Organisation collaborative du travail

Nathanal Jourdane et Victor Adam

7 octobre 2014 - 10 mai 2014

Table des matières

Introduction

...

On peut observer chez certaines espèces animales, notamment les fourmis, un grand nombre de participants à un projet. Par conséquent, ces sociétés animales disposent d'une excellente organisation.

Partie 1 : Notions d'organisation

...Introduction...

1.1 Les gains de productivité

...

1.2 Efficacité de la collaboration

Pour réaliser des produits et des services, une entreprise a besoin d'une collaboration optimale entre le client et le prestataire, entre les équipes de travail, ainsi qu'entre les membres au sein de chaque équipe.

La qualité de cette collaboration dépend de plusieurs paramètres tels que la personnalité des individus, les conditions de travail, l'entente au sein de l'équipe, etc.

Il est important de dissocier l'efficacité de l'équipe et l'efficacité de ses membres. En outre, il faut savoir que l'efficacité de l'équipe n'est pas nécessairement égale à la somme de l'efficacité de ses membres. Dans *Le mythe du mois homme*, Frédéric Brooks nous énonce .

Ainsi, en terme de performance, la réussite de la collaboration de l'équipe peut être définie par le ratio :

$$\frac{\textit{efficacité de l'équipe}}{\textit{efficacité de ses membres}}$$

On peut dès lors observer trois niveaux de collaboration :

- **Efficacité de l'équipe < efficacité de ses membres** (faible collaboration) : Le groupe est moins performant qu'il le pourrait. Il n'est peut-être pas assez soudé ou certains éléments troublent le travail de l'équipe, ou alors l'organisation et la gestion du travail de l'équipe ne sont pas assez efficace.
- **Efficacité de l'équipe = efficacité de ses membres** (bonne collaboration) : Les membres du groupe s'entendent suffisamment bien pour travailler ensemble dans de bonnes conditions. Le groupe peut faire face à certains problèmes qui pourrait perturber le travail, il y a plusieurs interactions positives et la gestion du travail est convenable.
- **Efficacité de l'équipe > efficacité de ses membres** (excellente collaboration) : Schéma idéal et particulièrement difficile à atteindre, il est toutefois possible d'y parvenir grâce à une équipe particulièrement soudée et organisée. Il est également nécessaire de favoriser l'échange de connaissances entre les différents membres de l'équipe et d'utiliser des outils de travail collaboratif adaptés.

1.2.1 Loi de Brooks

Le livre de Brooks a donné naissance à *la loi de Brooks*, qui est une prédiction sur la productivité des projets informatiques :

> *Ajouter des personnes à un projet en retard accroît son retard.*

Pour l'énoncer, Brook se base sur deux postulats :
— La plupart des tâches d'un projet ne sont pas partitionnables ;
— les nouveaux arrivants vont faire perdre du temps aux équipes en place.
Ce temps perdu étant proportionnel à

$$n(n-1)$$

(où n est le nombre de personnes impliquées) : la taille d'une équipe influe donc dans la productivité. Par exemple, en étant 10 dans une cuisine, il y a peu de chance de pouvoir faire un aussi bon travail qu'à trois personnes. En entreprise, un temps de formation et de compréhension est nécessaire avant que les employés s'impliquent dans un projet.

Cas d'application

Dans le cas de GFI, les équipes de recherche sont très petites par rapport aux autres équipes (1 ou 2 personnes), alors que le client attend des résultats dans les plus brefs délais. Les chefs de projets considèrent qu'augmenter la taille de l'équipe ne ferait pas gagner de temps, car celles-ci travaillent sur des projets dont la formalisation des besoins varient régulièrement.

1.3 La collaboration en dehors de l'entreprise

On peut noter l'existence de projets réalisés par de nombreuses personnes, sans aucun rapport avec une quelquonque entreprise. Par exemple le développement du noyaux Linux (un système d'exploitation open-source), est réalisé par des milliers de développeurs de par le monde, de manière indépendante et sans qu'aucune entité, entreprise ou association, ne gouverne ce travail. Il est ici évident que pour que ce projet avance, ces personnes doivent collaborer ensemble et suivre un système d'organisation adapté et rigoureux.

Le développement d'Internet a grandement facilité le partage d'information entre personnes à travers le monde. Ainsi, certains projets collaboratifs ont pu voir le jour, faisant intervenir différents acteurs géographiquement distants, sans qu'il n'y ait d'entité juridique ni d'établissement physique définis. Dans le cadre de ces projets, qui peuvent faire intervenir des milliers de personnes, une bonne collaboration est indispensable.

1.4 Collaboration selon la taille du projet

...

1.5 Application

1.5.1 Le cas de GFI Informatique

GFI Informatique est une Entreprise de services du numérique (ESN) [1] implantée sur de nombreux sites en France et en Europe, elle compte plus de 10000 salariés. Les secteur d'activités développée sont la banque, industrie, secteur public, télécom, énergie et enfn transport, ce dernier étant très actif à Toulouse avec Airbus et son écosytème. Ainsi sur le site de Toulouse, comprenant près de 500 salariés, le client principal est Airbus, suivi par EDF, la mairie de Toulouse et ATR.

En règle générale, les petits clients n'imposent ni les outils ni les méthodes : ils se content d'expliciter les objectifs attendus, qu'importe les moyens mis en oeuvre pour y parvenir.

En revanche, concernant Airbus, le rapport de force entre le client et le fournisseur est inversé. Les outils et méthodes à appliquer sont imposées, ce qui est souvent le cas pour des clients importants. En effet, la taille importante du projet oblige le client à traiter avec plusieurs fournisseurs, d'autre part Airbus comporte également des équipes de déveoppement en interne. Tous ces acteurs du projets utilisent ainsi les mêmes méthodes et les mêmes outils, ce qui facilite grandement la collaboration.

1.5.2 Le cas de Ineo

1.6 Conclusion

1. ESN : Société de services numériques spécialisée en génie informatique.

Partie 2 : Les méthodes de travail

Lorsque un projet devient assez conséquent, sa gestion devient de plus en plus complexe. Un produit informatique comportera un grande quantité de code, dont il faut faciliter la maintenance le plus tôt possible. Il fera intervenir un grand nombre de personnes, réparties dans plusieurs équipes de travail. En outre, le prestataire devra collaborer avec un client qui n'a pas les mêmes notions techniques et dont les besoins peuvent évoluer.

Une bonne organisation du travail est donc essentielle pour la réussite d'un tel projet.

L'organisation d'un projet passe tout d'abord par une méthode de travail à appliquer. Le choix d'une méthode de travail est généralement la première décision à prendre avec le client avant de commencer le développement d'un produit. En outre, elle aura une grande incidence sur sa réussite.

2.1 Méthodes appliquées à l'informatique

Il existe de nombreux modèles de gestion de projet dans le milieu industriel. Nous allons ici nous pencher sur deux méthodes de travail : le cycle en V et les méthodes Agiles.

2.1.1 Le cycle en V

En gestion de projet, il est important de considérer les erreurs comme faisant partie intégrante d'un projet (*Errare Humanum est*). Dans l'industrie en général, plus un problème est détecté tôt, plus il sera facile de le corriger et moins grande en seront les conséquences (effets de bord). Ainsi, pendant le développement d'un produit, l'objectif n'est pas tant de limiter les erreurs, mais davantage de parvenir à les détecter le plus tôt possible.

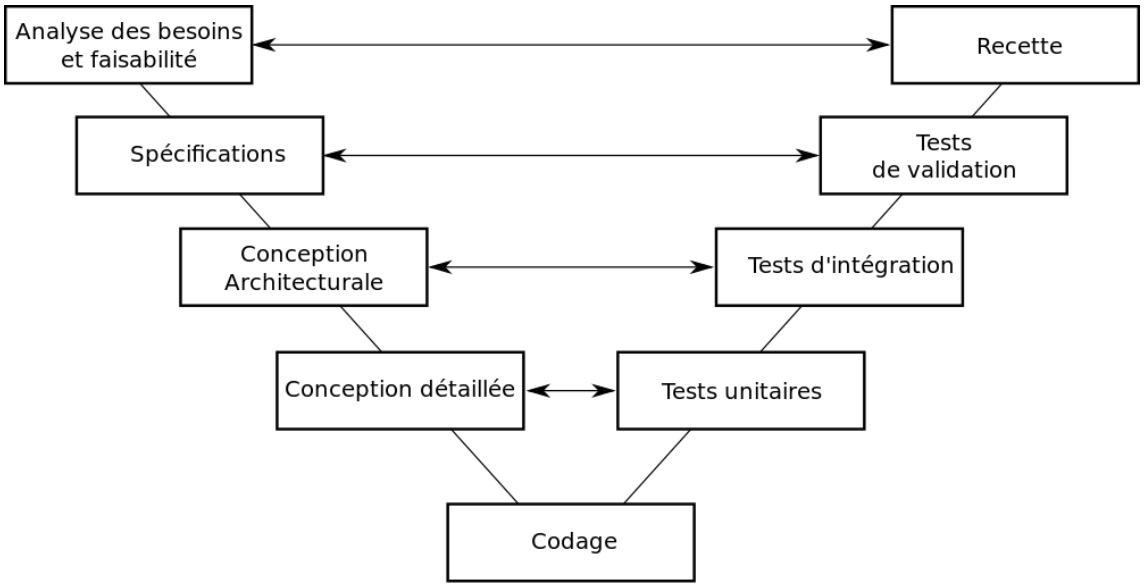
Le cycle en V est devenu un standard de l'industrie logicielle et dans les autres domaines de l'industrie en général. C'est un modèle de gestion de projet permettant de limiter ces effets de bord, en découpant la réalisation d'un projet en plusieurs étapes de manière séquentielle.

Le cycle en V comporte 3 phases : Conception, Développement et Tests. Chacune de ces phases peut comprendre plusieurs étapes.

On peut noter une correspondances entre les étapes situées sur le même niveau : par exemple, si il existe un problème lors d'un test d'intégration, cela va affecter la conception architecturale (cf. fig. 2.1 p. 9).

- **La phase de conception**
  - Analyse des besoins et faisabilité ;
  - Spécification ;
  - Conception architecturale ;
  - Conception détaillée.

FIGURE 2.1: Schéma du cycle en V



Les étapes de la phases de conception commencent par une approche très globale du projet, et augmentent progressivement le niveau de détail jusqu'à la phase de codage. Chaque étape de conception s'appuie sur l'étape précédente.

— **La phase de développement logiciel (codage)** Il s'agit du développement du produit, qui s'appuie sur la conception détaillée.

— **La phase de tests**
  — Tests unitaires ;
  — Tests d'intégration ;
  — Tests de validation ;
  — Recette.

Les tests sont des étapes très importantes dans la réalisation d'un produit, car c'est ce qui permet de valider leur bon fonctionnement et la conformité aux attentes du client. Ils commencent par un niveau de détail élevé, puis offrent une vue de plus en plus globale sur le produit final.

Pour la réalisation d'un projet informatique, ce modèle de gestion de projet à l'avantage de prévoir et de quantifier les besoins, de choisir l'architecture logicielle à adopter, de penser à l'intégration des différentes fonctionnalités, avant de commencer son développement. Cela permet notamment d'anticiper certains problèmes de conception pouvant survenir pendant la phase de codage, et donc de réduire développement de fonctionnalités inadaptées.

Par exemple, dans le cadre de la réalisation d'un site Internet, il sera bien utile aux développeurs de savoir que le site devra proposer plusieurs langages avant de commencer la réalisation. En effet, cette fonctionnalité va influencer l'architecture générale du site, et il sera difficile d'implémenter une telle fonctionnalité en cours de développement si elle n'a pas été prévue au départ.

Ainsi, avec une approche théorique, le cycle en V possède de nombreux avantages et peut se révéler très utile dans le développement d'un projet informatique. Toutefois, la mise en pratique de ce modèle de gestion de projet a mis en valeur certains défauts.

2.1.2 Les méthodes agiles

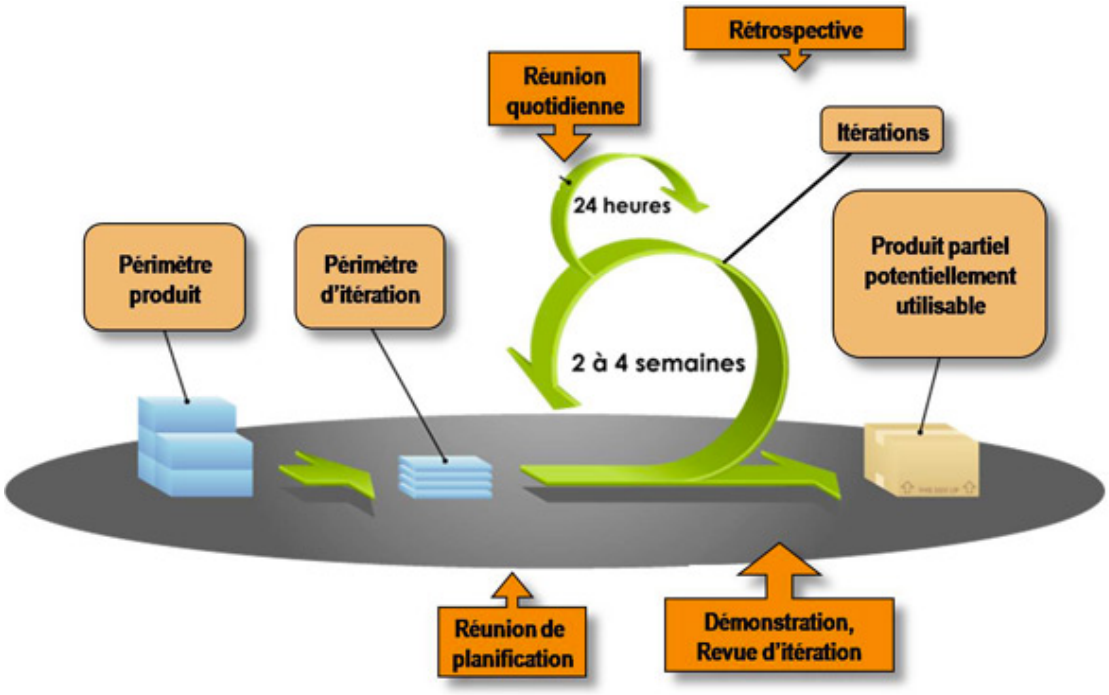
La méthode du cycle en V, bien qu'elle soit intéressante d'un point de vue théorique, possède en réalité de gros inconvénients, pouvant mettre en péril la réussite du projet :

- Les documents de conception sont réalisés par différentes personnes qui ont chacune leur point de vue. Par ailleurs, il n'y a aucun moyen de vérifier la bonne concordance entre ces différents documents. Ainsi, les développeurs peuvent se trouver face à des incohérences considérables dans le dossier de conception.
- Les personnes qui réalisent la conception ont souvent un point de vue trop théorique et n'ont pas forcément en tête les problème techniques qui pourront survenir en utilisant leurs choix de conception.
- La rédaction des différents documents de conception prend du temps et aura donc un impact considérable sur le coût du projet.
- En cas d'arrêt de la production, pour diverses raison, le produit est inutilisable.
- Il est courant que le client change d'avis pendant la réalisation d'un projet. Avec le modèle du cycle en V, un tel changement impactera toutes les étapes de conception, de développement et de tests, ce qui a une forte incidence sur le coût du projet.
- Enfin, le problème le plus important du cycle en V est l'effet dit **Tunnel**. En effet, le client n'a aucune visibilité sur le projet pendant sa réalisation : il est sollicité uniquement au début (pour l'analyse des besoins) et à la fin (pour la recette). Ainsi, si une confusion apparaît sur le cahier des charges, le client s'en apercevra uniquement pendant la recette. En prenant également en compte les confusions pouvant exister entre les différentes étapes de conception, le client peut se retrouver face à un produit ne correspondant pas du tout à ses attentes.

Ainsi, l'industrie informatique a parfois connu des scénarios catastrophiques en utilisant la méthode du cycle en V. Dès lors, pour limiter ces dangers, d'autres modèles de gestion de projet ont vu le jour.

Les méthodes agiles, apparues dans les années 1990, sont un groupe de pratiques de développement de projets informatiques reposant sur un même philosophie.

FIGURE 2.2: Schéma du circuit agile



Elles consistent à développer le produit de manière **itérative, incrémentale et adaptative** (cf. fig. 2.2 p. 10). Les fonctionnalités sont développées les unes après les autres, les plus importantes en premier. Le client est sollicité régulièrement afin de vérifier la conformité entre ses attentes avec ce qui a été développé.

Le manifeste Agile (cf. annexe 3.6 p. 22) définit les méthodes agiles par ces quelques principes :

Extrait de *Agile Manifesto*

> *Nous découvrons comment mieux développer des logiciels par la pratique 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.*

En outre, les méthodes agiles permettent :
— une bonne conformité entre les attentes du client et le produit développé ;
— une grande réactivité à ses demandes, même pendant la réalisation du produit ;
— d'obtenir un produit partiellement fonctionnel, quelque soit l'état d'avancement du projet.

Il existe plusieurs méthodes Agiles, qui ont leurs propres spécifications et qui doivent être adaptées en fonction du contexte. Ainsi, il ne suffit pas de choisir de développer un projet au moyen des méthodes agiles, il faut également se pencher sur la méthode à adopter. Parmi les plus connues, on retrouve :

La méthode Scrum

Le terme *Scrum* (signifiant « mêlée ») provient du rugby à XV, sport qui a pour objectif d'atteindre un but grâce à une équipe soudée. Le projet est découpé en plusieurs phases appelées *sprints*, d'une durée d'environ un mois, pendant lesquelles l'équipe a comme objectif de développer un ensemble précis de fonctionnalités. Au début du sprint, l'équipe se rassemble pour une *réunion de planification de sprint*, pendant laquelle l'équipe cherche à prévoir ce qui sera développé durant le prochain sprint et sur la manière dont ses membres atteindront cet objectif. À la fin du sprint, l'équipe se retrouve pour une réunion appelée *revue de sprint*, pour valider ensemble les fonctionnalités développées. Pour davantage de dynamisme, la mêlée quotidienne (Daily Scrum) permet également aux développeurs de faire un point de coordination d'environ 15 minutes sur les tâches en cours et sur les difficultés rencontrées.

Les fonctionnalités sont souvent représentées physiquement au moyen de *Post-its* à coller sur un tableau composé de trois colonnes correspondant à leur état d'avancement : *à faire*, *en cours* et *réalisé*. Au sein de l'équipe, une personne désignée *ScrumMaster* a la charge de motiver l'équipe, de vérifier les tâches développées et proposer celles du prochain sprint, ainsi que de former le directeur de produit et l'équipe à la méthode Scrum. Ce statut n'a aucune signification hiérarchique et un nouveau ScrumMaster peut être désigné à chaque sprint.

Cette méthode facilite le dynamisme et le travail d'équipe, elle est adaptée aux projets qui peuvent évoluer.

Elle répond aux besoins d'évolution des projets car le client teste régulièrement la partie fonctionnelle du produit et en donne son avis au minimum à chaque sprint : ainsi il met en avant les problèmes et les amélioration attendues de manière régulière.

La méthode XP (Extreme programming)

La méthode XP consiste en différents principes à appliquer pendant la réalisation d'un projet. Ces derniers existent depuis de nombreuses années, toutefois ils sont ici poussés à l'extrême :

Extrait de *Extreme Programming Explained*

- *Puisque la revue de code est une bonne pratique, elle sera faite en permanence (par un binôme) ;*
- *puisque les tests sont utiles, ils seront faits systématiquement avant chaque mise en œuvre ;*
- *puisque la conception est importante, elle sera faite tout au long du projet (refactoring) ;*
- *puisque la simplicité permet d'avancer plus vite, nous choisirons toujours la solution la plus simple ;*
- *puisque la compréhension est importante, nous définirons et ferons évoluer ensemble des métaphores ;*
- *puisque l'intégration des modifications est cruciale, nous l'effectuerons plusieurs fois par jour ;*
- *puisque les besoins évoluent vite, nous ferons des cycles de développement très rapides pour nous adapter au changement.*

Cette méthode repose sur des cycles de développement rapides, elle est adaptée aux équipes réduites avec des besoins qui peuvent évoluer.

2.2 Critères de choix

2.2.1 Rapport avec le client

2.2.2 Préférences au sein de l'entreprise

2.3 Application

2.3.1 Le cas de GFI Informatique

L'entreprise GFI comprend un Service Qualité. Son but est de contrôler chaque livraison au quotidien afin de déterminer si les règles de qualité sont appliquées au sein de chaque projet. Par ailleurs elle met en place des plans et des processus à appliquer, de manière à harmoniser les différents projets GFI.

Une norme qualité est souvent nécessaire pour répondre aux appels d'offre, la plus utilisée étant la CAGR 3.1.

Ces méthodes et usages imposés varient en fonction du client. Dans le cas d'Airbus, elles varient même en fonction du service, car chacun a différentes exigences : un projet industriel, dont le code sera intégré à l'avion, mettra en application des processus assez lourds, avec une documentation importante, de nombreux tests et procédés de validation, tandis qu'un projet de recherche sera bien plus léger.

Les méthodes et normes de développement sont souvent impactées par le cycle de vie du logiciel et l'incidence qu'entraînera une éventuelle anomalie.

Pour un grand nombre d'application, comme les logiciels bureautiques ou les jeux vidéos, le logiciel sera en service quelques années avant que le client passe a la version supérieure. D'autre part un bug aura un effet indésirable pour client mais n'aura pas une grande incidence.

En revanche, l'aéronautique est un secteur assez particulier puisque d'une part une anomalie informatique peut avoir d'importantes conséquences (jusqu'au crash d'un avion),

d'autre part le cycle de vie est très long : la conception elle-même de l'avion peut prendre de nombreuses années, et l'avion sera en service encore pendant plusieurs décennies.

Exceptions :

Une ESN doit s'adapter aux clients. Si celui-ci n'impose aucune méthode ou norme à appliquer, elles sont décidées par GFI au cas par cas et dans ce cas ne comportent pas de Service Qualité.

Pour les équipes de recherche, qui ne se conforment pas à ces normes et dont les projets ne comportent pas de Service Qualité. En effet le code fournit par ces équipe a pour objectif de valider la conception et le fonctionnement d'un produit et obtenir un résultat au plus vite. Il ne sera pas intégré à l'application, mais sera repris par une autre équipe, qui développera la version industrielle en respectant les normes en vigueur.

2.3.2 Le cas de Ineo

2.4 Conclusion

Nous avons vu les différentes méthodes de travail pouvant être adoptées pour la réalisation d'un projet informatique. Nous avons également vu comment le rapport avec le client et les préférences au sein de l'entreprise peuvent influencer le choix de ces méthodes.

Partie 3 : Les outils d'aide au travail collaboratif

Intro avec Google Drive : outils collaboratifs devenus omniprésents

3.1 Les ERP

Un Progiciel de Gestion Intégré (PGI) [1] , est une solution logicielle regroupant différents outils ayant pour but d'assister les différentes composantes de l'entreprise. Il permet une gestion globale et simplifiée, via un support organisationnel unique pour toute l'entreprise. L'usage d'une base de données commune facilite grandement la gestion des différents domaines de gestion de l'entreprise.

L'entreprise CXP, spécialisée dans le conseil et l'analyse de solutions logicielles d'entreprise, définit ce progiciel ainsi :

> *Un PGI est un progiciel qui intègre les principales composantes fonctionnelles de l'entreprise : gestion de production, gestion commerciale, logistique, ressources humaines, comptabilité, contrôle de gestion. À l'aide de ce système unifié, les utilisateurs de différents métiers travaillent dans un environnement applicatif identique qui repose sur une base de données unique. Ce modèle permet d'assurer l'intégrité des données, la non-redondance de l'information, ainsi que la réduction des temps de traitement.*

Il existe des PGI pour de nombreux corps de métiers : l'informatique, la santé, l'éducation, le commerce de détail, etc. D'autre part, ces progiciels sont modulaires et permettent d'activer uniquement les fonctionnalités nécessaires, ils sont également hautement paramétrables. Ainsi, un PGI pourra s'adapter afin de correspondre exactement aux besoins de l'entreprise. Le paramétrage d'un tel progiciel peut donc se révéler complexe, aussi certaines entreprises préfèrent sous-traiter l'installation du PGI.

On peut notamment retrouver dans la base de données d'un PGI :

- Une table pour les produits, comportant leurs nomenclatures, leurs matières premières, leurs quantités, etc. ;
- Une table pour les clients, comportant leurs commandes et livraisons ;
- Des tables pour les stocks, les durées de conservations, les délais d'acheminement des transporteurs ;
- Des tables relatives aux aspects financiers de l'entreprise.

On remarque ainsi qu'à travers une unique base de données, plusieurs domaines rentrent en jeu : la table des produits comporte à la fois les nomenclatures et matières premières, qui sont des informations relatives à la fabrication ; mais également la quantité, qui est une information relative à la vente.

1. PGI : aussi appelé ERP, de l'anglais Enterprise Resource Planning.

3.2 Outils utilisés en informatique

Le développement informatique en équipe pose de nombreuses difficultés spécifiques, en plus de celles qui sont commune à toute forme de collaboration. Afin de faire face à ces difficultés, de nombreux outils spécifiques ont été créés. Certains de ces outils sont largement utilisés en entreprises.

3.2.1 Systèmes de Gestion de Versions

Dès que plusieurs développeurs travaillent sur un même logiciel, le partage des modifications est l'une des premières difficultés rencontrées. On peut les décomposer en plusieurs problèmes :

- Chacun doit disposer de la dernière version du logiciel, afin de ne pas baser ses modifications sur un code qui n'est plus d'actualité.
- Si plusieurs personnes apportent des modifications différentes à leur copies respectives d'un même fichier, plus personne ne dispose d'une version incluant toutes les dernières modifications. Pour cela, il est nécessaire de créer une version fusionnant ces modifications.
- Parfois, plusieurs versions d'un même logiciel doivent coexister (par exemple, une version stable à laquelle ne sont apportées que des corrections de bugs, et une version de développement proposant de nouvelles fonctionnalités expérimentales). Cela peut rapidement devenir source de confusion.
- Suite à la découverte d'une régression (apparition d'un bug qui n'existait pas précédemment), il est important de déterminer exactement quelle modification a entraîné ce bug afin de pouvoir le corriger au plus vite.

Les outils de contrôle de version permettent de répondre à ces problèmes. Parmi ces logiciels, les deux qui sont le plus couramment utilisés sont SVN et Git. Bien que leur fonctionnements internes soient fondamentalement différents, ils offrent des fonctionnalité similaires :

- Regroupement d'un ensemble de modifications en unités atomiques (baptisées « révisions » par SVN et « commits » par Git)
- Publication de ces modifications
- Accès aux informations utiles concernant chaque modification (auteur, date, différences ligne à ligne)
- Fusionnement (« merge ») automatique des modifications à un même fichier, lorsque c'est possible
- Création de « branches », permettant d'isoler plusieurs versions divergentes
- Possibilité de retourner à n'importe quel état antérieur

Ces fonctionnalités permettent de répondre aux problèmes formulés ci-dessus.

3.2.2 Rapport de bugs et gestion des feuilles de route

Un logiciel est destiné à évoluer constamment. De nouveaux bugs sont découverts, et le client demande de nouvelles fonctionnalités. Une gestion de projet efficace nécessite d'identifier ces différentes tâches (qu'il s'agisse de corrections ou d'évolutions), d'en estimer le coût et d'y affecter des ressources humaines et matérielles. Il existe des outils prévus à cet effet, tels que Trac ou Redmine.

Ces outils associe à chaque tâche un « ticket ». Chaque ticket peut être affecté à une ou plusieurs personnes, ce qui permet au chef de projet de suivre l'avancement global du projet ainsi que la répartition des tâches *via* une interface Web.

Lorsque cela est souhaité, ces outils permettent à des personnes extérieures au projet de créer de nouveau tickets. Cette pratique est largement répandue dans les projets open-source, mais elle est également utilisée par certaines entreprises afin d'offrir au client un cadre formel pour effectuer des demandes.

De plus, ces outils offrent une intégration avec les systèmes de gestion de versions. Il est par exemple possible d'afficher la liste des modifications se rapportant à un ticket donné, ou d'imposer des restrictions sur les modifications autorisées (par exemple, interdire à un développeur de publier des modifications relatives à un ticket auquel il n'est pas affecté).

3.2.3 Outils de contrôle de qualité du code

Tout développeur est capable d'écrire du code compréhensible pour l'ordinateur. Faire en sorte que ce code soit également compréhensible par les autres développeurs participant au projet est tout aussi important, mais peut s'avérer plus difficile. En effet, chacun a son propre style et ses idiosyncrasies auxquelles les autres ne sont pas habitués.

C'est pourquoi il existe des *conventions de code* - des ensembles de règles de programmation. Certaines sont des standards internationaux, d'autres des règles internes à une entreprise. Lorsque tous les participants à un projet comprennent et respectent les mêmes conventions de code, la compréhension mutuelle est grandement facilitée, ce qui à son tour facilite la collaboration.

Ces conventions ne sont utiles que si elles sont appliquées, et vérifier cela manuellement serait laborieux. Heureusement, il existe des outils informatiques permettant de vérifier automatiquement le respect des conventions. Ces outils d'analyse du code ont généralement d'autres fonctionnalités, comme la détection d'erreurs de programmation potentielles.

Une fois mis en place, ces outils permettent de détecter et de corriger au plus vite certains problèmes.

3.3 Les outils de communication

On peut également noter l'ensemble des outils servant à la communication : on dissocie alors la communication inter-entreprise et la communication avec le client.

On peut citer avant-tout les outils classiques comme les mails et les messageries instantannées. Ces outils, qui sont très génériques, peuvent dans certains cas se présenter comme un frein dans le développement du projet, car leur usage peuvent s'étendre à du divertissement (essentiellement lorsqu'il s'agit de communication interne). Si de tels outils sont mis à disposition, il est alors de la responsabilité de chacun d'en faire un usage pertinent.

Il y a ensuite des outils conssacrés au développement informatique, qui servent aussi à la communication au sein du projet, tels que les programmes de rapport de bugs et de feuille de route. Comme vu précedemment, ils permettent au client de communiquer les éventuelles anomalies présentes sur le produit. En interne, ils permettent également au chef de projet de lister les différentes tâches à réaliser avant la livraison d'un produit.

3.4 La documentation

La documentation d'un projet de présente pas un outil à part entière mais est un composant essentiel pour une bonne collaboration dans l'avancement d'un projet.

Dans le développement de produits informatiques, on distingue deux types de documentation :

— **La documentation utilisateur**
— **La documentation technique**

3.5 Application

3.5.1 Le cas de GFI Informatique

SVN est l'unique système de gestion de version est utilisé dans toute l'entreprise. En effet le choix d'un tel logiciel concerne essentiellement le fournisseur.

La suite Microsoft Office est également utilisée : Microsoft Word pour la rédaction des dossiers, comme la documentation utilisateur, la documentation technique, les spécifications, etc. Ce logiciel intègre dans sa dernière version un système de versionnement, ce qui facilite la rédaction de document de manière collaborative. Microsoft Outlook est également utilisé, d'une part pour la communication par mail (concernant l'organisation interne de l'entreprise, la communication entre membres de l'équipe et la communication avec le client), d'autre part pour l'ogranisation des réunions d'équipe.

3.5.2 Le cas de Ineo

3.6 Conclusion

Ainsi, il existe de nombreux outils informatiques permettant de faciliter la collaboration. Cependant, aussi sophistiqués que soient ces outils, ils ne font pas tout. Le facteur humain reste primordial, et une bonne entente est nécessaire afin d'atteindre une synergie efficace entre les participants.

Conclusion

Bibliographie

Pro Git

Auteur : Scott Chacon
Editeur :
Date de publication :

Le Mythe du Mois Homme

Auteur : Frederick Brooks
Editeur :
Date de publication :

Le Manifeste agile

Titre original : Agile Manifesto
Auteurs = Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas
Editeur :
Date de publication :

Extreme Programming Explained : Embrace Change

Auteurs : Kent Beck et Cynthia Andres
Editeur :
Date de publication :

ERP et conduite des changements

Auteurs : Jean-Louis Tomas et Yossi Gal
Editeur :
Date de publication :

Glossaire

ESN Entreprise de services du numérique. 7, 13

PGI Progiciel de Gestion Intégré. 14

Annexes

Manifeste pour le développement Agile de logiciels

Nous découvrons comment mieux développer des logiciels par la pratique 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.

Principes sous-jacents au manifeste

Nous suivons ces principes :

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