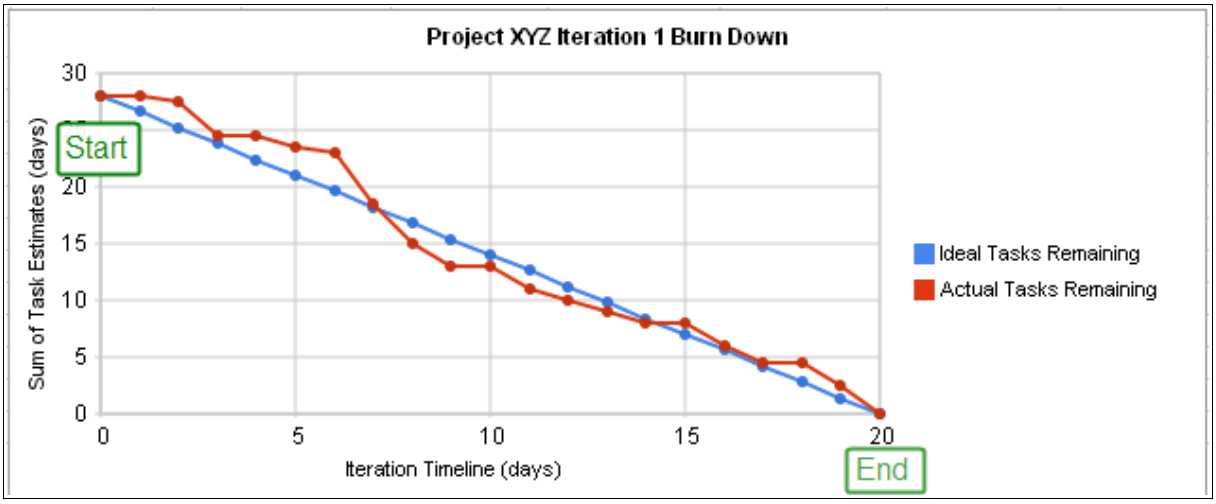


Figure 27: Graphique *burndown* représentant l'effort restant.

- Le facteur de charge - Le facteur de charge est utilisé pour déterminer la vélocité de l'équipe sur une itération. C'est un ratio entre une durée idéale et une durée réelle. Si le facteur de charge est de 2,5 et que l'itération est de 10 jours, la vélocité est de 4 jours.

- *Lead time* - Il représente la durée écoulée entre l'idée ou la découverte d'un défaut et la livraison du produit modifié. C'est l'indicateur de « *time-to-market* » que la vélocité ne peut pas donner. Il est représenté graphiquement par un diagramme de flux cumulés. (figure 28).

- Cycle time - C'est le temps écoulé sur une demande, entre le début des travaux et la livraison.

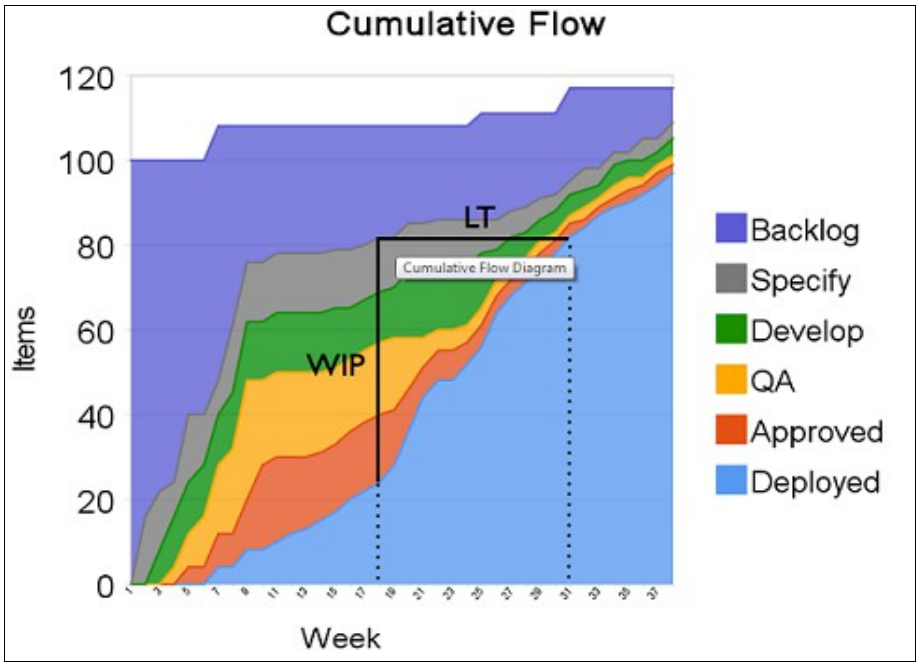


Figure 28 : Diagramme de flux cumulé.

La **qualité du produit** peut se mesurer selon les indicateurs ci-dessous :

- le nombre de tests : Le nombre de tests exécutés avec succès est considéré à la fois comme un indicateur de qualité et comme un indicateur de pilotage du projet. Il doit suivre la progression de la valeur acquise notamment si le succès des tests fait partie de la définiton du « fini » (*Definition of Done*).

- Nombre de défauts après développement : Le nombre de défauts découverts après la livraison du produit est retenu comme un indicateur de qualité du produit [BEC04]. Kent Beck considère qu'une équipe XP mature ne devrait pas générer plus d'une poignée de défauts par an (1 défaut pour 1000 lignes de code).

2.13. Pilotage et décision

Les projets menés par l'approche traditionnelle sont en général définis par la qualité et le contenu (figure 29). C'est la contrainte non négociable. Même si les coûts et les délais sont définis au lancement, ces derniers sont généralement les variables d'ajustement. En cas de glissement, les délais seront allongés et les coûts augmentés au delà des estimations initiales. Pour l'approche traditionnelle on parle aussi de conduite par les plans (*plan driven software development*) [ROT10][BOS04] ou bien de conduite par les documents (*document driven*) [BOE88]. Avec l'approche agile, l'ajustement se fait sur le contenu de la livraison. On parle alors de pilotage par la valeur. Ce pilotage par la valeur (*value driven software development*) et par le périmètre fonctionnel permet de contenir les coûts et les délais qui sont les contraintes.

Le pilotage par la valeur nécessite un changement de comportement de la part du client. Dans l'approche agile, il est de la responsabilité du client de maximiser la valeur produite tout au long du projet. La valeur est maximisée, entre autres, au travers d'une émergence des besoins et des spécifications associées, tout au long du développement (principe fort en agilité, présent dans le manifeste agile). La responsabilité du client est engagée et ce dernier doit se rendre disponible et participer activement au projet qu'il a initié.

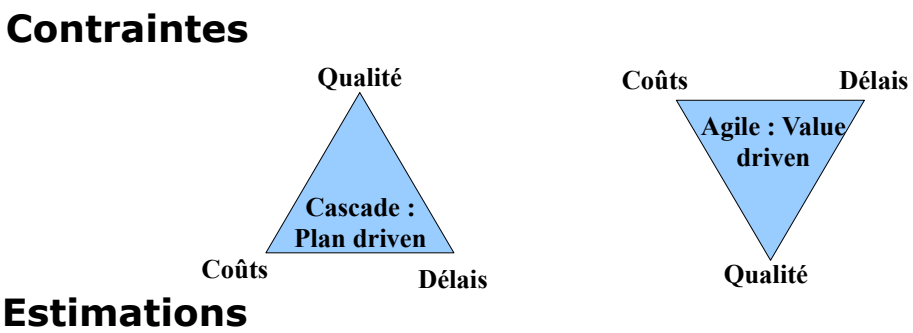


Figure 29 : Différence des approches

La décision d'engager des moyens sur un projet est prise sur la base d'éléments tangibles. La difficulté est bien de produire cette base qui permettra aux décideurs de lancer

ou non le projet. L'analyse en avant-projet permet de créer les conditions favorables à cette prise de décision. La différence entre l'approche traditionnelle et l'approche agile réside dans les moyens engagés et dans la manière de conduire ses activités d'exploration et d'estimation. Plus cette étude sera précise et plus elle engagera de moyens en avant-projet. Ces derniers seront fonction de l'envergure du projet et des risques identifiés.

Avec une approche agile, les besoins sont étudiés en avant-projet à un niveau plus macroscopique. Ce niveau d'étude permet de dégager une architecture, un découpage des grandes fonctionnalités du systèmes et une estimation suffisamment fiable pour permettre la prise de décision [BOS04]. En avant-projet, une *roadmap* est établie et fixe les livraisons estimées du produit. Il appartient au client de fixer les priorités sur les grandes fonctions qu'il souhaite voir implémentées dans le produit et surtout à quel moment il souhaite en disposer.

Chaque livraison peut faire l'objet d'un avenant à contrat cadre. Les livraisons sont normalement opérables et les fonctions qui engendrent le plus de valeur pour le client sont normalement celles développées en priorité. En cas de problème et en fonction de ce que le contrat prévoit, le client peut décider de stopper le projet ou de poursuivre. Le processus favorise et encourage le *feedback* du client et permet de conserver l'alignement du produit sur les exigences du client.

3. Première expérience de l'agilité : Le projet « CCR ».

3.1. Présentation du projet Catalogue Chèques Ressources

Au printemps 2009, le ministère de l'éducation nationale lançait le projet « Ecoles Numériques Rurales», dit projet ENR. Ce projet faisait partie d'un plan de relance en deux volets. Le premier visait à doter les écoles rurales du premier degré, de matériels numériques tels que des tableaux blancs interactifs ou des classes mobiles. Le deuxième volet consistait en la dotation de ressources pédagogiques associées. C'est sur ce deuxième volet que le projet ENR fut lancé. Il prévoyait le développement d'un catalogue de ressources, accessible sur internet qui permettrait aux écoles sélectionnées de venir choisir les ressources qui les intéressaient. L'achat de ces ressources était subventionné par une enveloppe de 9 millions d'euros allouée dans le cadre du projet. Le CNDP a construit la plate-forme et en a été l'opérateur. L'inspection générale (voir note inspection générale n°2011-073) a dressé un bilan globalement positif de l'opération avec toutefois quelques réserves sur la plate-forme, notamment sur le moteur de recherche. Les problèmes fonctionnels suivants ont également été relevés :

- Les écoles devaient consommer leur subvention en une seule commande.
- La sémantique des libellés était parfois confuse.
- Le catalogue était trop ouvert et proposait trop de ressources conventionnelles en dehors du champ numérique.

En 2010, les acteurs du projet ont demandé le renouvellement de l'opération en l'étendant aux écoles du second degré. L'opération ENR a donc pris fin au début de l'été 2011 avec le lancement d'un nouveau dispositif qui devait s'adresser cette fois aux écoles du premier et du second degré. Ce nouveau projet a alors été baptisé « Catalogue Chèque Ressources » ou projet CCR.

Le dispositif CCR visait donc à permettre aux écoles du premier degré et aux établissements du second degré, l'achat de ressources numériques à l'aide d'un chèque ressource. Le montant de ce chèque ressource dépendait de la nature du projet pédagogique porté par l'établissement. Il était prévu que le ministère et les collectivités locales abondent les comptes des établissements en fonction des projets pédagogiques retenus par les inspections académiques. De plus, le catalogue était constitué par les éditeurs. Ces derniers saisissaient dans un *back office* les fiches descriptives de ressources qu'ils souhaitaient voir apparaître dans le catalogue. Ces notices descriptives étaient ensuite validées par les équipes du ministère. Le projet CCR consistait donc à réaliser trois modules spécifiques.

Le premier était le catalogue accessible depuis internet. Ce catalogue devait permettre aux établissements de naviguer parmi les nombreuses ressources, de prendre connaissances de toutes les informations disponibles sur ces ressources et enfin de constituer une sorte de panier d'achat.

Le second module était un *back office* de gestion des notices descriptives. Les éditeurs devaient pouvoir ajouter de nouvelles ressources et modifier celles existantes. Le ministère quant à lui, devait disposer d'un *workflow* de validation pour traiter les nouvelles ressources. Il s'agissait ici de valider la ressource pour la faire apparaître au catalogue ou au contraire de la

rejeter et de motiver la raison du rejet.

Enfin, le troisième module devait permettre de gérer l'enveloppe de 10 millions d'euros allouée à l'opération. Cette enveloppe était ventilée sur les différentes académies retenues dans l'opération. Les collectivités locales devaient également abonder les comptes des établissements. Les comptes des établissements étaient également gérés dans ce module.

La plate-forme a été ouverte à la rentrée 2011 et devait être opérationnelle jusqu'à la fin de l'année scolaire 2014, elle sera finalement fermée à la fin de l'année 2013.

3.2. Démarrage

3.2.1. Élaboration de la charte projet

Le projet CCR n'a pas fait l'objet d'un lancement de projet conventionnel sponsorisé. Ce qui signifie qu'aucune charte projet n'a véritablement été produite. Le CNDP travaillait depuis plusieurs semaines sur le projet dans l'attente de se voir confier l'exécution. Les différents personnels du CNDP sont intervenus sur le projet au gré des besoins et des ateliers de travail.

Puis, c'est au cours du comité de pilotage du 16 mars 2011, que le ministère a demandé au CNDP de fournir une première maquette de ce que pourrait être le *back office* de saisie des notices de ressources. J'ai donc présenté des maquettes de principe au comité de suivi du 29 mars. J'intervenais alors en tant que représentant de la DSI et référent technique.

A l'issue de ce comité, la DGESCO produisit quelques jours après une note dans laquelle figurait les principaux besoins fonctionnels au sujet de la plate-forme. Pour le CNDP, cette note faisait office de cahier des charges et sonnait le « *go* » sur le projet. A partir de cet instant le développement a pu commencer.

3.2.2. Identification des parties prenantes

Le projet CCR réunissait des acteurs importants du domaine de l'édition et de la diffusion de ressources pédagogiques. Il faut garder à l'esprit qu'il s'agit d'un marché concurrentiel fortement influencé par l'environnement politique [ARC01]. Ce projet a nécessité une communication travaillée auprès des acteurs de la filière.

Le premier acteur était donc le ministère de l'Éducation Nationale. Plus précisément, pour le ministère les services impliqués étaient :

- la Direction Générale de l'Enseignement Scolaire (DGESCO) et son bureau (A3-2) des usages du numérique et des ressources pédagogiques (voir organigramme du ministère et de la DGESCO en annexes 3-a et 3-b) pour le pilotage du projet. Ce service est habituellement celui pour lequel le CNDP assure l'exécution de projet divers. Dans le cadre du projet CCR, le bureau A3-2 était donc notre donneur d'ordre.
- L'Inspection Générale intervenait pour émettre un avis et des recommandations sur la

conduite du dispositif. Les inspecteurs généraux intervenaient également en qualité d'experts pédagogiques.

- Le Cabinet du Ministre était également partie prenante et pouvait être impliqué en cas de désaccord avec les éditeurs privés. Un conseiller pouvait siéger au comité de pilotage du projet.

Le deuxième acteur est le réseau Scéren. Le CNDP ayant déjà géré le projet ENR, le ministère a naturellement fait appel à l'établissement pour le projet CCR. Plusieurs services ont été sollicités.

- La Division des Systèmes d'Information est intervenue sur les développements et l'hébergement de la plateforme.
- La Direction de la Production a été sollicitée pour la production d'éléments graphiques.
- La Direction commerciale s'est exprimée sur un certains nombre de points relatifs à la conduite du dispositif puisqu'elle était déjà en responsabilité sur le sujet pour le projet ENR et que les processus de gestion étaient semblables.
- Le CRDP de Marseille intervenait en conseil sur le projet car il développait à la même période une solution logicielle de catalogage de ressources numériques.

Enfin, le troisième acteur était constitué du secteur privé. Les éditeurs privés étaient représentés aux instances par leurs syndicats.

- Les représentants des plates-formes de diffusion « Le kiosque numérique de l'éducation » et « le canal numérique des savoirs » sont intervenus sur le projet pour émettre leurs exigences vis à vis de la plate-forme CCR.
- Les représentants des éditeurs sont principalement intervenus dans les instances pour surveiller le déroulement du projet.

Le projet a été rythmé à partir du mois de mars, par les réunions des instances de pilotage et de suivi. Ces instances ont été constituées par le ministère et leur réunion s'effectuait dans les locaux du ministère à Paris. Côté CNDP, il n'existait pas d'instance particulière hormis un groupe de travail qui se réunissait quotidiennement au début du projet. La composition de ce groupe a varié tout au long du projet en fonction des besoins. Globalement, le groupe était constitué des personnels qui avaient participé en avant-projet aux réflexions et préparation entre l'automne 2010 et janvier 2011. Les travaux étaient pilotés et animés par le directeur adjoint du CNDP.

Entre décembre 2010 et janvier 2011, en tant qu'adjoint au DSI et responsable de l'équipe de développeurs, j'intervenais de plus en plus fréquemment sur ce dossier pour travailler au sein de ce groupe sur la proposition fonctionnelle et technique que le CNDP pourrait être conduit à exprimer. Je suis également intervenu en soutien du directeur général et du directeur adjoint. Aussi, j'ai finalement été désigné pour travailler sur ce dossier comme référent technique et fonctionnel car je connaissais le fonctionnement de la précédente plate-forme développée dans le cadre de l'opération ENR.

Les rôles ont été partagés de la manière suivante. Le directeur adjoint du CNDP participait au comité de pilotage et au comité de suivi. Pour ma part, je l'accompagnais au

comité de suivi. Ce comité s'appelait aussi comité technique en référence à la nature des questions traitées. Nous avons principalement abordé, en son sein, les aspects fonctionnels et techniques de la plate-forme.

Enfin, le Directeur adjoint possédait la vision stratégique du projet. En soutien, je représentais le ministère et les éditeurs privés en portant le besoin fonctionnel de la plate-forme auprès des développeurs.

Le comité de pilotage traitait davantage de stratégie et de pilotage global de l'opération. Il traitait les questions qui ne pouvaient l'être en comité de suivi. Le comité de pilotage s'est par exemple exprimé sur l'opportunité de diffuser sur la plate-forme des ressources conventionnelles imprimées, comme dans le cadre de l'opération ENR.

Le comité de pilotage était composé :

- du chef de bureau A3-2,
- du sous-directeur des programmes d'enseignement et du développement numérique,
- d'un conseiller du cabinet du ministre,
- du directeur adjoint du CNDP,
- des représentants des éditeurs privés,
- d'inspecteurs généraux.

Le comité de suivi se situait davantage sur le côté opérationnel du projet et sur la réalisation de la plate-forme. Nous traitions des questions de plannings et de disponibilités des fonctionnalités. C'était pour moi l'occasion d'obtenir des réponses plus précises à mes questions. Le développement de certaines fonctions dépendait de la capacité des éditeurs à répondre à ces questions. Aussi, le comité de suivi permettait au ministère d'observer le niveau d'implication des acteurs sur le projet.

Le comité de suivi était composé :

- Du chef de bureau A3-2 et de l'un de ses collaborateurs.
- Du directeur adjoint du CNDP et de moi-même,
- Des représentants des équipes techniques des éditeurs privés.

3.3. Planification

3.3.1. Élaboration de l'échéancier

Le projet n'a pas véritablement été lancé de manière officielle. Les premières réunions de travail au ministère et au CNDP se sont tenues entre l'été et l'hiver 2010. Le projet était alors en « gestation ». Il était simplement demandé au CNDP de faire des propositions fonctionnelles et techniques autours d'une nouvelle plate-forme. Puis nous avons glissé au fur et mesure des réunions vers la réalisation du projet.

Le calendrier prévisionnel de réalisation établi à la fin de l'hiver 2010 était celui-ci :

- **Février 2011** : Le cahier des charges de la plate-forme CCR devait être rédigé. Le chef de projet et les intervenants côté ministère et côté CNDP devaient être désignés. Les instances de pilotage et de suivi du projet devaient également être constituées et les membres identifiés.

- **Mars 2011** : Le cahier des charges devait être validé par le comité de pilotage.

- De **Mars à Mai 2011** : La plate-forme devait être développée et opérationnelle pour permettre la saisie des notices descriptives des ressources.

- De **Juin à Juillet 2011** : Le ministère devait lancer les actions de communication vers les acteurs concernés (éditeurs, collectivités locales, établissements). Les fiches des ressources devaient également être saisies sur la plate-forme pour que la validation puisse s'effectuer.

- Fin **Juin 2011** : L'opération ENR devait prendre fin pour laisser la place au nouveau dispositif CCR et à sa nouvelle plate-forme.

- **Septembre 2011** : Le module permettant de gérer les subventions et les comptes des écoles devait être opérationnel. La plate-forme CCR devait également permettre aux établissements de commander.

3.3.2. Identification des risques

Plusieurs réunions préparatoires ont eu lieu entre l'automne 2010 et le printemps 2011. Le CNDP et son donneur d'ordre étaient en réflexion sur la nature du dispositif et sur la manière dont il serait conduit. Il faut comprendre qu'à cette période la plate-forme ENR est toujours en service. Par ailleurs le dispositif ENR a fait l'objet de critiques par les éditeurs et l'inspection générale. Il était reproché à cette plate-forme de ne pas mettre tous les éditeurs sur un pied d'égalité. La navigation dans le catalogue de ressources était jugée fastidieuse. De plus les éditeurs se plaignaient d'inertie sur le règlement des commandes par l'opérateur (les écoles ne payaient rien puisque c'est l'opérateur de la plate-forme qui réglait les éditeurs).

Aussi, le ministère tout en sachant qu'une deuxième opération serait lancée sous un format un petit peu différent, était dans l'attente et dans la réflexion. Il s'agissait de déterminer les modalités financières et organisationnelles de cette deuxième opération et d'obtenir un « *go* » politique. A cette période, le CNDP sait qu'une deuxième opération sera certainement lancée et tente de se tenir prêt à en être de nouveau l'opérateur. Plusieurs services de l'établissement sont consultés et mobilisés par la direction générale pour produire un document sous forme de proposition fonctionnelle et technique. Toutefois ce travail est réalisé au CNDP sans qu'aucune commande formelle ne soit passée et sans aucune certitude sur la conduite d'une seconde opération.

Pendant cette même période, plusieurs risques ont été identifiés et levés avant que ne débute le développement de la plate-forme. Toutefois, ces risques n'ont pas fait l'objet d'une gestion du risque conventionnelle. Ils ont été abordés au cours des réunions de travail et levés

grâce aux décisions prises par la direction et aux informations apportées par les participants à ces mêmes réunions.

Le premier risque concernait l'éventuelle reprise d'éléments logiciels (*back-office* de gestion des notices, moteur de recherche) de la plate-forme de l'opération ENR. Le développement de cette plate-forme, avait été piloté par le service d'ingénierie documentaire du CNDP. A l'époque, le service travaillait en parallèle sur un projet de portail de diffusion de ressources d'envergure nationale. Ce projet se nommait « Canal Scéren ». Le développement de la plateforme « Canal Scéren » avaient été confié à un prestataire. Aussi, lorsque le projet ENR débuta, le service d'ingénierie documentaire proposa que ce même prestataire exécute les travaux de développement. L'architecture de la plate-forme ENR a donc été influencée par l'architecture de Canal Scéren dans la mesure où les pilotes techniques du projet souhaitaient une interopérabilité entre les deux plate-formes. Malheureusement, ce choix d'anticipation sur le besoin exprimé a engendré des choix d'architecture qui n'ont pas facilité la maintenance et l'exploitation de la plate-forme par la suite. Aussi il fut décidé assez tôt qu'aucun élément logiciel de la plate-forme ENR ne serait repris soit parce qu'il ne correspondait pas au besoin, soit parce que les composants n'étaient pas suffisamment fiables et opérables dans de bonnes conditions.

Le second risque concernait la disponibilité de la plate-forme pour la rentrée 2011. Nous savions que si l'opération CCR était lancée, la plate-forme devrait être opérationnelle dès la rentrée. Ce risque a été levé au fur et à mesure du développement par les choix méthodologiques (voir paragraphe 3.7.).

Le troisième risque se situait sur une orientation trop documentaire de la plate-forme alors qu'elle devait s'apparenter davantage à un site de commerce en ligne sur le plan fonctionnel. Le risque a été levé en maîtrisant le niveau d'intervention et d'implication du service d'ingénierie documentaire.

Le quatrième risque concernait le rejet de la plate-forme par les éditeurs si celle-ci n'était pas suffisamment ergonomique en *back-office* pour permettre la saisie et l'import des notices descriptives. Ce risque a également été levé rapidement au travers des choix méthodologiques, en fournissant dès le début du développement des éléments de spécification sur le fonctionnement de la plate-forme.

Enfin, l'intervention du CRDP de Marseille sur les opérations de développement a été jugée risquée. A cette même période le CRDP travaillait sur le développement d'une plate-forme similaire en utilisant des outils de la suite ORI-OAI. Il souhaitait mutualiser ses travaux en cours avec ceux qu'il fallait réaliser pour l'opération CCR. Le risque se situait d'une part sur le manque de souplesse en développement induit par l'utilisation des composants choisis et d'autre part sur les contraintes organisationnelles du développement dans la mesure où le CRDP ne prenait pas en charge la totalité de la problématique. Le CNDP était donc obligé de mener les travaux de développement complémentaires et cette répartition aurait ajouté de la complexité alors qu'il fallait aller vite. Le risque a été levé en confiant le développement à une seule équipe basée au CNDP(voir paragraphe 3.4.).

3.3.3. Estimation des coûts

Le projet a fait l'objet d'un suivi en comptabilité sur les dépenses engagées à la fois

pour le développement de la plate-forme et pour son exploitation. Cependant, aucune estimation de ces coûts en amont n'a été réalisée. Le CNDP s'est vu confier la réalisation de la plate-forme et son exploitation dans le cadre de ses missions de service public ce qui signifie que l'exploitation de ce dispositif a été financée par le budget de fonctionnement de l'établissement.

3.3.4. Définition du contenu

L'exécution du projet a été rythmée par les comités de suivi et de pilotage qui se réunissaient fréquemment pendant les premières semaines du projet. Un certain nombre de livrables ont été produits de manière régulière.

- Les comptes rendus de comité - Ils étaient produits par le directeur adjoint et destinés à la direction générale du CNDP. J'ai effectué la rédaction des comités de suivi. Ces comptes rendus étaient diffusés à l'équipe projet, au directeur adjoint et à la direction générale. Les échanges et la transmission de comptes rendus pour validation entre le ministère et le CNDP ont été très ponctuels.

- Le code source - Tout au long du projet, les développeurs ont été en mesure de fournir un code source stabilisé à chaque fin d'itération. Ce code source était documenté par les commentaires et les tests unitaires qu'il contenait.

- La documentation utilisateur - Les trois modules constituant la plate-forme ont fait l'objet d'une rédaction de documentation. J'ai assuré la mise à jour de cette documentation jusqu'en décembre 2011 avant que la cellule fonctionnelle du CNDP prenne le relais sur la maintenance de ce livrable.

- Le *backlog* et les *stories* - Le livrable le plus important du projet avec les comptes rendus des comités était le *backlog* de la plate-forme dans lequel était stockées toutes les *stories*. C'est autour de ce *backlog* que l'équipe a organisé ses activités.

- Des spécifications fonctionnelles et techniques - Nous avons produit des éléments de spécifications pour les modules de la plate-forme sous forme de *mockups*, de cas d'utilisation textuels (annexe 6), et de scénarii de test pour apporter plus de précisions aux *user stories* du *backlog*.

3.4. Exécution

En Avril 2011, les instances étaient définies et la note émise par la DGESCO permettait au CNDP d'engager les travaux de développement de la plate-forme. Le CNDP était désigné pour être l'opérateur de la plate-forme mais il fallait encore que l'établissement décide à qui il allait en confier la réalisation. A ce moment du projet, le choix devait s'opérer entre :

- La société PASS'Tech qui avait travaillé à l'origine sur le dispositif ENR.
- Le CRDP de Marseille qui exploitait un dispositif similaire et qui était en 2011 en

cours de développement d'une nouvelle plate-forme de diffusion.

- Une équipe du CNDP constituée de personnels ayant l'expérience du précédent dispositif.

Par ailleurs, il était trop tard pour publier un appel d'offre sur le projet. Le CNDP décida finalement de réaliser le développement de la plate-forme en interne. Les travaux avaient déjà commencé en mars. Nous avions décidé d'un commun accord avec le directeur adjoint de les débuter, en attendant qu'une décision soit prise, pour ne pas perdre de temps et limiter les risques de retard.

3.4.1. Constitution de l'équipe projet

Lorsque les maquettes proposées ont été acceptées au comité de suivi du 29 mars 2011, la direction du CNDP a dû prendre une décision rapide sur la composition de l'équipe qui assurerait la réalisation de la plate-forme. La direction générale décida de confier l'intégralité de la réalisation de la plate-forme à l'équipe de développeurs de l'établissement pour s'affranchir des risques identifiés (voir paragraphe 3.3.2.) et pour produire rapidement les premiers éléments demandés.

Nous avions peu de visibilité sur notre capacité à produire les différents éléments de la plate-forme dans les délais imposés. J'ai constitué une équipe de développement avec trois développeurs parmi lesquels il y avait deux développeurs seniors, l'un ayant un niveau moyen en java. Les deux autres avait davantage des profils de débutant sur cette technologie mais il était important qu'il puissent acquérir des compétences en travaillant sur ce projet. Le choix du nombre de développeurs et leur mobilisation temporelle sur le projet était indéterminée au début du projet. Le nombre de développeurs impliqués correspondait simplement au nombre de modules à développer (un module catalogue, un module de saisie des notices descriptives et un module de gestion des comptes écoles et des subventions) et leur durée de mobilisation était fonction du degré d'urgence. Lorsque le premier module commença à être opérationnel, l'équipe fut réduite à deux développeurs même si le projet était prioritaire sur le reste de l'activité. Les projets initialement planifiées ont été décalés.

Des personnels de la direction de la production sont intervenus sur le projet ponctuellement pour réaliser la conception graphique de la plate-forme et l'intégration HTML. Ces personnels n'ont pas pu travailler de manière « intégrée » à l'équipe projet. Ils ont reçu les consignes de la part de leur responsable hiérarchique.

3.4.2. Pilotage de l'exécution

L'exécution du projet a été réalisée en utilisant des pratiques issues des *frameworks* agiles. Ces pratiques sont présentées dans le paragraphe 3.7. Les outils utilisés pour supporter le pilotage de l'exécution sont présentés dans le paragraphe 3.8.

3.5. Surveillance

3.5.1. Vérification du contenu

Pour permettre la vérification des livrables et en particulier de l'application, nous avons produits des scénarios de tests. Ces scénarios avaient pour but d'aider nos collègues du ministère à prendre en main l'application et à vérifier que le comportement des fonctionnalités attendues était correct. Par ailleurs, il devait permettre de combler le manque d'automatisation des tests fonctionnels en permettant aux utilisateurs d'effectuer des tests manuels sur les fonctions de haut niveau. Ces scénarios de tests concernaient :

- L'inscription de l'éditeur, sa validation et sa connexion.
- La description d'une ressource par un éditeur et sa demande de validation.
- L'attribution d'une demande de validation à un expert du ministère et la validation

3.5.2. Compte rendu de performance

L'avancement des travaux de développement était communiqué aux différentes parties prenantes lors des comités de suivi. La performance de l'équipe était communiquée en nombre de fonctionnalités et de modifications implémentées par rapport à ce que nous avions prévu au comité précédent.

Les données issues des outils de suivi (Jira) n'ont pas été utilisées dans le cadre de cette communication. Nous passions également en revue les problèmes que nous pouvions rencontrer et qui pouvaient affecter notre capacité à fournir le travail dans les temps. A la fin des comités de suivi, je m'engageais en fonction des discussions tenues, sur le contenu des livraisons suivantes.

3.6. Clôture

En janvier 2012, nous sommes entrés dans un mode de maintenance évolutive et corrective, la cellule fonctionnelle du CNDP a alors assuré le support de l'application et mes activités en suivi de projet ont pris fin avec le transfert de toutes les informations dont je disposais. Ce transfert d'informations s'est opéré en plusieurs séances de travail.

Dans un premier temps, nous avons passé en revue les livrables de la plate-forme et la documentation associée. Nous nous sommes donc concentrés sur le fonctionnement de l'ensemble.

Dans un deuxième temps, j'ai transmis toutes les informations concernant le pilotage et l'exécution du projet. J'ai communiqué à la cellule fonctionnelle un compte d'accès sur les outils de suivi (Jira, Confluence). Ce compte permettait d'accéder, aux comptes rendus de comités de suivi et de comités de pilotage, aux scénarios de tests, à la documentation produite et aux prototypes (*mockups*) créés en cours de projet.

3.7. Pratiques agiles déployées

A l'issue du comité de pilotage du 16 mars 2011, je savais que le CNDP n'obtiendrait pas, de la part du ministère, d'expression du besoin formalisée. Compte tenu des indisponibilités des différents acteurs du projet pendant la période allant de juillet à août, il fallait que la plate-forme soit opérationnelle en juin pour permettre la saisie des notices de ressources. A cette époque, j'avais déjà étudié les méthodes agiles et il me semblait que mes connaissances étaient suffisantes pour en promouvoir l'usage au sein de l'équipe. J'ai donc proposé aux développeurs d'expérimenter certaines pratiques. L'idée générale était de mettre en place un cycle itératif et de produire une version opérable toutes les deux à trois semaines. Ceci devait nous permettre de faire régulièrement une présentation de notre travail et d'obtenir en retour les remarques des éditeurs et du ministère, ces derniers étant, comme le CNDP, les futurs utilisateurs de la plate-forme. Cette organisation coïncidait avec les réunions planifiées du comité de suivi. Le comité de suivi me permettait donc d'une part de montrer l'avancement du développement, et d'autre part d'obtenir des réponses aux questions fonctionnelles les plus critiques qui se posaient au cours du développement. Sur le plan méthodologique les pratiques de pilotage du développement étaient issues des méthodes agiles. Ces pratiques sont venues en complément des pratiques de gestion de projet habituellement exécutées au CNDP. Ces dernières sont en général articulées autour de l'organisation des instances, des groupes de travail ainsi que du *reporting*.

Pour la réalisation du projet CCR, j'ai donc proposé aux développeurs d'expérimenter certaines pratiques agiles. La configuration du projet permettait cette expérimentation dans la mesure où notre seule contrainte était de fournir une version opérable en juin. Nous avions en quelque sorte « carte blanche » et au regard des délais, personne ne nous a imposé une méthode de travail en particulier d'autant plus que notre méthode permettait au ministère de suivre en continu les travaux de développement et de l'impliquer sur les choix fonctionnels.

Nous avons expérimenté des pratiques issues des méthodes Scrum et XP. Cette expérimentation des pratiques et l'utilisation du champ lexical associé sont restées quasiment confidentielles. Autrement dit, le déploiement de ces pratiques est resté complètement transparent pour nos collègues du ministère qui n'ont pratiquement pas eu à acquérir de savoir faire particulier sur ces aspects méthodologiques.

J'ai proposé à l'équipe certaines pratiques Scrum pour la gestion du cycle de développement. Puis nous avons utilisé en parallèle des pratiques de génie logiciel davantage tirées de la méthode XP. Il m'a semblé difficile de déployer tout le processus Scrum. Je craignais que les développeurs ne trouvent la méthode inadaptée et trop éloignée de nos pratiques quotidiennes. Je craignais qu'ils finissent par rejeter l'idée même de travailler sur un cycle de développement en accord avec les valeurs de l'agilité. Et enfin, débutant, je craignais moi même de rater notre entrée en matière en visant trop haut. J'ai donc choisi de ne déployer et de ne proposer que les pratiques qui me semblaient les plus utiles. Dans les paragraphes suivants je concentre mes explications sur les pratiques que nous avons exécutées sans parler des logiciels qui les ont outillées. Ces outils sont présentés dans le paragraphe 3.8.

Les pratiques sélectionnées étaient les suivantes :

- Expression des besoins à l'aide des *user stories*.
- Estimation des travaux à réaliser en groupe de travail à l'aide du *planning poker*.

- Gestion des travaux au sein d'un *backlog* de produit.
- Intégration continue des développements.

Ces pratiques étaient concrètes pour l'équipe. Elles reposaient sur la création et la gestion d'artefacts identifiés (*stories*, code source, *backlog* etc.). A un niveau plus général, j'ai essayé d'être fidèle aux valeurs de l'agilité en facilitant la communication et la collaboration entre les acteurs du projet, et en acceptant l'incertitude et les changements tout au long de l'exécution du projet.

3.7.1. Postures managériales adoptées

En tant que responsable de l'équipe de développeurs, j'étais au gré des réunions identifié comme une sorte de référent fonctionnel et technique sur le projet. Il me fallait encore trouver la bonne posture face aux développeurs. Je ne voulais pas seulement expérimenter quelques pratiques agiles. Je voulais à l'occasion de ce projet me rapprocher le plus possible des valeurs de l'agilité. Autrement dit, il me fallait, dans le cadre du projet, sortir de mon rôle de responsable d'équipe pour en jouer un autre. J'ai, en fait, dû assurer deux rôles. J'ai tenu une position certainement peu courante en Scrum en assurant à la fois le rôle de *Scrum master* et celui de *product owner*.

Si le rôle de *Scrum master* n'est pas toujours reconnu dans les organisations, il était très important pour le projet. J'avais proposé aux développeurs d'expérimenter des pratiques mais j'étais le seul à avoir les connaissances pour expliquer, déployer et mettre en scène ces pratiques. Aussi, j'ai assuré ce rôle en essayant d'être au plus proche des préconisations des méthodes et *frameworks* pour l'exécution des pratiques sélectionnées, en m'appuyant sur mes connaissances et évidemment sur ma perception des pratiques.

J'ai également assuré le rôle de *product owner* en recueillant les besoins auprès du ministère tout au long du projet, et en les formalisant pour les développeurs. J'ai porté la vision fonctionnelle de la plateforme tout au long du développement.

Avant le projet CCR, nous avions dans l'équipe l'habitude du consensus. Au quotidien, les choix d'architecture système et logicielle sont discutés en groupe et ne sont pas effectués par une seule personne. Nous fonctionnons ainsi d'abord pour limiter les risques d'incompréhension et de mauvaise conception et ensuite pour impliquer chaque développeur et faire en sorte que l'information et les choix techniques soient partagés. En général je participe à ce travail, mais dans le cadre du projet CCR, je me suis efforcé de rester à l'écart des choix techniques réalisés par les développeurs pour me concentrer sur mon rôle de *product owner*. Cette posture devait aussi me permettre d'observer le comportement des développeurs. Dans un cycle de développement agile il est important que les développeurs soient en capacité de prendre des initiatives techniques de manière responsable, en étant respectueux de la qualité des développements réalisés. Enfin, j'ai insisté auprès des développeurs sur la nécessité d'accepter un haut niveau d'incertitude, sans quoi le développement ne commencerait jamais puisque nous n'avions pas d'expression formalisée du besoin.

3.7.2. Les *user stories*

La première pratique que j'ai présentée à l'équipe concernait l'expression du besoin. Je leur ai présenté le formalisme des *user stories*. J'ai expliqué aux développeurs qu'à partir de la note émise par la DGESCO nous pourrions effectuer un premier travail d'analyse. Cette note (annexe 4) donnait des informations sur les trois modules de la plate-forme. De plus nous avions l'expérience fonctionnelle de la précédente opération. Cette expérience nous permettait d'avancer des hypothèses de fonctionnement que nous pourrions faire valider très facilement avec ce formalisme.

J'ai donc écrit les premières *stories* à partir de la note émise par la DGESCO en m'appuyant sur l'expérience du projet ENR. Je me suis d'abord concentré sur le module « saisie éditeur » qui devait permettre aux éditeurs de saisir les notices et au ministère de les valider. J'ai écrit au moins une *story* pour chaque fonctionnalité attendue dans le *back-office*. J'ai également produit des prototypes d'interfaces à l'aide de notre outil de *sketching* pour accompagner les *stories*. Ainsi, j'avais au démarrage du projet, et avant le démarrage de la première itération, une trentaine de *stories* sur lesquelles je pouvais communiquer avec les développeurs. Par la suite, les échanges en comité de suivi avec nos collègues du ministère et les éditeurs privés me permettaient d'écrire de nouvelles *stories* pour préparer les itérations suivantes. La collecte s'est faite sur le mode de l'interview. Lorsque j'avais besoin d'information pour l'écriture des *stories* et que je ne pouvais pas attendre la prochaine réunion du comité de suivi, je posais mes questions par mail ou par téléphone à ses membres. A la fin du projet, au mois de janvier 2012, nous avions plus de 80 *stories* référencées et traitées. Ces *stories* ont été écrites et utilisées uniquement au sein du CNDP (voir le paragraphe 3.11.2.).

Les premières *user stories* formulées étaient les suivantes.

Exemple 1 :

```
CCR-4  En tant qu'éditeur, je peux saisir une notice descriptive pour alimenter le
                          catalogue avec cette fiche.
```

Exemple 2 :

```
 CCR-8 En tant qu'éditeur, je peux rendre une fiche indisponible sur le catalogue
pour me laisser le temps de la corriger en cas d'erreur importante sur la fiche ou
                 bien pour simplement l'enlever du catalogue.
```

Exemple 3 :

```
CCR-30 En tant qu'utilisateur MEN, je peux affecter un montant de subvention à une
 école ou à un ensemble d'écoles pour que ces dernières puissent commander sur le
                                  catalogue.
```

Les *user stories* n'ont pas subi de modifications en cours d'itération, même s'il a parfois été nécessaire de rediscuter de leur périmètre avant leur implémentation.

3.7.3. Le planning meeting

Une fois les premières *user stories* écrites, nous avons pu organiser la première

itération. Pour cela, j'ai présenté à l'équipe la pratique du *planning meeting*. C'est véritablement autour de cette cérémonie que tout le processus de développement du projet CCR s'est organisé. Nous avons organisé un *planning meeting* par itération jusqu'à la livraison du mois de juin, soit quatre *planning meetings*. La cérémonie se décomposait en trois temps, la présentation, l'estimation et la planification. Comme le projet CCR était un projet critique pour l'équipe et pour le service, toute l'équipe de développeurs assistait à cette cérémonie. De plus, c'était l'occasion pour chaque développeur d'expérimenter les pratiques, notamment celles liées à l'estimation. La cérémonie de *planning meeting* aussi appelée *sprint planning* marquait le début de l'itération.

- La présentation

Pendant la présentation, je passais en revue les *user stories* devant les développeurs. J'avais produit ces *stories* sur l'étude de la note émise par la DGESCO et sur la base de mes échanges avec les interlocuteurs du ministère. Pour la plupart, elles ont été par la suite travaillées avec les développeurs. Le but de cette lecture commune était ici de vérifier que nous avions tous la même compréhension du besoin. Si quelqu'un ne comprenait pas ce qui était demandé dans la *story*, je devais ré-expliquer et reformuler la fonctionnalité attendue jusqu'à ce que l'ambiguïté ou l'incompréhension soit levée.

- L'estimation

La création du planning pour l'itération en cours reposait sur l'estimation réalisée par les développeurs. J'ai présenté la pratique du *planning poker* aux développeurs. Il s'agissait d'évaluer le temps nécessaire pour implémenter dans le produit la fonctionnalité exigée. Pour préparer cette étape de travail, j'avais confectionné un jeu de cartes par développeur. Chaque jeu était composé de onze cartes numérotées de 0 à 100. L'estimation consistait à faire deux tours de table. Chaque développeur devait estimer la durée en choisissant une carte de son jeu. Le nombre porté par chaque carte correspondait à des durée « idéales », c'est-à-dire à un temps de travail sans interruption. Le développeur choisissait la carte correspondante ou la carte immédiatement supérieure et la gardait sur la table, face cachée. Lorsque l'estimation était terminée, les développeurs retournaient les cartes et dévoilaient leur estimation. Un premier tour de table commençait alors. Chaque membre de l'équipe s'exprimait rapidement sur sa cotation si le besoin s'en faisait sentir, particulièrement lorsque des différences de cotation existaient. Puis, l'équipe procédait à une deuxième estimation sur la base des cotations argumentées au premier tour de table. La cotation était ensuite portée sur la *story* et devait permettre par la suite la planification.

- La planification

Chaque *user story* était cotée grâce à la pratique du *planning poker*. Cette cotation permettait par la suite de mieux sélectionner les *user stories* qui feraient partie du *sprint*. Pour effectuer une sélection appropriée des *stories*, il nous fallait estimer la vélocité sur la durée de l'itération. Pour cela nous avons estimé le temps dont nous disposions en développement sur l'itération. Par exemple, sur la première itération, pour trois développeurs, nous avons estimé quinze jours de développement auxquels nous avons soustrait les jours de formation, les congés, le temps passé sur les cérémonies et une marge de manœuvre pour traiter d'éventuels demandes d'intervention sur d'autres produits. Nous avions finalement onze jours de développement sur la première itération. A partir de l'estimation réalisée sur les *user stories*, je

pouvais, en tant que *product owner,* choisir les fonctionnalités que je voulais voir implémentées en priorité. Ce travail de priorisation se faisait en concertation avec les développeurs pour tenir compte d'éventuelles contraintes d'architecture logicielle.

Le but de ces pratiques est de limiter les erreurs de planification et les glissements en travaillant sur des périodes courtes et des cycles très prévisibles. De plus, il est ainsi plus facile de déterminer la capacité réelle de développement de l'équipe pour les *sprints* futurs.

Il ne faut pas oublier que la *story* est une description de fonctionnalité centrée sur la valeur pour l'utilisateur. En complément du travail de cotation sur les *user stories*, les développeurs peuvent donc procéder à un découpage des *stories* en tâches techniques plus petites (de 2 à 8 heures). Par exemple, une tâche pour la base de données, une tâche pour l'interface, ou bien encore une tâche d'écriture de code etc. Ce découpage aide à mieux calibrer la cotation des *stories* sur les premiers sprints. Nous n'avons pas procédé à ce découpage.

3.7.4. Le *sprint backlog*

En Scrum, le *sprint backlog* est la partie du *backlog* dédiée au *sprint*. C'est un sous-ensemble du *backlog* qui lui, est global au produit. Le *backlog* est un référentiel. On y stocke les *user stories* qui sont la véritable expression du besoin fonctionnel et les tâches associées. Dans le cadre du projet CCR, nous avons utilisé un *backlog* numérique (logiciel JIRA). Nous sommes restés sur une utilisation assez standard et très proche de ce que la méthode Scrum préconise. Nous avons utilisé le *workflow* natif de l'outil dans lequel nous avons ajouté un seul état pour stocker des *stories* en cours d'élucidation. La figure 30 montre ce *workflow.* Les *stories* sont présentes dans la colonne « Terminé » du tableau Scrum.

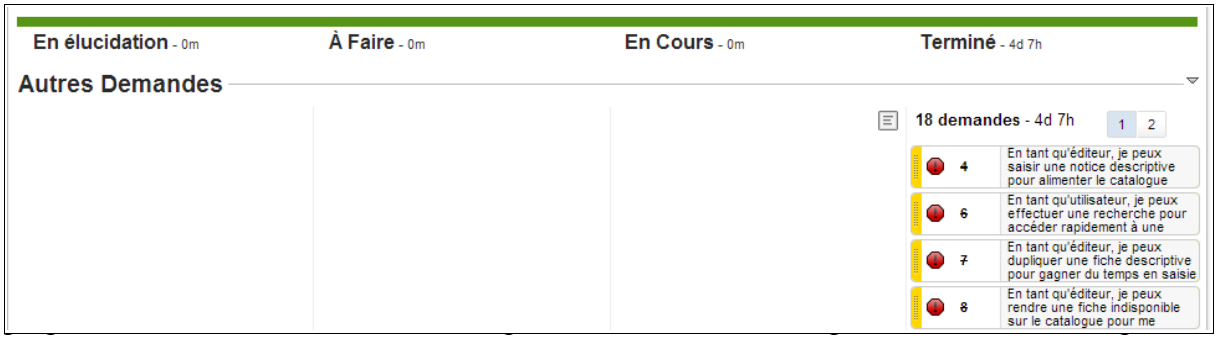


Figure 30 : Tableau numérique kanban et son workflow utilisé sur le projet CCR

Le *sprint backlog* était pour l'équipe l'outil de référence après la cérémonie de *planning poker.* Au travers de l'outil chaque membre de l'équipe pouvait suivre l'avancement du *sprint*. Les *stories* étaient écrites directement dans l'outil. Dès la création, elles étaient placées dans l'état « En élucidation ». C'était l'état par défaut. Les *stories* nouvellement écrites ou en cours d'écriture n'étaient pas planifiées. Le *backlog* numérique que nous avons utilisé, proposait la fonctionnalité de version, que nous avons fait correspondre à la notion de *sprint*. Chaque *sprint*, autrement dit chaque version avait une date de début et une date de fin (figure 31).

Figure 31 : Informations stockées sur les versions au sein du *backlog*

Ensuite, pour préparer le *planning meeting* du prochain *sprint*, je passais en revue les *stories* pour choisir celles que je souhaitais voir figurer dans la prochaine version. Je choisissais donc dans les *stories* non planifiées celles qui devaient être estimées en cérémonie de *planning meeting*. A ce stade, les *stories* étaient toujours dans l'état « En élucidation » même si elles étaient placées dans un *sprint*. Puis, le *planning meeting* avait lieu. Nous commencions par déterminer la vélocité de l'équipe pour le *sprint* qui allait démarrer, en tenant compte des dates de début et de fin. La vélocité était estimée et exprimée en nombre d'heures « idéales ». Ensuite, lorsque les *stories* étaient estimées, nous changions leur état en les faisant glisser dans la colonne « A faire » du tableau Scrum. Chaque ajout de *story* dans cette colonne avait pour effet de faire grimper le temps total estimé. Lorsque ce temps était égal ou juste inférieur au temps que nous avions défini pour le *sprint*, ce dernier était planifié. Les *stories* restantes étaient ensuite enlevées du sprint *backlog* et retournaient avec les *stories* non planifiées dans l'état « En élucidation ».

3.7.5. L'intégration continue.

En complément des pratiques Scrum, nous avons également expérimenté des pratiques de génie logiciel préconisées par la méthode XP. L'une de ces pratiques est l'intégration continue. Elle permet un haut niveau d'automatisation du processus de développement. Elle permet de réaliser l'assemblage du produit tout au long du développement au lieu de le reporter dans des phases finales du projet avant la livraison. Elle permet également de contrôler la qualité du développement en automatisant l'exécution des tests unitaires, des tests d'intégration et des tests utilisateurs. Ces tests peuvent être lancés à volonté pour contrôler la non régression fonctionnelle du produit. L'intégration continue participe donc très fortement à l'agilité au travers des aspects qualitatifs. Elle soutient le processus Scrum en permettant la livraison fréquente d'un produit opérable. L'intégration continue nécessite de travailler avec un gestionnaire de code source. Cette pratique était déjà exécutée dans l'équipe sur tous les projets de développements. Ici la consigne, que j'ai passée dans l'équipe, était d'effectuer des mise à jour du dépôt de code le plus fréquemment possible pour limiter les problèmes d'intégration et les conflits.

Nous avons pu fournir une version stable du produit en fin de *sprint* comme le

préconise la méthode Scrum. Pour s'assurer que la livraison est opérable, Scrum préconise également aux développeurs de s'entendre sur le DoD (*Definition Of Done*) [ROT10]. Il s'agit ici de définir ce que l'on entend par « Fait ». Nous avons considéré qu'une *story* était implémentée et qu'une version était livrable à partir du moment ou les actions ciblées dans le Dod étaient réalisées.

Ci-dessous, notre définition du « Done » appliquée aux *stories* et au *sprint*.

- Il ne devait pas y avoir de fonctionnalités à moitié développées dans la version, même si l'exécution ne provoquait pas d'erreurs et que les fonctionnalités étaient bouchonnées. Une fonctionnalité à moitié implémentée ou un processus à moitié opérationnel dans l'application aurait perturbé les utilisateurs même si aucune erreur n'était levée.

- Le dépôt de code devait être à jour. La construction du produit qui faisait foi était celle produite par l'outil d'intégration continue à partir du dépôt de code source.

- Les fonctionnalités devaient être testées sur le serveur de validation puisque c'est ce serveur que j'utilisais en tant que *product owner* pour tester les *stories* implémentées. Ce serveur était également utilisé par les éditeurs et le ministère pour effectuer des tests.

- Il fallait contrôler et modifier la documentation utilisateur que nous avions produite pour chacun des trois modules. Des changements ont eu lieu en cours de développement et nous devions nous assurer de la cohérence de ces documents.

En cours de *sprint*, je procédais aux tests des fonctionnalités implémentées. Les démonstrations au ministère se sont pour la plupart effectuées par téléphone.

3.8. La plate-forme de développement et les outils utilisés

La plupart des pratiques agiles ne nécessitent aucun outil particulier. Au maximum, un mur, quelques feutres et crayons et du papier. C'est le cas de l'ensemble des pratiques que l'on retrouve dans le *framework S*crum. Dans la réalité et dans la majorité des situations où Scrum est implémenté, il se retrouve combiné aux pratiques XP qui nécessitent des outils spécifiques. C'est notamment le cas de la pratique d'intégration continue qui fonctionne de pair avec le code géré à l'aide d'un gestionnaire de code source.

Avant que le projet CCR soit lancé nous avions commencé le déploiement de ces outils. Ces logiciels devaient venir supporter notre processus de développement. Il s'agissait d'un serveur d'intégration continue que nous avions branché sur nos dépôts de code source, d'un outil de suivi de demande de développement de type *bug tracker* et d'un outil facilitant le travail collaboratif et s'interfaçant avec les autres logiciels. Les outils étaient installés depuis l'automne 2010 mais le projet CCR était le premier à nous permettre de les éprouver en situation de développement agile.

3.8.1. L'intégration continue avec Hudson.

La pratique d'intégration continue a été soutenue par le serveur d'intégration continue Hudson. Ce serveur fonctionne avec des technologies variées pourvu que les extensions appropriées soient installées. Le projet CCR a été développé en Java. C'est une technologie avec laquelle Hudson s'interface nativement.

Le projet a été découpé en trois sous-projets. Ces sous-projets pouvaient être construits de manière indépendante par Hudson. Ils sont également stockés dans le dépôt de code source indépendamment les uns des autres.

Nous avions donc :

- Une base de code pour le noyau. C'est le code source commun à l'ensemble du produit.
- Une base de code pour les web services consommés par les applications tierces.
- Une base de code pour la gestion des subventions.

L'intégration continue permet d'automatiser l'intégration de ces codes tout au long du développement et de repérer très tôt et donc de corriger très tôt d'éventuels dysfonctionnements. Cette manière de développer du logiciel est une garantie de qualité lorsqu'elle est appuyée par le développement de tests unitaires dont l'exécution sera elle aussi automatisée. Mais le processus d'intégration continue permet également d'incorporer d'autres traitements. En effet, elle permet d'automatiser les audits de code et le déploiement. C'est la solution que nous avons déployée sur le projet CCR pour surveiller la qualité du développement tout au long du projet. Nous avons configuré l'outil logiciel Sonar. A chaque construction, Sonar produisait un rapport (figure 32) sur la qualité du code produit.

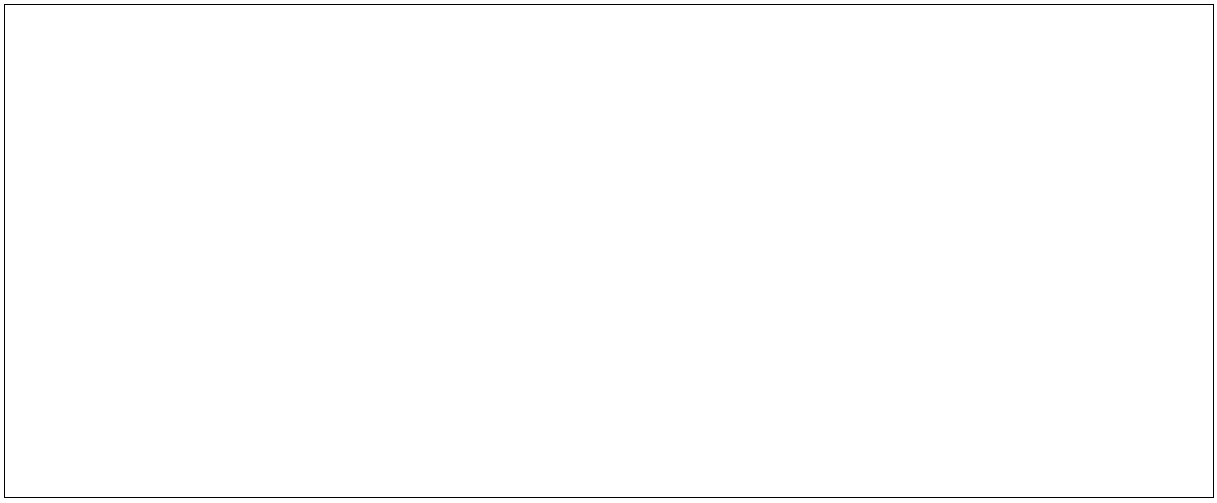


Figure 32 : Rapport Sonar sur la qualité logicielle.

Parmi les indicateurs fournis par Sonar, nous nous sommes intéressés principalement aux chiffres suivants :

- Le facteur de complexité du code. Il nous a renseigné sur la maintenabilité et

l'évolutivité du code.

- Le taux de couverture du code par les tests unitaire. Il nous a renseigné sur le niveau de test du code.

- Le taux de respect des règles de codage. Il nous a renseigné sur la qualité d'écriture du code.

3.8.2. Le suivi des demandes avec Jira

Le suivi des demandes a été réalisé avec Jira, produit développé par la société australienne Atlassian. Il est fréquent de trouver sur internet des produits logiciels dont le développement est suivi grâce à Jira. Jira est plus qu'un *bug tracker*. Il existe une multitude de *plugins* pour étendre les fonctionnalités de l'outil au delà des fonctions de gestion de demandes. De plus, Jira s'interface très bien avec les outils de génie logiciel. Par exemple Hudson pour l'intégration continue, Subversion pour le dépôt de code source, Eclipse/Netbeans pour les environnements de développement, et enfin Confluence pour la documentation. Il permet d'assurer la traçabilité d'une demande de bout en bout, de l'émission jusqu'à l'implémentation dans le code et au déploiement.

Jira était installé depuis l'automne 2010, mais nous avions utilisé uniquement les fonctionnalités natives de l'outil, c'est à dire la gestion de demande de développement au travers du *workflow* proposé par Jira. Dans Jira, un utilisateur crée une demande par rapport à un projet. Le projet doit être préalablement créé dans le système. Ensuite, l'outil permet une interaction entre le développeur et le rapporteur sur une demande de développement. Dans le *workflow*, cette demande passe par plusieurs états (« ouverte », « en cours », «fermée ») jusqu'à la résolution.

Dans le cadre du projet CCR, j'ai proposé que nous utilisions le *plugin* Greenhopper développé par Atlassian. Ce plugin implémente Scrum dans Jira et permet de gérer un *scrumboard* numérique (le scrumboard est un tableau kanban représentant la chaîne de valeur). Le *workflow* est modifié et adapté en conséquence par l'installation du *plugin*. Dans notre cas nous avons ajouté un état pour faciliter la gestion des *stories* dans le *sprint backlog*.

Sur le projet CCR, Jira a été utilisé uniquement au sein de la DSI. Comme je jouais le rôle de *product owner*, nous avions tous les rôles réunis au sein du service. Nous avions donc toute latitude pour configurer l'outil et l'adapter à notre processus de développement.

Les fonctionnalités que nous avons utilisées sur le projet CCR sont les suivantes :

- La gestion des demandes est une fonctionnalité native de Jira. Elle permet grâce au navigateur de demandes de suivre les développements. Toutefois, il faut manipuler les critères de sélection et les filtres pour retrouver les demandes sur le projet.

- Le suivi temporel nous a permis de construire les *sprints*. Cette fonctionnalité a été utilisée en cérémonie de *planning meeting* lors de l'estimation et de la planification des *stories*.

- Le *scrum board* est une fonctionnalité spécifique au *plugin* Greenhopper. Elle nous a permis d'afficher et de gérer l'avancement du développement sur le *sprint* en cours, en déplaçant les *stories* dans le *workflow* Scrum.

- Enfin, Le *burndown chart* est aussi une fonctionnalité spécifique à Greenhopper. Elle a permis la construction en automatique du graphique *burndown*. Sans cette fonction, il aurait fallu travailler sur les données manuellement et produire le graphique pour l'afficher à l'ensemble de l'équipe.

J'ai utilisé la fonctionnalité de *burndown chart* pour suivre l'avancement du développement par rapport à nos estimations réalisées en cérémonie de *planning meeting*. J'envisageais alors d'utiliser le pour communiquer cet avancement au ministère. Ce graphique présente quatre courbes (figure 33).

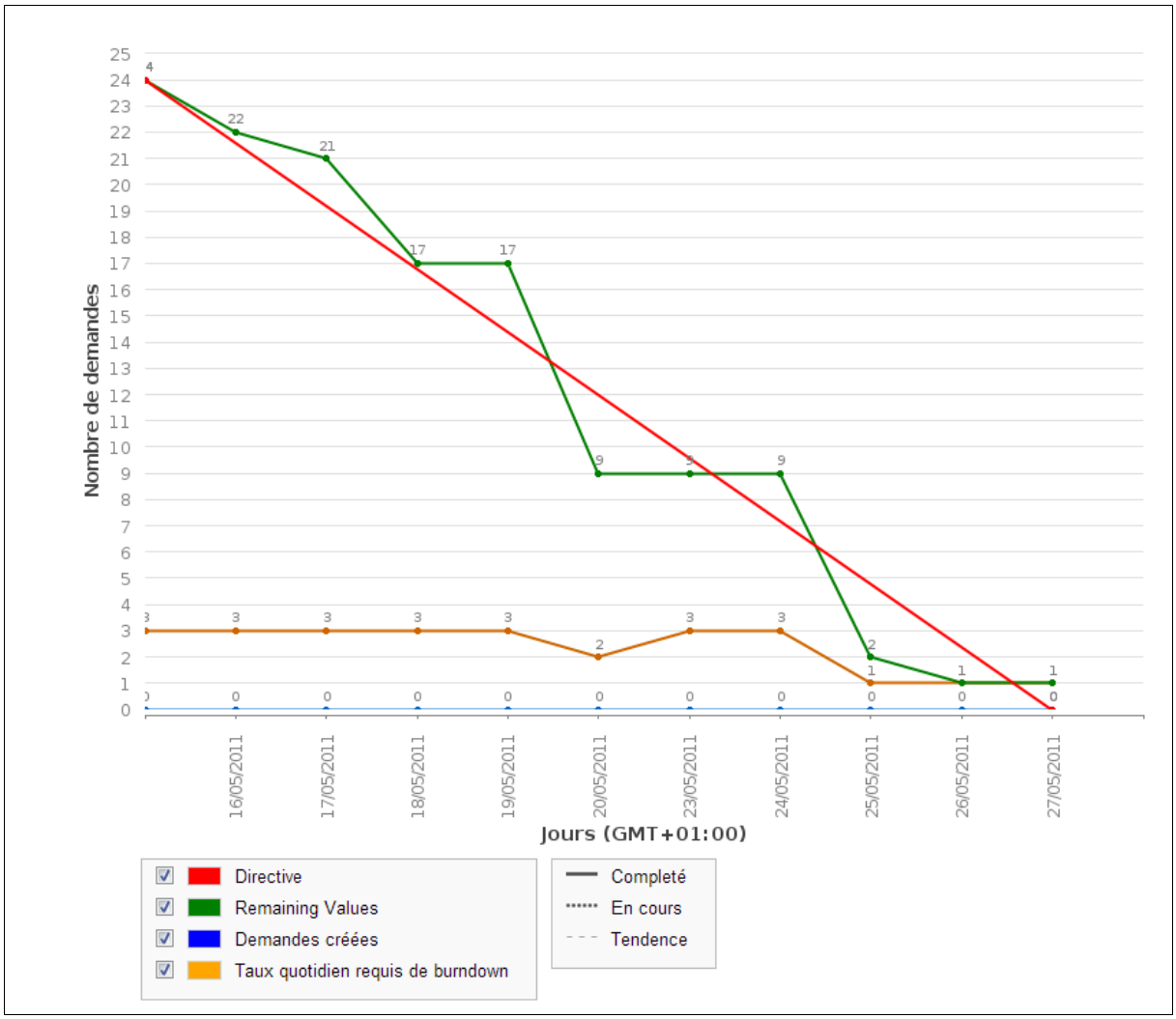


Figure 33 : *Burndown chart* produit sur le projet CCR.

La courbe rouge nommée « directive » donne la tendance à respecter sur le *sprint*

défini. Elle part de la date de début et s'effondre de manière linéaire jusqu'à la date de fin du *sprint*. Autrement dit, c'est l'effort restant lissé jusqu'à la date de fin. La courbe verte nommée « *value remaining* » représente la valeur restante à développer. Cette courbe s'effondre de manière non linéaire au cours du *sprint* au fur et à mesure que les demandes (*stories*, tâches etc..) sont traitées. Si la courbe verte est en dessous de la courbe rouge, l'équipe est en avance, si elle est au dessus, elle est en retard. La courbe bleue est plate car dans le *sprint* en cours aucune demande supplémentaire ne doit normalement être exprimée. Dans le cas contraire, la courbe verte peut également représenter un sursaut. Enfin la courbe orange représente le taux de *burndown* requis. Autrement dit c'est l'effort à faire pour terminer le *sprint*. Dans le cas du troisième *sprint*, la courbe orange est restée stable. Elle suit l'évolution de la valeur restante. Lorsque l'équipe est en avance le taux de *burndown* requis baisse. Sur un *sprint* normalement exécuté, si les *stories* et tâches ont été correctement estimées et découpées et si les développeurs avancent normalement sans rencontrer de problèmes (autrement dit la valeur restante reste jour après jour proche de la directive), alors le taux de *burndown* requis reste stable.

3.8.3. Le travail collaboratif de documentation avec Confluence

Une des valeurs essentielles de l'agilité est la communication entre les individus. Mais pour ne pas perdre les informations qui sont échangées à un instant donné, il était tout de même nécessaire de les stocker pour pouvoir les retravailler. Dans le cas contraire, l'équipe pouvait se trouver dans l'obligation de reproduire une discussion qui avait déjà eu lieu et dans laquelle des décisions avaient été prises. Le manifeste agile dit que la documentation n'est pas prioritaire mais il ne dit pas qu'elle est inutile et qu'il faut s'en passer. Nous avons utilisé un outil logiciel qui devait nous permettre de travailler efficacement sur cette documentation, tout du moins de manière plus efficace que sur un document produit à l'aide d'un traitement de texte traditionnel. Là encore, nous avions déjà déployé un outil. Il s'agissait du wiki Confluence. Comme Jira, Confluence est développé et diffusé par la société australienne Atlassian. Confluence est un wiki professionnel, facile d'accès, ergonomique et proposant des fonctionnalités intéressantes à mi-chemin entre la gestion électronique de documents et le wiki. Il s'interface également très bien avec Jira puisqu'il permet d'incorporer des macros permettant l'affichage de demandes.

J'ai utilisé Confluence pour documenter plus précisément certaines *stories*. Pour cela j'ai produit des cas d'utilisation textuels [ROT10] et des esquisses d'interface homme machine. Ces esquisses que nous appelons *mockup* ont été produite à l'aide de l'outil de *sketching* nommé Balsamiq. Cet outil existe en version plugin pour Confluence. Il m'a permis de produire directement dans Confluence les *mockups* pour les *stories* qui le nécessitaient (voir figure 34).

Confluence a également permis à l'équipe d'accéder à l'ensemble des informations dont nous disposions pour le projet. Confluence embarque des fonctionnalités similaires à celles d'une GED. Il permet de stocker des documents produits avec des outils extérieurs (word, excel) et de gérer le *versionning* de ces documents. Il est également possible de modifier directement dans le navigateur les fichiers stockés.

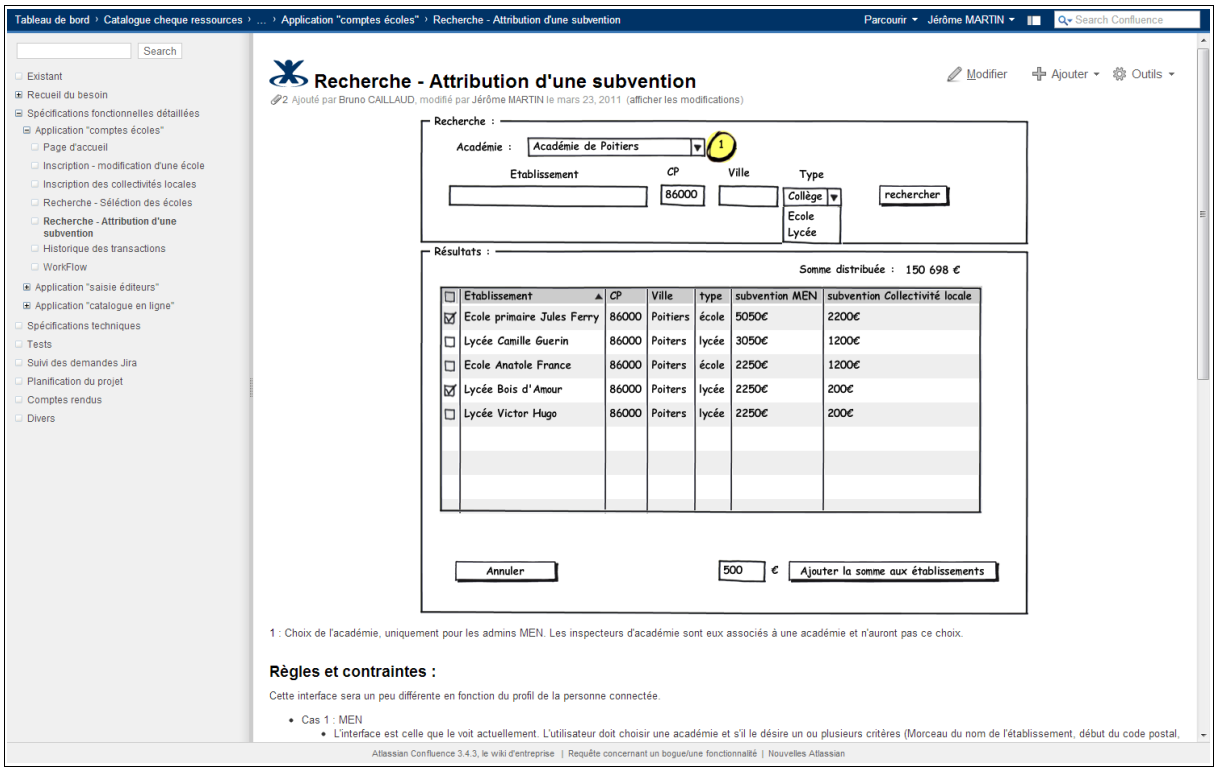


Figure 34 : Interface du wiki Confluence

Nous avons utilisé Confluence pour mieux structurer et hiérarchiser l'information qu'il nous semblait utile de stocker pour une consultation ultérieure. Les éléments stockés dans Confluence étaient les suivants :

- Les comptes rendus des comités de pilotage et des comités de suivi.
- Les éléments de spécification, sujets à discussion (*mockups*, cas d'utilisation, scénarios de test)
- Les éléments de planification.
- Des informations diverses permettant de prendre en main la plate-forme (comptes utilisateurs, mot de passe etc..)
- Les guides utilisateur.
- Les questions en suspens d'ordre fonctionnel.

Enfin, nous ne savions pas quelle serait la quantité d'informations échangées au cours du développement. Travailler avec des fichiers de traitement de texte est toujours délicat surtout lorsque les développeurs veulent repérer les ajouts récents dans le document. C'est un des points sur lequel le wiki se montre plus performant qu'un fichier word et permet d'être réellement agile sur la documentation du projet. Toutes les modifications sont stockées par l'outil et consultables par les utilisateurs.

3.9. Déroulement du projet

Les figures 35 et 36 présentent le déroulement réel du projet. Pour simplifier le

diagramme, j'ai choisi de ne faire apparaître que les *sprints* de développement, les comités de pilotage et les comité de suivi. Le développement de la plate-forme s'est articulé en trois temps. Alors que certaines parties de l'application devaient être disponibles en exploitation, d'autres parties de la plate-forme étaient toujours en développement. C'était le cas du module de gestion des subventions. Il a été finalisé après que le *back-office* du catalogue fut opérationnel. La priorité était de fournir pour le mois de juin, un *back-office* qui devait permettre aux éditeurs de renseigner leurs notices de ressources pédagogiques. La gestion des subventions pouvait attendre.

Le premier temps de développement concerne les sprints 1 à 4. L'objectif était alors de produire le *back- office* de gestion des notices. Les éditeurs devaient très tôt saisir leurs notices pour alimenter le catalogue qui devait être consultable à la rentrée 2011. Le ministère devait examiner et valider ces notices pendant l'été.

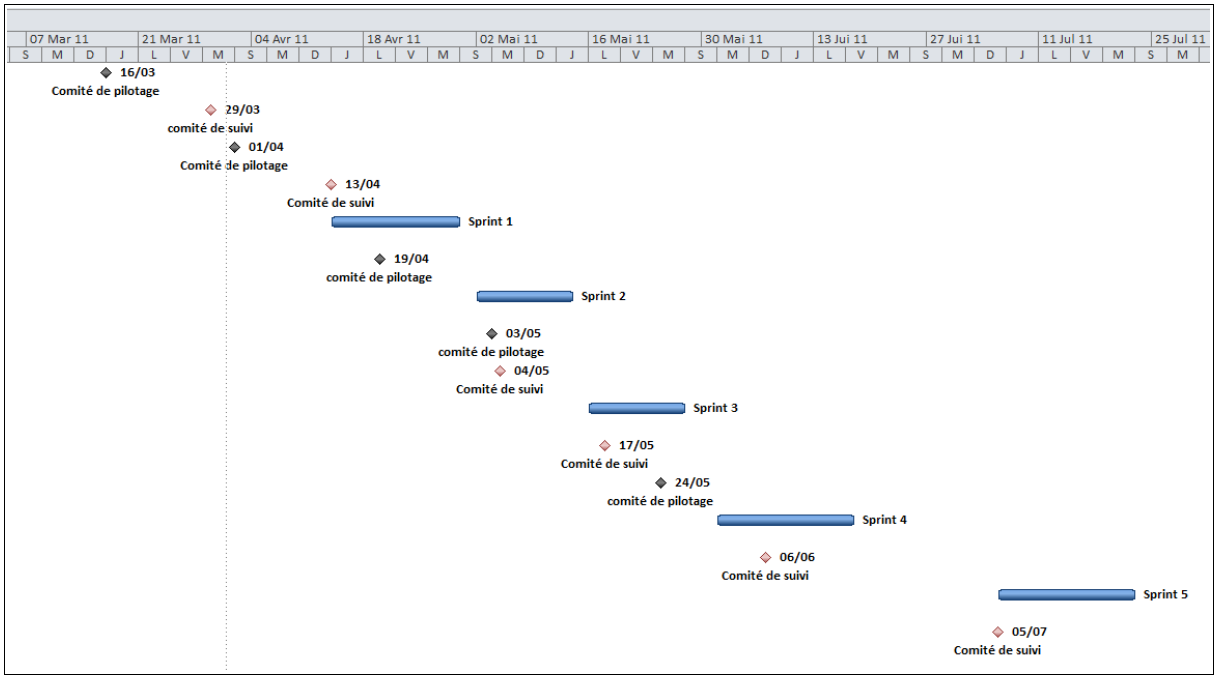


Figure 35 : Première période de travail sur le projet CCR

Le deuxième temps de développement a été consacré à la mise au point du catalogue pour l'ouverture à la consultation en octobre (figure 36).

Et enfin, dans un troisième temps allant d'octobre à fin décembre, nous sommes entrés dans une période de maintenance tout en finalisant le module de gestion des subventions et de gestion des comptes écoles. Sur cette période le processus de développement a subi quelques adaptations. Nous avons dû pour des questions organisationnelles (congés et indisponibilités de certains acteurs du projet pour tester la plate-forme) raccourcir certains *sprints* et en allonger d'autres.

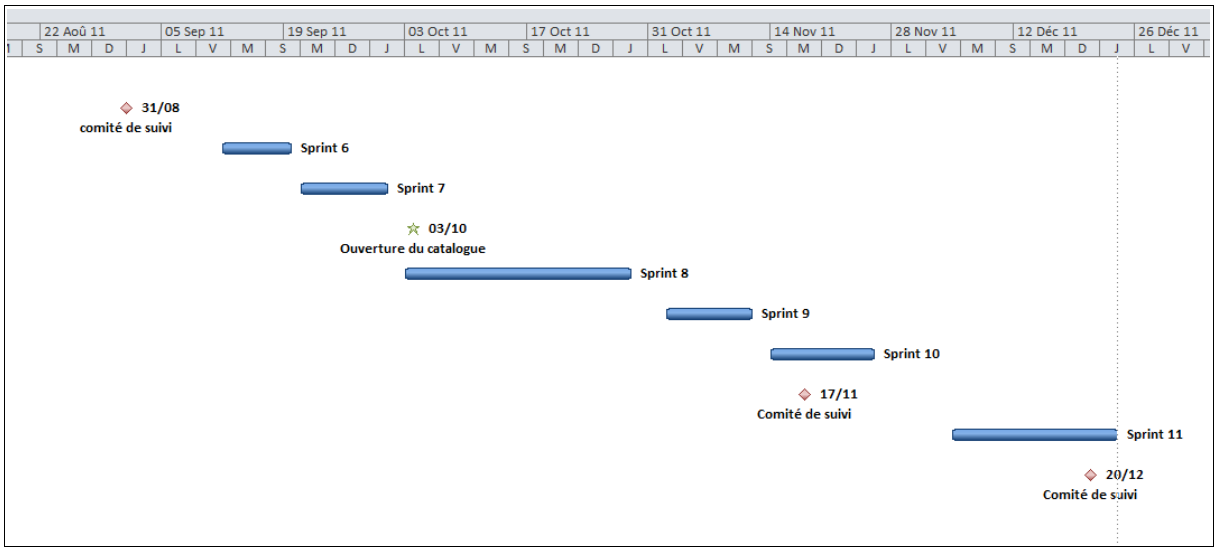


Figure 36 : Deuxième période de travail sur le projet CCR

Les comités de suivi fréquents m'ont permis de communiquer sur l'avancement du développement auprès des collègues du ministère et des éditeurs. Nous avons été très souvent dans l'attente de réponses à des questions fonctionnelles. Chaque comité de suivi était pour moi l'occasion d'alimenter le *backlog* de développement avec de nouvelles *stories*. L'équipe de développeurs a quasiment été en avance sur toute la durée du projet. Seule la période de septembre à octobre a été un peu plus difficile. Il s'agissait de la mise au point de la navigation dans le catalogue et celle-ci avait fait l'objet de critiques sur la précédente opération. Les éditeurs exigeaient que la plate-forme n'avantage personne au travers du moteur de recherche.

Mon rôle sur le projet s'est achevé à la fin du mois de décembre 2011. Nous sommes passés en maintenance applicative à partir de Janvier 2012, plusieurs évolutions ont été commandées par la suite.

3.9.1. Présentation des itérations

Le tableau suivant (figure 37) présente les douze itérations exécutées jusqu'au transfert de la plate-forme à la cellule fonctionnelle. La vélocité a été estimée en séance jusqu'au cinquième *sprint* inclus.

Sprint	Date de début	Date de fin	Nombre de stories / tâches / améliorations / bugs	Vélocité estimée (en j/h)	Descriptions / Modules fonctionnels
1	15/04/11	29/04/11	14 / 4 / 0 / 0	11	Module saisie éditeurs Module compte école
2	02/05/11	13/05/11	11 / 3 / 0 / 0	10	Module saisie éditeurs Module compte école
3	16/05/11	27/05/11	19 / 5 / 0 / 0	14	Module saisie éditeurs Module compte école Module Catalogue
4	01/06/11	17/06/11	11 / 6 / 0 / 3	9	Module saisie éditeurs Module Catalogue

5	06/07/11	22/07/11	3 / 2 / 0 / 3	2,5	Module saisie éditeurs Module Catalogue
6	12/09/11	19/09/11	4 / 2 / 5 / 3	N/C	Module saisie éditeurs Module compte école Module Catalogue
7	21/09/11	30/09/11	3 / 4 / 4 / 1	N/C	Module saisie éditeurs Module compte école Module Catalogue
8	03/10/11	28/10/11	5 / 9 / 44 / 25	N/C	Module saisie éditeurs Module compte école Module Catalogue
9	02/11/11	11/11/11	3 / 2 / 1 / 5	N/C	Module saisie éditeurs Module compte école
10	14/11/11	25/11/11	7 / 5 / 21 / 12	N/C	Module saisie éditeurs Module compte école Module Catalogue
11	05/12/11	23/12/11	1 / 13 / 0 / 3	N/C	Module saisie éditeurs Module compte école Module Catalogue
12	02/01/11	31/01/11	0 / 3 / 8 / 2	N/C	Module saisie éditeurs Module compte école Module Catalogue

Figure 37 : Tableau de présentation des *sprints*

3.9.2. *Sprint* 1

Sur le plan fonctionnel, les *stories* de cette itération se répartissaient sur le module « saisie éditeurs » et sur le module « comptes écoles ». Deux développeurs ont travaillé sur le premier et un seul sur le second module. Le module « Saisie éditeurs » devait être prêt en Juin pour permettre aux éditeurs de travailler sur les notices descriptives. Le ministère au travers d'un *workflow* de validation devait traiter ces notices pour une ouverture du catalogue à la consultation en octobre. Les développeurs devaient donc implémenter les fonctions de gestion de fiches (recherche, création, modification) rapidement. Le module « comptes écoles » était celui sur lequel il y avait le plus d'incertitudes. Nous ne savions pas comment les comptes des établissements seraient abondés. Il était prévu une alimentation par une subvention d'état complétée par des subventions locales. Dans ce *sprint*, les principales fonctionnalités implémentées pour ce module concernaient la ventilation de la subvention d'état sur les comptes des écoles sélectionnées pour l'opération.

Les tâches faisaient référence à l'initialisation du projet. Il s'agissait de préparer les serveurs, les espaces Subversion, Hudson, Jira et Confluence. L'équipe a eu du mal à démarrer le projet. Les étapes d'initialisation ont pris plus de temps que prévu, notamment la construction du modèle à partir des diagrammes de classes. A la fin du *sprint*, 14 des 18 demandes ont dû être reportées sur le *sprint* 2. Aucune des fonctions n'avait été implémentée sur le module « comptes écoles ». Le *burndown chart* (annexe 7-a), produit automatiquement par Jira, reflète bien la situation avec une courbe quasiment plate pour la valeur restante.

3.9.3. *Sprint* 2

Sur le plan fonctionnel, les *stories* étaient réparties entre le module « saisie éditeurs » et le module « comptes écoles ». Les développeurs ont repris les *stories* qui n'avaient finalement pas été implémentées dans le premier *sprint*.

Le *sprint* s'est déroulé normalement sans incidents majeurs bien que du retard se soit accumulé dans le traitement des demandes Jira. Les transitions entre états n'étaient pas effectuées correctement. Le sursaut que l'on observe dans le *burndown* du *sprint* (annexe 7-b) est dû à la création de deux demandes supplémentaires en cours de sprint, ce qui n'est pas conforme au *framework* Scrum. Il s'agissait d'une tâche de développement sur le noyau de l'application et d'une *story* ajoutée au dernier moment sur le module « comptes écoles ». Cette *story* devait donner plus de droits aux utilisateurs des collectivités locales. On remarque aussi que la courbe « *remaining values* » ne rejoint pas la « directive ». Ceci est dû au fait que les transitions entres les demandes n'étaient pas respectées. Certaines demandes sont restées ouvertes en fin d'itération alors qu'elles étaient traitées.

3.9.4. *Sprint* 3

Sur le plan fonctionnel, les *stories* étaient réparties entre le module « saisie éditeurs », le module « comptes écoles » et le module « catalogue ». C'est dans ce sprint que le développement du catalogue et de la gestion des commandes a débuté. Ces travaux n'étaient pas prioritaires mais en les commençant assez tôt, nous avons pu prendre en compte les besoins du CNDP. L'établissement était désigné pour être l'opérateur de la plate-forme. A ce titre, il avait des besoins particuliers au sujet de la gestion des commandes effectuées sur la plate-forme et des paiements des éditeurs. Cette partie n'avait pas été correctement implémentée dans le précédent dispositif et les traitements manuels avaient engendré beaucoup d'attentes sur ces fonctionnalités. Nous avons donc travaillé en collaboration avec les utilisateurs du CNDP en charge de la gestion des commandes. Ces derniers ont présenté les difficultés rencontrées sur la précédente plate-forme (opération ENR) et ont émis des souhaits sur la nouvelle que j'ai traduite sous forme de *stories*.

Sur ce *sprint*, les développement étaient bien engagés. Nous arrivions à la fin mai et le formulaire de saisie des fiches était attendu pour juin. Ce *sprint* a été celui pour lequel le processus agile que nous avons monté, a été exécuté avec le plus de discipline. Cette discipline se traduit par un *burndown* bien formé. Les demandes sont bien passées d'un état à l'autre dans le Scrum *board* numérique, au fur et à mesure de leur implémentation. Nous avons veillé à introduire dans ce *sprint,* les *stories* les mieux travaillées et celles pour lesquelles il y avait le moins d'incertitudes. On peut observer sur le *burndown chart* (annexe 7-c) que le taux requis de *burndown* baisse lorsque la courbe « r*emaining values* » passe en dessous de la « directive ».

3.9.5. *Sprint* 4

Dans ce *sprint*, les *stories* se répartissaient sur les modules « saisie éditeur » et « catalogue ». Nous n'avions plus qu'un seul développeur disponible pour travailler sur le projet. Le quatrième *sprint* était le dernier avant la livraison d'une première version en

production. Le ministère voulait effectuer pendant l'été, l'examen et la validation des notices de ressources saisies par les éditeurs. A ce stade, le ministère disposait d'une plate-forme de test pour suivre l'avancement des développements. En parallèle, j'ai testé les travaux réalisés dans le *sprint* 3 concernant le module « catalogue ». Ce module allait devenir notre prochaine priorité puisque l'ouverture à la consultation devait être opérationnelle pour la rentrée 2011. J'ai profité de cette phase de tests pour créer dans le *backlog* de nouvelles *stories* sur ce module.

Au début de ce quatrième *sprint*, huit demandes on été fermées par le développeur car elles étaient en fait implémentées dès le début du *sprint*. Ce dernier a commencé avec deux jours de retard, deux jours pendant lesquels les développements n'ont pas cessé. Le *burndown chart* (annexe 7-d) de ce *sprint* est relativement bien formé et traduit un avancement normal des développements à ce stade du projet. Nous avons livré à l'issue du *sprint* une première version du module « saisie éditeurs », pour permettre la saisie des notices descriptives.

3.9.6. *Sprint* 5

Ce *sprint* a été exécuté pendant la période estivale. L'activité de développement était à son minimum. Le *sprint* a surtout permis de faire quelques corrections sur la plate-forme. Le *sprint* n'a pas démarré dans la continuité du quatrième. La période allant de fin juin à septembre est assez perturbée par les formations et les congés de l'équipe.

Le *burndown* (annexe 7-e) est ici assez mal formé à cause des 3 *bugs* incorporés au référentiel au cours du *sprint*. De plus les demandes n'ont pas suivi le *workflow* Scrum de manière à représenter fidèlement l'activité de développement. Notre premier temps de développement, qui avait pour but de fournir le *back-office* aux éditeurs, pour leur permettre de constituer le catalogue, était terminé.

3.9.7. *Sprint* 6

Les travaux ont repris à la mi-septembre et visaient tous les modules. Le deuxième temps de développement commençait. Il s'agissait alors de travailler à la mise au point du catalogue qui devait ouvrir au mois d'octobre. Les demandes sur ce *sprint* étaient de natures diverses. Nous reprenions les développements et j'ai choisi de planifier un *sprint* d'une semaine seulement pour permettre à l'équipe de revenir dans le projet plus rapidement. De plus, j'étais dans l'attente d'éléments de la part du ministère pour nous permettre d'avancer sur les développements.

Ici le *burndown* (annexe 7-f) est bien formé et représente une légère avance sur le développement. Cette avance n'était pas réelle. A partir de cette sixième itération, nous avons abandonné la pratique d'estimation en *planning meeting*. Aussi, la vélocité et le nombre de demandes à traiter dans le *sprint* ont été estimés de manière approximative. De plus, le principe qui veut que le périmètre du *sprint* ne soit pas modifié pendant son exécution, est lui aussi tombé. Cet abandon de pratiques est dû à une conjonction de différentes situations et de différents facteurs (voir le bilan du projet paragraphe 3.11.).

3.9.8. *Sprint* 7

Ce *sprint* (annexe 7-g) comportait les premières demandes d'améliorations résultant de la prise en main de l'application par les utilisateurs au cours de l'été. En parallèle, les travaux sur le catalogue devaient commencer. Les trois *stories* exprimées concernaient de nouvelles fonctions sur les trois modules.

Un seul développeur est intervenu au cours de cette itération en raison de formations suivies par les autres membres de l'équipe. Au cours de ce *sprint*, des travaux de mise à jour des composants de la plate-forme ont été effectués sur les environnements de développement et de test. Pendant plusieurs jours, les plate-formes de développement, de test et de production, n'étaient donc plus identiques et ces travaux ont rendu difficile l'évaluation des travaux de développement.

3.9.9. *Sprint* 8

Les améliorations demandées concernaient principalement des modifications de libellés, des ajouts de colonnes dans les grilles de données ou bien encore des tris de données. Les défauts rencontrés étaient principalement liés à des problèmes de moteur de recherche, dont certains avaient été introduits sur le *sprint* précédent. Certains de ces défauts étaient de véritables régressions fonctionnelles. Certains défauts corrigés sur les sprints précédents réapparaissaient et des fonctions précédemment opérationnelles ne l'étaient plus. Ces régressions concernaient principalement le module « catalogue » et le module « saisie éditeurs ». Elles ont engendré beaucoup de tests manuels de la plate-forme à cause de carences dans l'automatisation des tests fonctionnels.

3.9.10. *Sprint* 9

A ce stade, les travaux de développement devaient porter principalement sur le module « comptes écoles » pour implémenter les fonctionnalités de gestion des subventions. Mais des demandes d'améliorations portant sur le module de saisie des notices étaient toujours émises. Nous devions toujours faire face à des défauts et des régressions fonctionnelles.

3.9.11. *Sprint* 10

Sur cette dixième itération, les défauts relevés concernaient le module « saisie éditeur » et le module « catalogue ». Nous avons rencontré des problèmes sur les notices des éditeurs. Toutes les notices n'ont pas été saisies au travers de l'application et nous avons dû procéder à des imports de données fournies dans des fichiers XML. Ce dixième *sprint* a été marqué par un nombre important de travaux de corrections et d'améliorations (33 demandes au total). Les nouvelles fonctions implémentées concernaient le module « comptes écoles » et le module « saisie éditeurs ».

Beaucoup de ces demandes ont été exprimées pendant le comité de suivi qui s'est tenu le 17 novembre. Aussi à l'issue du comité, nous avons dû prendre en compte les demandes dans le *sprint* ce qui explique le deuxième sursaut dans le *burndown chart* (annexe 7-j).

3.9.12. *Sprint* 11

Pendant le mois de décembre 2011, ce onzième *sprint* (annexe 7-k) fut consacré à l'amélioration de la plate-forme sur les trois modules. Un quatrième axe était alors à l'étude. Il s'agissait d'implémenter des fonctions minimales permettant le suivi du service après-vente. Ces opérations de gestion étaient effectuées manuellement en dehors de la plate-forme. Le CNDP et le ministère n'intervenaient pas sur le support des ressources distribuées. L'idée était de permettre une mise en relation de l'acheteur de la ressource avec l'éditeur tout en permettant aux gestionnaires du dispositif de s'assurer que les acheteurs obtenaient bien les réponses à leurs questions. Les *user stories* ont été produites en collaboration avec le ministère mais elles ne furent pas implémentées par la suite.

3.9.13. *Sprint* 12

Nous étions dans un mode de fonctionnement plus proche de la maintenance évolutive et corrective depuis plusieurs semaines, aussi à partir du mois de janvier nous avons cessé le traitement des demandes au sein d'itération. Ce douzième *sprint* (annexe 7-l) marque la fin du développement de la plate-forme au travers du mode projet. En janvier 2012, la cellule fonctionnelle a pris en charge le support de la plate-forme et mon intervention sur le projet CCR était terminée.

3.10. Le projet en chiffres

Les chiffres présentés ci-dessous ont été obtenus à partir des relevés de temps adressés au contrôle de gestion et à partir des outils utilisés par l'équipe de développeurs. Pour rappel, le CNDP a conçu la plate-forme avant de l'exploiter en tant qu'opérateur du ministère, tout en étant également partie prenante dans le dispositif CCR en tant qu'éditeur public.

3.10.1. Les demandes JIRA

Les développeurs ont traité en 2011 (année du projet) 279 demandes JIRA sur 200 jours/hommes de développement d'avril à décembre. Les demandes sont réparties de la manière suivantes :

- 84 *User stories*
- 57 *Bugs*
- 96 Améliorations
- 42 Tâches

La figure 38 montre la répartition des demandes par type et par *sprint* ainsi que l'évolution de cette répartition jusqu'à la fin du projet. On observe qu'à partir du sprint 6 le nombre de nouvelles fonctionnalités à implémenter était moins important que les opérations d'évolution et de correction.

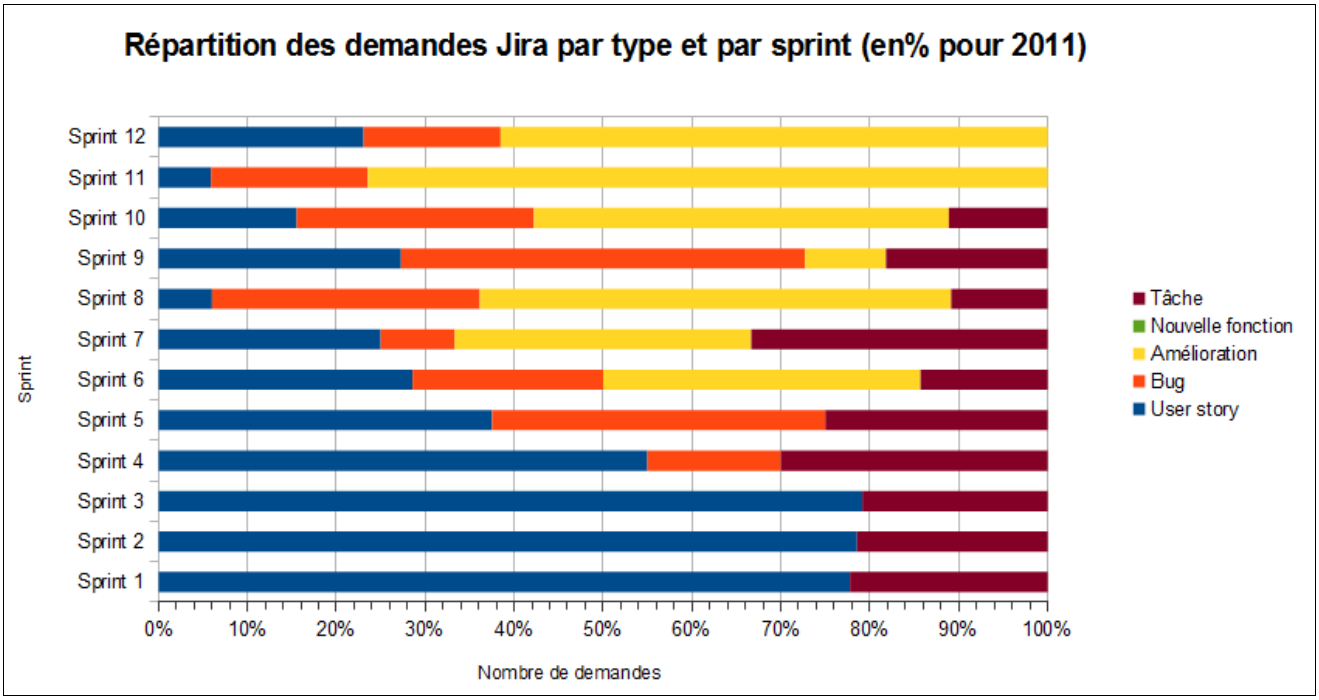


Figure 38 : Répartition des demandes selon leur typologie

En dehors des demandes de développement, les services qui sont intervenus pour assurer le rôle d'opérateur et d'éditeur du CNDP, ont émis 128 demandes d'interventions techniques et-ou fonctionnelles dans l'outil logiciel GLPI sur la même période.

3.10.2. Les coûts

Les éléments chiffrés présentés dans le tableau ci-dessous (figure 39) concernent le développement de la plate-forme. Ces charges ont été consommées sur les trimestres 2, 3 et 4 de l'année 2011. Les relevés de temps ne font pas la distinction entre le temps passé sur le projet en conception de la plate-forme et le temps passé sur l'exploitation du dispositif dans son ensemble. Les relevés ne font pas apparaître non plus, le temps passé en tant qu'éditeur. Par ailleurs, je n'ai pas eu accès au temps consommé par la direction de l'établissement et à ses charges externes sur le projet.

	Nombre j/h	Coûts en €
Développeurs	198	45 147,00 €
Intégrateurs web	34	6 068,00 €
Graphismes	3	573,00 €
product owner	73	16 152,00 €
total	**308**	**67 940,00 €**

Figure 39 : Charges en développement de la plate-forme CCR pour l'année 2011

Le nombre de jours consommés par les autres services du CNDP est de 104 jours/hommes pour un coût de 19 621 €. Au total, tous services confondus pour l'année 2011,

411 jours/hommes ont été consommés pour un coût de 87564 €.

Les dépenses externes sur l'année 2011 comprennent les frais de déplacement, d'hébergement, de restauration et toutes les autres dépenses réalisées en dehors des ressources humaines. Le montant de ces frais s'est élevé à environ 3000€. Ces frais concernent principalement les déplacements au ministère pour les comités de pilotage et de suivi.

Les frais de serveurs s'élèvent à environ 15 €/mois et par machine. Ce coût n'intègre pas l'électricité, la climatisation et les frais d'exploitation. L'architecture web du CNDP est virtualisée et les machines sont mutualisées.

3.10.3. Les indicateurs sonar.

Le tableau suivant (figure 40) montre l'évolution, au cours du projet, des indicateurs sonar.

sprint	Nombre de lignes de code	Complexité (method/class/ file)	Tx de couverture par les tests	Respect des règles de codage
1	1164	1,2/5,0/5,0	0,00%	83,50%
2	3382	1,7/8,6/8,6	0,00%	86,50%
3	5702	1,9/10,8/10,8	0,00%	86,80%
4	6449	2,0/12,5/12,5	0,00%	86,10%
5	7393	2,2/13,7/13,7	0,00%	86,40%
6	8292	2,3/15,7/15,7	0,00%	85,80%
7	8767	2,4/15,6/15,6	8,90%	85,20%
8	10775	2,7/19,8/19,8	8,90%	83,70%
9	11541	2,9/20,2/20,2	8,90%	83,20%
10	12169	2,9/20,5/20,5	8,90%	82,80%
11	13625	3,1/21,9/21,9	8,90%	82,00%
12	14441	3,0/21,8/21,8	8,90%	85,50%

Figure 40 : Evolution des indicateurs sonar au cours du projet.

3.11. Le bilan du projet

L'exécution du projet s'est déroulée d'avril à fin décembre 2011. Nous avons rencontré sur cette période plusieurs soucis au cœur de notre processus mais ce dernier nous a tout de même permis de respecter nos engagements. Les pratiques Scrum déployées nous ont permis d'apprendre beaucoup sur l'équipe et sur notre maturité face à l'agilité. Nous avons également

beaucoup appris au travers des dysfonctionnements rencontrés au cours du projet. Je peux lier certains de ces dysfonctionnements à l'absence de certaines pratiques ou à leur mauvaise exécution. En général les agilistes conseillent d'appliquer Scrum à la lettre (au début tout du moins) pour en retirer tous les avantages [BOD11]. Pour ma part, j'ai choisi de n'exécuter que certaines pratiques pour obtenir plus facilement l'adhésion de l'équipe. Les pratiques déployées ont été riches d'enseignements. Certaines de ces pratiques ont été exécutées avec plus ou moins de discipline par rapport aux préconisations des méthodes. Cependant, toutes les pratiques que j'ai proposées et que nous avons implémentées, ont été utiles au processus. Certaines ont engendré une vraie dynamique de groupe.

3.11.1. Généralités

Au cours du projet, le CNDP a eu une positon délicate à tenir. En effet, il intervenait comme opérateur d'une plate-forme mise à disposition des éditeurs alors qu'il était lui-même partie prenante en tant qu'éditeur public. Il avait donc un rôle d'opérateur et d'utilisateur en tant qu'éditeur. Cette position a demandé à l'équipe intervenant en réalisation une grande attention dans le recueil du besoin. Il s'agissait ici de rester impartial vis à vis des éditeurs privés et de ne pas avantager le CNDP. Lors des comités de suivi, il nous a fallu faire la preuve que les développements réalisés n'avantageaient pas le CNDP. C'était le cas par exemple pour le fonctionnement du moteur de recherche. Nous avons expliqué sur quels paramètres nous avions la main et sur quels autres nous n'intervenions pas (les algorithmes du moteur) pour rassurer les parties prenantes. Les besoins fonctionnels des éditeurs ont guidé le développement de la plate-forme. Le CNDP étant le concepteur et l'opérateur de la plate-forme, il était délicat pour le Directeur Général adjoint et moi-même de porter, en comité de suivi, les besoins fonctionnels de l'établissement sur la partie « éditeur ». Le CNDP en tant qu'éditeur s'est donc adapté à ce fonctionnement comme d'autres éditeurs non représentés en comité de suivi.

L'organisation suivante aurait pu être mise en place. Deux *product owners* auraient pu être désignés, un pour représenter les intérêts du CNDP en tant qu'éditeur, celui-ci aurait pu travailler en interne avec l'équipe projet sans se rendre au comité de suivi au ministère, et un autre pour représenter le CNDP en tant qu'opérateur de la plate-forme. Ce dernier aurait eu pour rôle de recueillir les besoins des éditeurs et de travailler en étroite collaboration avec le ministère sur la conception de la plate-forme et du dispositif.

3.11.2. Les *stories*, bilan sur le formalisme

Sur le projet, j'ai assuré le rôle de *product owner.* Il était donc de ma responsabilité d'alimenter le *product backlog* en écrivant les bonnes *user stories*. Globalement, je n'ai pas rencontré de difficultés. Le formalisme est assez simple à assimiler une fois que l'on se place au bon niveau de granularité fonctionnelle, ce qui demande un peu d'expérience tout de même. La collaboration avec les développeurs permet de se placer assez naturellement au bon niveau. De mon point de vue, il est nécessaire d'avoir parmi ces développeurs des profils suffisamment matures qui oseront poser des questions. Les agilistes disent que les *stories* sont l'assurance d'une discussion entre le *product owner* et les membres de l'équipe. Le *product owner* doit bien connaître et bien comprendre le domaine fonctionnel sur lequel il écrit les *stories*.

Lorsque qu'une *story* engendrait trop de questions, nous nous rendions compte rapidement que le périmètre était trop large et qu'elle devait être découpée. Le premier symptôme étant l'impossibilité de l'estimer.

Les *stories* doivent normalement être découpées par les développeurs en unité de travail de deux à quatre heures, voire huit heures maximum. Ce découpage permet de créer des tâches associées pour l'implémentation de la *story*. Ce découpage n'a pas été réalisé et je pense qu'il n'aurait pas donné, en terme de suivi sur le *burndown chart*, un niveau d'information vraiment utile. Cela n'a pas facilité le suivi de l'implémentation des fonctionnalités, mais je pense que cette difficulté de suivi réside plus dans le manque de discipline à faire passer les *stories* d'un état à l'autre sur le *scrumboard* (« à faire », « en cours », « fait ») (voir paragraphe 3.11.5.).

De plus, chaque *story* aurait normalement du être accompagnée d'au moins un critère d'acceptation utilisateur (exemple en annexe 5). Ce critère aurait permis aux développeurs de mieux comprendre la *story*, et d'implémenter des tests réellement pertinents en cours de développement. Nous aurions également pu éviter la création de demandes supplémentaires avec plus de discipline sur l'écriture de ces critères d'acceptation. En effet, certaines *stories* ont été acceptées dans un premier temps et ont fait l'objet, par la suite, d'une demande de correction. Du travail a été fait, défait et refait. Si les critères d'acceptation n'accompagnent pas la *story*, il est difficile de dire sur quelle base on considère que celle-ci est bien implémentée. Si plusieurs personnes participent aux tests utilisateurs, toutes les interprétations sont possibles puisque rien n'est spécifié en acceptation. Une *story* peut alors être acceptée par un développeur et refusée par un autre. Malgré ces manquements, toutes les fonctions à développer ont fait l'objet d'une écriture de *user story* et ce formalisme a vraiment permis aux développeurs de se focaliser sur les fonctionnalités attendues à la fin du *sprint*.

A moi, en tant que *product owner*, elles m'ont permis d'aborder tous les aspects fonctionnels de la plate-forme avec plus de méthode et de discipline en endossant le rôle d'utilisateur. Personnellement, j'ai trouvé cette manière de travailler le besoin extrêmement efficace dans la répartition des rôles et nous aurions gagné encore davantage en efficacité si nos collègues du ministère avaient complètement partagé cette pratique avec nous. La seule occasion de leur présenter le formalisme a été le travail effectué avec eux sur les fonctionnalités de gestion du service après vente (*sprint* 11).

3.11.3. Bilan du *planning meeting*

Le *planning meeting* est la pratique Scrum (*sprint planning meeting*) que nous avons exécutée avec le plus de discipline. Cette pratique a immédiatement remporté l'adhésion du groupe, peut être grâce au côté ludique du *planning poker*.

Le *planning meeting* a engendré sur le projet CCR un très bon niveau de collaboration entre les développeurs. Les cotations de *stories* avec la technique du *planning poker* ont permis un vrai partage d'expérience, chacun faisant par exemple un retour d'expérience sur une problématique similaire à la *story* en cours d'estimation. Les cotations ainsi ajustées se sont révélés justes dans la mesure où, sur les premiers *sprints*, les développeurs ont réussi à implémenter les *stories* de l'itération.

Nous avons, par contre, manqué de régularité. Nous avons pratiqué le *planning meeting* jusqu'en juin. Ensuite lorsque les itérations ont comporté plus de demandes d'améliorations et de corrections nous n'avons pas poursuivi sur cette pratique. Certaines demandes devaient être traitées immédiatement. Le principe Scrum qui veut que le périmètre soit défini pour une itération et gelé le temps de cette même itération était abandonné. L'abandon de la pratique résulte de plusieurs facteurs. A la fin de l'été 2011, la plate-forme était en production depuis quelques semaines pour permettre au ministère de valider les notices saisies par les éditeurs. En septembre et en octobre, le rythme s'est accéléré sous l'effet de la rentrée scolaire, du salon Educatice qui approchait et des engagements pris par notre donneur d'ordre. Nous n'avons pas eu d'autres choix que de prendre en compte, après une transcription rapide dans Jira, les demandes qui étaient formulées, soit par téléphone, soit par mail. Comme les pratiques n'étaient pas partagées avec nos collègues du ministère et que ces derniers ne connaissaient pas tout le processus, de l'écriture des *stories* jusqu'à leur implémentation par les développeurs, ils auraient eu du mal à comprendre que leur demandes ne soient pas traitées immédiatement.

Je n'avais donc plus le temps d'avance qui me permettait de travailler les *stories*, de les discuter avec les développeurs et de planifier avec eux les itérations. A partir du sixième *sprint*, nous pouvons même considérer que la notion d'itération n'avait plus beaucoup de sens puisque nos activités de développement étaient parfois plus proches de pratiques de déploiement journalier et continu.

Je pense qu'il est important de tenir sur les principes pour être conforme au *framework*, à condition que cela ait un sens et que ce soit adapté à la situation. Lorsque la situation change, le risque est grand de perdre d'autre pratiques importantes et de perturber tout le processus de développement si l'équipe ne s'adapte pas assez vite sur les pratiques à conserver ou à remplacer.

Enfin, nous aurions peut-être pu passer un peu moins de temps en séance avec un meilleur travail sur les *stories*, notamment sur les critères d'acceptation. Le *planning meeting* doit rester une cérémonie assez rythmée. Le *product owner* ne doit pas négliger la préparation de la séance et l'écriture des *stories*. Je pense que le temps passé en séance de planification diminue avec l'expérience acquise sur le projet. Nous avons passé en moyenne trois heures en séance pour une durée d'itération de dix jours.

3.11.4. Le *backlog* produit et le suivi du changement (*changelog*)

Pour gérer les *user stories* et constituer notre *backlog* nous avons utilisé l'outil logiciel Jira avec le *plugin* GreenHopper. Ce *plugin* permet d'implémenter le *framework* Scrum dans l'outil. Les développeurs ont bien assimilé le fonctionnement mais nous avons rencontré deux difficultés importantes dans le suivi du changement (*changelog*).

La première difficulté était liée à notre manière d'utiliser l'outil et au fait que cet outil n'était pas partagé. J'ai choisi de créer et de travailler la mise au point des *stories* directement dans Jira, ce qui signifie qu'il stockait, d'une part, des *stories* en cours d'écriture et d'élaboration et d'autre part des *stories* acceptées et estimées par les développeurs. Je pense qu'il aurait été préférable de travailler les *stories* en dehors de Jira et de les saisir uniquement après leur acceptation pour constituer le *backlog* de *sprint*. L'outil n'était pas partagé avec le

ministère et ce manque de visibilité sur la production des *stories* n'a pas favorisé le partage du formalisme.

De plus, ce manque de partage et de visibilité a généré beaucoup de communications par messagerie et par téléphone. Lorsque les demandes d'améliorations et de corrections sont apparues, le temps de travail important consacré à cette communication a pris beaucoup d'ampleur. Un compte dédié à nos collègues leur aurait permis de renseigner directement dans l'outil ces demandes. Pour ma part, j'aurais pu me concentrer davantage sur l'écriture de nouvelles *stories*.

La deuxième difficulté résidait dans le fait que seuls les développeurs possédaient un compte sur l'outil, il n'a donc pas été possible d'utiliser l'outil de suivi de demandes pour gérer les interventions de tous les personnels intervenant sur le projet. Cette équipe était pluridisciplinaire puisqu'elle était composée de développeurs mais aussi d'un graphiste et d'un intégrateur web. Ces deux derniers profils étaient dans un service différent de celui de la DSI et chacune de leurs interventions s'est déroulée sous contrôle du chef de service concerné. Que le *backlog* soit stocké dans un outil dédié ou pas, il faut qu'il soit partagé et visible par tous les membres de l'équipe projet.

3.11.5. Suivi des demandes, *scrumboard* et *burndown chart*.

Pour piloter le développement et suivre l'avancement du projet j'ai principalement utilisé l'indicateur de vélocité estimé en séance de *planning meeting* et le *burndown chart* produit automatiquement par Jira. Les *burndown charts* ne se sont pas révélés suffisamment précis pour me permettre de suivre l'évolution du développement au cours du *sprint*. Ce n'était pas réellement un problème car sur une itération de deux semaines et au regard du nombre de *stories* traitées, le *scrumboard* était suffisant pour suivre les travaux des développeurs. Dans Jira, nous avons planifié les itérations au fur et à mesure avec un niveau de granularité pour les *stories* qui ne dépassait pas la journée, et nous n'avons pas produit d'autres livrables que le *burndon chart*. L'utilisation de ce dernier est restée confidentielle au sein de l'équipe. Je pense que nous avons comblé ce manque en livrant rapidement une version opérationnelle de la plate-forme dès le mois de mai. Je pense également que communiquer les *burndowns* n'aurait pas apporté aux parties prenantes le niveau d'information requis. Les pratiques employées comme le formalisme des *user stories* n'était pas partagé. L'effort à produire et le temps à passer pour expliquer la démarche aux parties prenantes auraient eu un impact sur le pilotage du développement. Par contre, je pense que le graphique de *burndown* est un artefact intéressant pour communiquer l'avancement à des parties prenantes initiées.

La difficulté rencontrée dans le suivi des demandes réside dans le fait que les développeurs n'ont pas toujours traité les demandes une par une dans le *workflow* de Jira. Au lieu de cela, ils ont parfois attendu d'en avoir traité plusieurs pour les déclarer terminées. Ceci a rendu parfois difficile la lecture du *scrumboard* et pour réellement savoir ce qui était en cours, terminé ou à faire. De plus, cela a provoqué un effet d'escalier sur le *burndown* et comme aucune demande ne semblait être traitée pendant plusieurs jours, la courbe « *remaining value* » stagnait. D'un coup, le changement de statut de plusieurs demandes provoquait un brusque effondrement de la courbe. L'équipe a pu apprécier l'efficacité du *burndown chart* comme indicateur du travail accompli. Mais plus de discipline dans la gestion

des demandes du *backlog*, notamment sur les transitions entre états, aurait permis d'obtenir des *burndowns* bien formés et réellement significatifs de l'activité sur le *sprint*.

Nous avons donc utilisé un *scrumboard* numérique en remplacement du traditionnel tableau blanc et des post-its. Je pense que les deux types de *scrumboard* ont leurs avantages et leurs inconvénients. Le *scrumboard* traditionnel est plus ludique, plus visible et invite davantage au dialogue, à la collaboration et à l'engagement. Les membres de l'équipe peuvent se rassembler et discuter autour de lui. C'est un peu plus compliqué et moins pratique avec un *scrumboard* numérique. Il faut alors se munir d'un vidéo projecteur et d'un ordinateur. Ce dernier est alors l'interface de communication sur le *scrumboard* et il n'est pas toujours possible de laisser cette installation en fonctionnement toute la journée. De mon point de vue, le *scrumboard* traditionnel a pour inconvénient de ne pas permettre de traitement automatique des données. Il faut alors compléter l'exploitation de ce tableau par des saisies sur un tableur (outil le plus utilisé selon l'étude *state of agile survey* 2012). Sur ce point des outils comme Jira apportent un réel gain. Le partage des informations est également facilité par les acteurs. De plus, la tracabilité entre le code et les *stories* est assurée en branchant les dépôts de code sur le *backlog* et en se dotant d'un minimum de discipline en gestion de versions, qui consiste à placer la référence de la *story* dans les commentaires associés à la mise à jour des dépôts de code. Cette mise à jour des dépôts est désignée par le terme *commit*.

Les informations stockées et associées aux *stories* sous forme de commentaires (ici il s'agit de commentaires sur la *story* dans Jira, à ne pas confondre avec les commentaires du *commit*) sont alors très utiles en maintenance lorsque cette traçabilité est effective. On reproche parfois aux méthodes agiles une capitalisation plus difficile. Je pense que le stockage dans les outils des points d'effort estimés et des temps passés peuvent aider à mieux chiffrer un futur projet. Cependant, les projets, les fonctions, le contexte ne sont jamais exactement les mêmes d'un projet à l'autre et les chiffrages deviennent très vite approximatifs sur des projets importants. Pour chiffrer un projet, une séance de cotation avec les développeurs, sur un ensemble de *stories* me semble plus pertinente. Cette cotation peut éventuellement être ajustée en la rapprochant des référentiels pour communiquer avec des décideurs.

3.11.6. Le *daily meeting*

Lorsque j'ai proposé aux développeurs intervenant sur le projet d'utiliser Scrum, ils se sont globalement montrés enthousiastes. Malgré cela j'ai choisi de ne pas déployer tout le processus Scrum. Je pensais qu'il pourrait y avoir un risque de rejet des valeurs de l'agilité et des pratiques associées si nous rations cette première expérience. J'ai donc sélectionné les pratiques qui m'ont semblé les plus incontournables et j'ai laissé celles qui risquaient de générer un rejet avec un risque au final pour le projet.

Parmi les pratiques que j'ai choisi de ne pas exécuter sur le projet CCR, il y a d'abord le *daily meeting*. Je pense que le choix de ne pas exécuter le *daily meeting* s'est révélé être contre productif. Il permet normalement aux membres de l'équipe d'échanger, autour du tableau Scrum (*scrumboard*), sur les tâches en cours. Dans notre cas, comme la mêlée quotidienne autour du *scrumboard* n'avait pas lieu, nous n'avions pas cette vision partagée de l'avancement des développements et de l'implémentation des *user stories*. Les *stories* étaient traitées en bloc et les changements d'états enregistrés ne correspondaient pas à l'activité réelle puisque rien ne motivait le bon suivi du *workflow*. Je pense que l'exécution du *daily meeting*

aurait permis un suivi journalier des éléments du *backlog* avec plus de transparence et qu'il aurait encouragé le traitement des éléments de *backlog* de manière plus disciplinée. Le fait de devoir s'exprimer quotidiennement sur le degré d'avancement des tâches et sur les problèmes que l'on rencontre donne une sorte d'impulsion au groupe et encourage cette discipline quotidienne autour du *backlog* produit. Cette discipline a fait défaut entre septembre et octobre lorsque les demandes d'améliorations ont été plus nombreuses.

3.11.7. La rétrospective

J'ai choisi de ne pas déployer la pratique de rétrospective issue de Scrum (*sprint retrospective*) car elle m'a semblé trop éloignée culturellement des habitudes des développeurs. Je pense que nous sommes passés à côté de l'amélioration continue visée par la pratique. Les problèmes de suivi du *sprint* que j'ai rencontrés par l'absence de *daily meeting* ont été identifiés dès les premières itérations. En travaillant la rétrospective en fin de *sprint* nous aurions pu résoudre ce problème collectivement et peut être décider d'exécuter des *daily meeting*. D'une manière générale, nous avons accordé peu d'importance à l'amélioration du processus au cours du projet alors que cette pratique est fondamentale en agilité [BEC04] et plus largement en ingénierie.

3.11.8. L'intégration continue

Avant le projet CCR, nous avions déjà expérimenté l'intégration continue sur de petits projets développés en Java. Ces projets nous avaient permis de nous familiariser avec l'outil et la pratique. Sur des développement de produits logiciels à très faible niveau d'intégration, l'intérêt de la pratique se situe davantage sur l'automatisation des activités complémentaires à la construction. L'intégration continue en tant que telle a eu un intérêt limité sur cette phase de construction car celle-ci ne faisait pas l'objet d'un assemblage complexe de composants avec tests d'intégration à l'appui. Par contre, l'intégration continue a libéré les administrateurs des tâches de déploiement en environnement de validation. Ces tâches n'apportaient pas de « valeur » dans le processus. Elle a permis aux développeurs de déléguer la construction du produit pour le déploiement, de sorte que le contenu du serveur d'intégration faisait foi. Sans intégration continue, un développeur et un administrateur auraient été mobilisés pour chaque déploiement.

3.11.9. Le développement piloté par les tests.

Si nous n'avons pas retiré tout le bénéfice de l'intégration continue c'est parce que nous n'avons pas exécuté les pratiques de génie logiciel liées aux tests. L'ingénierie du test est un pilier de la qualité logicielle et de l'agilité. Sans tests unitaires, sans tests d'intégration et sans tests fonctionnels automatisés, il est très difficile de maintenir et de garantir la qualité du logiciel dans un contexte agile qui par nature génère des déploiement fréquents liés aux changements.

C'est ainsi que nous avons connu, pendant le projet, des phases de régression qui ont été particulièrement stressantes pour tout le monde. Ces phases de régression sont apparues entre septembre et octobre lorsque nous devions livrer le catalogue consultable par les

enseignants et les inspecteurs. La mise à jour du moteur de recherche en était en partie la cause mais plus globalement, ces régressions auraient pu être évitées si un patrimoine de tests unitaires avait été codé dès le début du projet,.

De manière générale, sur l'ensemble du projet, j'ai consommé en tant que *product owner* beaucoup trop de temps sur les tests utilisateurs. Par exemple les tests en *back-office*, sur le *workflow* de validation des fiches descriptives de ressources ont presque été rejoués manuellement à chaque *sprint* alors qu'ils auraient pu être automatisés.

3.11.10. Le rôle de *product owner*

En tant que *product owner*, je n'ai pas toujours été capable de répondre aux questions des développeurs, y compris lors des phases de préparation du *backlog* suivant. Ce manque d'information était dû au fait que même nos collègues du ministère n'avaient pas vraiment d'informations sur certains points du dispositif. C'était le cas par exemple, pour la gestion des subventions allouées aux établissements. Il y avait beaucoup de questions sur la manière dont devaient être constituées les subventions (participation de l'état et participation des collectivités locales) d'un établissement.

J'ai assumé mon rôle de *product owner* en formalisant des hypothèses que j'ai fait développer puisqu'il nous fallait avancer. Cette posture a généré quelques problèmes au sein des développeurs. De manière générale, pour l'ensemble des *user stories* implémentées, il m'a fallu présenter chacune d'elles comme des exigences du ministère car les développeurs ont eu parfois tendance à mettre ma parole en doute. Ce doute naissait aussi du fait que les utilisateurs du ministère, que je représentais, ne savaient pas toujours ce qu'ils souhaitaient, d'où un manque d'assurance dans le portage du besoin. Je pense aussi que je n'étais pas complètement légitime en tant que *product owner* car les développeurs ont l'habitude que je tienne un rôle de manager. Habituellement, lorsque je tiens ce rôle, je laisse les chefs de projet gérer leur projet et porter le besoin fonctionnel qu'ils doivent faire implémenter par les développeurs. Dans le cadre du projet, puisque j'étais le *product owner* par défaut, la direction générale aurait pu légitimer officiellement mes activités dans ce rôle auprès de l'équipe. J'aurais également pu partager la pratique d'écriture des *user stories* avec le ministère pour faire en sorte que les demandes soient plus légitimes et pour impliquer davantage les utilisateurs dans l'expression du besoin tout au long du projet.

3.11.11. Bilan sur l'organisation et les rôles.

Sur le plan organisationnel, l'équipe de développement s'est mise en place au travers du prisme du management hiérarchique. Aucun chef de projet en tant que tel n'a été désigné et les opérations ont finalement été conduites par des managers hiérarchiques, dont je fais partie, qui en ont endossé officieusement le rôle et les fonctions. Le directeur général adjoint de l'établissement avait géré l'opération « Ecoles Numériques Rurales ». Il a donc naturellement géré le dispositif « Catalogue Chèques Ressources » puisque les deux opérations étaient très semblables. Le directeur adjoint a tenu un rôle proche de celui d'un directeur de projet. J'emploie le terme de directeur de projet car il est davantage intervenu en comité de pilotage sur les questions politiques et stratégiques. Nous avons fonctionné en binôme et je me suis chargé de piloter la réalisation de la plate-forme. Je pense que ce tandem a bien fonctionné

tout au long du projet car nous avons toujours très bien communiqué. J'ai eu toute la liberté nécessaire pour organiser le processus de développement, ce qui m'a permis de déployer les pratiques agiles que j'avais sélectionnées. J'ai pu communiquer librement avec mes collègues du ministère et avec les éditeurs privés pour le développement de la plate-forme tout en rendant compte régulièrement au directeur adjoint. Je pense que l'organisation aurait été complètement optimale si toutes les compétences avaient été vraiment déléguées au sein de l'équipe projet.

Dans le cadre du projet, j'ai assuré un rôle inhabituel auquel les développeurs de l'équipe ne sont pas habitués. Ils sont généralement placés dans une équipe projet et travaillent en collaboration avec un chef de projet extérieur au service. Ils disposent d'une certaine autonomie pour travailler le besoin en collaboration avec le chef de projet. A mon avis, ils tirent d'une certaine manière de cette organisation, de la reconnaissance et de l'estime qu'ils n'ont pu obtenir de la même façon sur le projet CCR. L'exécution du projet est restée finalement assez confidentielle au sein de la DSI. De plus ayant adopté un double rôle de *product owner* et de *Scrum master*, j'étais de ce fait leur seul contact tant sur les aspects méthodologiques que sur les différents aspects du projet, ce qui a pu générer parfois un peu de frustation, là où les développeurs travaillent habituellement avec nos collègues des services métiers. L'organisation mise en place sur le projet CCR était exceptionnelle et ne sera pas réitérée. En général, les managers ne tiennent pas le rôle de chef de projets.

3.11.12. Principe agile de communication et de collaboration

Les principes et pratiques de l'agilité qui tendent à favoriser la communication et la collaboration n'ont pas été totalement appliqués, ce qui a provoqué des retards et une qualité en dessous des attentes sur les aspects graphiques et ergonomiques de la plate-forme. L'identité graphique de la plate-forme est par exemple est très pauvre, comparée à la précédente opération ENR.

Toutes les compétences requises n'étaient pas intégrées dans l'équipe projet. Aussi la conception web graphique a été exécutée en dehors de l'équipe projet tout comme les travaux d'intégration web. Les développeurs n'ont pas pu travailler correctement avec le graphiste et les intégrateurs. Ces derniers n'ont pas eu toutes les informations utiles pour mener à bien leurs activités. Aussi les produits d'activité du graphiste et des intégrateurs ont donc été récupérés en l'état par l'équipe projet. Que l'exécution du projet soit agile ou conventionnelle, il est important que l'équipe projet intègre toutes les compétences et que les membres de cette équipe puissent communiquer sans barrière organisationnelle et-ou hiérarchique.

3.11.13. Bilan sur les pratiques de gestion de projet

En dehors des pratiques agiles sélectionnées pour le développement logiciel, les pratiques et les processus de gestion de projet exécutés sur le projet CCR n'ont pas fait l'objet d'une sélection préalable, ni d'une préparation en amont. Elles ont été exécutées de manière informelle au fil du projet, ce qui a engendré des difficultés de gestion et d'exécution du projet.

Identifier et choisir les pratiques en se basant par exemple sur un guide comme le

pmbok ou un référentiel comme le CMMI facilite, de mon point de vue, l'installation d'une certaine discipline sur le projet. Dès lors, l'objectif n'est plus seulement le produit ou le service, mais bien également d'exécuter le projet avec un certain niveau de qualité sur les activités de gestion.

3.11.14. Conclusion

Choisir et déployer des pratiques Scrum n'a pas été si simple. Il a fallu expliquer et rester vigilant, tout au long du projet sur l'adhésion de l'équipe aux pratiques. L'enthousiasme dominait, mais j'ai tout de même fait attention à conserver cette adhésion intacte. Je pense que la nouveauté et le côté ludique de certaines pratiques, comme celle du *planning poker* y ont contribué.

Les pratiques sont vraiment interdépendantes. Elles ne délivrent toute leur efficacité qu'en étant combinées les unes aux autres. Je pense que nous aurions été plus efficaces en adoptant l'ensemble du *framework* Scrum quitte à abandonner certaines pratiques par la suite grâce à l'expérience acquise.

Malgré tout, notre processus agile, bien que de faible maturité, nous a permis de tenir nos engagements. Si nous avions suivi un cycle de développement classique, la plate-forme n'aurait jamais été livrée à temps. Nous n'aurions tout simplement pas commencé le développement car nous aurions attendu une expression des besoins formalisée de la part du ministère qui ne serait jamais arrivée à temps. Nous aurions passé le printemps et le début de l'été à écrire des spécifications, à les faire valider et corriger. Au lieu de cela, nous avons anticipé sur le caractère d'urgence de la situation en nous engageant au plus tôt dans le développement, tout en repoussant les décisions non critiques pour le développement du produit. Nous avons commencé le produit, sur la base de la note émise par la DGESCO, tout en acceptant l'incertitude sur les fonctionnalités à développer. Par la suite, la plate-forme ne s'est construite qu'à l'aide du *feedback* des utilisateurs.

4. Les pratiques depuis le projet CCR

Depuis Janvier 2012 et le développement du projet CCR, nous continuons à exécuter des pratiques agiles empruntées aux f*rameworks* Scrum et XP. Si l'expression et la gestion du besoin sont le cœur de ces pratiques, d'autres sont désormais exécutées pour favoriser le partage et la communication au sein de l'équipe. De nouveaux outils, sans être prédominants, sont également utilisés dans le cycle de développement pour assurer la traçabilité et le partage du code.

4.1. Le recueil et la gestion du besoin par les *user stories*

Depuis le projet CCR, l'équipe n'a pas eu l'occasion de pratiquer de manière globalisée et institutionnalisée les pratiques liées à l'expression du besoin et à sa gestion. Je fais allusion ici à la discipline dans son ensemble, qui s'étend de l'écriture des *stories* jusqu'à leur estimation et planification dans un plan d'itérations. En revanche, les pratiques subsistent et sont exécutées de manière unitaire en fonction des projets.

4.1.1. Pénétration de la pratique

Depuis mon intervention sur le projet CCR, j'effectue la promotion des pratiques auprès de mes collègues des services métiers. Cette présentation du formalisme et des pratiques induites est effectuée au fil du temps lorsque j'interviens sur les phases de préparation et de planification des projets. A ce jour, j'ai communiqué principalement sur un principe et une pratique. D'abord, j'ai insisté sur un suivi plus étroit du développement de la part du chef de projet et j'ai ensuite proposé que les cahiers des charges se limitent aux informations strictement utiles, pour le lancement du projet et pour la communication avec les décideurs. Cette dernière proposition est accompagnée d'une présentation du formalisme de la *user story* et des pratiques associées. Ces propositions et suggestions ont conduit à l'écriture de *stories* sur 14 projets depuis janvier 2012. Cinq sont en cours, un a été abandonné et un autre est en attente et sera exécuté en 2014. Environ 300 *stories* sont référencées sur ces projets et environ 70 sont en cours d'écriture et en attente de saisie dans Jira (voir figure 41).

Projet	Etat du projet/produit	*Stories* référencées	*Stories* non référencées	Estimation en *planning poker*	Planification et plan d'itération
ABCD de l'égalité	En cours	58	0	**oui**	**oui**
Antigone	En attente	0	43	**oui**	non
Corpus	En cours	0	32	prévue	prévue
Carte des ressources culturelles	En cours	0	N/C	**oui**	**oui**
Cndp.fr (espace philo, musagora, divers)	En maintenance	7	N/C	non	non
Concours dis moi dix mots	En maintenance	34	0	non	non

Dictionnaire des écoliers	En maintenance	8	0	non	non
Eduthèque	En cours	7	0	non	non
Modèle Commun	Abandonné	24	0	**oui**	non
Numéribase	En maintenance	14	0	non	non
Convention	En maintenance	11	0	non	non
Gestion des abonnées	En cours	3	0	non	non
Site de l'agence des usages	En maintenance	35	0	non	non
Vérificateur url	En maintenance	55	0	non	non

Figure 41 : Tableau de présentation des projets support à la pratique des *user stories*

4.1.2. Exécution de la pratique

L'écriture des *stories* est effectuée par les chefs de projets à partir du cahier des charges qu'ils ont rédigé. En général, le cahier des charges contient une description des grandes fonctions du produit. A partir de ces fonctions, un découpage fonctionnel plus précis est opéré. En fonction du projet et des collègues en présence, cet exercice peut prendre différentes formes. Soit le chef de projet s'empare du formalisme et écrit seul les *stories*, soit celles-ci sont écrites en atelier. Dans ce dernier cas, nous assurons un accompagnement pour aider le chef de projet à les rédiger. Parfois, le porteur du besoin est un référent en charge du produit. Pour effectuer le découpage, nous prenons une fonction de haut niveau du système et nous amenons le porteur du besoin à imaginer plus précisément les sous-fonctions en lui posant des questions. Le but est ici de trouver le bon niveau de granularité pour l'écriture des *stories*. Lorsque le niveau semble correct nous traduisons le besoin en *story*. Après quelques traductions le porteur est capable de les exprimer lui-même. Sur les projets que nous exécutons aujourd'hui, le bon niveau de granularité est atteint lorsque les *stories* sont implémentées en 2 à 3 jours environ.

Les *stories* sont ainsi écrites lorsque le projet est accepté par le comité de programmation mais l'écriture peut intervenir à n'importe quel stade du projet. Dans le cas du projet de refonte du produit « Antigone », le comité de programmation demandait une estimation du temps nécessaire pour développer en HTML5, une version du produit comportant les mêmes fonctions. Une trentaine de *stories* ont donc été écrites par le chef de projet en se basant sur le produit existant. Nous avons ensuite exécuté une séance de *planning poker* pour estimer les *stories* et dégager une estimation du temps de développement sur ce premier lot.

Les *stories* et les critères d'acceptation constituent les principales spécifications du besoin. Elles peuvent être accompagnées de prototypes d'IHM sous forme de *mockup*, de cas d'utilisation textuels (annexe 6) ou bien encore de diagrammes UML (diagramme de cas d'utilisation, diagramme de classe).

4.2. Les *stand-up*

Nous effectuons deux *stand-up* par semaine le mardi et le jeudi. L'ensemble de

l'équipe y participe soit dix personnes. Chaque membre s'exprime rapidement sur ce qu'il a réalisé depuis le *stand-up* précédent, ce qu'il est en train de faire ou va faire et les problèmes qu'il rencontre. Le *stand-up* a une durée comprise entre vingt et trente minutes. Il se tient en tout début de journée. Les tâches présentées par les uns et les autres appartiennent aux différents projets en cours d'exécution, le *stand-up* est donc une réunion qui agrège ces échanges d'information. C'est aussi une réunion assez rythmée effectuée sans support, sans document ; les membres de l'équipe sont debout et dans une posture qui encourage la communication.

Nous terminons parfois la séance par un vote ROTI (*Return On Time Invested*) à main levée pour contrôler le niveau et la qualité des informations échangées pendant la réunion. Chacun s'exprime sur une échelle de 1 à 5 sur la pertinence des informations échangées en *stand- up* (voir figure 42).

1 doigt	Je n'ai rien appris, j'ai perdu mon temps.
2 doigts	J'ai appris des choses mais j'en retire peu de bénéfices au regard du temps passé.
3 doigts	Réunion moyenne.
4 doigts	J'ai récupéré des informations utiles qui vont m'aider. Je n'ai pas perdu mon temps.
5 doigts	Les informations échangées valaient très largement le temps passé.

Figure 42 : Échelle de notation ROTI (*Return On Time Invested*).

4.3. La rétrospective

Nous effectuons une autre réunion plus longue (2 heures) dite rétrospective pendant laquelle l'équipe s'interroge sur ses pratiques et sur l'exécution des projets en cours et terminés. Les échanges sont disciplinés autour de la méthode des six chapeaux d'Edward de Bono [BON12].

J'anime cette réunion de travail en poursuivant plusieurs objectifs :

- Partager les avis et l'expérience des uns et des autres sur les pratiques et sur les projets.
- Identifier ce qui peut être amélioré.
- Abandonner les pratiques inutiles et éviter les gaspillages.

A la fin de la réunion nous choisissons un axe de travail sur lequel nous engageons des actions correctives.

4.4. Les outils

Pour supporter nos pratiques, nous utilisons des outils logiciels commerciaux de l'éditeur Atlassian en complément d'outils libres et gratuits. Deux de ces outils nous

permettent d'assurer la traçabilité du besoin, de son expression sous forme de *story* jusqu'à son implémentation dans le code source. Il s'agit de Jira et de Fisheye.

4.4.1. Jira

Nous avons utilisé Jira sur le projet CCR jusqu'en janvier 2012 mais ce n'est qu'en septembre 2012 que son utilisation a été généralisée à l'ensemble des projets de développement logiciel. Aujourd'hui, Jira supporte donc toutes les activités de développement logiciel que ce soit en maintenance ou en mode projet. Plus précisément, il est le support des pratiques de gestion du besoin. D'une part, il sert de *backlog* et donc de référentiel d'exigences et d'autre part il permet la planification et le suivi de l'implémentation de ces exigences. Aujourd'hui, on dénombre 64 utilisateurs, tous rôles confondus pour un total de 1400 demandes environ au mois de Juillet 2013.

Nous avons modifié la configuration native de Jira pour l'adapter à notre flux de travail (voir figure 43) . Aussi depuis le projet CCR nous utilisons un nouveau tableau Kanban. Ce dernier est commun à l'ensemble des projets et est partagé par toute l'équipe mais son affichage peut être personnalisé par des filtres pour permettre à chaque membre de l'équipe de ne voir que les demandes qui le concerne.

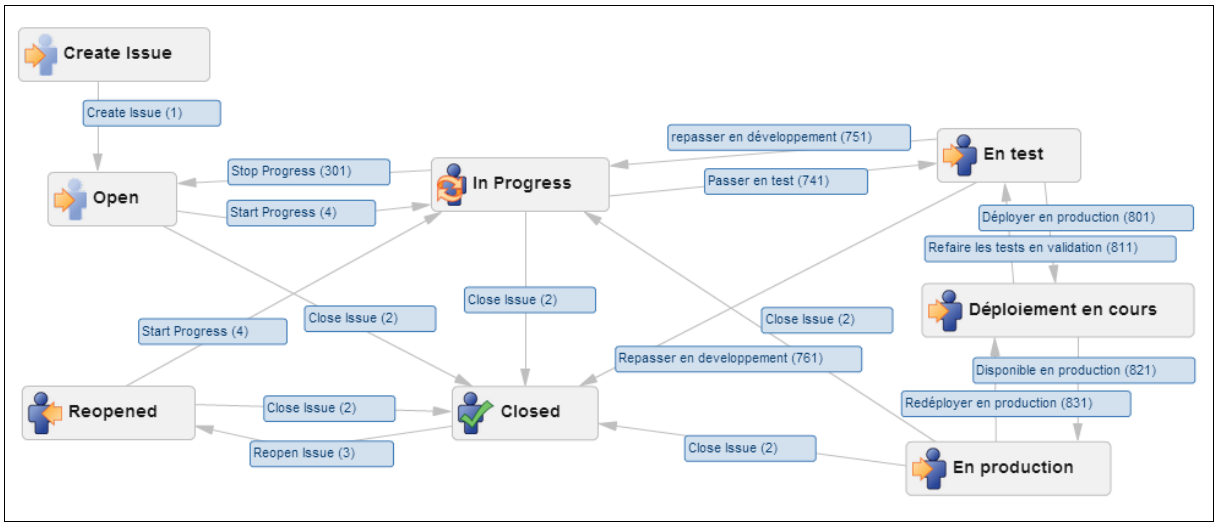


Figure 43 : *Workflow* suivi par les demandes de développement.

Nous avons placé une limite sur l'état « en-cours ». Chaque membre de l'équipe ne devrait pas avoir en théorie plus de deux demandes dans cette colonne ce qui fixe le nombre de demandes traitées en simultané à 20 quel que soit leur type (*bug, story*, tâche etc.) et quel que soit le projet. L'objectif de cette limite est d'améliorer la visibilité sur le flux de travail conformément aux préconisations Kanban.

Enfin, Jira nous permet de contrôler notre temps de cycle sur le traitement des demandes, en particulier sur les défauts relevés par les utilisateurs. Ce temps de cycle est compris entre 3 et 5 jours pour des défauts non bloquants. Il peut s'agir par exemple d'un problème d'affichage dans un navigateur ou bien d'un contrôle de saisie mal implémenté. Les défauts plus urgents sont traités sur le champ car en général la demande Jira est doublée d'un appel téléphonique ou d'un mail. Les temps de cycle importants visibles sur le graphique ci-

dessous (figure 44) concernent des défauts sur des produits dont la maintenance n'est plus assurée ou alors avec beaucoup de délai. C'est par exemple le cas de la plate-forme P@irformance V2 qui est en cours de refonte ou bien encore l'application d'indexation documentaire Numeribase.

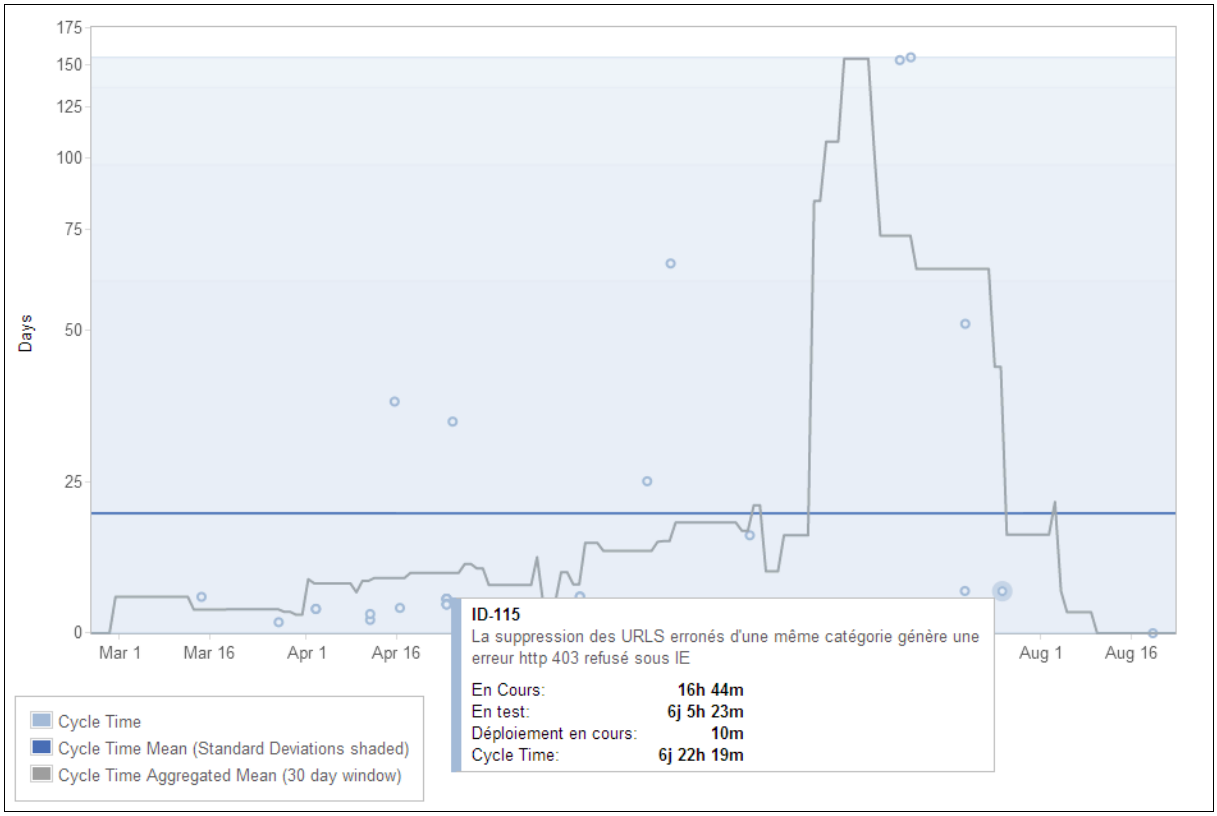


Figure 44 : Graphique représentant le temps de cycle sur les défauts.

4.4.2. Fisheye

Nous utilisons depuis janvier 2013 le logiciel Fisheye. C'est un outil d'exploration de code. Il est développé et commercialisé par la société Atlassian. Il nous permet de supporter deux pratiques importantes en XP que sont la base de code unique et le code partagé (voir paragraphe 2.8.3.). Il est important pour l'équipe entière d'identifier rapidement toute les versions existantes d'un code source. De plus il est important que le code puisse être modifié par un développeur qui n'en est pas l'auteur. Ces modifications peuvent se faire d'autant plus rapidement que le code et l'activité sur le code sont rendues visibles au quotidien.

Pour cela, l'outil est branché sur les dépôts de code Subversion. Il permet d'accéder à l'ensemble du code sur tous les dépôts configurés mais surtout, il donne une visibilité accrue sur l'activité au sein des dépôts. Cette visibilité est supérieure à ce qu'il est possible de voir dans un environnement de développement comme Netbeans ou Eclipse. Le code est consultable depuis un navigateur sans utiliser ces environnements et sans récupérer le code en local sur son poste. Il permet à chacun de voir et de retrouver ce qui a été réalisé sur les projets au travers de l'historique des dépôts Subversion. Le pré-requis est que chaque développeur doit penser à commenter son activité sur les dépôts en remplissant le champ

commentaires lors du *commit* Subversion. C'est précisément ici que la traçabilité de l'implémentation des demandes Jira est assurée. Il suffit aux développeurs de placer la référence de la demande dans le commentaire du *commit*. Toute l'activité Subversion est indexée par Fisheye, il est très simple par la suite de faire une recherche de tous les changements intervenus sur la base de code pour une demande précise (*bug, story*, etc.).

4.5. Devops

Le mouvement Devops est né aux alentours de l'année 2010. Il s'agit d'un mouvement respectueux des valeurs de l'agilité. Le principal point de convergence avec les méthodes agiles réside dans l'importance accordée à la valeur apportée aux utilisateurs. Ce mouvement pose le postulat de départ suivant : les développeurs et les administrateurs systèmes ne poursuivent pas les mêmes buts. Les développeurs produisent de nouvelles fonctionnalités pour les utilisateurs, potentiellement porteuses d'instabilité pour le système d'information alors que les administrateurs travaillent justement sur la stabilité et la sécurité du système. Cet antagonisme entre les équipes de développeurs et d'administrateurs est souvent source de conflits dans les DSI. Le mouvement Devops prône le rapprochement des équipes de développeurs et d'administrateurs en leur faisant partager au quotidien leurs problématiques et en les faisant collaborer étroitement autours d'objectifs partagés. Le but étant bien ici de sortir d'une logique d'optimums locaux. L'optimum global n'étant pas la somme des optimums locaux, il y a fort à parier que l'utilisateur final ne sera pas satisfait du travail réalisé par des équipes soucieuses uniquement de leurs problématiques.

La DSI du CNDP n'a pas échappé à cet antagonisme entre développeurs et administrateurs mais en travaillant sur les valeurs de l'agilité nous avons dégagé certaines pratiques dans l'esprit du mouvement Devops.

La géographie du service permettait une collaboration étroite entre administrateurs et développeurs. La proximité entre les développeurs et les administrateurs est maintenue. Une attention particulière est désormais portée au maintien de cette proximité d'autant que la DSI devrait faire l'objet d'une réorganisation géographique durant l'hiver 2013/2014. Cette proximité permet aux uns et aux autres de se porter mutuellement assistance en cas de besoin. Il peut s'agir de trouver la source d'une instabilité, ou bien d'un manque de ressources systèmes, ou bien encore d'un manque de droit sur une partie du système. Dans ce cas il est très fréquent de voir un administrateur et un développeur collaborer en face à face à la résolution d'un problème.

Ensuite, nous organisons des points mensuels entre les développeurs et les administrateurs pour faire le point sur les pratiques. Chacun peut s'exprimer sur les difficultés rencontrées à son poste. Ces points de dialogue sont souvent l'occasion pour les administrateurs de présenter ou de préciser des points d'architectures systèmes importants. De manière générale, nous abordons lors de ce point mensuel, tous les éléments techniques qui doivent être partagés et connus par les administrateurs et les développeurs. Il peut s'agir par exemple de changement d'adresses IP ou de nom de domaine de certains serveurs. Il peut aussi s'agir de virtualisation, de nouvelles règles de sécurité, ou bien de règles de déploiement.

Enfin, nous travaillons à l'automatisation des tâches répétitives apportant peu de valeur ajoutée pour les développeurs mais aussi pour les administrateurs. Ces travaux

d'automatisation rejoignent l'intégration continue déjà implémentée sur les développements et tendent désormais vers une logique de déploiement continu.

4.6. Bilan des pratiques et orientations

Les pratiques présentées sont exécutées dans l'équipe depuis le printemps 2011 si j'inclus le projet CCR. J'ai décidé de les promouvoir au sein de la DSI mais aussi auprès de nos collègues des services métiers car j'ai la conviction qu'elles nous permettent de développer dans de meilleures conditions et de produire au final du logiciel de meilleure qualité. Plus le nombre de projets exécutés avec ces pratiques sera important et plus nous pourrons mesurer notre évolution au travers des indicateurs que nous utilisons (nombre de défauts, temps de cycle etc.). Cependant, il reste difficile d'évaluer scientifiquement l'apport réel de certaines pratiques comme le *stand-up* par exemple. Il est possible de mesurer le nombre de défauts par projet, de mesurer le temps de cycle ou la qualité du code mais quel sera l'impact sur ces indicateurs si nous supprimons le *stand-up* ou la rétrospective ? Est-il évident et pertinent de chercher à faire le lien entre ce type d'indicateur et les pratiques ? Faute de critères réellement objectifs, la conviction, basée sur l'expérience immédiate des membres d'une équipe, que la pratique est bénéfique, justifie sans aucun doute son utilisation.

4.6.1. La *user story*, une pratique phare.

Je pense que notre pratique de l'agilité s'est structurée autour de l'écriture des *user stories* et que c'est le point de convergence de toutes les autres pratiques que nous exécutons comme la conception incrémentale ou bien l'intégration continue. Peu à peu notre processus de développement s'oriente vers un mode « piloté par les *stories* » ou « *story driven* » [COH04]. Cette pratique est très efficace dans l'élucidation du besoin. Elle donne un côté presque ludique à son expression. Nous avons pu constater comment elle a permis à certains de nos collègues métiers, qui se sont véritablement emparés du formalisme, de débloquer cette expression du besoin. Par contre, il n'est pas toujours évident au début de se placer au bon niveau de granularité. L'écriture des critères d'acceptation nécessite un accompagnement. La plupart des *stories* produites ne sont pas toujours fournies avec ces critères (voir paragraphe 3.11.2.).

De plus, l'écriture des *user stories* met en évidence la nécessité de confier cette écriture à des personnels qui possèdent le niveau de connaissances requis en fonction du domaine ou de la cible concernée. Par exemple, il sera difficile de produire des *stories* sur un système d'indexation documentaire si l'on ne connaît pas ce type d'application et que l'objectif n'est pas assimilé. Lorsque le porteur du besoin ne possède pas assez de connaissances sur le domaine, l'expression du besoin, quelle que soit sa forme, ne permet pas aux développeurs d'aborder le développement dans de bonnes conditions. Lorsque les développeurs commencent à poser des questions, l'expression du besoin, quelle que soit sa forme, ne permet pas aux développeurs d'aborder le développement dans de bonnes conditions. Lorsque les développeurs commencent à poser des questions le porteur peut se retrouver alors en difficulté. Il a alors du mal à répondre aux questions portant sur un produit ou un service qu'il ne connaît pas vraiment.

Enfin, nous travaillons plutôt au quotidien dans une émergence des spécifications tout au long du projet, même si ce mode de fonctionnement ne s'inscrit pas encore systématiquement dans un vrai cycle de développement discipliné, itératif et incrémental. Le formalisme de la *user story* est très adapté et permet une traçabilité du besoin au code qui n'était pas possible auparavant.

4.6.2. Les *stand-up* et la rétrospective

Concernant les *stand-up* et la rétrospective, le bilan est plus mitigé. Dans la mesure où les *stand-ups* sont communs à l'ensemble des projets, ils permettent à chaque membre de l'équipe de savoir ce que font les uns et les autres. Le *stand-up* participe à la transparence dans l'équipe. C'est une pratique très installée et elle sera exécutée même si l'équipe n'est pas au complet. Cependant elle est exécutée uniquement au sein de l'équipe de développeurs et nos collègues des services métiers (porteurs de besoins, chefs de projet etc.) ne sont pas présents. Je pense que cette pratique devrait être partagée avec eux.

La rétrospective est également une pratique qui devrait être exécutée avec nos collègues des services métiers pendant, ou au moins à la fin des projets. A défaut, nous l'exécutons dans l'équipe pour tenter d'améliorer nos pratiques. Nous manquons de régularité sur l'organisation des rétrospectives et nous manquons également de méthode et de discipline pour appliquer les actions correctives identifiées. Si bien que le bénéfice de cette pratique est pauvre même si c'est un moment de communication et de partage apprécié et demandé par les membres de l'équipe.

4.6.3. Retours sur les outils.

« Les individus et leurs interactions plus que les processus et les outils ». C'est ce que prône le manifeste agile. Nous veillons à garder la main sur les pratiques. Il est important que les logiciels viennent supporter nos pratiques et que ces dernières ne soient pas induites par les outils. Jira est un de ces outils et il est utilisé dans cet esprit. Cependant il a créé de la confusion parmi certains utilisateurs qui ne comprennent pas la différence entre un outil comme celui-ci et un outil de gestion de projet traditionnel comme Sciforma. Je pense que ces deux outils sont complémentaires, chacun supportant des pratiques précises. Jira permet de suivre des activités de développement assez fines, en fonction d'un découpage fonctionnel précis effectué tout au long du projet. Sciforma est un outil de planification et de gestion de ressources.

4.6.4. Appréciation sur nos pratiques Devops

Les pratiques Devops nous ont permis de rapprocher les développeurs et les administrateurs système. Ils travaillent aujourd'hui dans une bonne entente et dans un bon état d'esprit. La collaboration est de mise. Les uns et les autres apprennent à s'écouter et à se comprendre sans défiance. Comme pour les rétrospectives, nous avons manqué de régularité au printemps 2013 sur nos rendez-vous Devops à cause d'une activité importante qui a laissé peu de place à ces temps d'échange.

4.6.5. Orientations des pratiques

Désormais, d'autres pratiques vont venir compléter celles que nous exécutons déjà. Nous allons travailler sur les tests unitaires et les tests fonctionnels, sur tous nos développements quelle que soit la technologie. L'ingénierie du test est fondamentale en agilité. Elle seule nous permettra de maintenir un haut niveau de qualité sur nos réalisations. Ces tests seront automatisés sur le serveur d'intégration continue. En plus d'être un indicateur de suivi supplémentaire des développements, les tests nous permettront d'anticiper la découverte des défauts.

Nous allons également mettre l'accent sur la pratique XP de code partagé. Même si le code est aujourd'hui de plus en plus partagé, les corrections ou évolutions sont souvent produites par l'auteur du code. Pour cela un outil complémentaire à Fisheye et à Jira va venir en support. Il s'agit de Crucible de la société Atlassian. Nous possédons l'outil mais son utilisation n'est pas effective. Cet outil permet d'organiser des revues de code et d'en consigner les résultats. Par exemple, lorsqu'un développeur travaille sur un code, il peut demander dans l'outil, une revue du code par une ou plusieurs personnes. Cet outil pourrait nous conduire à revoir notre *workflow* dans Jira. Il pourrait par exemple y avoir un état supplémentaire de revue de code pour les *stories*. L'objectif ici poursuivi est de rendre possible l'intervention d'un développeur sur un code qu'il n'a pas produit. Crucible associé à Fisheye et Jira participera grandement à ce partage de connaissance du code.

5. Projection des pratiques sur l'organisation.

Depuis l'été 2012, l'établissement se réorganise et revisite son mode de fonctionnement en mode projet. Dans le cadre de cette réorganisation, les rôles et les responsabilités des uns et des autres sont parfois redéfinis ou précisés. L'outil de gestion de projet Sciforma acquis dans le Plan d'Évolution du Système d'Information sera opérationnel en Septembre 2013.

Les pratiques agiles peuvent s'inscrire dans les changements qui s'opèrent au CNDP. Elles sont complémentaires et ne remettent pas en cause le travail effectué sur la gouvernance des projets, sur la mise en place des instances, sur la configuration de Sciforma et de son utilisation.

Beaucoup de pratiques ne concernent que les développeurs mais elles font partie d'un ensemble plus large de pratiques exécutées sur les processus projet partagés avec les métiers. Ces pratiques n'ont de sens et ne sont efficaces que lorsqu'elles sont exécutées dans cette dimension collaborative. C'est par exemple le cas de l'écriture des *user stories* dont découlent les pratiques de test ou d'estimation.

Choisir d'adopter ces pratiques ne remettra pas en cause l'organisation mais certains ajustements pourraient être nécessaires notamment dans la définition des rôles et des responsabilités.

5.1. Principes et pratiques préconisées.

La compréhension et l'acceptation des principes ainsi que l'exécution des pratiques ci-dessous sont préconisées dans les services métiers et globalement pour tous les personnels qui ont un rôle à jouer dans la réalisation d'un projet de développement logiciel. Elles constituent le cœur des pratiques. Elles donnent du sens et structurent les activités des membres d'équipes projets (développeurs, intégrateurs, graphistes, ergonomes etc.). Ces pratiques devraient être institutionnalisées pour légitimer l'exécution de processus de développement agiles pilotés par les *stories*.

- Principes d'émergence du besoin et des spécifications tout au long du projet

  Parce qu'aujourd'hui ce principe est implicite sur la majorité des projets, il devrait être désormais expliqué et mis en avant comme un principe clé dans l'exécution des pratiques. Cette émergence des spécifications tout au long du projet se fait grâce au f*eedback* des utilisateurs, du responsable produit et du chef de projet sur la base d'éléments concrets qui sont le produit d'itérations.

- Ecriture des *user stories*

  Le découpage du produit en unités fonctionnelles permettrait de développer dans un cycle itératif et incrémental. Cette pratique est simple à assimiler et facilite grandement la spécification du besoin. L'acceptation du projet par le comité de programmation serait le point de départ de la production des *stories*.

- Estimation et planification

En fonction de la complexité et de l'envergure des projets, les pratiques d'estimation et de planification des *user stories* pourraient être exécutées. Il s'agirait ici de choisir quelles *stories* seraient implémentées dans l'enveloppe de temps allouée par le comité de programmation. En fonction de son envergure, le projet pourrait faire l'objet d'un découpage en itération. Ce découpage pourrait être porté dans Sciforma. Le contenu de chaque itération (les *stories*) étant géré dans Jira.

5.2. Rôles et responsabilités

Institutionnaliser des pratiques nécessite aussi de préciser les rôles et les responsabilités de chacun vis à vis des pratiques en question.

- Le chef de projet

  Quel qu'en soit le type (voir paragraphe 1.3.4.), le chef de projet gère la communication sur le projet. Il coordonne également l'ensemble des activités du projet (juridiques, financières, commerciales, techniques...) et lève tout obstacle qui pourrait se présenter. Il fait le lien entre la gouvernance et l'exécution. Il rassemble et maintient à jour toutes les informations utiles à la gestion du projet sur les axes coûts, qualité, délais. Sciforma est son outil. Il peut être affecté à n'importe quel service (édition, communication, marketing, documentaire etc.) mais il dépend d'un bureau d'étude projet qui lui apporte le support nécessaire sur les aspects méthodologiques.

- Le responsable produit

  Il est responsable de la spécification du besoin sur le projet. Il rédige les *user stories* et les critères d'acceptation associés. Pour cela, il peut se faire aider par des utilisateurs ou des concepteurs. Néanmoins, il connaît le domaine concerné par le projet. Il porte la responsabilité de maximiser la valeur du produit (nombre et pertinence des fonctions implémentées) dans l'enveloppe de moyens accordée par le comité de programmation. C'est donc lui qui travaille avec les développeurs en *planning poker* et *planning meeting*. Le plan d'implémentation des *stories* devra être conforme à la planification prévue avec le chef de projet avec qui il travaille en étroite collaboration ainsi qu'avec l'équipe de réalisation. Jira est son outil. Le responsable produit est un expert métier. En fonction de l'envergure ou de la complexité du projet, ce rôle peut être assuré par le chef de projet.

- Les membres d'équipes

  Les membres d'équipes sont tous les personnels qui participent à la réalisation du produit. Ils portent collectivement la responsabilité de l'implémentation des *stories* dans le produit conformément à ce qu'ils ont estimé en séance. Ils endossent cette responsabilité auprès du responsable produit et du chef de projet à partir du moment où ils acceptent les *stories* produites. Les membres d'équipes sont les développeurs, les intégrateurs, les graphistes et les ergonomes.

Dans cette projection, les rôles de responsable de produit et de chef de projet sont distincts pour plusieurs raisons. Sur les projets, les personnels peuvent actuellement se retrouver en difficulté parce qu'ils doivent de manière implicite assurer les deux rôles et qu'ils n'ont pas toujours la possibilité de le faire correctement. En effet, le chef de projet peut gérer plusieurs projets pour lesquels il a de multiples tâches à coordonner et à assurer comme la gestion des aspects juridiques, la production de documents pour les instances, la mise à jour d'informations dans les outils, il n'a pas toujours le temps (et c'est encore plus visible sur un projet court) d'écrire de bonnes *stories* pour des développeurs et de travailler en étroite collaboration avec eux sur les pratiques citées précédemment, ce qui au final nuit à la qualité d'exécution du projet et à la qualité du produit final. Néanmoins, il est certain que si les rôles de chef de projet et de responsable produit sont confiés à des personnes distinctes le temps d'un projet, ces dernières devront travailler en étroite collaboration pour sa réussite.

5.3. Cérémonies

- La rétrospective

  Cette pratique serait exécutée au moins une fois en fin de projet. L'objectif serait de placer les personnels intervenant sur les projets dans une démarche d'amélioration continue des pratiques. Les informations récoltées pendant cette rétrospective seraient consolidées et portées à la connaissance des chefs de projet et des responsables produit.

- La démonstration

  La présentation du produit par les membres de l'équipe auprès du responsable et du chef de projet viserait à montrer le respect des engagements des membres de l'équipe sur les *stories* prises en charge.

- Les *stand-up*

  Ils permettraient à l'équipe projet de faire le point plus fréquemment sur l'avancée de la réalisation. Le responsable produit pourrait être présent à ce *stand-up* dans la mesure où cette réunion journalière serait complètement dédiée au projet.

5.4. Démarche

Pour promouvoir les valeurs, les principes et les pratiques agiles au sein de l'établissement, plusieurs actions peuvent être envisagées.

- Former les personnels aux méthodes et aux pratiques. La cible serait constituée des chefs de projet, des responsables de produit, des chefs de service. Cette formation pourrait être effectuée par un coach agile extérieur à l'établissement.

- Accompagner sur les pratiques. La cible serait constituée des chefs de projet, des

responsables de produit. Cet accompagnement pourrait être effectué dans un premier temps par un coach agile extérieur à l'établissement puis dans un deuxième temps par les praticiens les plus expérimentés.

- Créer et animer une communauté de chefs de projets et de responsables produits formés aux pratiques agiles. L'objectif serait de favoriser le partage de connaissances et d'expériences autours des pratiques agiles et de la gestion de projet en général.

- Communiquer sur les valeurs de l'agilité. Cette communication serait assurée par le sponsor des pratiques agiles au travers des instances et de la communauté de chef de projet.

- Encourager et faciliter la participation des personnels concernés (chef de projet, responsable produit, développeurs, designers) aux événements de la communauté agile comme l'Agile tour ou les Scrum days.

Conclusion

L'agilité est une discipline qui ne s'adresse pas seulement à la technique, mais aussi aux services métiers. Elle n'est pas simplement l'objet de l'émancipation des développeurs dans les organisations. En complément des pratiques de génie logiciel qu'elle promeut, elle permet de mieux mener des projets de développements logiciels en proposant une alternative à des pratiques d'ingénierie de projet parfois inadaptées au développement logiciel. Elle permet de créer, entre les acteurs de ces projets, la synergie qui fait souvent défaut dans nos organisations, encore très tayloriennes, dont la règle de fonctionnement est basée sur le « *command and control* ». Par ses valeurs et principes, elle impacte à certains niveaux, l'organisation sociale du travail. Comment s'améliorer sans viser un idéal, même utopique, qui combine au sein des projets, l'excellence technique et l'humain ? C'est précisément la combinaison que les agilistes cherchent à améliorer en mettant l'accent sur la création de valeur pour les métiers.

Beaucoup de critiques sont adressées aux méthodes agiles. Certains reprochent le manque de documentation, d'autres reprochent les difficultés d'organisation ou de scalabilité sur les projets d'envergure. Depuis que j'étudie les méthodes et les pratiques agiles, j'ai lu et entendu beaucoup de critiques. « L'agilité n'est pas adaptée pour les gros projets », ou bien encore « l'application sera mal documentée et ingérable», ou bien « l'agilité n'est qu'un ensemble de pratiques réservées aux développeurs ». Peut-on reprocher ces affirmations à leurs auteurs ? Les approches agiles du développement comme d'autres approches méthodologiques souffrent certainement de mauvaises interprétations et implémentations. Bien que la discipline soit mise en avant par les agilistes, la grande liberté qu'elle permet n'y est peut être pas étrangère.

Jeff Sutherland dit de Scrum qu'il est facile d'en comprendre le fonctionnement mais qu'il est difficile à mettre en œuvre. De manière générale sur les pratiques agiles, je pense que les difficultés de mise en œuvre qu'une organisation peut rencontrer ont deux origines. La première est l'adaptation des pratiques au contexte. Les organisations restent libres de choisir leur manière de faire mais il y a un vrai travail à réaliser sur le choix des pratiques et leur intégration dans le contexte. La deuxième est l'acceptation du changement. Quelle est la capacité de l'organisation à accepter de nouvelles pratiques et jusqu'où est-elle prête à aller dans le changement ?

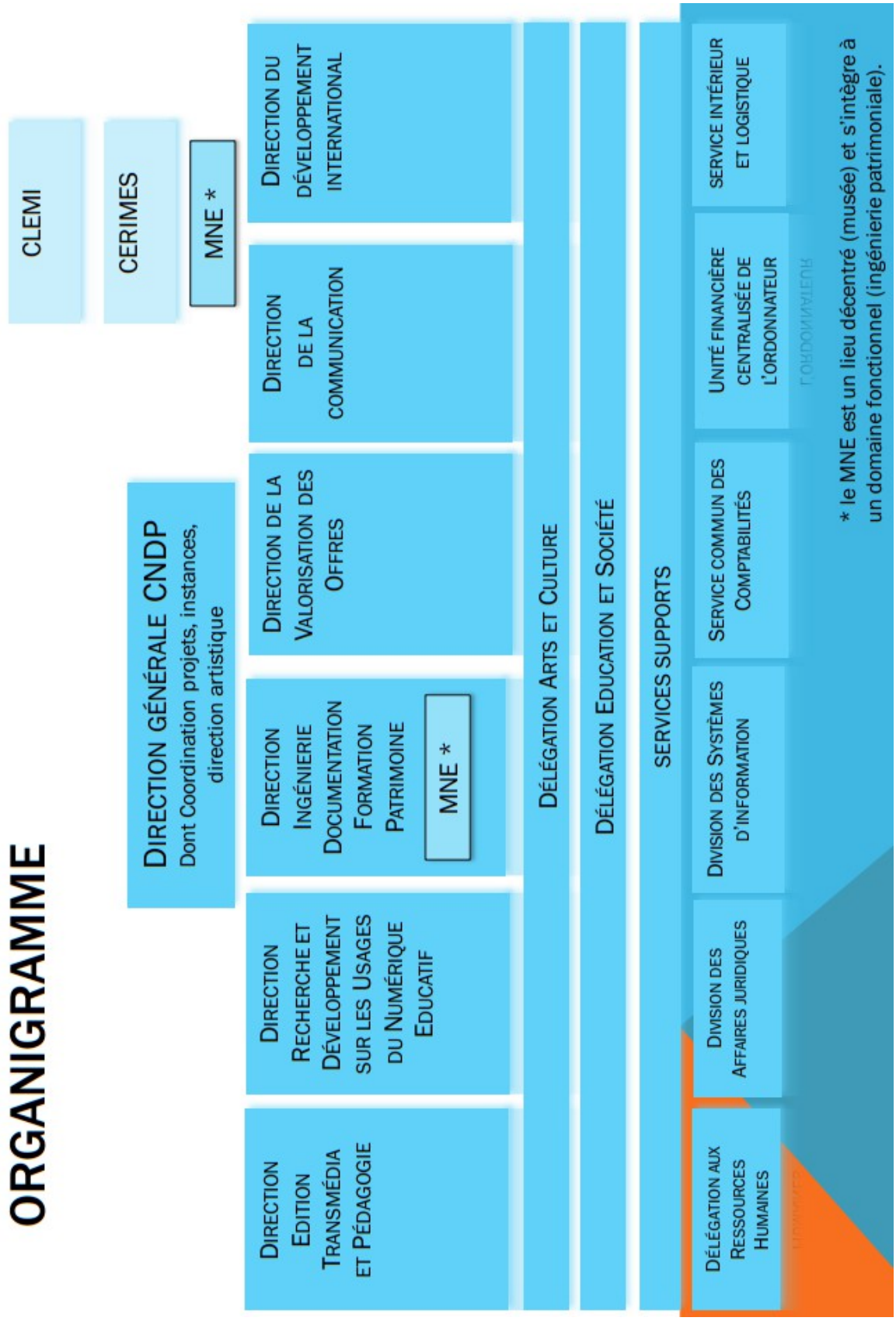
Au CNDP, serons-nous plus performants sur nos projets avec l'agilité ? Aujourd'hui, avec les données dont nous disposons il est difficile d'étayer nos convictions et de comparer, à typologie de projet semblable, l'approche agile et l'approche traditionnelle. Néanmoins dans le cadre du projet CCR, nous avons pu expérimenter et pratiquer l'agilité. Même si cette expérience n'était pas parfaite, nous avons appris. Nous avons choisi quelques pratiques et nous avons structuré et adapté notre manière de faire autour de ces pratiques. C'est certain, nous avons fait des erreurs en choisissant par exemple de ne pas exécuter certaines pratiques comme le *stand-up*. Nous avons aussi rencontré des problèmes de régression qui nous ont montré les progrès à accomplir sur l'ingénierie du test. Le projet CCR a nécessité 300 j/h de développement, ce qui représente pour le CNDP un projet de taille importante comparé aux petits projets de 20 à 30 j/h que nous avons l'habitude d'exécuter. Aujourd'hui, au-delà des chiffres, que reste t-il de cette expérience ? Le projet CCR et d'autres depuis ont montré à l'équipe et aux collègues des services métiers avec qui nous partageons les pratiques qu'il

existe une autre manière de produire du logiciel. Et que cette autre manière nous permet d'être autant, voire plus performant, d'être plus innovant et plus collaborant. Sur le projet CCR, cette autre approche du développement a permis au CNDP de répondre au besoin du ministère dans un temps très court tout en acceptant les nombreux changements et les incertitudes entre avril et septembre 2011. Cette réactivité a reposé en très grande partie sur notre processus piloté par les *stories*. Aussi, ces pratiques autour des *user stories* sont importantes et structurantes pour l'équipe et nous espérons qu'elle vont s'imposer durablement, tout comme le principe de développement itératif et incrémental préconisé par les méthodes.

Je ne crois pas à l'efficacité du « *big design up front* ». Je pense que cette approche génère trop de gaspillages et qu'elle bride l'innovation et la participation. Dans les projets d'envergure nécessitant un travail plus important sur l'architecture, je pense que des cycles de conception itératives, très courts, dans lesquels l'analyse et l'écriture de code exploratoire sont menées en parallèle sont plus efficaces car ils donnent un meilleur *feedback*.

Dans la mesure où l'établissement présente le développement de ses activités numériques comme un axe important de sa stratégie et de ses missions, il est primordial de créer les conditions qui favoriseront l'innovation et la création de valeur autour de nouveaux services et produits. Dans un contexte changeant et incertain, les valeurs, les principes et les pratiques agiles sont une réponse tout en composant avec les fondamentaux de la gestion de projet.

Annexe 1 : Organigramme de la nouvelle organisation du CNDP



Annexe 2 : Relevé d'informations sur les activités des développeurs de 2010 à 2012

820 j/h au total pour 4 développeurs
205 jours travaillés en 2010

nombre de demandes jira	14
nombre de demandes mantis	123
nombre de demandes glpi	331

Nom du projet ou de la séquence	Technologie	Type	Charges	Nombre de demandes créées	Nombre de demandes résolues	Catégorie
Aspect	php	Maintenance évolutive et corrective	5	15	15	étude
Base des diplôme	asp	Maintenance évolutive et corrective	23	4	4	édition
Callimaques	php	Maintenance évolutive et corrective	36	3	2	édition
Caren	php	Maintenance évolutive et corrective	20	6	6	édition
Dictionnaire des écoliers	java	Projet	84	16	8	tutelle
Ecole numérique rurale	php	Maintenance évolutive et corrective	4	42	42	tutelle
Education prioritaire	php, typo3	Maintenance évolutive et corrective	5	19	19	tutelle
Etwinning	php, typo3	Maintenance évolutive et corrective	15	3	3	étude
Expérience fondamentale	php, typo3	Projet	10	0	0	édition
Extension typo3	php,typo3	Maintenance évolutive et corrective	10	0	0	édition
Flux logiciel commercial	java	Maintenance évolutive et corrective	91	1	1	SI
Gestion des abonnements	java	Projet	18	4	4	SI
Implementation nouvelle nomenclatures	php	Projet	74	2	2	doc
Lettre et tice	php	Projet	15	0	0	édition
LIJ	java	Projet	27	4	4	édition
Maintenance modele commun	asp	Maintenance évolutive et corrective	28	10	10	doc
Motbis	php	Maintenance évolutive et corrective	31	23	24	doc
Outre mer	php, typo3	Projet	10	0	0	édition
Pairformance	php	Maintenance évolutive et corrective	5	7	7	tutelle
Planete chinois	php, typo3	Projet	3	1	1	édition
Poste ta carte	php	Projet	10	3	3	édition
Refonte graphique MDC	asp	Projet	70	0	0	édition
Sialle	php	Projet	10	2	2	SI
TDC	php	Maintenance évolutive et corrective	14	0	0	édition
Tenue de classe	php	Projet	14	0	0	tutelle
VEI	php	Maintenance évolutive et corrective	5	13	13	édition
Vocabnomen	php	Projet	40	0	0	tutelle
	total		677	178	170	

Année 2010

1224 j/h au total pour 6 développeurs
204 jours travaillés en 2011

nombre de demandes jira	217
nombre de demandes mantis	0
nombre de demandes glpi	622

Nom du projet ou de la séquence	Technologie	type	Charges consommées	Nombre de demandes créées	Nombre de demandes résolues	cat
Antigone	php, flash	Maintenance évolu	7	12	12	tutelle
Architecture SI documentaire	php	Projet	14	5	5	doc
Carolyn Graham	php	Projet	10	0	0	édition
Cartes des ressources	asp	Maintenance évolu	2	3	3	édition
Catalogue chèques ressources	Java	Projet	149	273	273	tutelle
Collections	php, Typo3	Projet	8	23	23	édition
Dictionnaire des écoliers	Java	Projet	20	18	18	tutelle
Document unique	Java	Projet	50	1	1	si
Etwinning	typo3	Maintenance évolu	17	17	17	étude
Expériences fondamentales	php, Typo3	Maintenance évolu	23	1	1	édition
Gestion des abonnées	java	Maintenance évolu	30	20	20	si
ItunesU	php	Projet	5	3	3	édition
Logiciel commercial nouvelle calédonie	Java	Maintenance évolu	6	2	2	si
Maintenance modele commun	asp	Maintenance évolu	25	8	8	doc
Media sceren	php	Projet	149	16	16	doc
Motbis	php	Maintenance évolu	4	9	9	doc
Musique prim	php, typo3	Projet	11	0	0	tutelle
Nomenclatures	asp	Projet	5	19	19	doc
OAI	Java	Projet	46	1	1	doc
Outre mer	php, Typo3	Maintenance évolu	5	2	2	édition
Pairformance	php, moodle	Maintenance évolu	84	0	0	tutelle
Portails disciplinaires	php, typo3	Projet	98	8	8	tutelle
Refonte educasources	php	Projet	47	24	24	doc
Refonte modele commun	Java	Projet	243	24	23	doc
Savoir cdi	php, typo3	Projet	16	4	4	doc
sceren.com	asp	Maintenance évolu	5	27	27	si
sceren.com rayonnage	asp	Maintenance évolu	5	0	0	si
Sialle	php	Maintenance évolu	24	6	6	tutelle
vocabnomen	php	Projet	51	0	0	tutelle
Voyage avec les mots	php, typo3	Maintenance évolu	13	11	11	édition
Conventions	Java	Maintenance évolu	20	15	15	si
	total		1172	537	536	

Année 2011

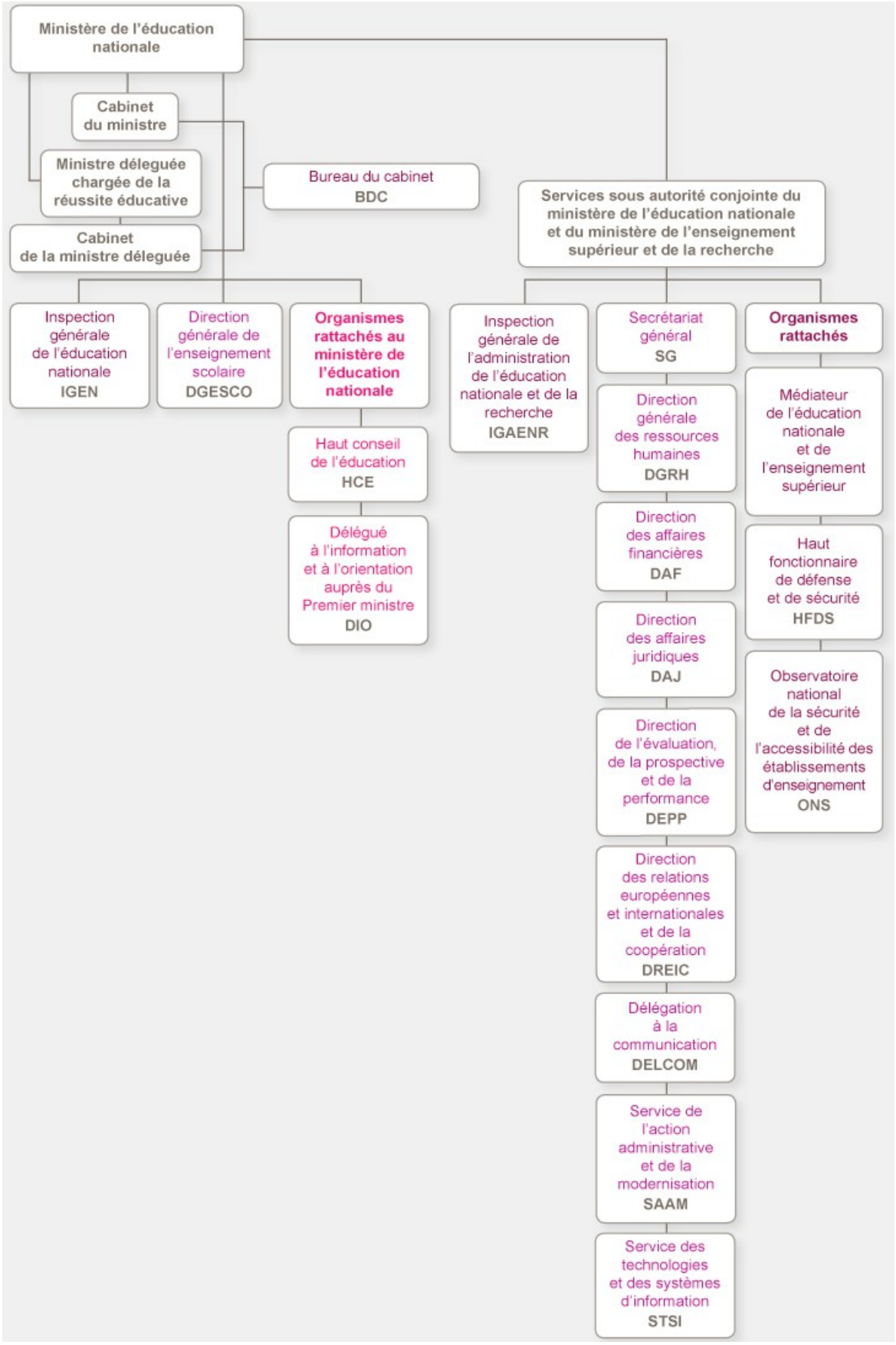
1218 j/h au total pour 6 développeurs
203 jours travaillés en 2012

nombre de demandes jira	345
nombre de demandes mantis	0
nombre de demandes glpi	1156

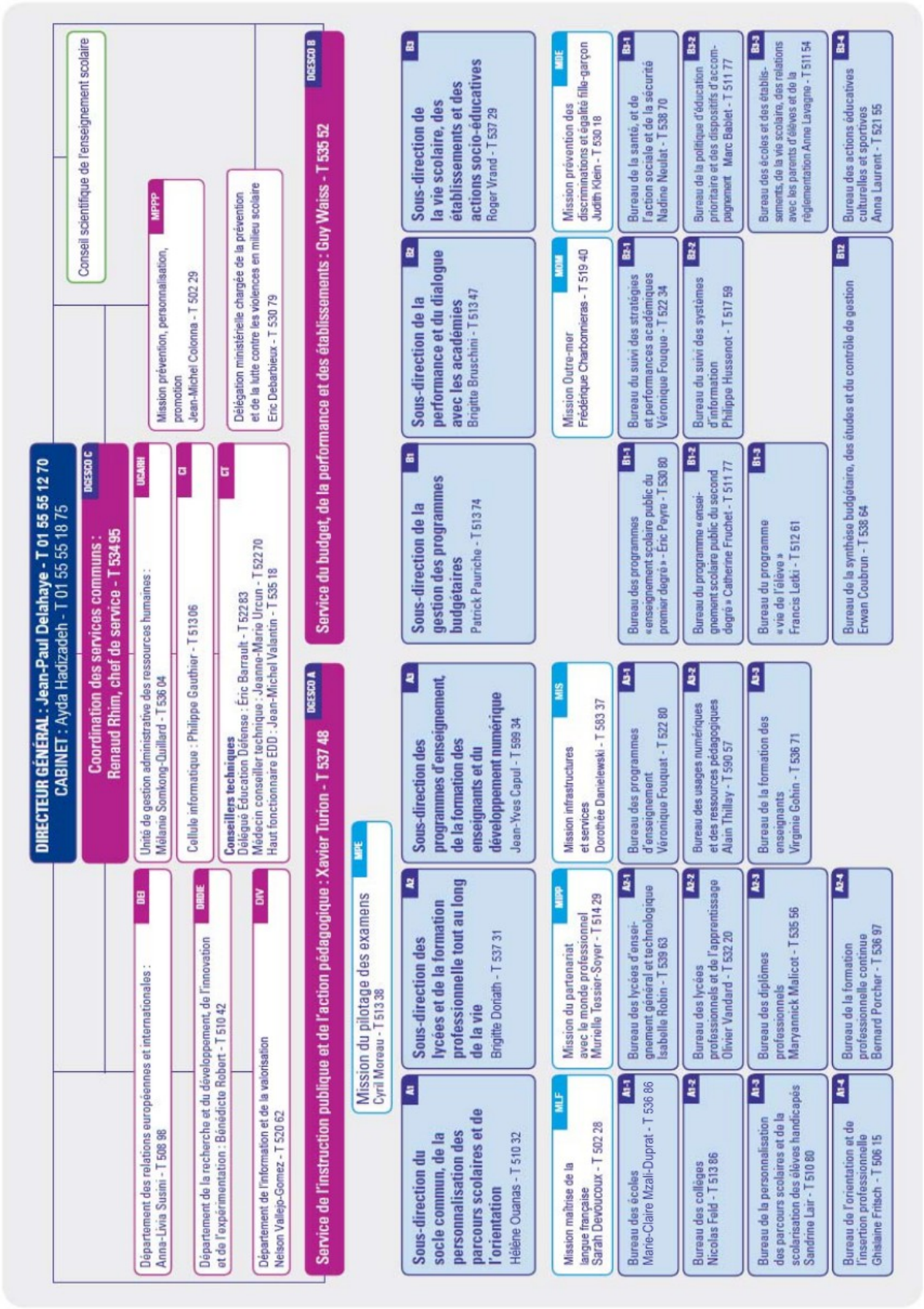
Nom du projet ou de la séquence	Technologie	Type	Charges	Nombre de demandes créées	Nombre de demandes résolues	cat
Anagène	visual basic	Maintenance évolutive et cc	15	10	10	édition
Application ios Agence des usages	objective c, ios	Projet	39	2	2	étude
Architecture SI Documentaire	java	Projet	48	80	80	doc
Carte des ressources locales	asp	Maintenance évolutive et cc	15	10	10	doc
Catalogue chèque ressources	java	Projet	75	257	256	tutelle
cndp.fr – base des diplômes	asp	Maintenance évolutive et cc	10	3	3	édition
cndp.fr – exp.fondamentales	php, typo3	Maintenance évolutive et cc	2	28	28	édition
Concours des 10 mots	php	Maintenance évolutive et cc	65	49	41	édition
Educameta	php	Maintenance évolutive et cc	3	3	3	doc
Enquete de sciences	php	Maintenance évolutive et cc	5	1	1	édition
Flux logiciel commercial	java	Maintenance évolutive et cc	15	5	5	si
Gestion des abonnés	java	Maintenance évolutive et cc	35	33	24	si
Internet Responsable	php, typo3	Projet	18	64	63	tutelle
Mag film	php, typo3	Projet	25	2	1	édition
Maintenance modele commun	asp	Maintenance évolutive et cc	10	6	6	doc
Media scéren	php	Maintenance évolutive et cc	10	13	12	doc
Musagora	php, typo3	Projet	2	9	6	édition
Musique prim	php, typo3	Maintenance évolutive et cc	28	11	7	tutelle
Outil de relecture vocabulaire	java	Projet	66	0	0	doc
Pairformance	php, moodle	Maintenance évolutive et cc	190	30	24	tutelle
Parcours d'exposition	php, typo3	Projet	15	5	3	édition
pertiweb	java	Projet	32	0	0	doc
Portails disciplinaires	typo3, java	Projet	39	29	22	tutelle
Primtice	php, typo3	Maintenance évolutive et cc	40	47	45	tutelle
Refonte MDC	java	Projet	15	2	2	doc
Savoirs CDI	php, typo3	Maintenance évolutive et cc	11	14	14	doc
Sialle	php	Maintenance évolutive et cc	7	13	13	tutelle
Thesaurus Thesagri	php	Projet	25	13	12	doc
Vocabnomen	java	Projet	22	6	5	tutelle
Voyage avec les mots	php, typo3	Maintenance évolutive et cc	18	18	16	édition
	total		900	763	714	

Année 2012

Annexe 3-a : Organigramme du ministère de l'Éducation Nationale



Annexe 3-b : Organigramme de la Direction Générale de l'Enseignement SCOlaire



Annexe 4 : Note émise par le ministère dans le cadre du projet CCR

Catalogue « chèque ressources »

Préalable

Le ministre a annoncé, dans le cadre du plan numérique, la mise en place d'un catalogue général « chèque ressources » (premier et second degré) qui permettra, dès la rentrée scolaire 2011-2012, aux établissements dont les projets académiques auront été retenus, l'acquisition de ressources numériques récentes (version de moins de deux ans), pédagogiques, interactives et accessibles en ligne, issues de l'édition publique, privée, associative.

Cette opération suppose la fin de l'opération ENR en cours dont les conditions diffèrent de celles énoncées ci-dessus (niveaux concernés, type de ressources, etc.).

Le catalogue « école numérique rurale », tel qu'il existe dans sa forme et son fonctionnement actuel ne sera donc plus accessible à la fin de l'année scolaire 2010/ 2011. Les écoles retenues dans le cadre de ce plan et n'ayant pas exercé leur droit de tirage, ne pourront plus le faire.

Il est donc nécessaire d'avertir rapidement les inspections d'académie afin que celles-ci en informent les écoles concernées (1900 écoles à ce jour).

Il est probable qu'il restera en fin d'opération un reliquat financier qu'il conviendra de réaffecter.

Nb : au 1er février 2011, 4800 écoles ont effectué leur commande sur les 6700 retenues dans le cadre du plan initial.

Mise en œuvre et fonctionnement du catalogue « chèque ressources » : principes et gouvernance

Principes

Le chèque ressources numériques permet à une école, un collège ou un lycée d'un projet académique retenu d'acquérir des ressources numériques pédagogiques, complément indispensable de l'équipement.

Le catalogue couvre tous les niveaux et toutes les disciplines ; il permet aux établissements retenus de constituer leur panier de ressources correspondant au montant autorisé et aux besoins définis par l'établissement.

Le montant autorisé dépend de l'abondement de l'Etat ou d'une collectivité territoriale à l'opération. Le back office du catalogue et sa gestion permettent donc ce paramétrage.

Le catalogue propose des entrées par niveau, discipline, par type de produits (manuels numériques, multimédias éducatifs, ouvrages de référence pour la classe) et par éditeurs. Il mentionne les produits ayant fait l'objet d'une reconnaissance d'intérêt pédagogique.

Pour le premier degré, il est précisé l'accès par cycle et par domaine des programmes.

Les produits doivent être accessible en ligne (accès ou téléchargement) et produits ou disposer d'une version de moins de deux ans.

Pour ce qui concerne les manuels numériques, suivant les niveaux d'enseignement et la provenance des financements l'offre sera différente et ce afin d'éviter les empiètements de compétence budgétaire :

- pour les écoles : version enseignant présente, version élève si financement municipalité
- pour les collèges : présence dans le catalogue des versions enseignant et élève
- pour les lycées : version enseignante présente, version élève si financement région.

Il est donc nécessaire que l'application permette un affichage différent suivant le niveau et l'origine de l'établissement ou l'école qui commande.

Les modalités d'aide à l'installation des ressources et à leur prise en main, les responsabilités en cas de défaillance du prestataire sont détaillées.

Leur licence d'exploitation doit être compatible avec une installation permettant l'utilisation optimale de la ressource en particulier version établissement.

L'éditeur s'engage à assurer un service après vente pour une durée d'au moins trois ans.

Une indexation multicritère permet de repérer rapidement un produit. Elle garantit l'égalité de traitement entre éditeurs.

Un moteur de recherche plein texte, multi-critères permet un filtrage efficace et rapide des notices.

La partie « informative » du catalogue est en accès public et gratuit.

Gouvernance

Un comité de pilotage du catalogue est mis en place sur la base de la constitution de la commission multimédia : représentants de la DGESCO, du Scéren/CNDP, des éditeurs et de l'Inspection générale.

Il est souhaitable que le cahier des charges de l'opération soit validé par ce comité de pilotage. De même, tout litige ou désaccord concernant l'opportunité d'intégrer une ressource sur le catalogue est arbitrée in fine par cette instance.

La maitrise d'œuvre est déléguée au Scéren/CNDP
La maitrise d'ouvrage est assurée par la sous-direction A3-2.

Processus d'entrée et d'inscription des ressources sur le catalogue

Chaque éditeur postulant renseigne une notice accessible en ligne pour chacune des

ressources qu'il souhaite voir figurer sur le catalogue. Cette notice s'appuie sur les normes de description des ressources pédagogiques (ScoLom). La maitrise d'ouvrage vérifie le respect du cahier des charges de l'opération avant la mise en ligne effective. Le refus fait l'objet d'une notification motivée au demandeur.

En interne au ministère, chaque notice est proposée aux responsables des pôles ressources numériques de la DGESCO A 3- 2 qui valide en fonction du respect du cahier des charges. Ces responsables sont réunis périodiquement afin d'harmoniser les décisions.

Une attention particulière est portée sur l'adéquation avec les programmes scolaires en vigueur dans chaque discipline.

Le choix des ressources et la commande

Les établissements concernés sont identifiés via leur code établissement. Ils choisissent un panier de ressources correspondant au montant alloué par l'opération. Un responsable pour chaque entité concernée « école ou établissement » certifiera, via le service, la commande, la bonne livraison et l'installation des ressources dans l'établissement. Ce « service fait » déclenche le processus de paiement aux éditeurs ressource par ressource.

Processus financier

Le CNDP, opérateur pour la mise en œuvre du catalogue, reçoit les sommes allouées à l'opération « chèque ressource ».

Il gère le flux financier entre les donneurs d'ordre (ministère, collectivités) et les éditeurs concernés.

A cette fin, un service est mis en place qui permet de tracer, bailleur de fonds par bailleur de fonds les flux financiers. Un bilan comptable est adressé régulièrement aux donneurs d'ordre. Celui-ci est accessible en ligne et précise les éditeurs, les ressources et les montants financés concernés.

Calendrier prévisionnel

Février 2011 : rédaction du cahier des charges de l'opération et désignation du chef de projet côté maitrise d'ouvrage et maîtrise d'œuvre. Mise en place du comité de pilotage.
Mars 2011 : validation par le comité de pilotage du cahier des charges du catalogue.
Mars-mai 2011 : constitution du catalogue « chèque ressource », développement de a plate-forme
Juin-juillet 2011 : publicité auprès des acteurs concernés (éditeurs, collectivités, établissements) entrée des fiches produits.
Juin 2011 : fin de l'opération ENR (écoles primaire)
Septembre 2011 : ouverture du catalogue

Annexe 5 : Exemple d'implémentation d'une user story avec ses critères d'acceptation.

La *user story* ci-dessous porte sur le module de gestion des comptes écoles et plus particulièrement sur la gestion des subventions allouées dans le cadre de l'opération CCR. Il s'agit d'un cas ou l'absence des critères d'acceptation et des tests unitaires sous-jacents a permis une implémentation erronée de la *story*. Le contrôle effectué dans le code sur le montant total alloué aux écoles du premier degré de l'académie ne fonctionnait pas. Autrement dit alors que le plafond de 20% de subventions pour le premier degré était atteint, le gestionnaire académique pouvait allouer d'autres subventions sans qu'il soit alerté du dépassement. L'implémentation de cette règle de gestion aurait dû commencer par l'écriture des tests unitaires correspondants.

Ci-dessous, la *user story* initiale et la correction du code apportée après la livraison de la fonction :

```
CCR-210

En tant que gestionnaire académique je ne peux pas attribuer aux écoles du 1er
degré de l'académie un total de subvention de plus de 20 % du montant total de la
subvention pour l'académie, afin de privilégier les établissements du second degré.
```

```
Critères d'acceptation initialement manquant :

1. Je dois obtenir un message d'erreur si le montant que je souhaite allouer pour
l'école donne un montant total de subvention pour les écoles supérieur à 20% du
total de l'académie.

2. Si le total est inférieur ou égal au 20%, la saisie est autorisée et un message
est affiché pour en avertir l'utilisateur.
```

A l'occasion de la correction du défaut, les tests suivants ont été écrits pour tester la classe contrôleur correspondante (`subventionAction`). Cette classe contient la méthode (`validerMontantTotalSubventionPrimaire`) elle aussi écrite à l'occasion de la correction et qui implémente le contrôle du montant. Les tests unitaires permettent de tester le cas ou le montant est supérieur, inférieur ou égal.

```
public class SubventionActionTest  {

    public SubventionActionTest() {
    }

    @Test
    public void testValiderMontantTotalSubventionPrimaire() {
       Boolean expResult = false;
       SubventionAction instance = new SubventionAction();
       assertEquals(expResult, instance.validerMontantTotalSubventionPrimaire(500.0,
50000.0, 9600.0));
    }

    @Test
    public void testValiderMontantTotalSubventionPrimaire2() {
       Boolean expResult2 = true;
       SubventionAction instance = new SubventionAction();
       assertEquals(expResult2,
instance.validerMontantTotalSubventionPrimaire(500.0, 50000.0, 9000.0));
    }
```

```
    @Test
    public void testValiderMontantTotalSubventionPrimaire3() {
       Boolean expResult2 = true;
       SubventionAction instance = new SubventionAction();
       assertEquals(expResult2,
instance.validerMontantTotalSubventionPrimaire(500.0, 50000.0, 9500.0));
    }
}
```

Ci-dessous, la classe contrôleur `SubventionAction` qui contient la méthode `validerMontantTotalSubventionPrimaire` sur laquelle portent les tests. Initialement le code erroné de cette méthode était dans le corps de la méthode `lancerTransaction` de la classe contrôleur.

```
public class SubventionAction  {

      public Boolean validerMontantTotalSubventionPrimaire(Double montant, Double
subventionTotalAcademie, Double subventionPrimaireTotalAcademie) {
      Double total = montant + subventionPrimaireTotalAcademie;
      if (total > 0.20 * subventionTotalAcademie) {
          return false;
      }
      return true;
    }

      public String lancerTransaction() {
      .
      .
      .
listeAcademieSubventionReste.put(etablissementTraite.getAcademie().getAcademieId(),
      (subventionRestant - montantTransaction));

      // 2.4 contrôle école est du premier degre
      if   (etablissementTraite.getTypeEtablissement().getTypeEtablissementId()   ==
TypeEtablissementImpl.getTYPE_ECOLE()) {

      Double subventionAlloueeAuxPrimaire = s.getSubventionService().
      getSubventionsAlloueesAuxEtablissementsPrimaire((AcademieImpl)
      etablissementTraite.getAcademie());

      // 2.4.1 Vérifie que le montant alloué ne dépasse pas 20% enveloppe allouée
aux   écoles du premier degré de l'académie
      if      (!this.validerMontantTotalSubventionPrimaire(montantTransaction,
      etablissementTraite.getAcademie().getSubvention(),
subventionAlloueeAuxPrimaire)) {
          addActionError("La transaction est impossible car la subvention ("
          + Fonctions.getTarifEnString(montantTransaction)
          + ") que vous souhaitez allouer \u00e0 l'\u00E9tablissement primaire '"
          + etablissementTraite.getNom()
            + "' d\u00E9passera les 20% des subventions distribu\u00E9es \u00e0
      l'Acad\u00E9mie '"
          + etablissementTraite.getDepartement().getAcademie().getLibelle()
          + "' ("
                                                                  +
      Fonctions.getTarifEnString(etablissementTraite.getAcademie().getSubvention())
          + ") !");
          rechercherEtablissementValides();
          return ERROR;
      }}
      .
      .
      .
      }
}
```

Annexe 6 : Prototype et cas d'utilisation de l'affectation de subventions aux écoles et aux lycées.

1 : Choix de l'académie, uniquement pour les admins MEN. Les inspecteurs d'académie sont eux associés à une académie et n'auront pas ce choix.

Règles et contraintes

• Cette interface comportera des différences en fonction du profil connecté.

• L'utilisateur doit choisir une académie et s'il le désire un ou plusieurs critères (Morceau du nom de l'établissement, début du code postal, début du nom de ville, type d'établissement).

Cas d'utilisation

Cas d'utilisation : Attribution des chèques/subventions

Résumé : ce cas d'utilisation permet à un utilisateur MEN ou collectivités d'attribuer un chèque ressource à un compte école.

Acteurs : inspecteur (principal), gestionnaire cndp (secondaire)

Date de création : 17/03/2011

Date de mise à jour : 17/03/2011

Version : 1.0

Rédacteur : Jérôme MARTIN
Responsable :
Contributeur : CaXXXXXX BXXXXXX, ZZZZZZ ZZZZZZ
Informés : YYYYYY YYYYYYY,

Préconditions

•L'inspecteur est identifié dans l'application

•Les écoles sont dans le système et elles ont fait l'objet d'une sélection pour recevoir une subvention.

Scénario nominal pour l'attribution de la subvention

1.L’inspecteur arrive sur le formulaire d'attribution de subvention.

2.L’inspecteur manipule des critères qui permettent de retrouver une ou plusieurs écoles.

3.L'inspecteur procède à une sélection (une case à cocher par ligne) des écoles et/ou établissement pour lesquels il souhaite attribuer un montant.

4.L’inspecteur saisit le montant.

5.L'inspecteur enregistre cette attribution.

6.Le cas d’utilisation est terminé.

Enchainements d'exceptions

E1: Le montant en cours d'attribution est supérieur au montant restant.

L'enchainement E1 démarre après le point 5 du scénario nominal.

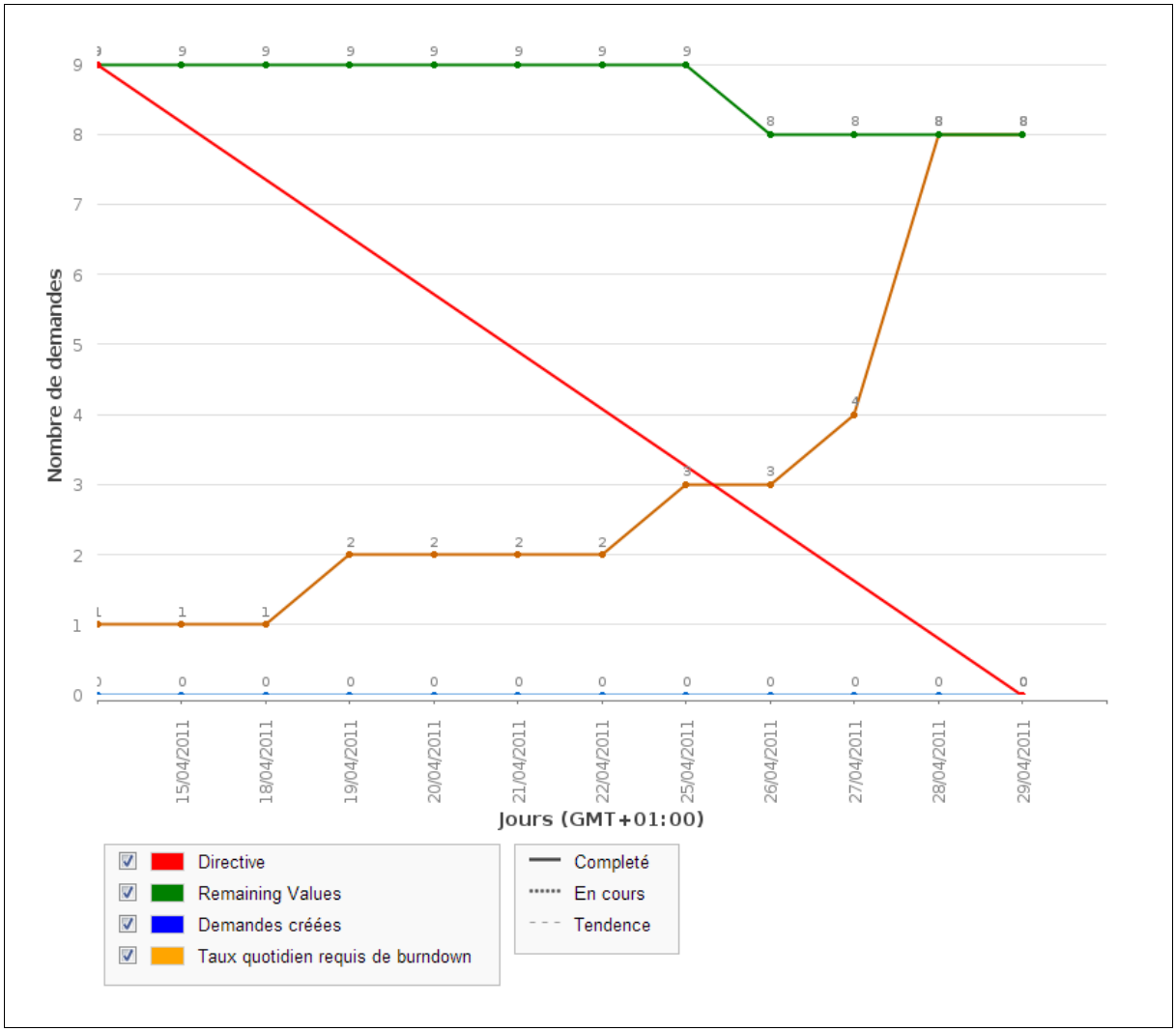
6. L'application informe l'inspecteur que l'attribution de subvention n'est pas possible et que le montant saisi ne sera pas enregistré.

le scénario nominal reprend au point 4.

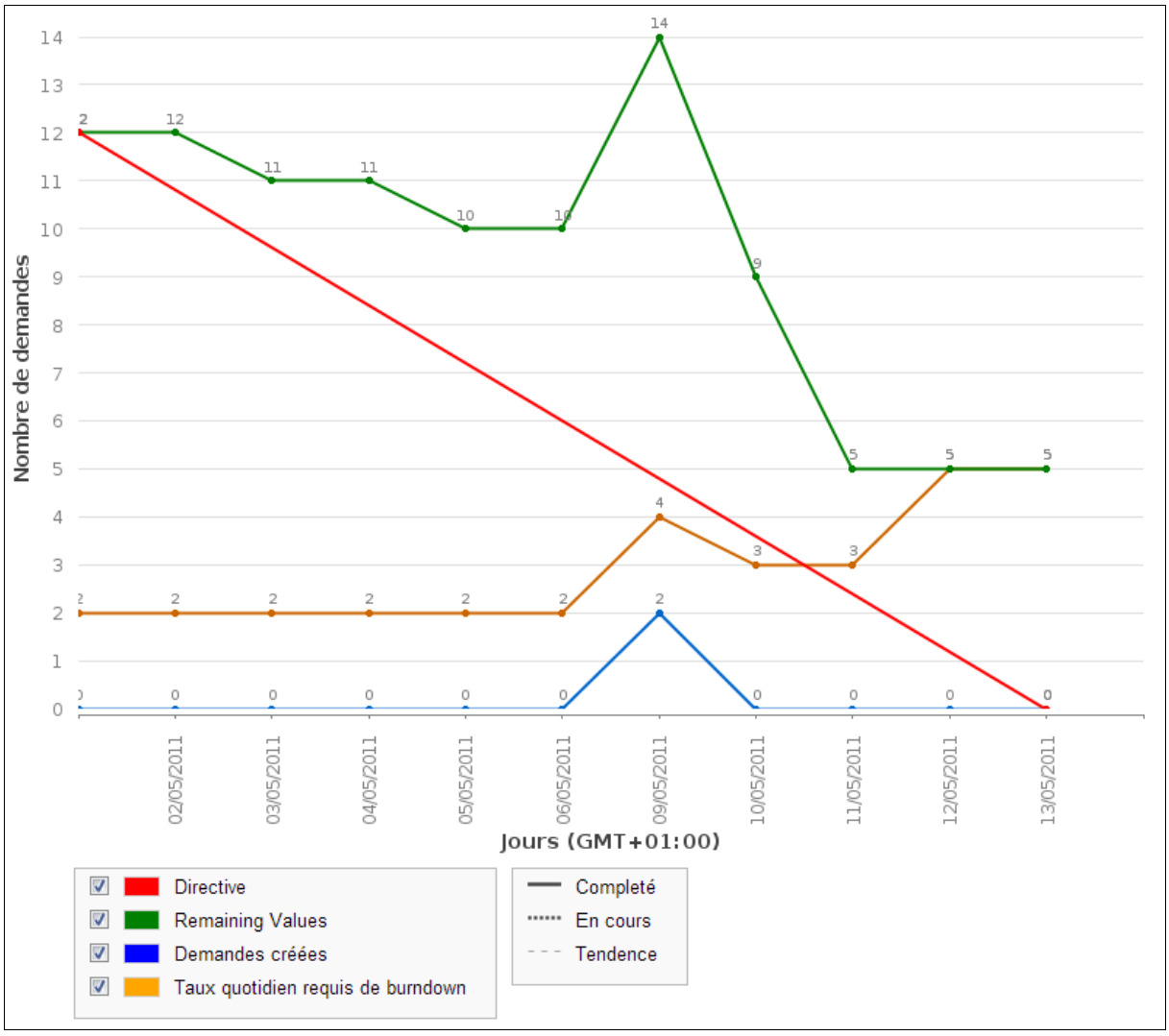
Post-conditions

•Les écoles ont sur leur compte un ou deux montants alloués par le ministère et/ou par une collectivité locale.

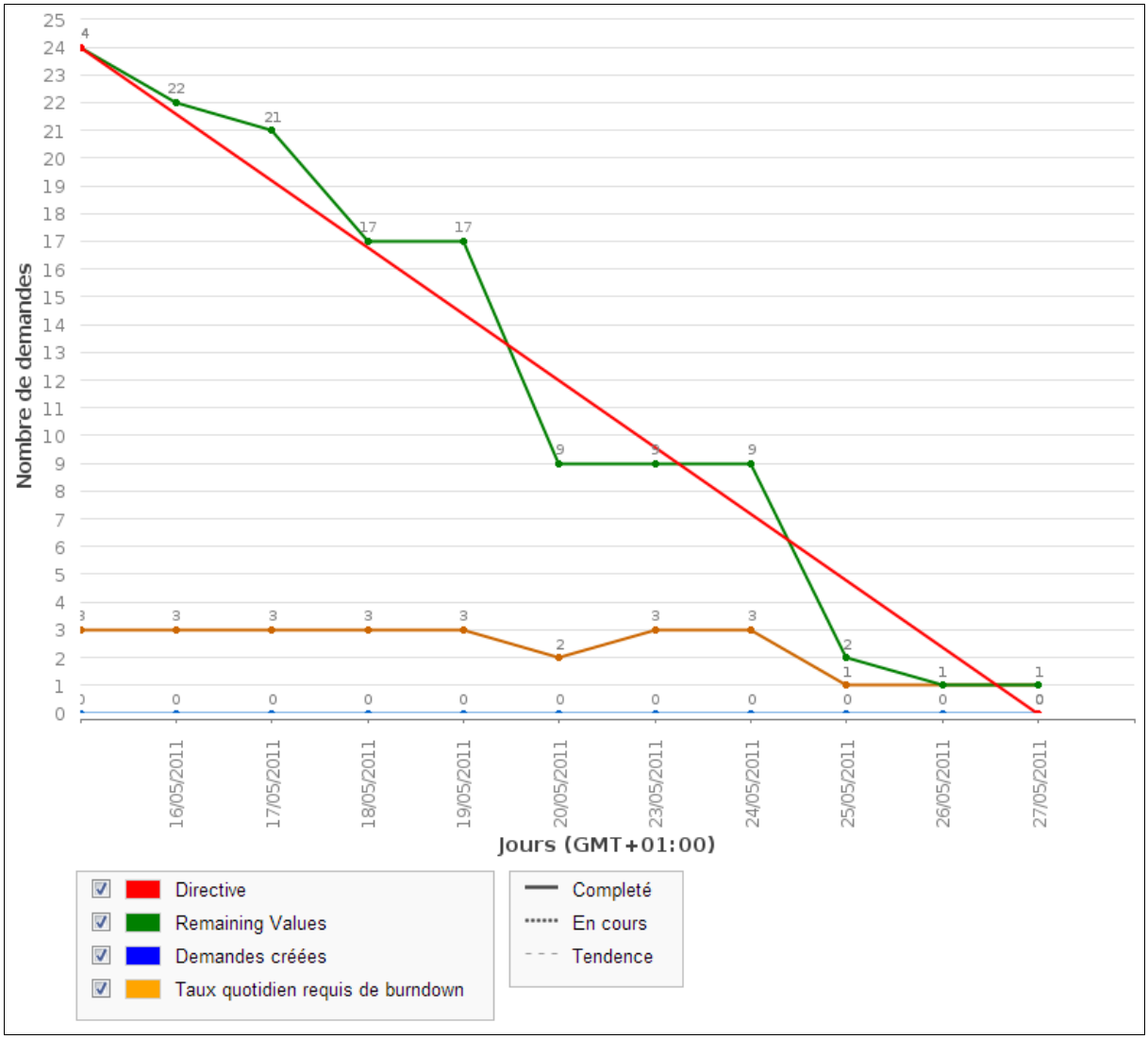
annexe 7-a : Graphique de *burndown* du *sprint* 1 du projet CCR.



annexe 7-b : Graphique de *burndown* du *sprint* 2 du projet CCR.

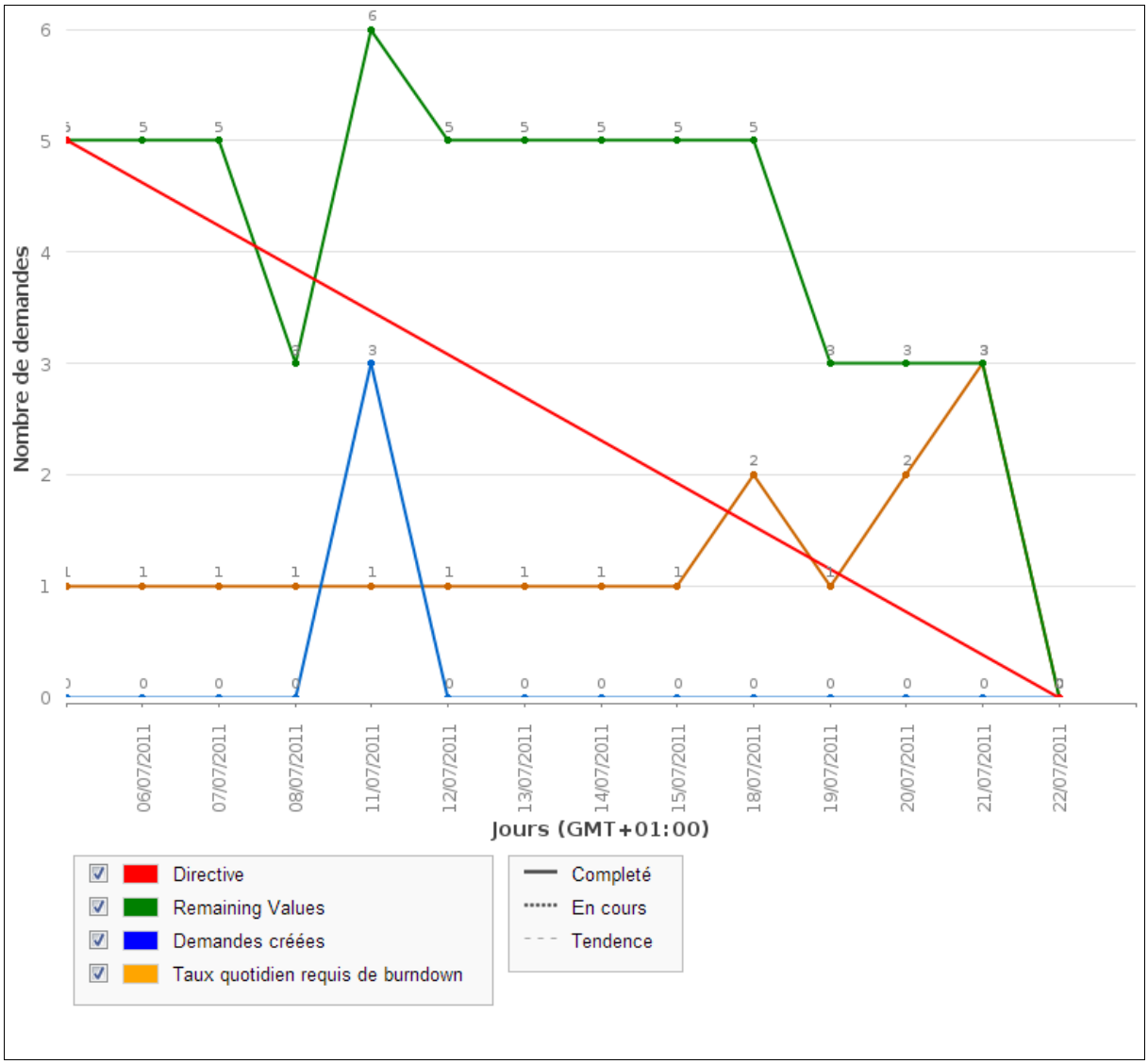


annexe 7-c : Graphique de *burndown* du *sprint* 3 du projet CCR.

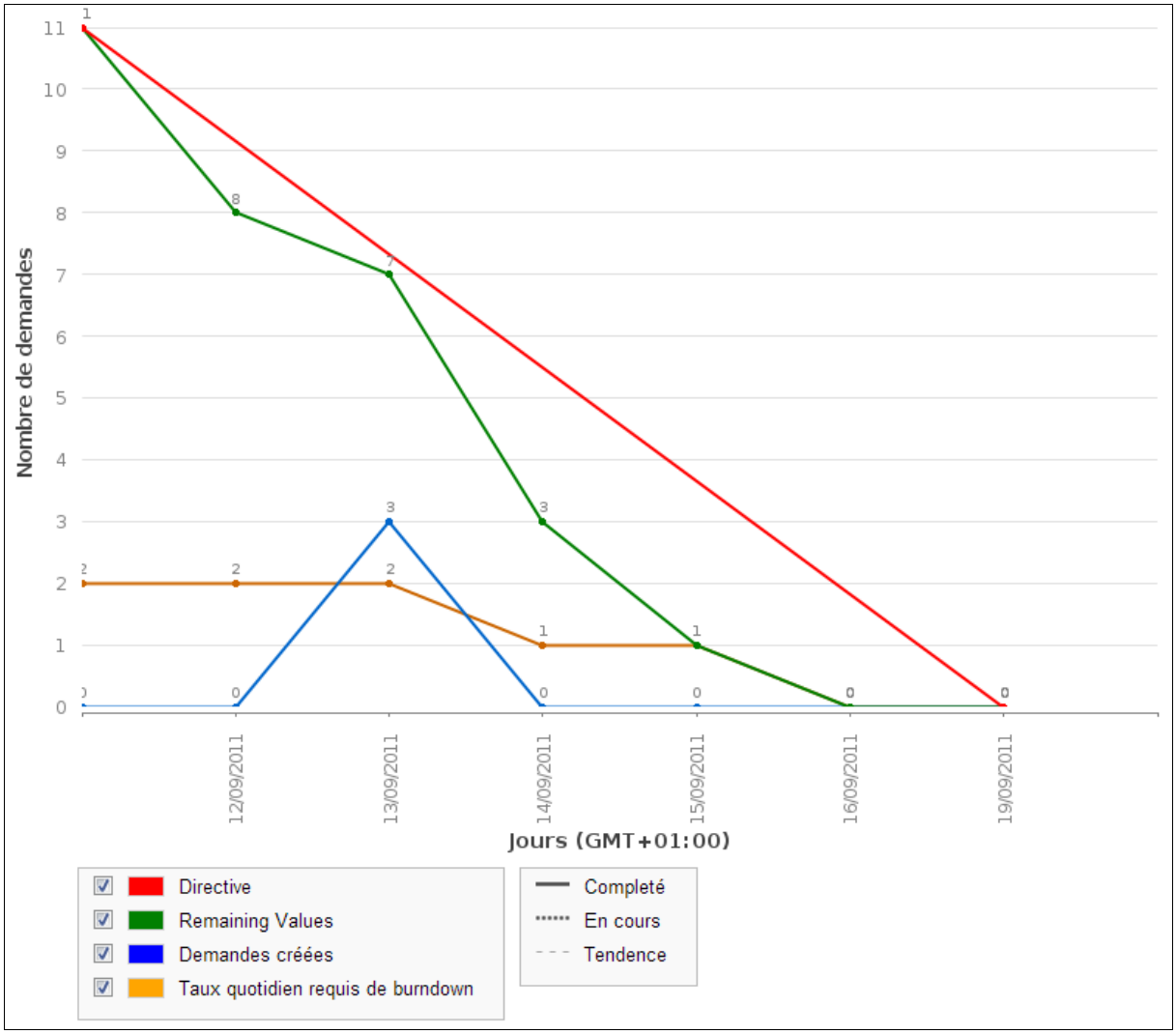


annexe 7-d : Graphique de *burndown* du *sprint* 4 du projet CCR.

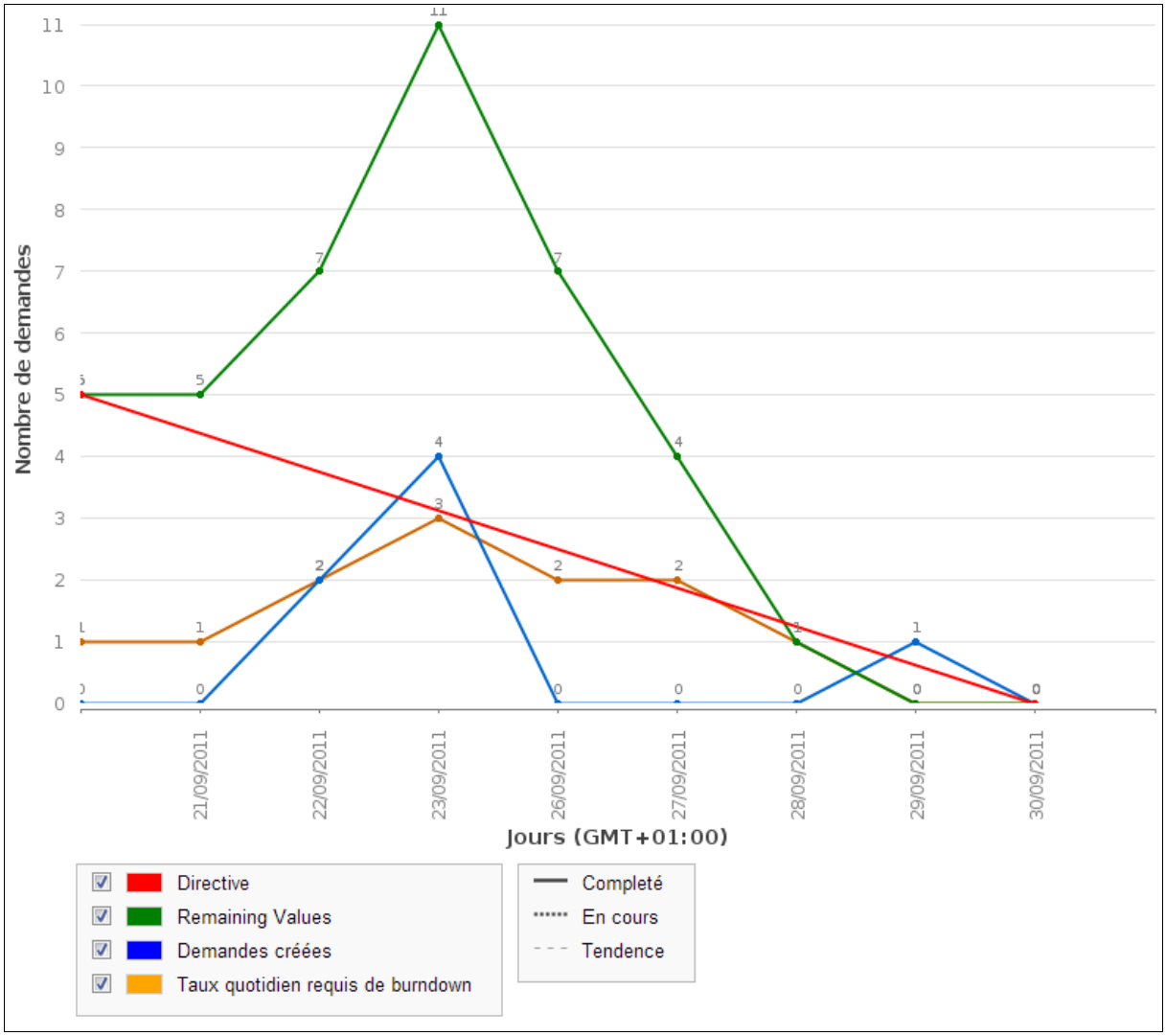
annexe 7-e : Graphique de *burndown* du *sprint* 5 du projet CCR.



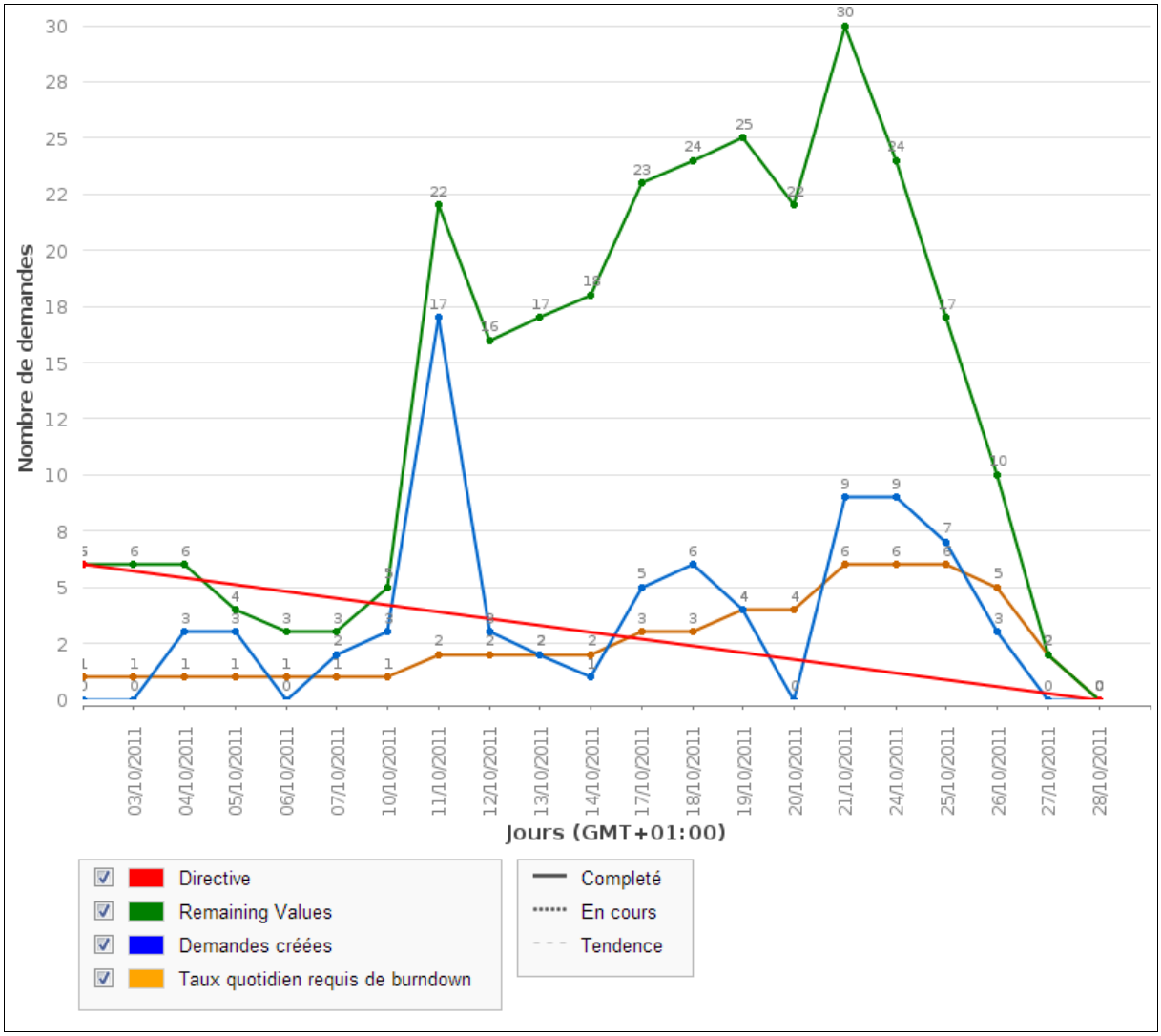
annexe 7-f : Graphique de *burndown* du *sprint* 6 du projet CCR.



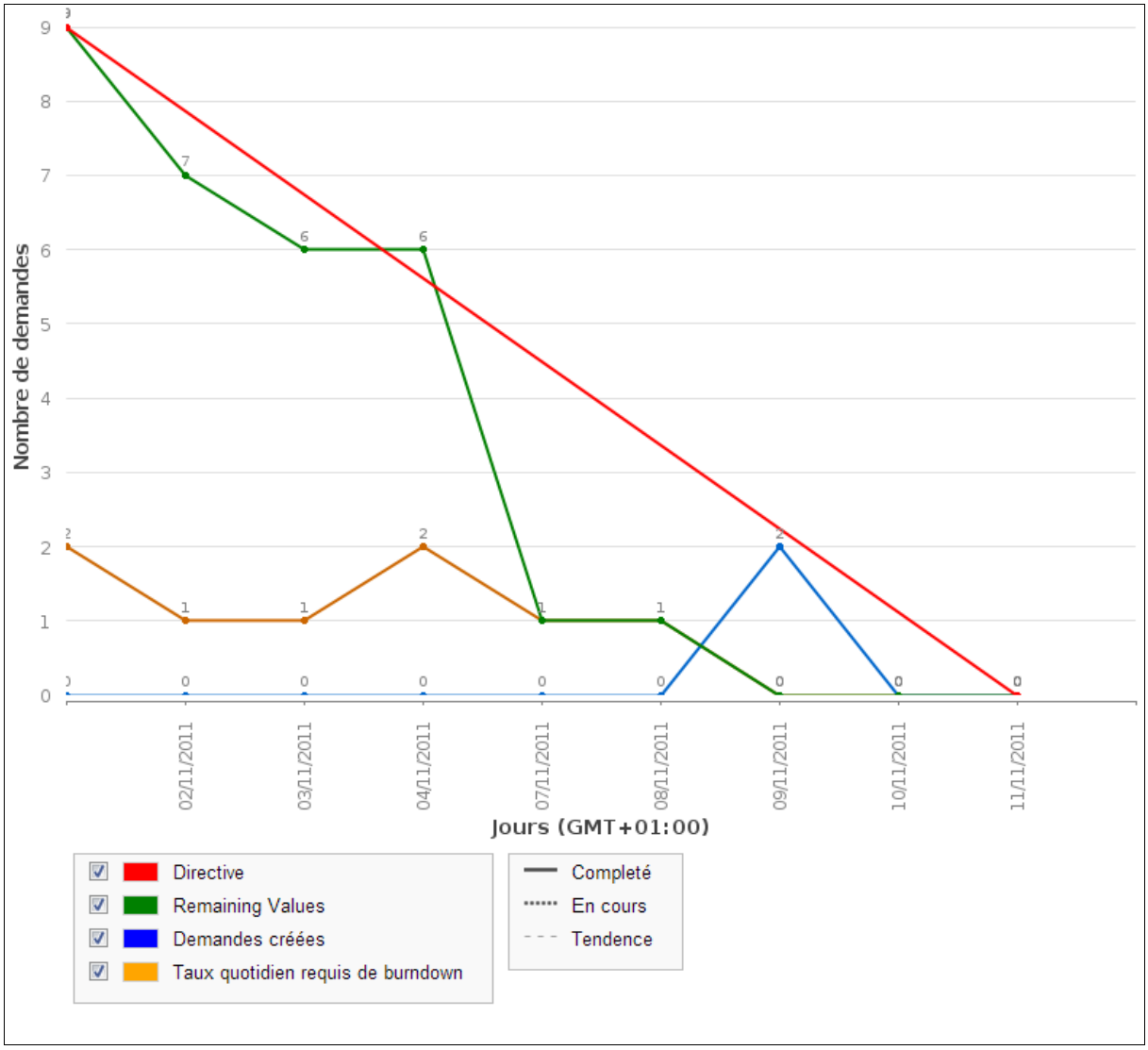
annexe 7-g : Graphique de *burndown* du *sprint* 7 du projet CCR.



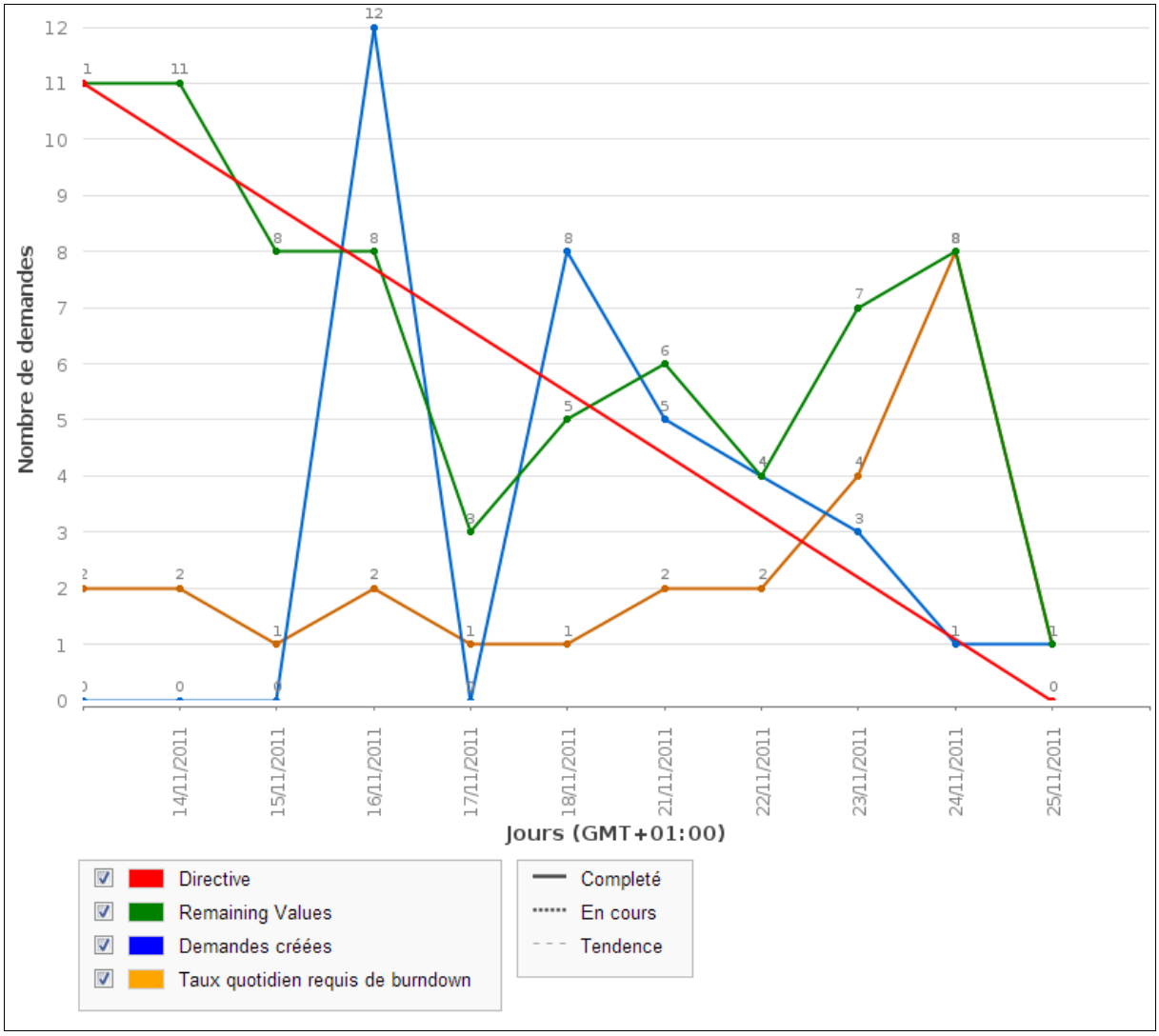
annexe 7-h : Graphique de *burndown* du *sprint* 8 du projet CCR.



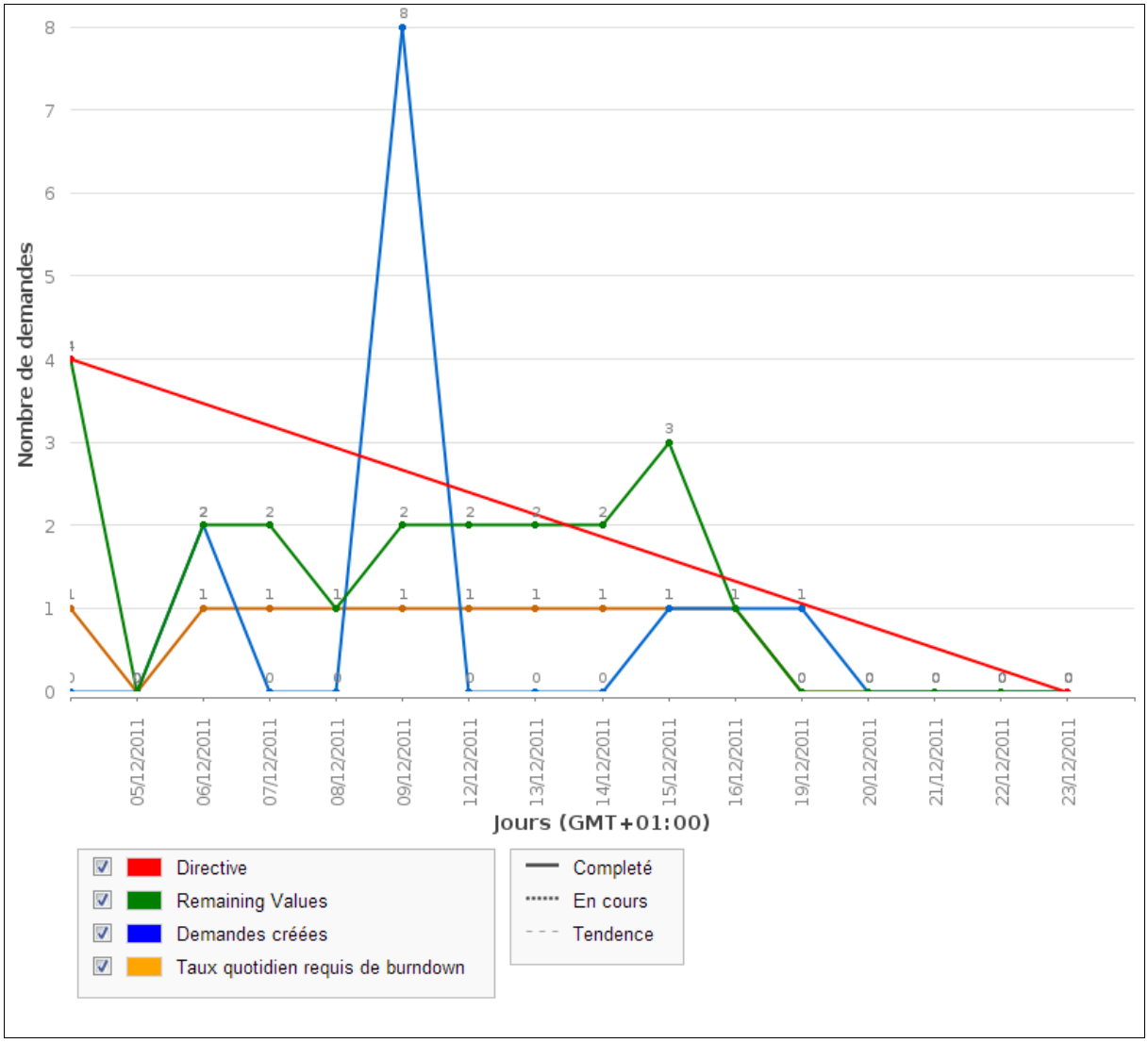
annexe 7-i : Graphique de *burndown* du *sprint* 9 du projet CCR.



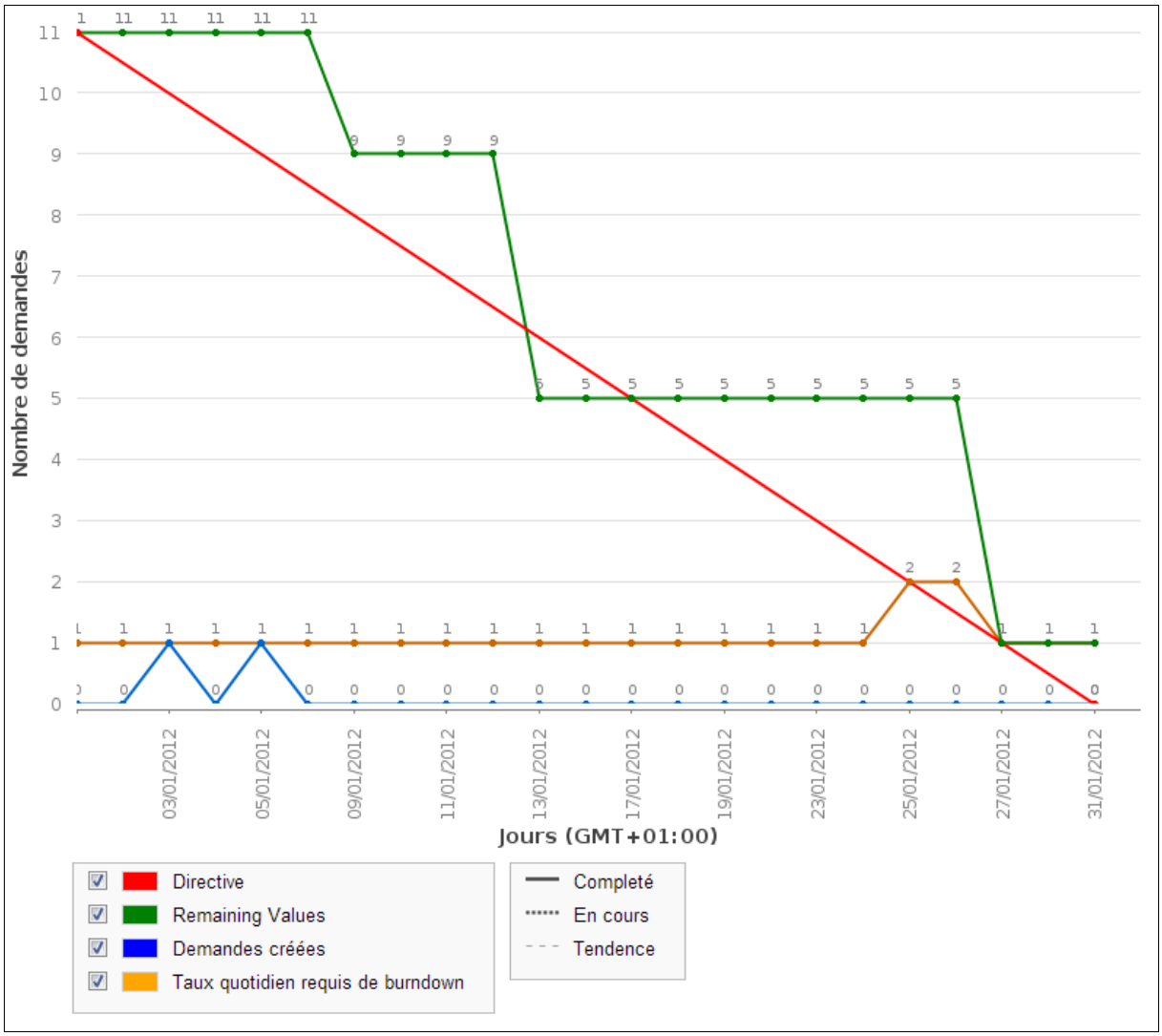
annexe 7-j : Graphique de *burndown* du *sprint* 10 du projet CCR.



annexe 7-k : Graphique de *burndown* du *sprint* 11 du projet CCR.



annexe 7-l : Graphique de *burndown* du *sprint* 12 du projet CCR.



Bibliographie

[ARC01] Archambault, J.-P. (2001). *L'édition scolaire au temps du numérique*. Disponible sur http://medialog.ac-creteil.fr/ARCHIVE41/ednumeriq41.pdf [consulté le 08 octobre 2013].

[BAR10] Barrand, J. (2010). *L'entreprise agile* (p. 201). Dunod.

[BAR12] Barrand, J. (2012). *Le manager agile* (2eme édition) (p. 182). Dunod.

[PEZ10] Pezziardi, P. (2010). *Lean management* (p. 160). Eyrolles.

[PRI10] Printz, J. (2010) *Les méthodes agiles* (p. 25) Dossier techniques de l'ingénieur.

[PMB08] Institute, P. M. (Ed.). (2008). *Project Management Body of Knowledge*. Project Management Institute.

[ROY70] W.Royce. (1970) *Managing the development of large software systems.* IEEE Wescon Disponible sur http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf [consulté le 08 octobre 2013]

[BOE88] Boehm B. (1988) *A Spiral Model of Software Development and Enhancement.* IEEE Computer Disponible sur http://csse.usc.edu/csse/TECHRPTS/1988/usccse88-500/usccse88-500s.pdf [consulté le 08 octobre 2013]

[BEC04] Beck, K., & Cynthia, A. (2004). *Extreme Programming Explained, Embrace Change* (p. 189). Addison Wesley.

[LAD08] Ladas, C. (2008). *Scrumban And Other Essays on Kanban Systems for lean Software Development* (p. 180). Modus Cooperandi Press.

[DER84] McDermid J, Ripken K. (1984) *Life cycle support in the ADA environment.* (p. 259) Cambridge University Press

[BOS04] Bénard, J.-L., Bossavit, L., Medina, R., & Williams, D. (2004). *Gestion de projet extreme programming* (p. 298). Eyrolles.

[ROT10] Rota, V. M. (2010). *Gestion de projets agile avec scrum, Lean, eXtreme programming...* (p. 274). Eyrolles.

[KNI10] Kniberg, H., & Skarin, M. (2010). *Kanban et Scrum - Tirer le meilleur des deux*. (C4Media, Ed.) (p. 128). InfoQ. Disponible sur http://www.infoq.com/resource/news/2010/01/kanban-scrum-minibook/en/resources/KanbanAndScrum-French.pdf [consulté le 08 octobre 2013]

[SUT12] Sutherland, J. (2012). *The Scrum Papers: Nut, Bolts, and Origins of an Agile Framework* (p. 216). Cambridge: Scruminc. Disponible sur http://jeffsutherland.com/ScrumPapers.pdf [consulté le 08 octobre 2013]

[AGM01] (2001). *Agile manifesto.* Retrieved May 20, 2013, Disponible sur http://agilemanifesto.org/ [consulté le 08 octobre 2013]

[DOI05] (2005). *Declaration of Interdependence.* Retrieved May 20, 2013, Disponible sur http://pmdoi.org/ [consulté le 08 octobre 2013]

[MSC09] (2009). *Manifesto for software craftsmanship* Retrieved May 20, 2013 Disponible sur http://manifesto.softwarecraftsmanship.org/ [consulté le 08 octobre 2013]

[SUT11] Schwaber, K., & Sutherland, J. (2011). *The Scrum Guide*. Disponible sur http://scrum.jeffsutherland.com/ [consulté le 08 octobre 2013]

[POP03] Poppendieck, M., & Poppendieck, T. (2003). *Lean Software Development - An Agile Toolkit* (p. 184). Addison Wesley -.

[KNI11] Kniberg, H. (2011). *Lean depuis les tranchées*. (K. Keepler, Ed.) (p. 76).

[COH04] Cohn, M. (2004). *User Stories Applied: For Agile Software Development* (p. 304). Addison Wesley.

[BOD11] Guillaume Bodet. (2011). *Scrum en action* (p. 192). Pearson education france.

[BON12] Bono, E. De. (2012). *Les 6 chapeaux de la réflexion* (p. 206). Eyrolles.

Liste des figures

Résumé

Ce mémoire traite des possibilités d'amélioration de l'exécution des projets de développement logiciel menés par le Centre National de Documentation Pédagogique. Après une présentation de l'établissement, de l'équipe de développeurs et de sa typologie de projet, il rappelle les origines de la gestion de projet et son guide de référence. Il aborde ensuite les valeurs, les principes et les pratiques des différents courants et frameworks agiles que sont Scrum, l'eXtreme Programming, le lean software développement et kanban.

Il présente une expérience de l'agilité vécue en 2011, par l'équipe de développeurs de cet établissement. Dans le cadre d'un projet de développement logiciel exécuté pour le compte du ministère de l'Éducation Nationale, l'équipe a organisé son cycle de développement en introduisant des pratiques agiles empruntées à Scrum et à l'eXtreme Programming. Un bilan présente les difficultés rencontrées et les enseignements retenus, pour une amélioration future des pratiques exécutées sur tous les projets de développement logiciel du CNDP.

Enfin, ce mémoire présente les pratiques installées depuis cette expérience. Il s'agit à la fois de pratiques de génie logiciel, de pratiques sociales et de pratiques axées sur l'expression du besoin. Des pistes pour poursuivre le déploiement de ces pratiques agiles et les promouvoir dans l'établissement sont également abordées.

mots clés : agilité, agile, scrum, xp, kanban, lean, gestion de projet, pmbok

Abstract

This memory presents the possibilities to enhance the execution of software development projects which are leaded by National Center for Education Documentation. After a presentation of organization, developper's team and their project's typology, it puts emphasis on the origins about project management and his principal guide. He focuses on the values, principles and practices of different agile way of thinking and frameworks which are scrum, eXtreme programming, lean software development and kanban.

He presents an experience about agility which started in 2011 by the developper's team of this organization. In the case of a software development project executed for the ministry of National Education, the team has implemented his development cycle by introducing agiles practices from Scrum and eXtreme programming. A retrospective presents difficulties and lessons learned from this experience for a next improvement of practices executed within all the software developpement project in NCED.

Finally, this memory presents practices installed since this experience. They are engineering software practices, socials practices and others practices focused on elucidation and writing requirements. Many ways to continue deployement of these practices and promote them in the organization are suggested.

Keywords : agility, agile, scrum, xp, kanban, lean, project management, pmbok