



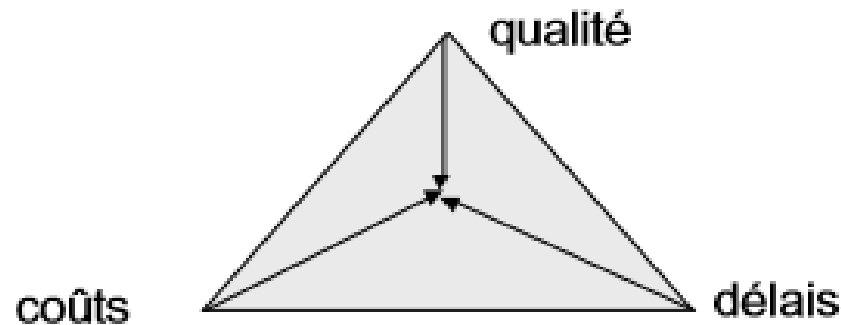
Gestion de Projets Informatiques



Introduction

Projet en général :

- **Définition :**
- Ensemble **d'actions** à entreprendre afin de répondre à un **besoin** défini (avec une **qualité** suffisante), dans un **délai** fixé, mobilisant des **ressources** humaines et matérielles, possédant un **coût**.

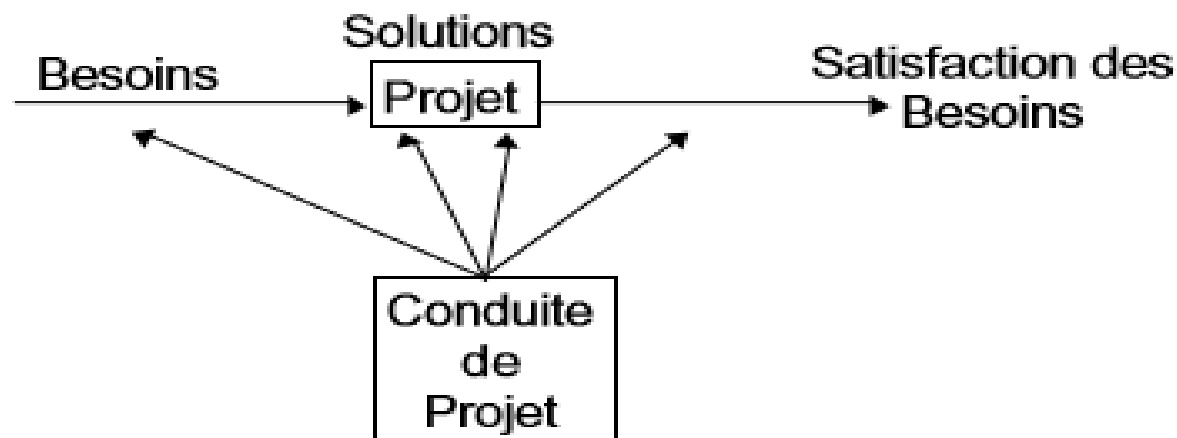


Acteurs d'un projet

- **Maître d'ouvrage :**
 - personne physique ou morale propriétaire de l'ouvrage. Il détermine les objectifs, le budget et les délais de réalisation.
- **Maître d'œuvre :**
 - personne physique ou morale qui reçoit mission du maître d'ouvrage pour assurer la conception et la réalisation de l'ouvrage.

Conduite de projet

- Organisation méthodologique mise en œuvre pour faire en sorte que l'ouvrage réalisé par le maître d'œuvre réponde aux attentes du maître d'ouvrage dans les contraintes de délai, coût et qualité.



Projet logiciel :

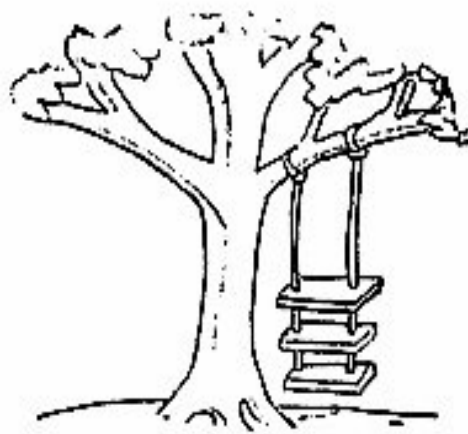
- **Qu'est-ce qu'un logiciel ?:**
- Un **logiciel** est un ensemble d'entités nécessaires au fonctionnement d'un processus de traitement automatique de l'information.
- Parmi ces entités, on trouve par exemple :
 1. des programmes (en format *code source* ou *exécutables*) ;
 2. Des *bases de données*;
 3. des documentations d'utilisation ;
 4. des informations de configuration.

Catégories de logiciels

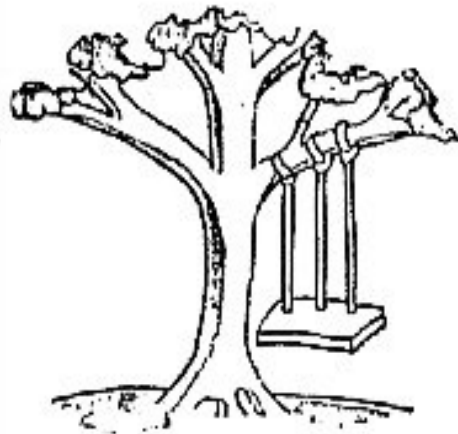
- Les deux principales catégories de logiciels sont:
- **le logiciel applicatif:** est destiné à aider les utilisateurs à effectuer une certaine tâche.
- **le logiciel de système:** est destiné à effectuer des opérations en rapport avec l'appareil informatique.

La crise du logiciel

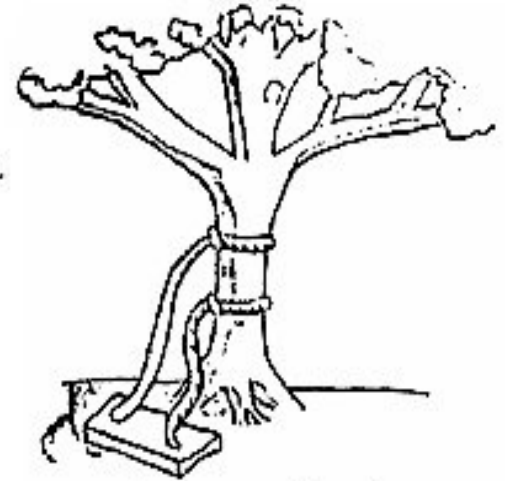
- ❖ les délais de livraison des logiciels sont rarement tenus, le dépassement de délai et de coût moyen est compris entre 50 et 70 % ;
- ❖ la qualité du logiciel correspond rarement aux attentes des acheteurs.
- ❖ le logiciel ne correspond pas aux besoins
- ❖ les modifications effectuées après la livraison d'un logiciel coûtent cher, et sont à l'origine de nouveaux défauts.
- ❖ il est rarement possible de réutiliser un logiciel existant pour en faire un nouveau produit.



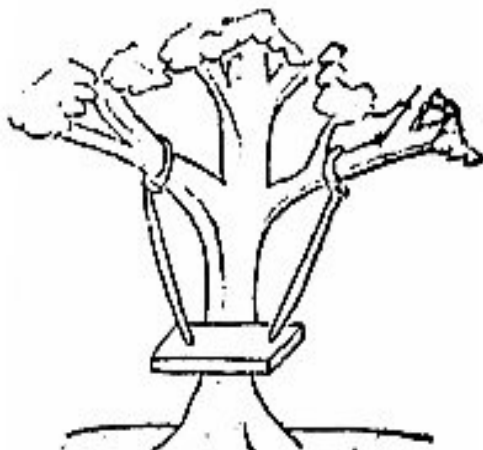
Proposé par le commercial



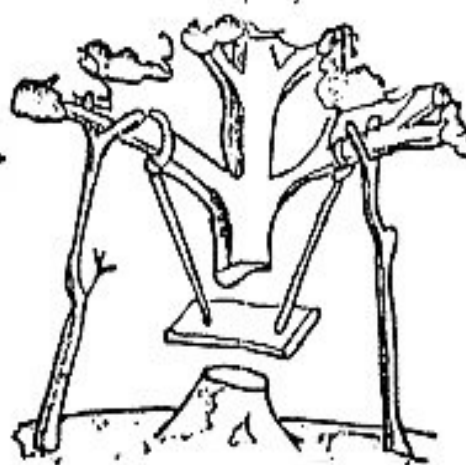
Spécifié par le chef de projet



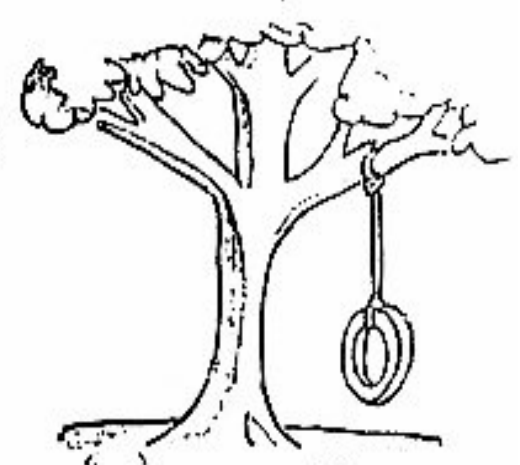
Conçu par l'équipe de conception



Réalisé par les programmeurs



Installé sur le site



Ce que voulait l'utilisateur

Génie logiciel

- **Définition:**
- Ensemble de **méthodes, techniques et outils** pour la production et la maintenance de composants logiciels de **qualité**.
- **Pourquoi:**
- logiciels de plus en plus gros, technologies en évolution, architectures multiples.

Génie logiciel

- **Principes:**
- Rigueur et formalisation, séparation des problèmes, modularité, abstraction, prévision du changement, généricité, utilisation d'incréments.

Cycle de vie d'un logiciel

- **Cycle de vie d'un logiciel:**
- Un cycle de vie est un ensemble de phase séquentiellement cohérente, dont le nom et le nombre de séquences sont déterminés à partir de **l'étude des besoins du projet à réaliser**.
- Ce projet est le plus souvent, soit le développement d'un service ou celui d'un produit à des fins commerciales.
- Il existe plusieurs manières de définir le cycle de vie d'un logiciel.

Description du cycle de vie d'un logiciel


- Le cycle de vie d'un logiciel indique les étapes par lesquelles doivent passer un logiciel de sa conception jusqu'à sa mort.
- Ce cycle de vie permet de détecter les erreurs tout au long du processus de réalisation et ainsi les corriger pour produire un **logiciel de qualité**.

Description du cycle de vie d'un logiciel

- **Les étapes sont les suivantes :**
 - **Pré-étude** : Cette étape permet de définir les objectifs du projet et de définir le domaine d'activité.
- En entrée on a les besoins et en sortie on a un cahier de charges.
- **Analyse** : Cette étape consiste à recueillir et à formaliser les besoins du client, de définir les contraintes et d'estimer la faisabilité de ces besoins.
- En entrée on a le cahier de charges et en sortie on a le dossier d'analyse.
- **Conception** : Cette étape permet d'élaborer la structure générale du système et de définir chaque sous-ensemble du logiciel à produire.
- En entrée, on a le dossier d'analyse et en sortie on a un dossier de conception.

Description du cycle de vie d'un logiciel

- **Codage** : Cette étape consiste à coder ou à programmer les fonctionnalités définies dans la phase de conception.
 - En entrée, on a le dossier de conception et en sortie on a des programmes.
- **Tests** : Cette étape permet de tester le logiciel conformément aux spécifications (fonctionnelle ou non fonctionnelle).
 - Il existe quatre types de tests à savoir : **le test unitaire, le test d'intégration, le test fonctionnel et le test de validation.**
- **Recette**: Cette étape permet au client de vérifier la conformité du logiciel avec les spécifications initiales.
 - En entrée on a un logiciel plus un cahier de charges et en sortie on a un procès verbal de réception (**acceptation ou refus du livrable**)

- 
- **Maintenance** : Cette étape permet de prendre en charge les actions collectives du système (**maintenance curative et évolutive**).
 - En entrée on a un logiciel et en sortie on a un logiciel modifié.

Modèles de cycle de vie d'un logiciel

- Deux types principaux de modèles:

- **Modèle linéaires:**

- en cascade et variantes



- **Modèles non linéaires**

- en spirale, incrémentaux, itératif



Modèle en cascade

- Ce modèle créé par **Winston W.Royce** en 1970 est basé sur deux éléments fondamentaux :
- On ne peut passer à l'étape suivante tant la précédente n'est pas terminée.
« Inutile de monter les murs tant que les fondations ne sont pas coulées »
- La modification d'une étapes a un impact important sur les étapes à venir.

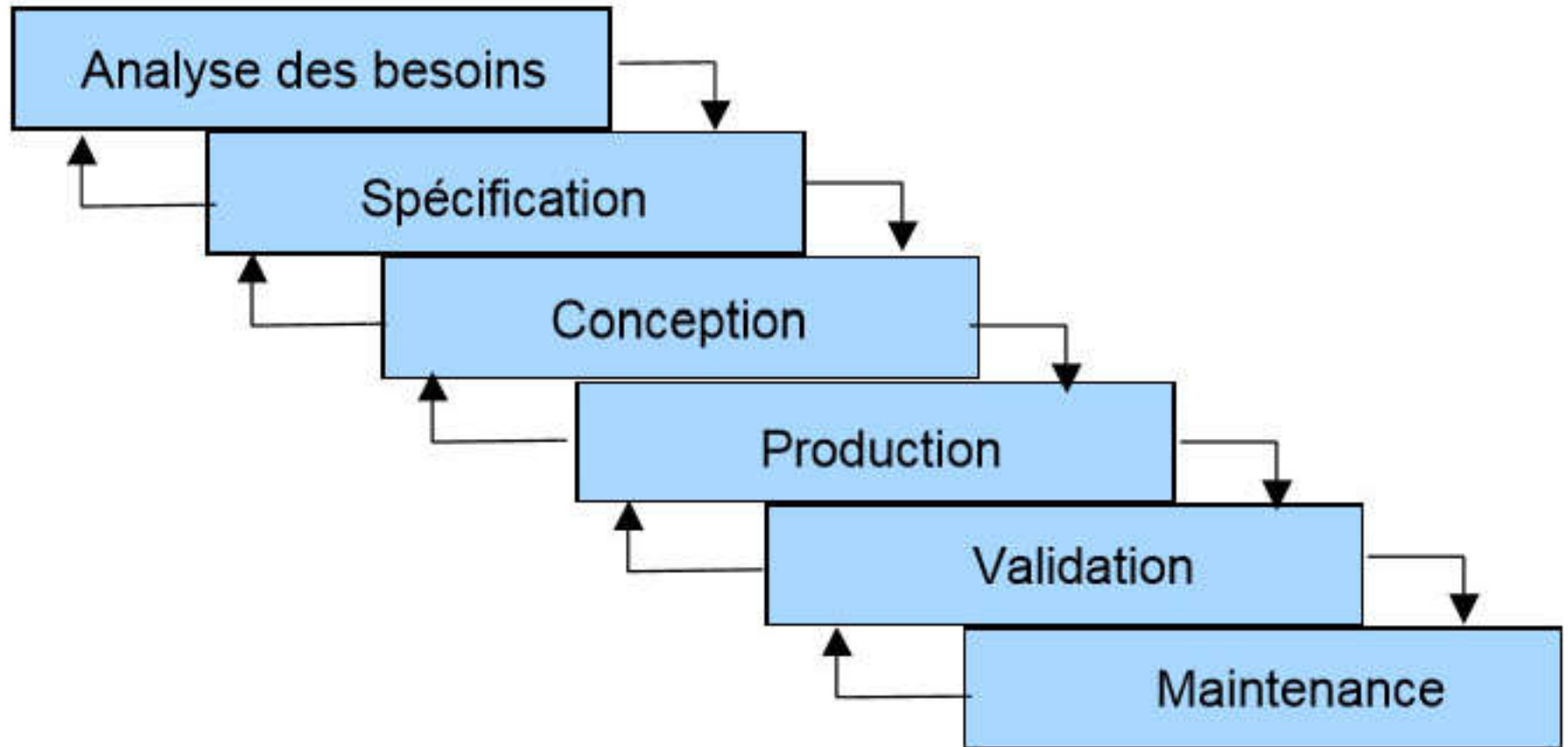
Modèle en cascade

- Lors du processus, chaque phase doit être définie précisément, c'est-à-dire, qu'il correspond au cahier des charges écrit ultérieurement, et possède une date d'échéance fixe.
- Lorsque l'étape est validée le processus continu, sinon l'étape est refaite. Si une erreur critique est rencontrée, il est possible de revenir à la première étape.

Modèle en cascade

- **Avantages du modèle en cascade :** Le planning est établi à l'avance et le maître d'ouvrage sait précisément ce qui va lui être livré et quand il pourra en prendre livraison.
- **Inconvénients du modèle en cascade :** ils sont assez nombreux mais le principal inconvénient est la très faible tolérance à l'erreur (les anomalies sont détectées tardivement) qui induit automatiquement un coût important en cas d'anomalie.

Modèle en cascade



Modèle en V

- Face aux problèmes de réactivité que pose l'approche en cascade, l'industrie informatique a adopté le cycle en v dans les années 80.
- Ce modèle ne se découpe en 9 phases qui se répondent 2 à 2 : à **chaque étape de conception correspond une phase de test** ou de validation, comme vous pouvez le voir ci-dessous.

Conception

Tests

1 Analyse du besoin

Les besoins du client sont analysés afin de définir les usages du produit final.
Que veut le client ?

2 Spécifications

Cette étape correspond à la rédaction du cahier des charge des spécifications fonctionnelles du produit final.
Que doit faire le produit ?

3 Conception architecturale

Elle est la traduction technique des spécifications fonctionnelles : elle décrit l'ensemble des briques de l'application et leurs interfaces.
Comment doit-il être fait ?

4 Conception détaillée

Elle peut être un début de code ou une documentation qui définit chaque brique fonctionnelle.

Recette fonctionnelle 9

Cette étape correspond à la dernière étape avant la mise en production. Le client vérifie que le produit fini répond aux besoins exprimés lors de l'analyse du besoin.

Tests de validation 8

Les tests de validation peuvent être faits par l'ensemble des futurs utilisateurs. Ils doivent vérifier la partie fonctionnelle de l'application.

Tests d'intégration 7

Ces tests sont réalisés sur l'ensemble du produit fini et assurent le respect du cahier des charges technique.

Tests unitaires 6

Les tests unitaires permettent de vérifier que chaque brique respecte le cahier des charges.

5 Réalisation

Les briques sont créées puis assemblées afin de créer le produit final

Avancement
du projet



Maitrise
d'ouvrage



Maitrise
d'œuvre

Modèle en V

- **Avantages du modèle en V :** La stricte structure en V permet d'espérer que le livrable final sera parfait, puisque les étapes de test sont aussi nombreuses que les étapes de réflexion.
- De plus, il est facile de prévoir les tests à réaliser au moment où l'on conçoit une fonctionnalité ou une interface, le travail s'enchaîne donc de façon assez naturelle.

Modèle en **V**

- **Inconvénients du modèle en **V** :**
Malheureusement ce modèle est rarement utilisé tel quel et le V est bien souvent déséquilibré, tantôt côté analyse, tantôt côté recette et la marge d'erreur est bien souvent proportionnelle à la marge de liberté prise par rapport au modèle théorique.

Problèmes du modèle séquentiel

- **Rigidité du processus:**
 - Analyse: le processus fait l'hypothèse que les besoins peuvent être complètement précises et captures au début du projet.
 - Les changements aux besoins sont très communs.
 - Le client peut changer d'avis après avoir utilisé le logiciel.

Problèmes du modèle séquentiel


- **Conception:** le processus fait l'hypothèse que la conception peut être complétée avant de débiter l'implémentations.
 - Irréaliste dans les cas où la solution n'est pas bien comprise.
 - Un changement des besoins peut invalider la conception.

Problèmes du modèle séquentiel

- **Difficulté à s'adapter aux changements**
 - Les changements imprévus peuvent invalider beaucoup de travail des phases antérieures.
- **Le client ne voit que le système final**
 - La quasi-totalité du développement s'effectue sans commentaires du client.
 - Le développement peut facilement dévier des besoins réels du client.

Problèmes du modèle séquentiel

- Le client modifie souvent sa perception du problème et par conséquent ses besoins après avoir utilisé le système.
- **Les problèmes difficiles ne sont pas attaqués en premier**
 - Peut mener à l'abandon de certains projets.



Développement itératif & méthodologies agiles

Développement itératif

- Le cycle de vie du logiciel est constitué de mini-projets appelés **itérations**.
 - Chaque **itération** inclut des activités d'analyse, de conception, d'implémentation et de test.
 - Le but d'une itération est de produire une version **stable**, testée et **partielle d'un logiciel**.
 - Le résultat d'une itération n'est pas un prototype, mais une version incomplète du système final.

Développement itératif

- Le résultat de la dernière itération est le logiciel complété.
- Chaque itération ajoute des fonctionnalités au logiciel.
- Occasionnellement, une **itération** peut être consacrée à l'amélioration (ex: performance), sans ajout de fonctionnalité.

Développement itératif

- Le système s'agrandit progressivement, itération par itération:
 - Le système peut ne devenir convenable pour un environnement de production qu'après plusieurs itérations.

Développement itératif

- Le cycle de vie du logiciel est basé sur le raffinement successif et l'évolution du système à travers une série d'itérations avec rétroaction et adaptation:
 - Les besoins doivent être stables à l'intérieur d'une itération.
 - Aucune activité ne devrait être ajoutée ou modifiée au cours d'une itération.
 - Des activités peuvent être remises à d'autres itérations si le travail ne peut être complété à temps.

Durée des itérations

- Chaque projet doit fixer une durée pour les itérations:
 - La durée typique d'une itération est 2-6 semaines
 - Une durée de 2-3 semaine est très répandue.
 - Des itérations courtes permettent la rétroaction et l'adaptation rapide aux changements.
 - Des itérations longues augmentent les risques

Durée des itérations

- Les itérations ont toutes la même durée
 - Si certaines tâches ne peuvent être effectuées à temps, elle sont remises à une itération ultérieure.
 - Une itération ne doit **jamais** se terminer en retard

Avantages du développement itératif

- **Permet les changements:**

- Le développement peut s'adapter aux changements au cours des itérations subséquentes.

- **Permet l'évolution des besoins:**

- Les utilisateurs peuvent donner leurs commentaires suite à l'utilisation du système opérationnel
- Les nouveaux besoins peuvent être pris en compte de façon incrémentale

Avantages du développement itératif

- **Réduction des risques:**
 - Les risques sont identifiés rapidement
- **Architecture robuste:**
 - L'architecture peut être évaluée et améliorée très tôt



Méthodologies agiles

Définition

- Les **méthodes agiles** sont des groupes de pratiques de pilotage et de réalisation de projets. Elles ont pour origine le **manifeste Agile**, rédigé en 2001, qui consacre le terme d'« agile » pour référencer de multiples méthodes existantes.
- Les méthodes agiles se veulent plus pragmatiques que les méthodes traditionnelles, impliquent au maximum le demandeur (client) et permettent une grande réactivité à ses demandes.
- Elles reposent sur un **cycle de développement itératif, incrémental et adaptatif**.

Caractéristiques

- **Agilité** : réponse rapide et flexible aux changements.
- Plusieurs méthodologies existent, mais ont des caractéristiques communes:
 - Itérations à durée fixe.
 - Développement évolutif (Raffinement progressif des besoins et de la conception).
 - Planification adaptive
 - Livraisons incrémentales

Le manifeste Agile

- Le Manifeste agile est constitué de quatre valeurs et de 12 principes fondateurs.
- **Les 4 valeurs:**
- *Nous découvrons de meilleures approches pour faire du développement logiciel, en faisant nous-mêmes et en aidant les autres à en faire. Grâce à ce travail nous en sommes arrivés à préférer et favoriser:*

Le manifeste Agile

- **Les individus et leurs interactions** plus que les processus et les outils.
- **Un logiciel qui fonctionne** 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.

<<

*Bien que nous reconnaissons la valeur des items de droite, vous
valorisons les items de gauche*

>>.

Les 12 principes agiles

1. Notre plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée.
2. Accueillez positivement les changements de besoins, même tard dans le projet. Les processus agiles exploitent le changement pour donner un avantage compétitif au client.
3. Livrez fréquemment un logiciel opérationnel avec des cycles de quelques semaines à quelques mois et une préférence pour les plus courts.
4. Les utilisateurs ou leurs représentants et les développeurs doivent travailler ensemble quotidiennement tout au long du projet.

Les 12 principes agiles

- 5. Réalisez les projets avec des personnes motivées. Fournissez-leur l'environnement et le soutien dont elles ont besoin et faites-leur confiance pour atteindre les objectifs fixés.
- 6. 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.
- 7. Un logiciel opérationnel est la principale mesure d'avancement.

Les 12 principes agiles

8. 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.
9. Une attention continue à l'excellence technique et à une bonne conception renforce l'agilité.
10. La simplicité – c'est-à-dire l'art de minimiser la quantité de travail inutile – est essentielle.
11. Les meilleures architectures, spécifications et conceptions émergent d'équipes auto-organisées.
12. À intervalles réguliers, l'équipe réfléchit aux moyens de devenir plus efficace, puis règle et modifie son comportement en conséquence.

Méthodologies agiles

- Scrum
- Extreme Programming (xp)
- Et bien d'autres:
 - Crystal
 - Evo
 - etc.



Méthode Scrum

Introduction

- **Scrum** est la **méthodologie** la plus utilisée parmi les méthodes Agiles existantes.
- Le terme Scrum (qui signifie mêlée) apparaît pour la première fois en 1986 dans une publication de **Hiroataka Takeuchi** et **Ikujiro Nonaka** qui décrit une nouvelle approche plus rapide et flexible pour le développement de nouveaux projets informatique.

Principe

- Evidemment, l'approche **Scrum** suit les principes de la méthodologie Agile, c'est-à-dire l'implication et la participation active du client tout au long du projet.
- Considéré comme un **cadre** (*framework* en anglais) de gestion de projet.

Caractéristiques

- **Processus itératif:**
 - Itérations plus longues que d'autres méthodologies (30 jours).
- **Équipe autogérée:**
 - Pas de processus rigide.
 - Le développement est adapté empiriquement.
- **Réunion debout quotidienne (mêlée, ou scrum).**

Caractéristiques

- Démonstrations du système après chaque itération.
- Planification impliquant le client après chaque itération.

Phases

➤ **Planification:**

- Établir la vision du projet, les attentes et assurer le financement.
- Activités : définition du carnet de produit initial, estimés, conception exploratoire, prototypes.

➤ **Mise en scène:**

- Identification de plus de besoins, priorisation suffisante pour une première itération

Phases

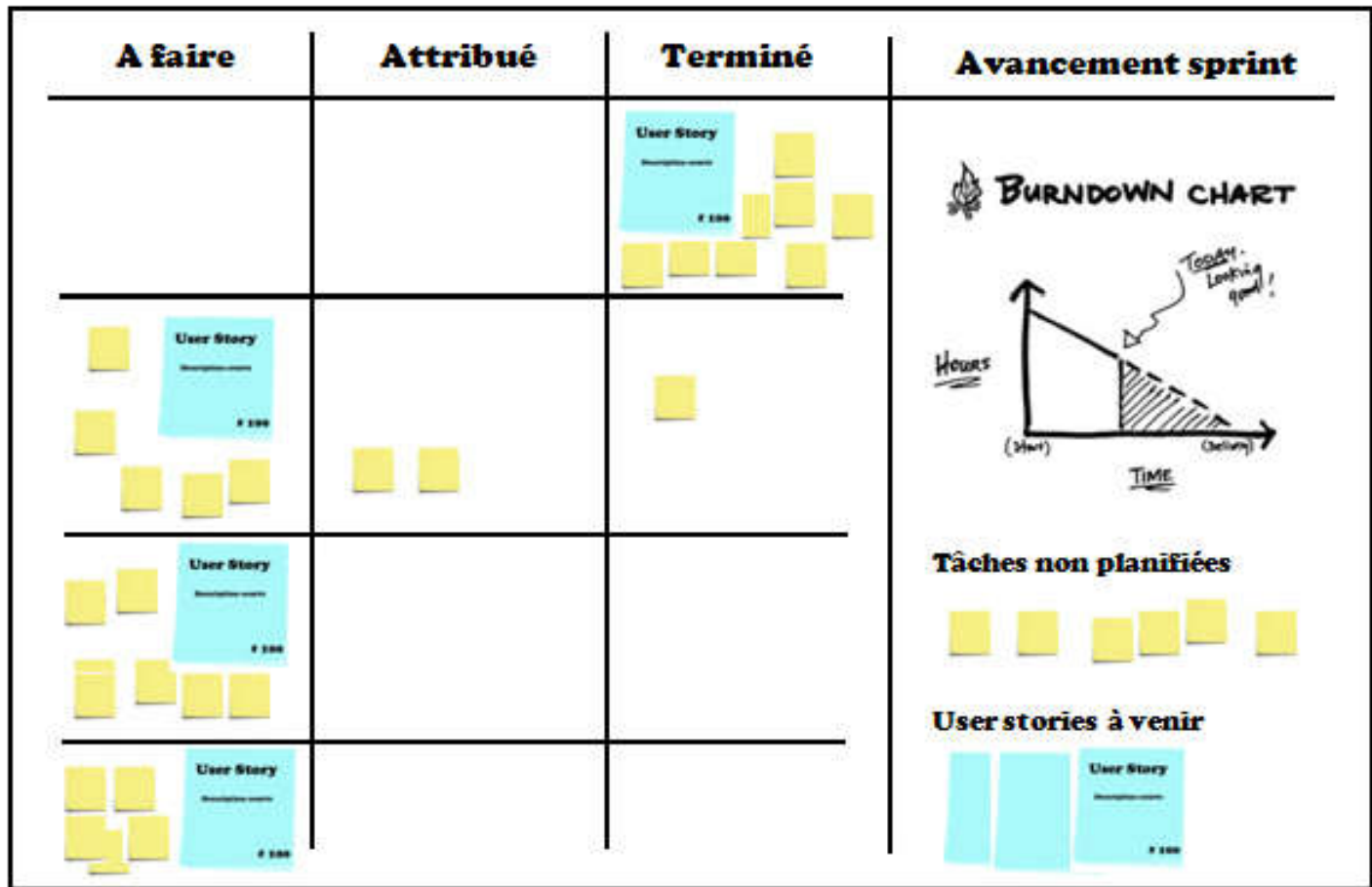
- Activités: planification (**Planning Poker**), conception ,prototypes, **Product Backlog** (liste ordonnancée des exigences fonctionnelles et non fonctionnelles du projet).

➤ **Développement:**

- Implémentation d'un système par une série d'itérations de 30 jours (**sprints**).
- **Activités** : planification de sprint, définition du carnet de sprint, mêlée quotidienne, revue de sprint

Phases

- Exemple du carnet de sprint :



Phases

- **Livraison d'une version du système (release):**
 - Déploiement
 - Activités : formation, documentation, commercialisation, etc.

Rôles

- **Scrum Master:**
 - **Élimine les obstacles:**
 - Idéalement en < 1 jour (avant la prochaine mêlée).
 - Prend des décisions lorsque nécessaire:
 - Idéalement en < 1 heure.
 - Agit comme écran (firewall) pour s'assurer que l'équipe n'est pas interrompue par des requêtes venant de l'extérieur.
 - Renforce la vision du projet

Rôles

- **Équipe:**

- Scrum recommande que les équipes soient limitées à 7-10 personnes.
- Les grands projets contiennent plusieurs équipes.

Rôles

- **Propriétaire du produit (Product Owner):**
 - Un représentant du client
 - Assigne les priorités dans le carnet du produit
 - Choisit les besoins à inclure dans une itération.

Carnets (*Backlogs*)

- **Carnet de produit:**
 - Appartient au propriétaire du produit
 - Contient les besoins de haut niveau, leur priorité, leur valeur d'affaire et une estimation de l'effort requis.
 - Durant la planification d'avant-match, tous les intervenants peuvent ajouter des fonctionnalités, des cas d'utilisation, des améliorations et des défauts au carnet de produit

Exemple – Carnet du produit

Besoin	Num.	Cat.	État	Pri.	Est.	Sprint
Enregistrement des paiements dans le registre	17	Fonctionnalité	En cours	5	2	1
Plan de financement à long terme	232	Amélioration	Terminé	4	38	1
Calcul de la commission de vente	12	Défaut	Pas débuté	2	14	2
Approbation de crédit lente	321	Problème	En cours	5	2	1

Carnets (*Backlogs*)

- **Carnet de sprint:**
 - Appartient à l'équipe
 - Contient des tâches et un estimé de l'effort requis / restant

Exemple – Carnet de sprint

			Heures de travail restantes				
Besoin	Resp.	État	6 362	7 322	8 317	9 317	10 306
Rencontre pour discuter des objectifs	JM/SR	Terminé	20	10	0	0	0
Déplacer les calculs hors du module ...	AW	Pas débuté	8	8	8	8	8
Obtenir les données ...	TN	Terminé	12	0	0	0	0
Créer et initialiser la base de données	GP	En cours	24	20	30	25	20
...							

Mêlée quotidienne (scrum)

- Tenue à chaque jour, à la même heure et au même endroit:
 - Doit débiter à l'heure, il est fréquent d'imposer des amendes aux retardataires qui sont ensuite utilisées comme dons de charité.
- Doit répondre à 5 questions :
 - Qu'avez-vous fait depuis le scrum précédant?.
 - Qu'allez-vous faire entre maintenant et le scrum suivant?.
 - Qu'est-ce qui entrave l'atteinte des buts de l'itération en cours?

Mêlée quotidienne (scrum)

- Y a t'il des tâches à ajouter au backlog?
 - Tâche manquantes, pas de nouveaux besoins.
- Avez-vous appris ou décidé quelque chose de nouveau qui serait utile aux autres membres de l'équipe?.
- Aucune autre discussion n'est permise
 - Le **Scrum Master** peut recentre la discussion au besoin.

Mêlée quotidienne (scrum)

- Idéalement tenue debout en cercle pour encourager la brièveté.
- Doit se tenir près d'un tableau où les tâches sont inscrites
- Dure en moyenne 15-20 minutes pour une équipe de 7-10 personnes
 - Des réunions plus longues sont communes au début d'une itération.
- Les membres qui sont absents doivent participer par haut-parleur.

Mêlée quotidienne (scrum)



Revue de sprint

- À la fin de chaque itération, le **Scrum Master** organise la revue de sprint.
 - Maximum 4 heures
 - Tous les intervenants participent à la revue
- L'équipe fait la démonstration du système au client:
- Le client est informé des fonctions du système, de la conception, des forces et faiblesses du système, de l'effort de l'équipe.
- Le client peut donner ses commentaires
- Pas de transparents, le produit doit être montré
- Les engagements sont pris lors de la planification du prochain sprint, et non pas lors de la revue.

Revue de sprint





eXtreme Programming (XP)

Introduction

- La méthodologie eXtreme Programming ou XP est une méthode de gestion de projet qui applique à **l'extrême les principes** du développement agile, c'est-à-dire se concentrer sur les besoins du clients, mettre en place un développement itératif et l'intégration continue.
- L'équipe projet et ses relations avec le client sont au coeur de XP.
- Cette méthode a été créée par Kent Beck entre 1996 et 1999, lorsqu'il travaillait sur un projet pour Chrysler.
- Elaborée à l'origine pour le secteur informatique, eXtreme Programming est aujourd'hui très populaire car **elle fonctionne pour tous types de projets, de toutes tailles et de tous secteurs confondus, partout dans le monde.**

Introduction

- Cette méthodologie est idéale pour de petites et moyennes équipes, c'est-à-dire pas plus d'une vingtaine de personnes environ.

Principe

- Les principes de la méthode **eXtreme Programming** ne sont pas nouveaux puisqu'il s'agit de ceux des méthodes Agiles.
- La différence et l'originalité résident dans le fait qu'ils sont poussés à l'extrême.
- La méthode eXtreme Programming s'appuie sur :
 - une forte réactivité au changement des besoins du client ;
 - un travail d'équipe;
 - la qualité du travail fourni ;
 - la qualité des tests effectués au plus tôt.

Principe

- XP repose sur cinq valeurs fondamentales :
 - **Communication** : il est essentiel que chaque membre de l'équipe communique quotidiennement avec ses collègues ainsi qu'avec le client.
 - C'est un moyen incontournable pour résoudre les problèmes.
 - **Simplicité** : la façon la plus simple d'arriver au résultat est privilégiée. L'équipe projet fait ce qui est nécessaire et demandé, rien de plus. Une application simple sera plus facile à faire évoluer ensuite.
 - **Feedback** : le retour d'information entre l'équipe projet et le client est essentiel. Chaque étape du projet est envoyée aussi rapidement et souvent que possible au client afin qu'il teste, donne son avis et valide l'étape. Chaque demande de modification est prise en compte immédiatement.

Principe

- **Respect** : le respect de chaque membre de l'équipe et de son travail sont primordiaux. Le management, l'équipe projet et le client se respectent mutuellement.
- **Courage** : Il faut du courage pour effectuer certains changements comme essayer une nouvelle technique, recommencer une itération non validée ou revoir l'organisation du projet. Le courage permet de sortir d'une situation inadaptée.

Fonctionnement

- Les cinq valeurs de XP se déclinent en **treize pratiques** qui se renforcent mutuellement :
- **Client sur site** : le client doit être représenté sur place pendant toute la durée du projet. Ce représentant doit avoir une vision globale du résultat à obtenir et être disponible pour répondre aux questions de l'équipe.
- **Jeu du planning (ou planning poker)**: le planning est réalisé en collaboration avec le client. Ce dernier crée des scénarios pour les fonctionnalités qu'il souhaite obtenir. L'équipe évalue le temps nécessaire pour les mettre en œuvre. Le client sélectionne ensuite les scénarios en fonction des priorités et du temps disponible.

Fonctionnement


- **Intégration continue** : lorsqu'une tâche est terminée, elle est tout de suite intégrée dans le produit complet. Cela permet d'éviter la surcharge de travail due à l'intégration de tous les éléments avant la livraison. Les tests facilitent cette intégration : quand tous les tests sont positifs, l'itération est terminée.
- **Petites livraisons**: les livraisons doivent être les plus fréquentes possible afin que le client donne son avis et que les modifications soient rapidement prises en compte par l'équipe.
- **Rythme soutenable** : aucune heure supplémentaire n'est tolérée. S'il y en a, alors le planning doit être revu. Un collaborateur fatigué travaille mal et fait plus d'erreurs.
- **Tests fonctionnels** : à partir des scénarios définis par le client, l'équipe crée des procédures de test qui permettent de vérifier l'avancement du développement. Lorsque tous les tests fonctionnels passent, l'itération est terminée.

Fonctionnement

- **Tests unitaires** : pour chaque fonctionnalité, un test est écrit afin de vérifier qu'elle fonctionnera comme prévu. Ce test sera conservé jusqu'à la fin du projet, tant que la fonctionnalité est requise. À chaque modification du code, tous les tests sont lancés afin d'identifier immédiatement s'il y a un problème de fonctionnement.
- **Conception simple** : on va droit à l'essentiel en se focalisant uniquement sur les besoins actuels du clients. Plus l'application est simple, plus il sera facile de la faire évoluer lors des prochaines itérations.
- **Utilisation de métaphores**: les équipes XP utilisent des métaphores pour décrire le système et son fonctionnement afin de clarifier les fonctionnalités à atteindre. Tout le monde parle le même langage.

Fonctionnement

- **Refactoring (ou remaniement du projet)**: le projet est amélioré régulièrement. Le but étant d'avoir de bonnes bases et de meilleures conditions de travail pour l'équipe.
- **Appropriation collective du projet** : la responsabilité du projet est collective. Chaque membre de l'équipe peut modifier toutes les portions du projet, mêmes celles sur lesquelles il n'a pas travaillé. L'objectif est d'être efficace et rapide.
- **Standards de langage** : puisque tout le monde travaille ensemble sur le projet, il est essentiel de faciliter le travail de chacun en utilisant les mêmes termes, le même style et des règles de communication claires.
- **Travail en binôme** : les collaborateurs travaillent en binôme. Le pilote et le copilote changent régulièrement afin d'améliorer la communication et la connaissance collective du projet.

- 
- Les projets gérés par la méthode eXtreme Programming reposent sur des cycles de développement (itérations) courts et rapides qui sont réalisés collectivement par l'équipe projet et le client dont l'implication est constante.
 - Les activités contre-productives ont été supprimées afin de réduire les coûts et la frustration de toutes les personnes impliquées.