3BIN - Synthèse Agile

Chef de projet - Produire plan, définir rôle et personnel, piloter et informer, décisions et motivation, contrôle progrès et coûts, informer sponsors et clients

5 conditions pour un projet :

- Le mandant (le client interne ou externe, le demandeur) doit exister et être clairement identifié
- Un chef de projet doit être nommé
- Le but est connu et formalisé
- Le délai est connu et formalisé
- Le budget est connu et formalisé

Eléments clefs :
Objectifs, Finalités, Déliverables, Parties prenantes, Méthode de gestion de projet, Bugdet-Délai, Risques, Gestion du changement
Méthodes :
• Chute d'eau (si erreur, retour en arrière)
Possible de formaliser besoins et ne changent pas
Projet court
Application produite en un run
• Incrémental (livrables)
Livrée en versions
Projet long
Acceptation logiciel en cours de dev
• Itérative (Moderne, évolution avec vie application)

Planning : mise en œuvre d'objectifs dans le temps - !domaine identifié - !objectifs précis - !moyens - !durée&étapes
Ordonnancement : organiser dans le temps la réalisation de tâches connues pour atteindre un objectif identifié, compte tenu de contraintes temporelles, telles que les durées estimées et les contraintes d'enchaînement, et des contraintes de ressources.
Contraintes d'enchaînement :

		A -> B
SF	Start-to-Finish	On doit commencer B avant d'avoir fini A
SS	Start-to-Start	On ne peut commencer B que si on a déjà commencé A
FS	Finish-to-Start	On ne peut commencer B que si on a déjà complètement fini A
FF	Finish-to-Finish	On ne peut finir B qu'après avoir fini A

PERT : Diagramme pour visualiser relations entre tâches. Cf ex cours
GANT : Diagramme pour visualiser une série de tâches dans le temps. Rectangle = tâche, longueur = durée.
Nivellement de capacité : Lisser bes besoins de ressources lors de l'ordonnancement.
Chemin critique : Succession de tâches sans marge début-fin projet. !retard
RACI : Spécifie responsabilité intervenant (Responsible, Acountable, Consulted, Informed)

Agile :

Valeurs
- Equipe soudée, communiquants, tous développeurs > experts, outils, …
- Logiciel fonctionnel avant le reste (doc, …)
- Feedback et implication client durant le dev
- Accepter et intégrer changements vàv du client

Principes
- Satisfaire clients avec livrables tôt et utiles
- Changement bienvenu. Livrables courts. Collaboration client. Personnes motivées et soutenues. Conversations face à face. Logiciel fonctionnel comme unité de mesure d'avancement. Rythme soutenable ∞. Excellente technique. Simplicité. Equipe auto-organisatrice. Réflexion régulière de l'équipe pour améliorer efficacité.

Planification
- Travaille comme une équipe
- Travaille en itérations courtes
- Délivre quelque chose à chaque itération (en principe au client)
- Se focalise sur les priorités "business" (en principe, les priorités du client)
- Mesure et s'adapte

Rôles
- Responsable produit
  - vérifier membres équipe partagent vision commune du projet
  - décider des priorités
  - prendre décisions nécessaires pour assurer rentabilité du projet
- Client
  - finance projet. souvent RP
- Développeurs
  - n'importe qui développe ou contribue à la réalisation
- Chef de projet
  - Membres équipe -> même définition du concept de réussite du projet
  - Principes Agile connus + respectés, !autonomie !auto-organisation

Les rôles de chef de projet et de développeurs peuvent être périodiquement échangés, mais pas trop souvent (toutes les 2 ou 3 itérations).

Itérations courtes 2sem à 2mois. Constant pour éviter procrastination. ±5it/release.
Une itération = testé, documenté, fonctionnel, parfois utilisable. Release -> complet.

Fonctions business écrites pour se focaliser sur les besoins client, "user stories"

Re-planification à chaque itération, ainsi que pour N+1 voire N+2. Communication client si retard impactant.

User stories (As a <type of user> , I want/must <capability> so that <business value>)

Non ambigüe	
Indépendante	autonome des autres US
Concise	
Négociable	contrat non figé client-dev
Valorisable	valeur pour le client
Estimable	qté de travail évaluable
Cohérente	voc unifié, pas de contradictions avec reste projet
De bonne taille	pas trop légère ou complexe
Vérifiable	

INVEST: Independent, Negotiable, Valuable, Estimatable, Sized appropriately, Testable

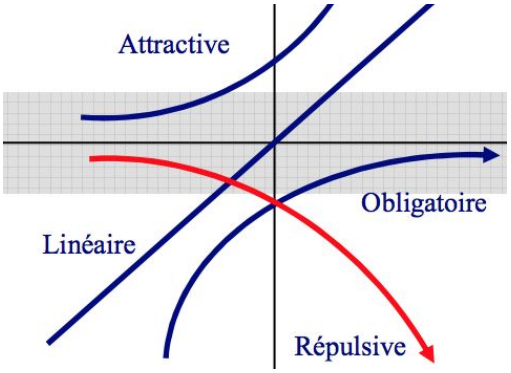
- Différents types d'US :

Fonctionnelle	règle métier
Non fonctionnelle	sécurité, confidentialité, …
Contrainte	lié aux règles (loi, procédures internes, …)
Conceptuelle	définir un concepte (classe, zone d'atterrissage, stockage…)

- Diviser US
  - séparer cas différents
  - trop grande pour une seule itération
  - isoler des fonctionnalités
  - séparer des spécifications de ≠ types (fonctionnelle, performance, …)
  - séparer fonctionnalités ± prioritaires
- **! Jamais** diviser une US en fonction de son implémentation (créer table X, tester X, …).
- Taille des US
  - jours/homme => calcul des besoins de l'us, pas sa taille
  - jour idéal = journée productive sans contre temps
  - points. arbitraire, relatif, consensuel, proportionnel au temps.
    - planning poker pour le côté consensuel. Se baser sur une suite à progression exponentielle pour gagner du temps. La vitesse est définie par les points réalisés/itérations.
  - ! sur/sous-estimer certaines tâches avec avis externe. cohérence primordiale.
  - ré-évaluer histoire si modifiée. (pas en cours d'implémentation de celle-ci)
  - jamais ré-évaluer pour des raisons de vitesse ou de non complétude.

Modèle Kano (!examen)

- 5 catégories d'US
  - Obligatoire (echec si absent, ok si présent)
  - Linéaire (+ y'a + il est content)
  - Attractive (il apprécie mais ne sera pas insatisfait si absente)
  - Répulsive (il n'en veut pas)
  - Indifférente (il s'en fout)



y = satisfaction, x = fonctionnalité

classer US :

- Si le produit, l'application peut faire cela… qu'en penseriez-vous ?
  - Cela me fait plaisir, j'aime ("Like")
  - C'est un minimum pour moi ("Expect")
  - Cela m'est égal ("Neutral")
  - Je l'accepte, je peux vivre comme cela ("Live with")
  - Cela me dérange ("Dislike")
- Si le produit, l'application ne possède pas la fonction … , qu'en penseriez-vous ?
  - Cela me fait plaisir, j'aime ("Like")
  - C'est un minimum pour moi ("Expect")
  - Cela m'est égal ("Neutral")
  - Je l'accepte, je peux vivre comme cela ("Live with")
  - Cela me dérange ("Dislike")

3) Classer chaque histoire dans le modèle de Kano en utilisant la grille d'analyse ci-contre

4) Analyser la distribution des réponses et décider.

Customer Requirements → / Functional Question ↓	Like	Expect	Neutral	Live with	Dislike
Like	Q	E	E	E	L
Expect	R	I	I	I	M
Neutral	R	I	I	I	M
Live with	R	I	I	I	M
Dislike	R	R	R	R	Q

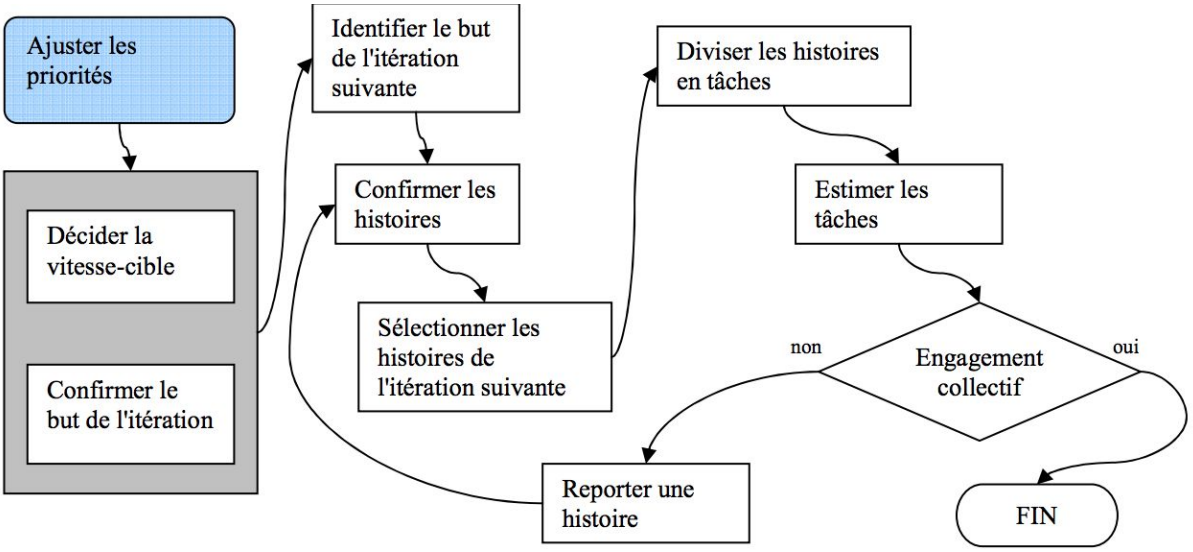
M Must-have
L Linear
E Exciter
R Reverse
Q Questionable
I Indifferent

Histoire	Must	Exciter	Reverse	Linear	?	Indifferent	Result
1	10%	20%	1%	74%	0%	5%	Linear
2	60%	15%	0%	10%	5%	10%	Must

- **Planifier releases**

dans planification du projet :
Déterminer conditions de satisfaction
Ecrire US
Estimer US
Décider longueur itérations / Estimer vitesse / Prioritiser US
Sélectionner US/realease (date-driven project ≠ feature-driven project)

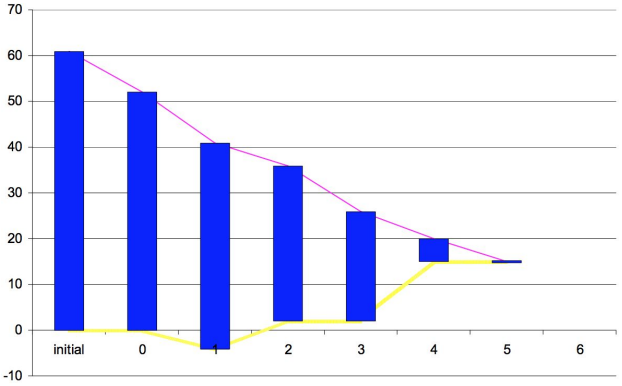
- Planifier itérations



Astuces : Pas assigner tâches à l'avance, Prendre bugs en compte, 'Petites' tâches, Ne pas modifier volume d'itération en cours, Pas terminer itération vendredi.

- Implémenter

Suivi tâche : TODO - In progress - On review - Done - Hours spent
Burndown chart pour le nombre de jours idéaux à effectuer.
Complétion binaire. Si 0, reporté en fonction priorité.
Prise de tâche par les devs.
Chef de projet -> vitesse, release burndown (! ajout/retrait par en dessous, complétion par au dessus).



- Communiquer

être transparant en fin de projet
(Key-figures, ratios, letter…)

- Evaluer risques

projet : Dépassement temps/budget/instatisfaction utilisateurs/clients/échec
opérationnel : disfonctionnement/mauvaise perf/insatisfaction utilisateurs/client

- Gérer risque

projet : buffer, gérer temps, priorité US, implication client
opérationnel : nouvelle release corrective

System characterization - Threat identification - Vulnerability identification - Control analysis - Likelihood determination - Impact analysis - Risk determination - Control recommendation - Results documentation

- Prévision de réserves

Fonctionnalités (max 70% must)
Temps (réserve globale, pas individuelle)

Pourquoi Agile

• Le client est très impliqué dans le projet
Le client participe à la gestion du projet, il la comprend puisque les concepts sont simples et centrés sur les
histoires écrites dans son vocabulaire.
• Le contenu du projet ne doit pas nécessairement être gelé longtemps à l'avance
Des histoires peuvent être changées, ajoutées, retirées à tout moment
• Il n'y a pas de confusion possible entre les points, les jours.hommes et les jours-calendriers
La vitesse lie pointes et itérations.
• Les évaluations sont indépendantes l'une de l'autre
Le problème de surévaluation ou sous-estimation systématique disparaît en utilisant le concept de vitesse
mesurée
• Le syndrome de l'étudiant est minimisé
Des itérations courtes mais constantes maintiennent une pression contrôlée, sans besoins de Pert ou Gantt
détaillés. La plupart des histoires et des tâches sont complétées aussi vite et aussi tôt que possible
• Délivrer souvent un logiciel qui fonctionne est au centre des préoccupations
Et non pas respecter la méthode ou le plan de projet
• La gestion de projet Agile demande moins de temps que la gestion de projet classique
• Un projet Agile est plus motivant pour tous les membres de l'équipe
• La probabilité de satisfaction des utilisateurs et du client est maximisée
Evident vu leur implication dans toutes les activités, y compris la gestion du projet.