Gestion de projet IT

ynov CAMPUS

1. World Café
2. Planification
3. Agile
4. Jeu
5. Scrum
6. Projet

1-World Café

1 hôte de table par sujet

x contributeurs

1 time keeper

3 sujets (3*20 minutes):

-Qu'avez-vous retenu du 1er jour sur la GDP ?

-D'après vous, en quoi l'IA pourrait être utile à la GDP ?

-Quels sont les facteurs qualités d'un projet réussi ?

2-Planification

La base de la planification c'est ?

La base de la planification c'est la communication

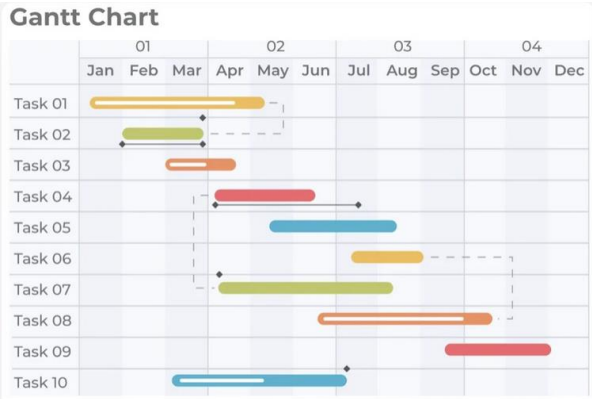
-Que doit on faire ?
-A quel moment ?
-Avec qui et quels moyens ?

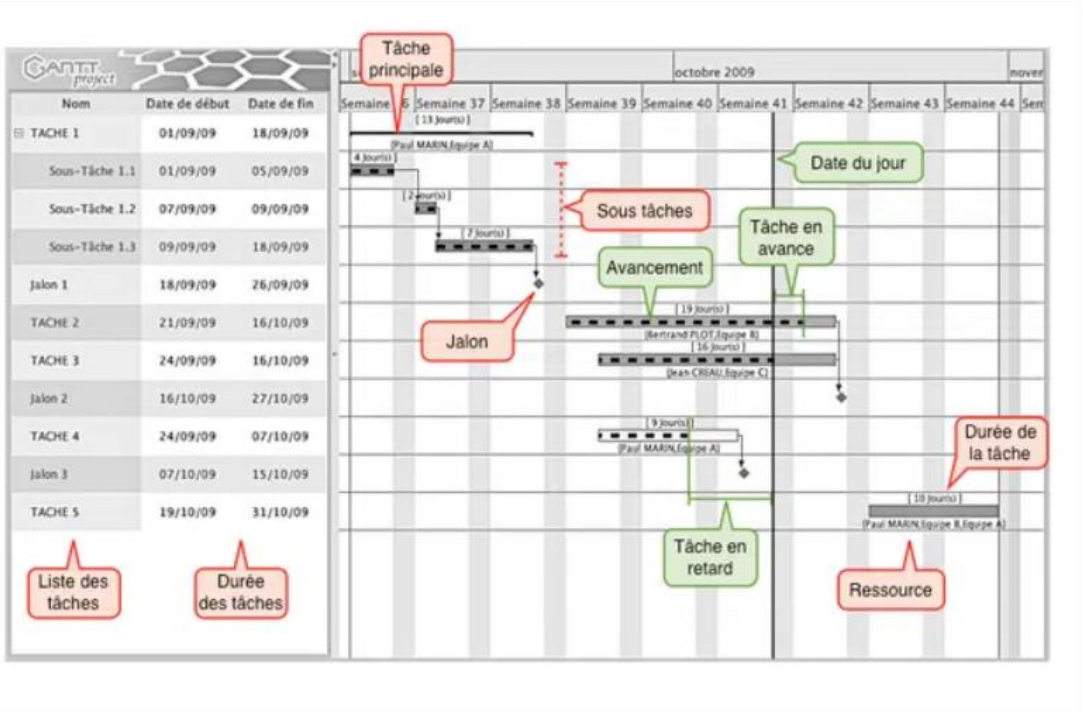
On retrouve les notions de ?

On va retrouver les notions de

-Tâches
-Ressources
-Date de début et date de fin
-Jalons (point d'arrêt qui permet de réaliser son suivi : réunion importante d'arbitrage, date de livrable, échéance)

Gantt: vision chronologique des tâches avec imbrication entres elles

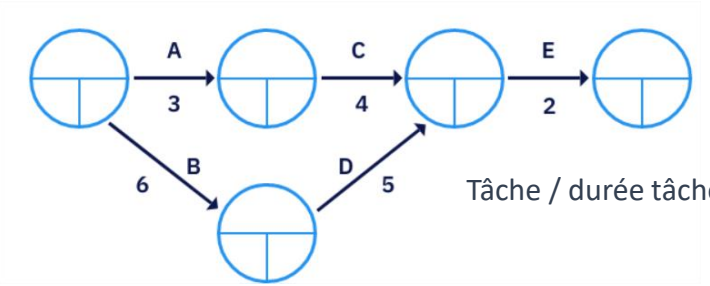




Vision d'ensemble
Efficacité pour gérer les ressources
Meilleur suivi
Chevauchement et interdépendance
Identifie les échéances

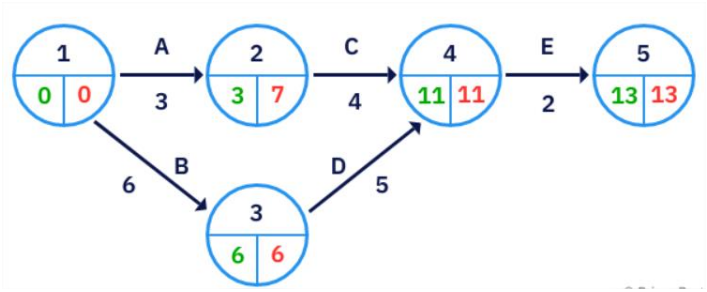
Prends du temps…
Peut être complexe et déroutant
Pas d'identification des priorités

PERT (*Program Evaluation Review Technique*): représentation en réseau des tâches, temps d'exécution et dépendances



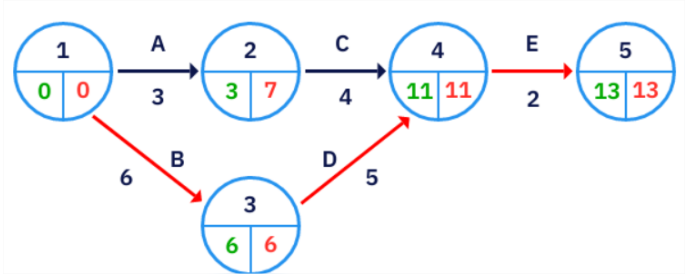
Evènement / jalon

Tâche / durée tâche



Date au plus tôt

Date au plus tard



→ Chemin critique

Au préalable, il est important d'établir le tableau des dépendances associées

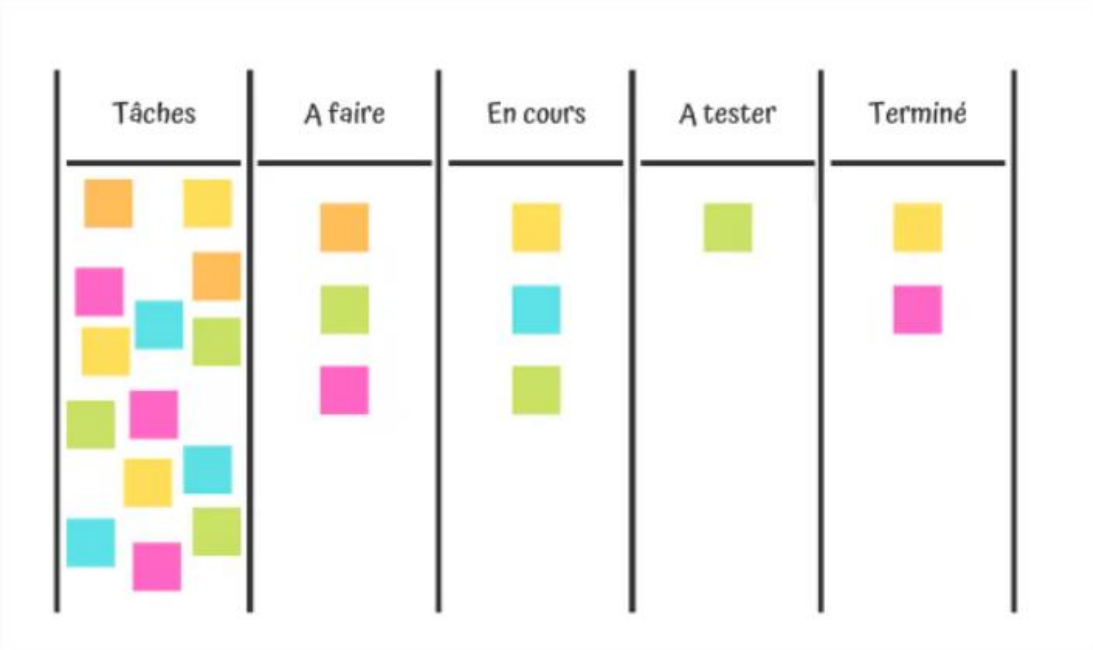
Tâche	Durée	Tâches précédentes
A	3	–
B	6	–
C	4	A
D	5	B
E	2	C, D

Mise en avant des dépendances entre tâches
Estimation plus précises des durées de tâches
Identification faciles des tâches hors chemin critique

Prends du temps…
Difficulté d'interprétation
Durée estimée (si mauvaise estimation, impact négatif sur la planification totale du projet)

https://prium-transition.com/diagramme-de-pert/

KANBAN: issu des années 1950 pour améliorer la production. Identification des tâches et de leur statuts



Circulation des tâches rapides
Lecture claire (en fonction nombre de tâches)
Communication facile
Adapté aux cycle itératif courts
Permet d'identifier si trop de tâches en cours

Adhésion des utilisateurs
Pas de vision des interdépendances de tâches
Pas de vision globale

WBS (Work Breakdown Structure): coûts, planning et références du projet

Organigramme des tâches de façon hiérarchique. En lien avec ma méthode PMBOK (cf cours GDP J1)

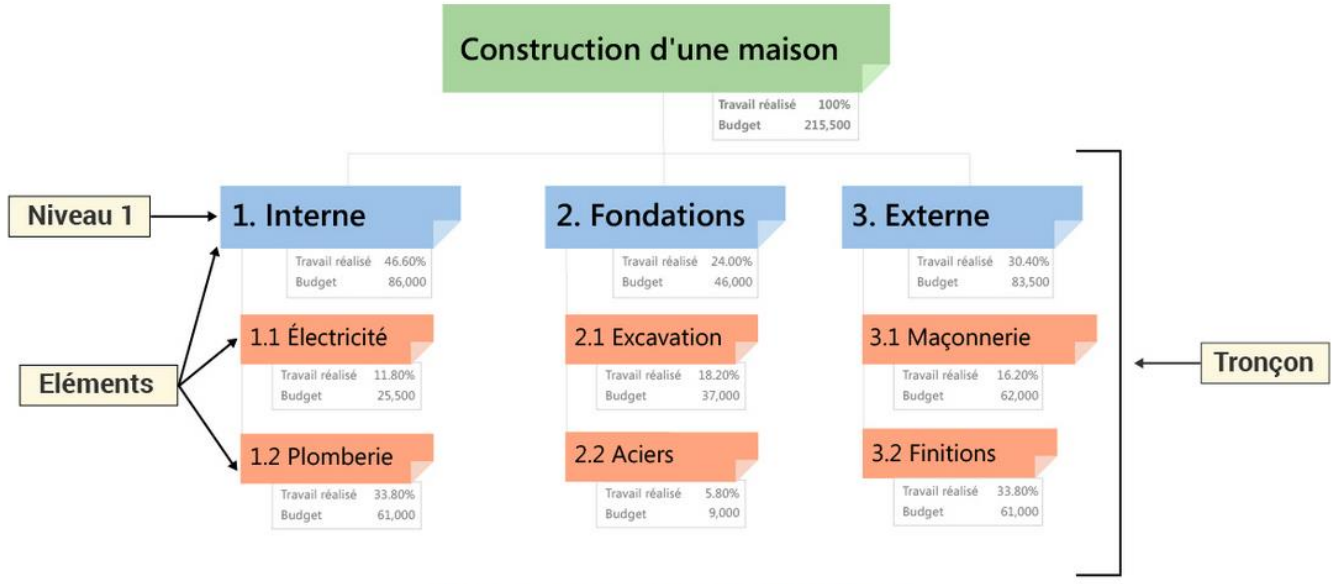
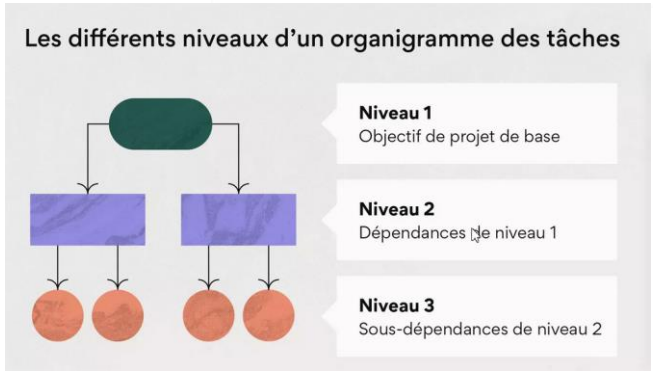


FIGURE 1 – STRUCTURE WBS PAR LIVRABLES



Identifie les responsabilité des chacun
Décrit les activités
Communication et compréhension du projet

Nécessite du temps
Lourd à mettre en œuvre

Exercice Gantt et Pert

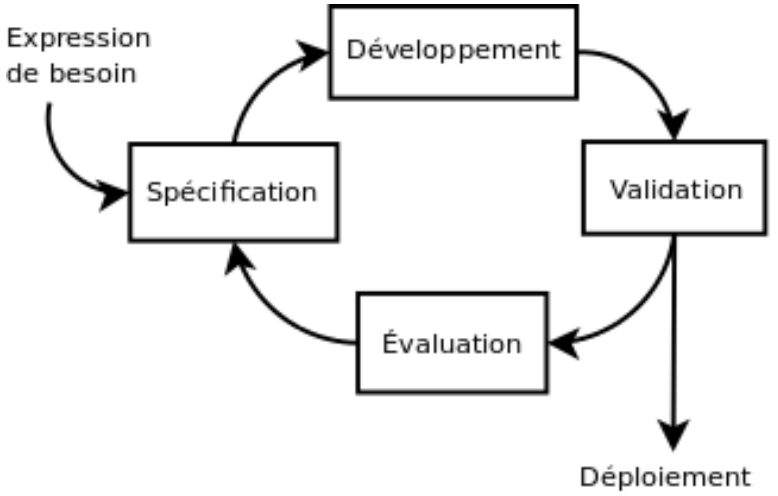
Refaire sa salle de bain

Taches	descriptif tâche	Durée (jour)
A	Estimer le budget	1
B	Connaitre son besoin	2
C	Acheter materiaux	3
D	Effectuer les travaux	10
E	Communiquer sur ce projet	3
F	Voir les salles de bains de mes amis	4
G	Finitions	2
H	Faire des photos et nettoyer	1

Contraintes
C doit être après A
D doit être après B et C
G doit être après D
H doit être après G et E
E doit être après F

4-Agile

- Les méthodes agiles sont des groupes de pratiques de projets de développement en informatique (Wikipédia)
- C’est un cycle de développement itératif
- Elles sont régies par un manifeste
- Ce manifeste est applicable à plusieurs méthodes dites Agiles



1976 : Méthode EVO de Tom Gilb

1986: Première approche du développement incrémental par Barry Boehm

1991: Méthode RAD (Rapid Development Application) par James Martin

1995-96: Méthode Scrum par Ken Schwaber qui décrit les principes de Scrum dans l'article *Controlled Chaos : Living on the Edge*

2001: 17 experts du développement se réunissent pour écrire et publier le *Manifeste Agile*

2011: Jeff Sutherland et Ken Schwaber décrivent les principes de la méthode dans le *Scrum Guide*

Agile – 4 valeurs

- L'équipe :
  - « **aux individus et leurs interactions**, plus que les processus et les outils »
- L'application
  - « **à un logiciel fonctionnel** plutôt qu'à une documentation exhaustive »
- La collaboration
  - « **à la collaboration avec les clients** plutôt qu'à la négociation contractuelle»
- L'acceptation du changement
  - « **à l'adaptation au changement** plutôt qu'à l'exécution d'un plan.»

Agile – 12 préceptes

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

2. Acceptation du changement:
Accueillez positivement les changements de besoins, même tard dans le projet.

3. Livraisons fréquentes:
Livrez fréquemment un logiciel fonctionnel, dans des cycles de quelques semaines à quelques mois, avec une préférence pour les plus courts.

4. Collaboration quotidienne:
Les utilisateurs ou leurs représentants et les développeurs doivent travailler ensemble quotidiennement tout au long du projet.

Agile – 12 préceptes

5. Motivation et encouragements:
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. Communication face-à-face:
Privilégiez la co-location de toutes les personnes travaillant ensemble et le dialogue en face à face comme méthode de communication.

7. Logiciel sert de mesure:
Un logiciel fonctionnel est la principale mesure de progression d'un projet.

8. Rythme soutenable:
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.

Agile – 12 préceptes

9. Attention continue à la technique et à la Conception:
Une attention continue à l'excellence technique et à un bon design.

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

11. Auto-organisation:
Les meilleures architectures, spécifications et conceptions émergent d'équipes auto-organisées.

12. Ajustements continus:
À intervalles réguliers, l'équipe réfléchit aux moyens possibles de devenir plus efficace. Puis elle s'adapte et modifie son fonctionnement en conséquence.

La méthode Agile doit être

Itérative

=>Affinement du besoin

Une méthode agile est avant tout itérative sur la base d'un affinement du besoin mis en œuvre dans des fonctionnalités en cours de réalisation et même déjà réalisées.

Cet affinement, indispensable à la mise en œuvre du concept adaptatif, se réalise en matière de génie logiciel sous deux aspects :

->fonctionnellement, par adaptation systématique du produit aux changements du besoin détecté par l'utilisateur lors de la conception-réalisation du produit (notion de validation permanente de l'utilisateur)

->techniquement, par remaniement régulier du code déjà produit (refactoring).

Adaptative

=>Contrôle de l'évolution du livrable

La notion d'adaptatif, quant à elle, nécessite au-delà d'un simple principe, la mise en œuvre de techniques de contrôle de l'évolution du livrable et d'une métrique formelle des modifications, avant, après et en cours de la production.
Il en découle une planification opérationnelle élémentaire, directement visible par le biais de l'affichage mural

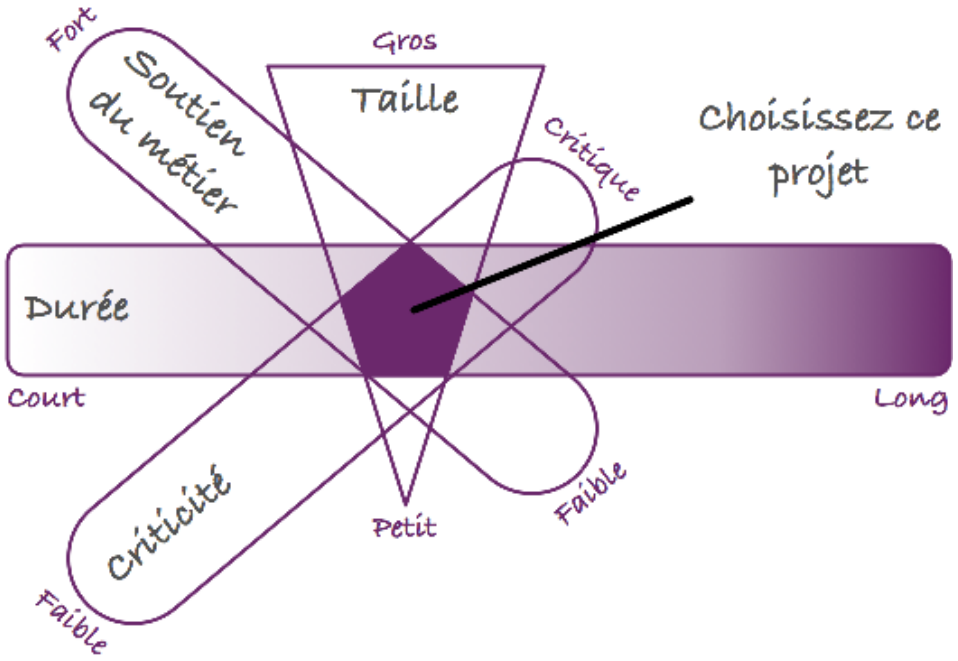
Incrémentale

=>Livraison régulière

Une méthode agile est ensuite, éventuellement, incrémentale.
Lorsque le projet, quel que soit le nombre de participants, dépasse en durée une dizaine de journées en moyenne, la production de ses fonctionnalités s'effectue en plusieurs incréments.

Tous les projets ne sont pas de bons candidats

Compromis: taille, durée et soutien métier



Critères d'applications Positifs

- Besoin rapide de mise à disposition du produit
- Imprévisibilité des besoins du client
- Nécessité de changements fréquents
- Besoin de visibilité du client sur l'avancement des développements
- Présence de l'utilisateur assurant un feedback immédiat

Principes d’applications Positifs

Accord sur la philosophie agile avant de commencer à travailler :
timeboxing (management par le temps), implication des utilisateurs, développement itératif, ...

Accord sur la délégation du pouvoir de décision aux utilisateurs et développeurs embarquées dans l'équipe.

Engagement du management client pour assurer la disponibilité et la Participation importante de certains utilisateurs finaux.

Soutien actif des commerciaux :
la confiance et la collaboration sont plus importants que la négociation d'un contrat.

Technologie utilisée pour le développement :
permet la livraison incrémentale, le prototypage rapide et le refactoring.

Critères d'applications Négatifs

- Indisponibilité du client ou de l'utilisateur
- Dispersion géographique des ressources humaines
- Inertie des acteurs du projet ou refus des changements
- Gouvernance complexe de la DSI

Critères d'applications Négatifs

- Il n'y a rien de nouveau dans l'agilité ce n'est qu'un ensemble de vieilles pratiques qui pour la plupart étaient utilisées dans le développement de logiciels des années 1960, mais qui ont été regroupées d'une nouvelle façon sous l'étiquette agile)

- Le manque de concentration sur l'architecture peut engendrer des mauvaises prises de décision par rapport au design du projet

- Les méthodes de développement agiles sont efficaces dans le contexte de petites équipes projet. Dans le cadre de grands projets, il est plus approprié d'utiliser d'autres méthodes de gestion.

- Difficulté de maintenir le rôle du client au sein de l'organisation dans l'éventualité où le projet se déroulerait sur une très longue période

5-Jeu

Jeu

Le but de ce jeu est de livrer une copie d'un dessin la plus exacte possible au client (moi)

- Spécifieurs et Artistes sont **séparés physiquement** (3 groupes de 6 avec 3 spécifieurs et 3 artistes)

- L'objectif est pour les artistes de reproduire aussi fidèlement que possible un dessin à partir des spécifications fournies par les spécifieurs de l'équipe.
    - Les spécifieurs sont les seuls à avoir accès au dessin.
    - La communication entre spécifieurs et artistes est uniquement écrite.
    - Les spécifieurs ne peuvent pas dessiner.
    - Les artistes ne peuvent pas se déplacer.
    - Les Spécifieurs peuvent communiquer oralement entre-eux, de même pour les Artistes.

- 3 phases de jeu de 10 minutes

- Le reste de l'organisation spécifieurs/artistes est libre : nombre d'allers-retours, format de communication écrite, attente d'un spécifieur aux côtés des artistes, etc.

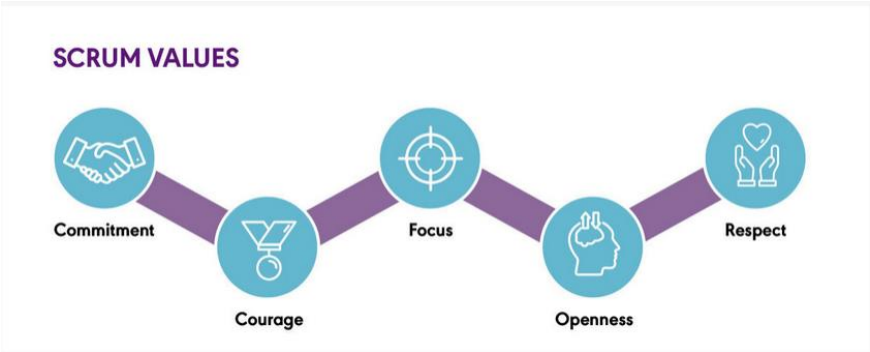
6-Scrum

https://www.youtube.com/watch?v=3qMpB-UH9kA

16 minutes

Piliers
Transparence. Ex: définition of done
Inspection. Ex: se consulter quotidiennement
Adaptation. Ex: Ajuster en permanence

Valeurs
Focus
Ouverture
Engagement
Respect
Courage



Idées clés
Le client au cœur du produit
Esprit d'équipe
La communication est la clé
Simplicité, efficacité et qualité
Flexibilité aux changements
Avancement basé sur du concret


SCRUM PROCESS
Scrum Master
Product Owner
Team
Daily Stand Up
24 H
Sprint 1-4 Weeks
Sprint Review + Sprint Retrospective
Product Backlog
Sprint Planning Meeting
Sprint Backlog
Finished Work

Acteur Product Owner

Rôle

Porter la vision du produit à réaliser
Gérer le backlog
Représentant du client

Droits

Décider de ce qui est fait ou non dans une version

Devoirs

Etre au contact régulier des utilisateurs
Participer au évènements
Etre disponible durant toute l'itération
Décider opérationnellement

Acteur: La Team

Rôle

Entre 3 et 9 personnes: pluridisciplinaire
Toutes les compétences nécessaires (dev, testeurs, architectes, analystes,…)
Etablie par le mangement, stable dans le temps, co-localisée

Droits

Etre protégée des interventions extérieures lors des itérations
Définir comment réaliser le produit
Refuser des développements trop lourds ou pas assez bien décrits

Devoirs

Estimer collectivement les tâches avant itération
S'engager collectivement sur l'estimation et la gestion des problèmes
Etre sincère sur l'avancement
Produire en respectant les engagements

Acteur: Scrum Master

Rôle

Garant de la bonne application de la méthode
Rôle de management non hiérarchique
Servant leader: Il facilite la réalisation du produit (maximiser ROI)

Droits

Parler à la hiérarchie des interférences empêchant l'équipe

Devoirs

Animer l'équipe, maintenir rythme soutenable
Mettre à jour l'avancement
Aider l'équipe à résoudre les problèmes
S'assurer de la démarche d'amélioration
Surveiller le back log
Aider le Product Owner

Les artefacts Scrum dans Agile sont des informations qu'une équipe Scrum et les parties prenantes utilisent pour détailler le produit en cours de développement, les actions à mettre en place pour y parvenir ainsi que les mesures prises durant le projet.

Le terme d'artefact est souvent associé aux sites archéologiques et aux reliques. Pourtant, dans le domaine du développement logiciel, le terme artefact fait référence aux informations clés nécessaires durant le développement d'un produit.

En scrum, nous définissons ainsi 3 artefacts classiques et 1 artefact de transparence :

- product backlog
- sprint backlog
- increment
- definition of done (artefact de transparence)

Le product backlog

Le product backlog est l'ensemble des besoins recueillis pour créer le produit désiré :

- fonctionnalités
- fonctions
- exigences
- améliorations
- correctif

Le sprint backlog

Le sprint backlog constitue l'ensemble des éléments qui ont été sélectionnés en démarrage de sprint (en sprint planning) avec pour but :

de répondre à l'objectif du sprint

d'incrémenter le produit de nouvelles choses :

- de nouvelles fonctionnalités
- des pré-requis
- de nouvelles fonctions
- des fixes
- des améliorations

le plan pour livrer les éléments sélectionnés (sous-tâches techniques, priorisation…)

L’incrément

L’incrément en scrum représente l’ensemble des éléments « done » du sprint en cours en plus de ceux déjà finalisé dans les sprint précédents.

Le Definition of Done

L'équipe de développement va définir ensemble l'ensemble des critères qui permettront d'affirmer qu'un item (user story par exemple) peut être considéré comme « done ». Nous utilisons couramment l'acronyme Dod pour définir cette pratique.

Cérémonies

Sprint

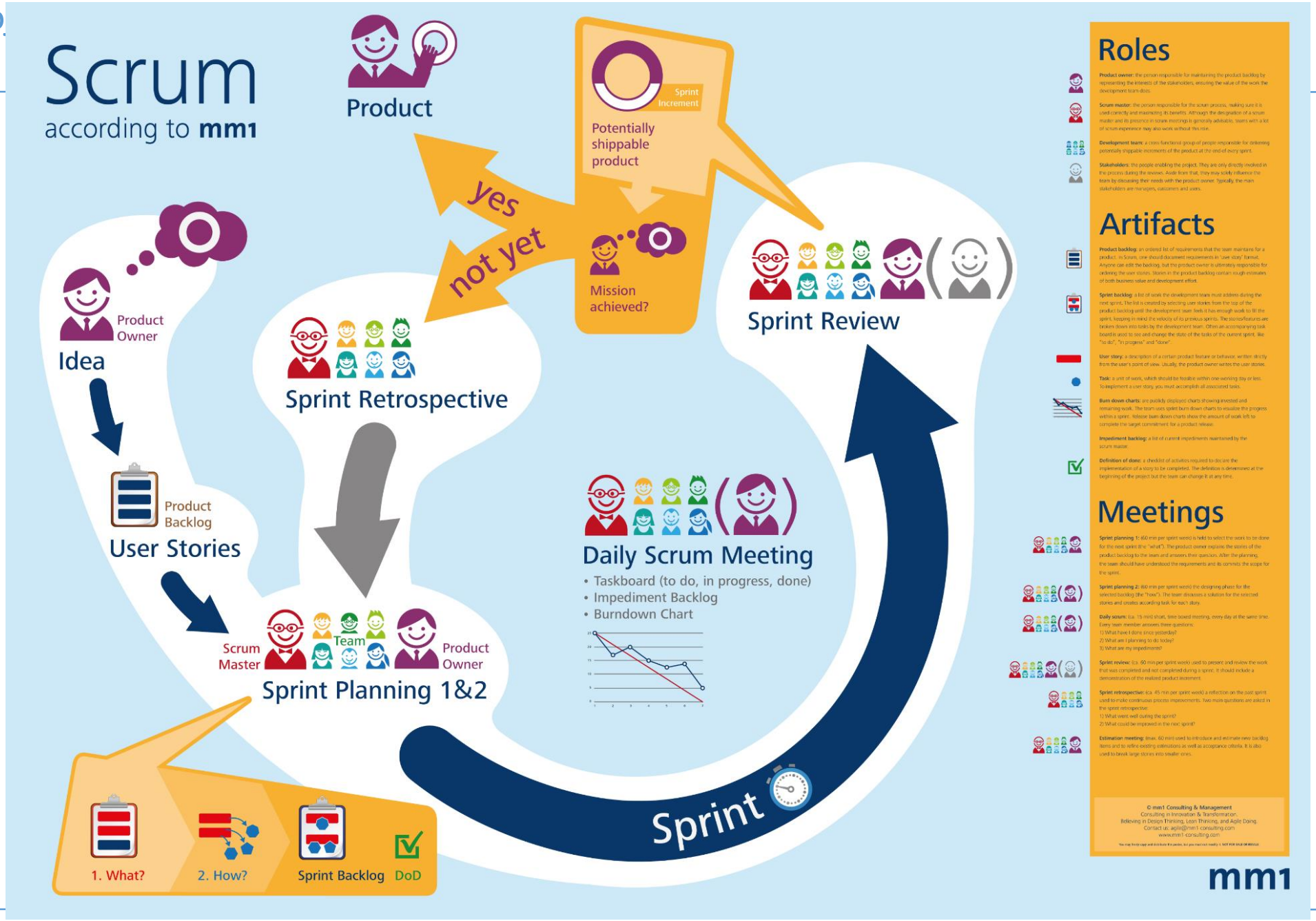
Sprint Planning

Daily Scrum

Sprint Review

Sprint Retrospective

Backlog Refinment


Scrum
according to mm1
Product
Sprint Increment
Potentially shippable product
yes
not yet
Mission achieved?
Sprint Review
Sprint Retrospective
Product Owner
Idea
Product Backlog
User Stories
Scrum Master
Team
Product Owner
Sprint Planning 1&2
Daily Scrum Meeting
• Taskboard (to do, in progress, done)
• Impediment Backlog
• Burndown Chart
Sprint
1. What?
2. How?
Sprint Backlog
DoD
Roles
Artifacts
Meetings
mm1

Chaque Sprint démarre avec un Sprint Planning lors duquel l'équipe doit définir :
what : l'objectif du Sprint ou Spring Goal,
how : puis comment atteindre cet objectif.

What : Sprint Planning 1 et définition du Sprint Goal
La Scrum Team se réunit en entier *(Product Owner, Development Team et Scrum Master)* pour définir le Sprint Goal.
1.Le Product Owner **décrit les User Stories** qu'il souhaiterait assigner au Sprint.
2.La Development Team **peut interroger le Product Owner** pour mieux comprendre le besoin.
3.Si une User Story est de taille importante, il s'agit alors d'une Epic et la Scrum Team peut alors décider de la décomposer en User Stories plus fines.
4.La Development Team **évalue** et **sélectionne à elle seule** les User Stories qu'elle **pense être capable de finir** d'ici la fin du Sprint.

How : Sprint Planning 2
Suite au Sprint Planning 1, la Scrum Team peut libérer le Product Owner.

La Development Team **réfléchit et présente la conception / architecture / design de chaque User Story**.
Cela peut se faire avec toute l'équipe ou par paire afin de paralléliser la réflexion si cela est vraiment nécessaire.
Dans ce cas, la paire doit présenter le résultat de sa réflexion au reste de l'équipe.

Chaque User Story est ensuite **décomposée en tâches** techniques.
Les membres de la Development Team doivent ensuite **se répartir naturellement** ces tâches.

Estimation de l'Effort
L'**effort** nécessaire pour réaliser chaque User Story est évalué avec un nombre de **points** suivant généralement une suite de Fibonacci *(1, 2, 3, 5, 8 du plus simple au plus complexe)*. En pratique, les valeurs 1, 2 et 3 suffisent et encouragent un découpage plus fin.

Planning Poker
Pour encourager tous les membres de la Development Team à participer à l'estimation de l'effort associé aux User Stories, l'équipe peut utiliser un jeu de cartes grâce auquel chacun réfléchit à son estimation puis tout le monde dévoile sa carte simultanément.

Répartition des Tâches
De nombreuses équipes ont le réflexe d'associer chaque tâche à l'expert du sujet.
Bien que cette approche puisse paraître optimale, cela :
•**décourage le partage** de connaissances au sein de l'équipe,
•**encourage le travail individuel** plutôt que l'appropriation du produit par l'équipe,
•**crée une dépendance forte** envers certaines personnes dans l'équipe,
•**n'encourage pas la remise en question** et aboutit souvent à de la **complexité artificielle**.

Daily Scrum

Le Stand-Up Meeting *(ou Daily Scrum)* est **un point informel** qui se tient tous les jours debout et à la même heure *(autour d'un café par exemple)*.

En présence du Scrum Master, durant le Stand-Up Meeting chaque membre de l'équipe aborde les points suivants :

- Ce qu'il a fait depuis le dernier Stand-Up Meeting.
- Ce qu'il prévoit de faire en attendant le prochain Stand-Up Meeting.
- **Les difficultés rencontrées**.

Objectifs

•d'encourager la **communication** dans l'équipe,
•**d'éviter l'isolement** des développeurs et donc les blocages et la complexité artificielle,
•de **déceler rapidement** des difficultés potentielles,
•de détecter **quand un membre de l'équipe à besoin d'aide**,
•de **décider de changer de paires** dans le cas du Pair Programming,
•de **propager la connaissance** dans l'équipe,
•de **réfléchir rapidement et en groupe** à des besoins de conception, d'architecture,
•d'éviter d'avoir à organiser des réunions supplémentaires dans la journée.

Sprint Review
A la fin de chaque Sprint, la Development Team, le Product Owner et le Scrum Master se réunissent.
Le(s) client(s) et les développeurs d'autres équipes peuvent être invités.
Comme les autres évènements d'un Sprint, **le Sprint Review doit rester informel** *(pas de slides etc...)*.

Objectifs
•Présentation *(par le Product Owner)* des **User Stories finalisées** et **celles qui ne le sont pas**.
•Présentation *(par la Development Team)* des **difficultés rencontrées** et des solutions adoptées.
•Récolte de "feedback" de tout le monde *(membres de l'équipe, client, autres équipes etc...)*.
•Présentation du Backlog et des dates projetées des releases à venir si nécessaire *(Cf. vélocité)*.
•Reflexion et sélection des objectifs futurs.
•Les User Stories non finalisées ne sont jamais présentées pendant le Sprint Review.
•L'une des principales raisons et que certaines personnes extérieures à l'équipe pourraient forcer la livraison ou "vendre" une User Story non finalisée.

Sprint Retrospective
Le Sprint Retrospective est **une réflexion autour du Sprint passé**.

Objectifs
Analyser le déroulement du Sprint passé avec un focus sur :
les **personnes** *(ex. : détecter la démotivation et en analyser l'origine)*,
les **relations** *(ex. : détecter les conflits, analyser les paires qui fonctionnent le mieux)*,
les **process** *(ex : déceler les faiblesses des process actuels ou ceux qui nuisent à la Developer eXperience)*
et les **outils** *(ex : est-ce que les outils utilisés sont adaptés et est-ce qu'ils sont utilisés convenablement ?).*

Analyser les indicateurs, **identifier** et **ordonner** ce qui s'est bien déroulé et les axes d'amélioration.
Définir un plan d'amélioration pour **réparer ce qui est cassé** dans le Scrum.

Backlog Refinement

Le Backlog Refinement *(ou Backlog Grooming)* regroupe toute la Scrum Team *(Development Team, Scrum Master et Product Owner)* afin de donner de la visibilité sur les Sprints à venir et le travail restant.

Objectifs

Clarifier et décomposer les Epics.

Evaluer ou réévaluer l'effort nécessaire pour réaliser des User Stories.

Retirer les User Stories superflues.

Ajouter de nouvelles User Stories.

Le Backlog Refinement peut avoir lieu une à deux fois par semaine *(et non par Sprint)*.

Si la Development Team finalise toutes les User Stories avant la fin du Sprint, le Backlog Refinement permet d'ajouter de nouvelles User Stories au Sprint Goal du Sprint actuel.

User stories
« En tant que [persona], je [souhaite que] [afin de] ».

User stories
« En tant que [persona], je [souhaite que] [afin de] ».

« En tant que [persona] » : pour qui développons-nous cette fonctionnalité ? Nous ne cherchons pas seulement l'intitulé d'une fonction, mais aussi le persona de cette personne. Notre équipe devrait avoir une compréhension commune de l'identité du persona. Il faut espérer que nous nous sommes entretenus avec beaucoup de « persona ». Nous comprenons comment cette personne travaille, comment elle pense et ce qu'elle ressent. Nous avons de l'empathie pour elle.

« Souhaite que » : c'est ici que nous décrivons l'intention de persona, et non les fonctionnalités qu'il utilise. Qu'essaie-t-il de faire réellement ? Cet énoncé ne devrait pas impliquer d'implémentation. Si vous décrivez une composante de l'interface utilisateur et non l'objectif de l'utilisateur, vous êtes hors sujet.

« Afin de » : comment son désir immédiat de faire quelque chose s'intègre-t-il à la vue d'ensemble ? Quel avantage global cette personne essaie-t-elle d'obtenir ? Quel est le principal problème à résoudre ?

User stories: exemples
« En tant que [persona], je [souhaite que] [afin de] ».

En tant que Max, je souhaite inviter mes amis afin de pouvoir profiter de ce service ensemble.

En tant que Sasha, je souhaite organiser mon travail afin d'avoir la sensation de mieux maîtriser les choses.

En tant que responsable, je souhaite être en mesure de comprendre l'avancement de mes collègues afin de pouvoir mieux parler de nos réussites et de nos échecs.

En tant que chef de produit, je souhaite que les membres d'équipe puissent comprendre en quoi leurs tâches individuelles contribuent aux objectifs commerciaux globaux afin de booster leur motivation.

En tant que client récurrent, je souhaite que mes informations soient sauvegardées afin que mon expérience de paiement soit plus fluide.

En tant qu'utilisateur régulier de l'application, je souhaite consulter les informations pertinentes le plus vite possible.

User stories: méthode 3C
« En tant que [persona], je [souhaite que] [afin de] ».

Les 3C signifient **C**arte, **C**onversation, et **C**onfirmation. Cette méthode permet de classer chaque user story en fonction de trois points de référence, de façon à suivre un processus plus organisé.

Carte: histoire courte, 1 ou 2 phrases et peut être écrite sur une carte de 13*8 cm

Conversation: les détails de l'histoire sont discutés par les équipes avec le métier, les ergonomes, les analystes et concepteurs

Confirmation: l'histoire est confirmée par des tests d'acceptation rédigés au même moment que celle-ci, au dos de la carte par exemple

User stories: méthode Invest
« En tant que [persona], je [souhaite que] [afin de] ».

I - Independent: Indépendance pour faciliter le traitement

N - Negotiable: Elle est négociée, dicutée dès les réunions d'estimation et de planification de l'itération

V - Valuable: Elle est source de valeur pour le Client final ou l'utilisateur

E - Estimable: Elle est estimée par les équipes de développement. Une estimation relative, c'est-à-dire les unes par rapport aux autres, en story points.

S – Sized Appropriately or Small: Le plus souvent petite car susceptible d'être traitée (livrée et testée) par l'équipe sur une seule itération de 2 à 3 semaines

T – Testable: Une User Story « de qualité » est avant tout testable, déjà dans sa forme et surtout dans le sens où les critères d'acceptation sont envisagés d'entrée

6-Projet

Contexte du PDG: société **Pizza New** veut innover en proposant des commandes de pizzas sur la base d'ingrédients
Principe: le client choisit des ingrédients et le site web propose des pizzas contenant ces ingrédients qu'il peut commander. Il peut aussi créer sa propre pizza.

Organisation: 4 groupes de 6 (pas les mêmes que vos groupes habituels). Marseille, Fonsorbes, Bayonne, Toulouse

1 product owner (en lien avec la stratégie du PDG)
1 scrum master
1 stakeholder (responsable du magasin (Marseille, Fonsorbes, Bayonne, Toulouse)
3 development team

J1: equipe et backlog / J2: iteration 1 / J3: iteration 2 / J4: documentation et restitution

Evaluation projet
-Dossier de synthèse (10pts - groupe): cadrage projet, détail des itération (backlog, expression de besoin des users stories, planification, risques, suivi global du projet, arbitrage des choix (on fait / on fait pas), architecture technique envisagée, maquette des écrans et des livrables, maintenance envisagée. Qualité de communication du document
-Fiche de bilan (5pts – individuel): bilan de votre participation dans votre rôle. Avantages et inconvénients d'avoir travaillé dans ce mode projet. 2 pages A4.
-Communication orale (5pts – individuel): capacité à présenter son action de manière claire et fluide

Itération 1 –
Faire un CR du planning poker
Montrer les réalisations du sprint backlog
Bilan oral de chaque participant dans le groupe