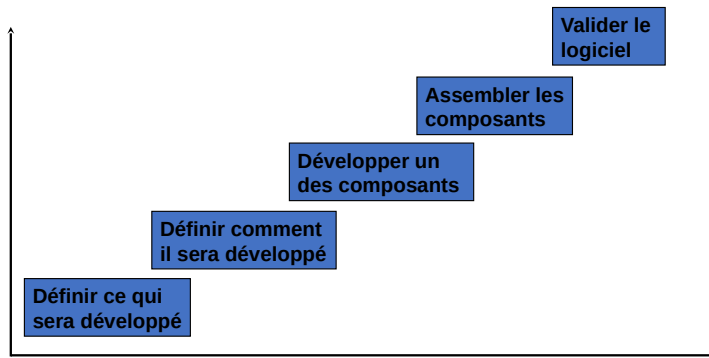


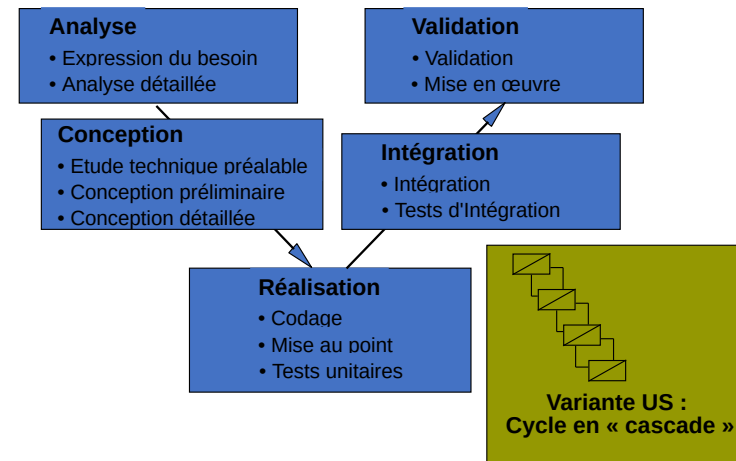


Pr. Jean-Marc Jézéquel
IRISA – Université de Rennes
Campus de Beaulieu
F-35042 Rennes Cedex
e-mail : jezequel@irisa.fr
<http://people.irisa.fr/Jean-Marc.Jezequel>
X @jmjezequel

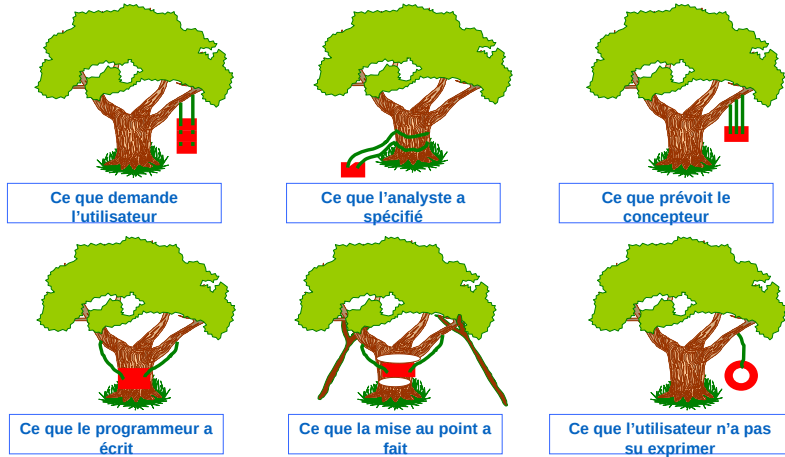
1 - Introduction



- L'organisation de ces activités et leur enchaînement définit le *cycle de développement* du logiciel



Problèmes avec le processus classique...



Problèmes du processus classique

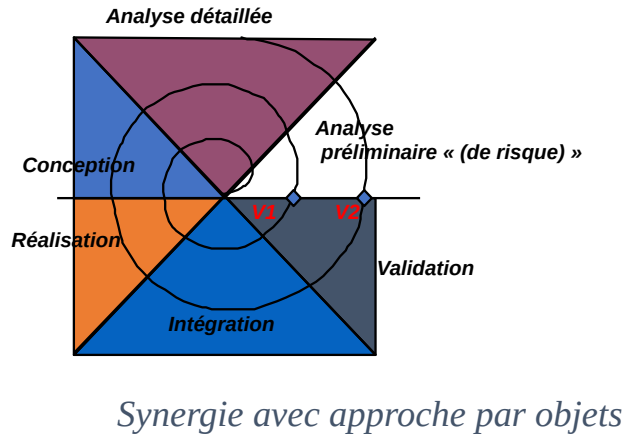
- Organisation « industrielle » héritée du XIX^{ème} siècle
 - rassurant pour les managers
 - hiérarchie malsaine dans les rôles
 - antinomie : Coplien 's organizational pattern
 - *Architects Also Implement*
- cycle management <> cycle développement
- linéarité implicite
 - temps d 'approbation des documents => effet tampon
 - coût de la (non-) modification d 'un document « final »
 - irréaliste pour un projet innovant, donc à risques

Causes principales d'échecs des projets

- Des objectifs de projet irréalistes ou non articulés
- Estimation inexacte des ressources nécessaires
- Des exigences système mal définies
- Mauvaise communication de l'état d'avancement du projet
- Risques non gérés
- Mauvaise communication entre les clients, les développeurs et les utilisateurs
- Utilisation d'une technologie immature
- Incapacité à gérer la complexité du projet
- Des pratiques de développement peu rigoureuses
- Mauvaise gestion des projets
- Jeux politiques des parties prenantes
- Pressions commerciales

2 – Processus itératifs

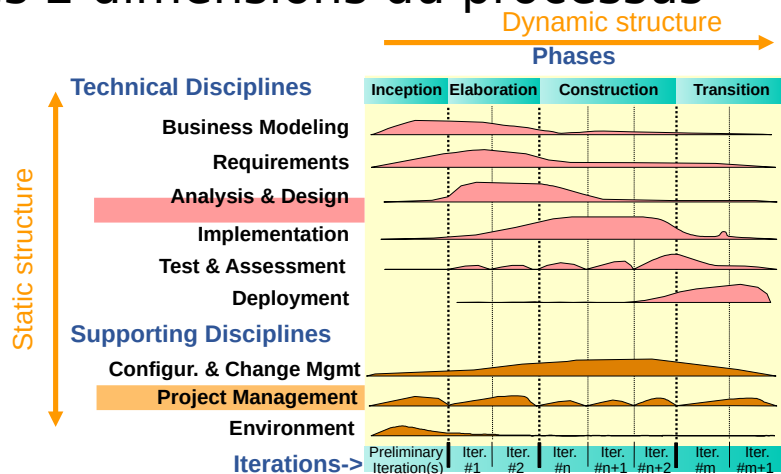
Barry Boehm, 1985



Intérêts du cycle de vie en « spirale »

- Bien adapté au développements innovants
 - les progrès sont tangibles : c'est du logiciel qui « tourne » et pas seulement des kilos de documents
 - possibilité de s'arrêter « à temps », i.e. avant que l'irréalabilité du projet ait créée un gouffre financier
- Moins simple à manager
 - difficile à gérer en situation contractuelle
 - mal contrôlé => on retombe dans le *hacking*
- Production des incréments asservie sur 2 parmi 3 :
 - période (e.g. release toutes les 2 semaines)
 - fonctionnalités (releases découpés suivant use-cases)
 - niveau de qualité (problème de la mesure)

Les 2 dimensions du processus



Phases du développement itératif



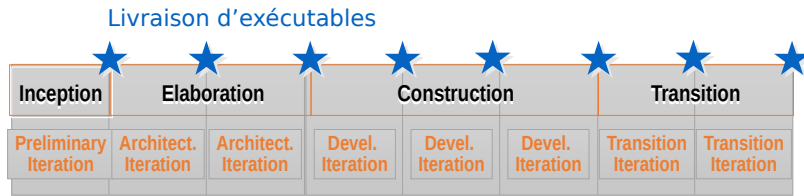
Inception: définition de la portée du projet

Elaboration: planification du projet, spécification des fonctionnalités, architecture de base

Construction: réalisation du produit

Transition: transfert du produit vers les utilisateurs

Itérations



Une **itération** est une séquence d'activités avec un plan bien établi et un critère d'évaluation, résultant en la livraison d'un logiciel *exécutable*.

Phases et itérations : 2 exemples

- Petit projet de commerce électronique
 - Intégration à un mainframe
 - 5 personnes
- Grand projet d'infrastructure
 - Gros travail d'architecture nécessaire
 - 20 personnes

	No. of Iterations				Project Length	Iteration Length
	Inception	Elaboration	Construction	Transition		
e-business	0.2	1	3	1	3-4 months	2-3 weeks
infrastructure	1	3	3	2	9-12 months	5-7 weeks

3 – Méthodes agiles

Exemple de Scrum

Points clés de l'Agilité

- Développer seulement ce qui est nécessaire
- Minimiser la paperasserie
- flexibilité
 - besoins, plan, utilisation des ressources, etc...
- Apprendre de ses erreurs précédentes
- Réévaluer les risques régulièrement
- Établir des critères de progrès
 - objectifs et mesurables
- Automatiser

Scrum

- Scrum est une **méthode agile** pour la gestion **itérative** de projet
 - exploitant la créativité des membres de l'équipe de développement.
- On peut distinguer trois phases :
 - La phase initiale au cours de laquelle les fonctionnalités du système sont listées et une architecture logicielle générale est définie
 - Suit une série de "sprints", chaque sprint correspondant à un incrément du système
 - La phase de terminaison du projet développe les derniers artefacts (manuel d'utilisation ...) et tire les leçons apprises durant le développement.

Terminologie Scrum (1/2)

Terme Scrum	Definition
Équipe de développement	Un groupe auto-organisé de développeurs de logiciels, qui ne devrait pas dépasser 7-10 personnes, responsables du développement du logiciel et d'autres documents essentiels du projet.
Incrément de produit potentiellement livrable	Incrément logiciel fourni à partir d'un sprint. « potentiellement livrable » signifie qu'il est dans un état fini et qu'aucun travail supplémentaire, tel que des tests, n'est nécessaire pour l'incorporer dans le produit final.
Product backlog	Liste d'éléments à faire auxquels l'équipe Scrum doit s'attaquer. Il peut s'agir de définitions de fonctionnalités pour le logiciel, d'exigences logicielles, de récits utilisateur (<i>user stories</i>) ou de descriptions de tâches supplémentaires nécessaires, telles que la définition de l'architecture ou la documentation utilisateur.
Product owner	Une personne (ou éventuellement un petit groupe) dont le travail consiste à identifier les caractéristiques ou les exigences du produit, à les hiérarchiser pour le développement et à examiner en permanence le backlog du produit pour s'assurer que le projet continue de répondre aux besoins critiques de l'entreprise. Le Product Owner peut être un client, mais aussi un chef de produit dans une société de logiciels ou un autre représentant des parties prenantes.

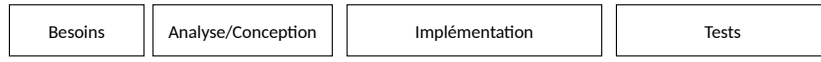
Terminologie Scrum (2/2)

Terme Scrum	Definition
Scrum	Une réunion quotidienne de l'équipe Scrum qui passe en revue les progrès et hiérarchise le travail à faire ce jour-là. Idéalement, il devrait s'agir d'une courte réunion en face à face qui inclut toute l'équipe.
Scrum Master	Le Scrum Master est chargé de s'assurer que le processus Scrum est suivi et guide l'équipe dans l'utilisation efficace de Scrum. Il est responsable de l'interface avec le reste de l'entreprise et de s'assurer que l'équipe Scrum n'est pas détournée par des interférences extérieures. Les développeurs Scrum sont catégoriques sur le fait que le Scrum Master ne doit pas être considéré comme un chef de projet. D'autres, cependant, ne trouvent pas toujours facile de voir la différence.
Sprint	Une itération de développement. Les sprints durent généralement de 2 à 4 semaines.
Velocity	Estimation de l'effort de backlog produit qu'une équipe peut couvrir en un seul sprint. Comprendre la vitesse d'une équipe l'aide à estimer ce qui peut être couvert dans un sprint et fournit une base pour mesurer l'amélioration des performances.

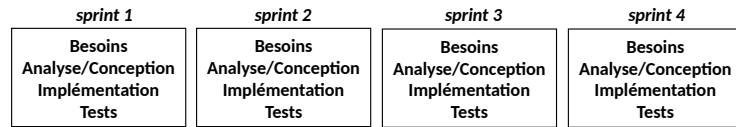
Processus logiciel SCRUM

- Une phase d'idéation
- Une succession de sprints de taille fixe
- Chaque *sprint* produit un logiciel fonctionnel
- Chaque *sprint* réalise toutes les activités de développement logiciel
- Les *sprints* sont groupés en *Releases* (de taille fixe)

Processus logiciel SCRUM (*sprints* et *releases*)

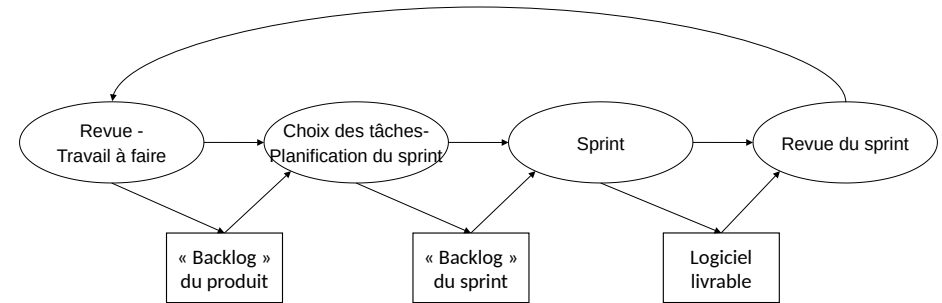


Cycle séquentiel



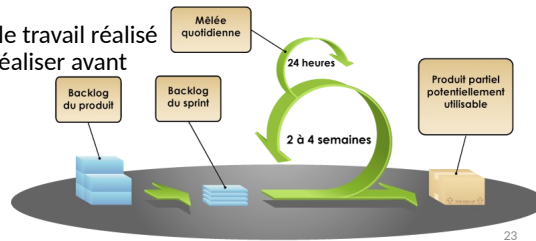
Cycle SCRUM d'une release

Cycle de vie d'un sprint SCRUM



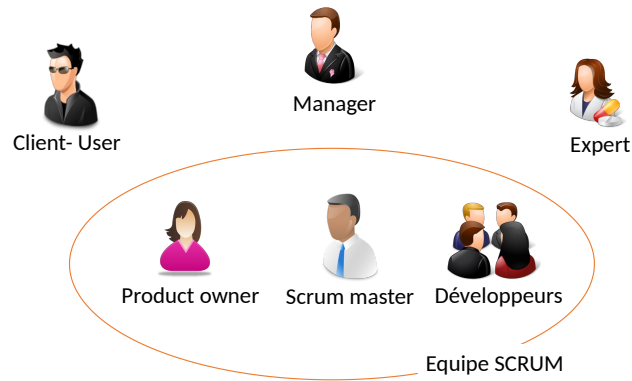
Sprint

- Chaque release est décomposée en *sprints* au cours de l'activité de planification de la release (la première fois « sprint zéro »)
- Un sprint est une période de développement de taille fixe
 - Durée fixe (en moyenne 2 à 4 semaines), équipe stable
- Un sprint se décompose en un ensemble de tâches
 - **Backlog**: carnet, liste ordonnée de « choses » à faire (*user stories*, tâches)
- Mêlée journalière (Scrum)
 - réunion d'équipe pour faire le point sur le travail réalisé depuis le début du sprint et le travail à réaliser avant la fin du sprint
- Chaque sprint termine par :
 - une revue du produit
 - une rétrospective sur le processus



4 - Les acteurs de Scrum

Rôles



L'équipe

- Composition:
 - 1 Product Owner
 - 1 Scrum Master
 - 2 à 7 développeurs
 - Le product owner et le Scrum master peuvent aussi prendre le rôle de développeur
- Principes
 - Auto-organisation
 - Pluridisciplinarité
 - Stabilité
 - Valeurs communes

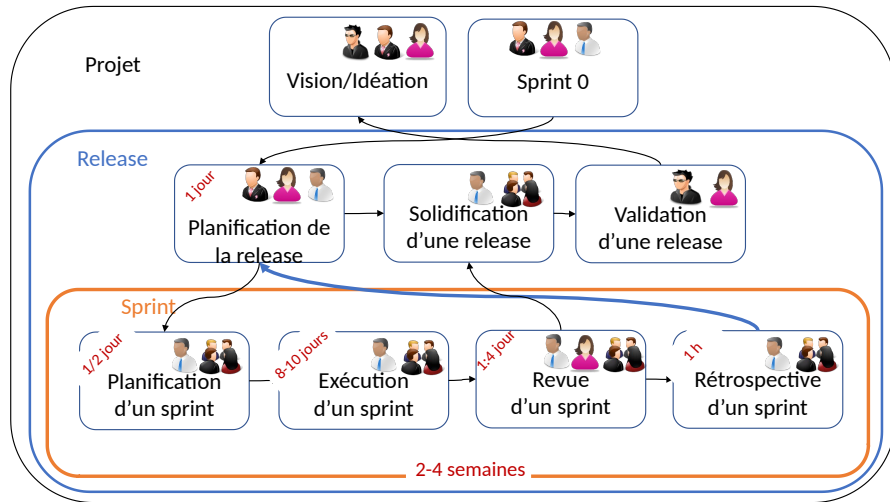
Product owner

- Responsabilités
 - Fait partager la vision globale du produit
 - Gère le backlog du produit (liste ordonnée des « choses » à faire)
 - Définit les priorités
 - Accepte ou rejette les *Releases* (livrables)

Scrum master

- Responsabilités
 - n'est pas « le chef », mais un facilitateur
 - Motive l'équipe
 - Fait appliquer les bonnes pratiques de Scrum
 - Gère les obstacles

Processus logiciel Scrum



ISTIC- L2GEN



Client- User



Manager



Product owner



Scrum master



Développeur

29

5 - Les objets de Scrum

Feature, Story, Tâche, Backlog

Feature

- Une *feature* (*fonctionnalité*) est un service ou une fonctionnalité du produit à développer
- Elle se décompose en *stories* (histoires) de tailles différentes. On distingue :
 - les *stories* complexes (épopées) qui seront affinées en *stories* plus simples
 - Les *stories* atomiques qui ne se décomposent pas et sont réalisées dans les *sprints*

ISTIC- L2GEN

31

Story

- Une *story* est une exigence du système à développer, formulée en une ou deux phrases dans le langage de l'utilisateur.
- Les *Stories* émergent au cours d'ateliers de travail menés avec le Métier, le Client et/ou les Utilisateurs.
- On distingue :
 - *Story* fonctionnelle
 - *Story* technique
 - Correction de bug
 - Remboursement de la « dette technique »

ISTIC- L2GEN

32

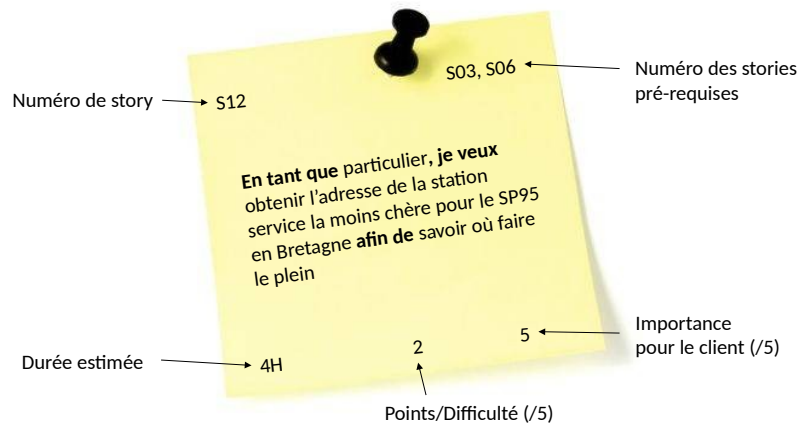
Story

- Les 3C
 - Carte : l'histoire est courte et sa description tient sur une carte (demi-page)
 - Conversation : l'histoire est définie avec les gens du métier
 - Confirmation : l'histoire est confirmée par des tests d'acceptation rédigés au même moment que celle-ci
- Workflow de la story
 - Idée d'une story rédigée sur une Carte (1/2 feuille)
 - Conversation dirigée par le *product owner* qui inclut les gens du métier
 - L'équipe apporte sa Confirmation que la story est prête
 - L'équipe réalise la story
 - Le *product owner* apporte sa Confirmation que la story est finie

Description type d'une story

- Plan type
 - **En tant que** <acteur>, **je veux** <un but> [**afin de** <une justification>]
 - **En tant que** client, **je veux** pouvoir accéder au site de ma banque **afin de** gérer mon compte sur Internet
- Priorité
- Nombre de points
 - Représente le niveau de difficulté intrinsèque, en fonction de la taille et de la valeur métier
 - Corrélées en moyenne, mais pas toujours (en fonction d'une forte valeur métier, de la réutilisation possible de composants ...)
- Conditions d'acceptation
 - **Etant donné** <le contexte> **quand je** <événement> **alors** <résultat>
 - **Etant donné** que je suis sur la page de connexion et que j'ai entré un login et un mot de passe dans le formulaire et que le login et le mot de passe correspondent à un utilisateur enregistré, **quand je** clique sur le bouton "Se connecter" **alors** j'arrive sur la page d'accueil du site.

Post-it de story



Tâche

- A l'exécution, une story se décompose en tâches

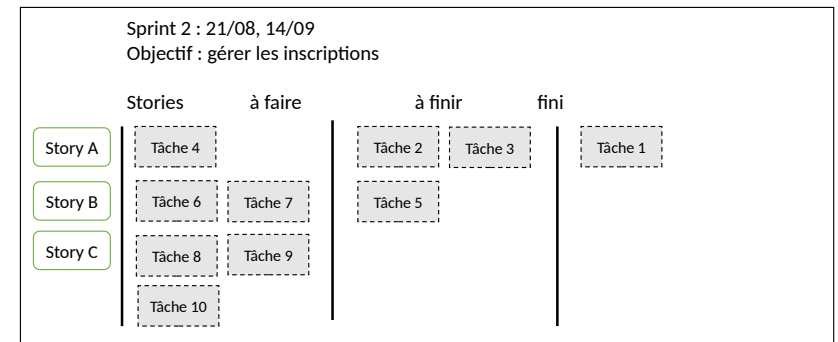
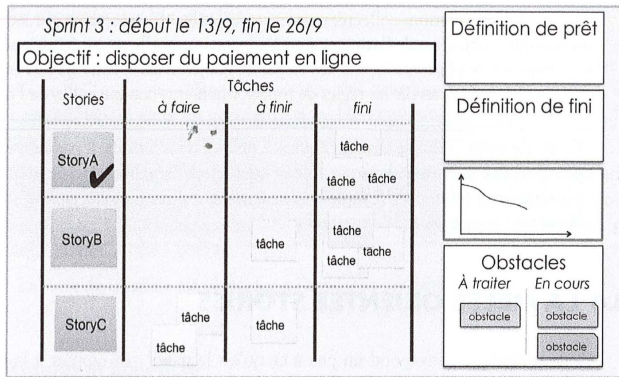


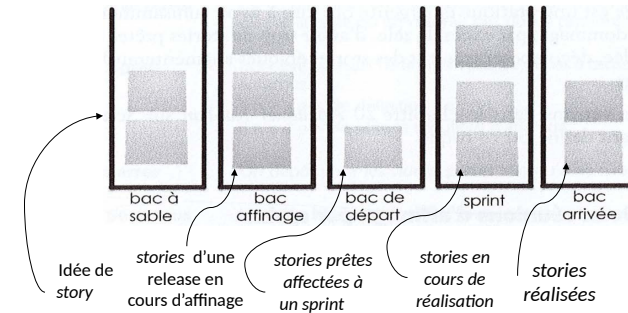
Tableau de la Story

- Le tableau de la story décrit son état.



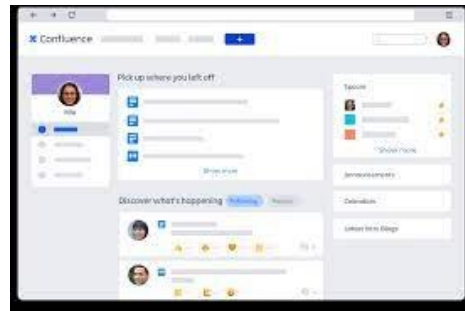
Le backlog (carnet de route)

- Liste ordonnée des choses (stories) à faire
- En pratique, plusieurs (sous-)backlogs
 - On peut distinguer le backlog de produit et le backlog de sprint

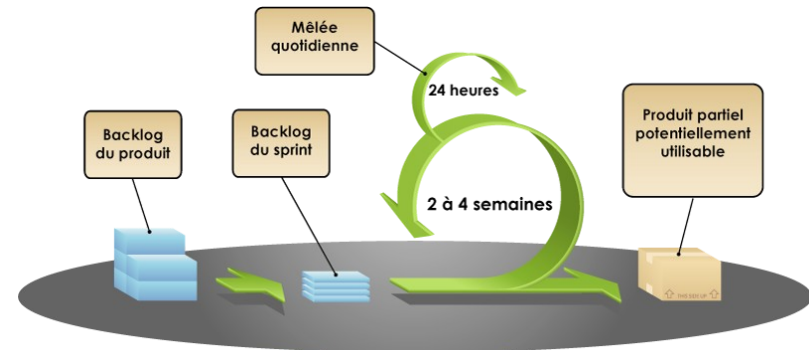


Outils SCRUM

- Jira, Confluence...



Backlog et Sprint

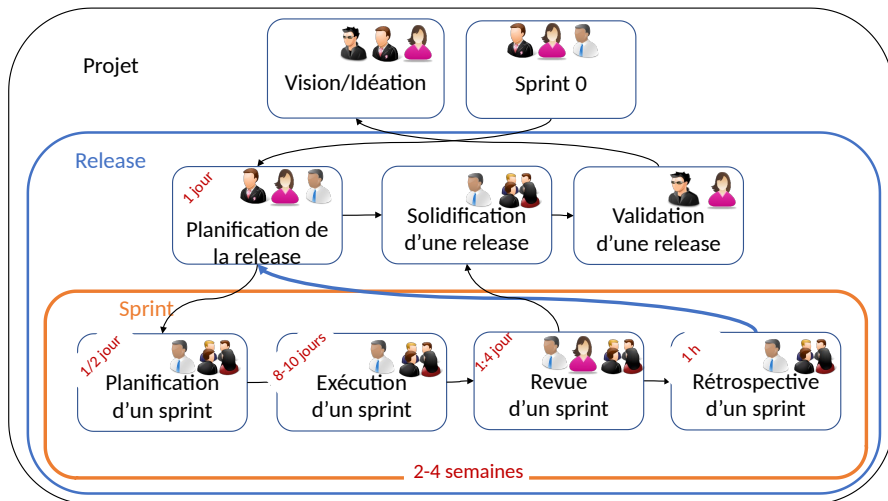


La mêlée quotidienne (Scrum)

- Courte : ~15mn
- Bilan: mise à jour du tableau de la story
 - Qu'est-ce que j'ai fait hier ?
 - Qu'est que je vais faire aujourd'hui ?
 - Quels sont les obstacles que j'ai rencontrés ?
- Objectif :
 - Rythmer le sprint (stories finies, prêtes)
 - Recenser les obstacles

6 – Gestion de projet Scrum

Processus logiciel Scrum



- Client- User
- Manager
- Product owner
- Scrum master
- Développeur

Idéation



- Définition d'une vision commune
 - Identification des *features*
 - Impact mapping
 - Identification des parties prenantes
 - Acteurs
- Création d'un *backlog* de haut niveau du produit
 - Tableau ordonné des features
 - (éventuellement Story map haut niveau)

Impact mapping

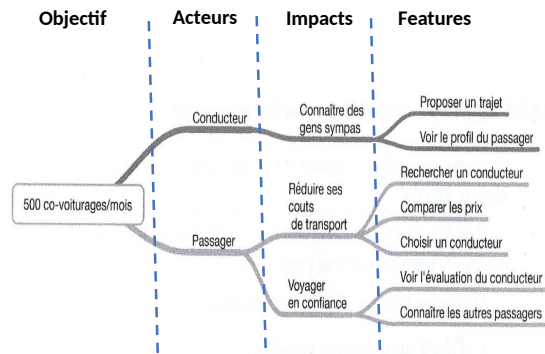


Figure 14.3 – Une carte qui donne la vision pour un site de co-voiturage

Sprint « Zéro »

- Affinage du backlog
 - Story mapping (feature stories)(partiel)
 - Description des stories (description, conditions d'acceptation)
 - Ordonnement des stories
 - Planification de la première release
 - Approvisionnement du bac d'affinage
 - Approvisionnement du bac de départ du sprint 1
- Peut durer plusieurs journées

Workflow d'une feature

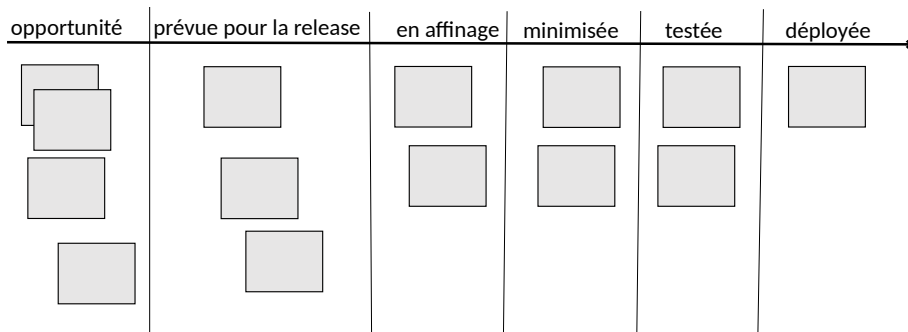


Tableau des features

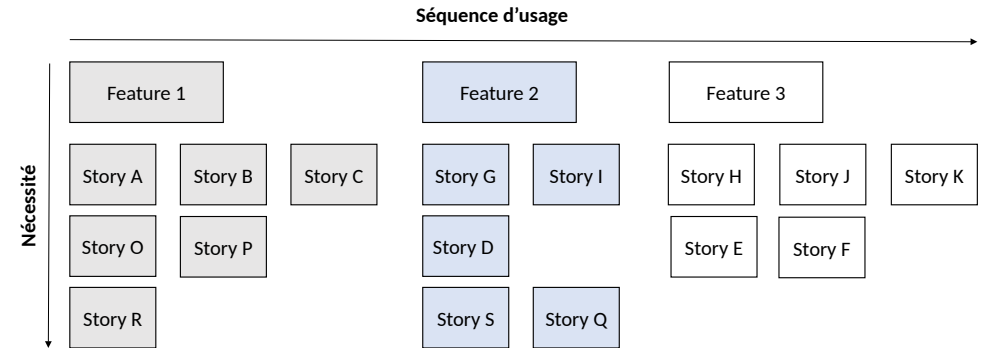
Workflow d'une feature

- Opportunité : idée de feature qui présente une opportunité pour le produit
- Prévue pour la release : l'étude d'opportunité a abouti, et la feature est prévue pour la release en cours
- En affinage : initialement sous forme d'épique, est décomposée en stories
- Minimisée : *Minimal Marketable Feature*
- Testée : ... avec des stories finies
- Déployée : utilisable par des utilisateurs qui peuvent fournir du feedback

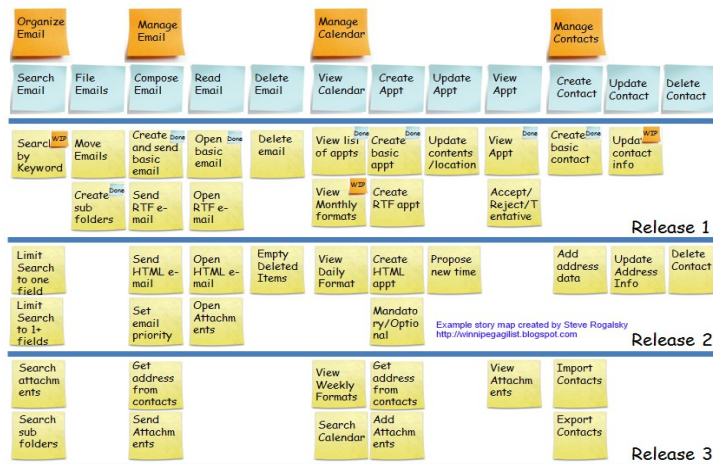
Story mapping

- Ordonnancement des features
- Décomposition en stories
- (Organisation des releases)

Story map : décomposition d'une feature en stories



Exemple



Estimation des points d'une story

- Difficulté intrinsèque
 - Taille et valeur métier
 - Corrélées en moyenne, mais pas toujours (en fonction d'une forte valeur métier, de la ré-utilisation de composants ...)
- Planning Poker (source Wikipedia)
 - Les participants s'installent autour d'une table, placés de façon que tout le monde puisse se voir.
 - Le responsable de produit explique à l'équipe un scénario utilisateur (*user story*).
 - Les participants posent des questions au responsable de produit, discutent du périmètre du scénario, évoquent les conditions de satisfaction qui permettront de le considérer comme "terminé".
 - Chacun des participants évalue la complexité de ce scénario, choisit la carte qui correspond à son estimation et la dépose, face vers le bas, sur la table devant lui.
 - Au signal du facilitateur, les cartes sont retournées en même temps.
 - S'il n'y a pas unanimité, la discussion reprend.
 - On répète le processus d'estimation jusqu'à l'obtention de l'unanimité.
 - Une procédure optimisée consiste, après la première "donne", de demander aux deux acteurs ayant produit les évaluations extrêmes d'expliquer leurs points de vue respectifs. Ces explications achevées et comprises de tous, une nouvelle estimation est produite et c'est alors la moyenne arithmétique de ces estimations qui est prise en compte.

Planification des *releases*

- Affiner les risques, les incertitudes en fonction des retours de la revue et de la rétrospective du dernier sprint
- Ajuster la vélocité de l'équipe (capacité de travail de l'équipe en nombre de points de story)
- Affiner, (re-)planifier le(s) prochain(s) sprint(s)
 - Définition de prêt et fini
 - Nombre de points
- Affiner, (re-)planifier la future release

Planification d'un *Sprint*

- Confirmer les stories prêtes
 - Définition de prêt et fini
- Evaluer la nombre de points d'une story
 - Points de récit ou journée idéale (homme/jour)
- Organisation de l'essaimage (plusieurs stories en parallèle, répartition des ressources)
- Décomposition des stories en tâches
- Affectation des tâches aux développeurs

Exécution d'un *sprint*

- Conception, réalisation et test des *stories*
- Organisation, affectation des tâches
- Inclut les *sprints* journaliers, dont les mêlées (Scrums)

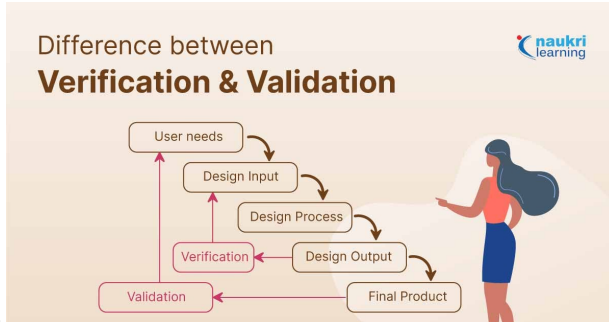
Solidification d'une *release*

- Documentation technique détaillée expliquant l'architecture
 - Par exemple à l'aide de diagrammes UML
- Tests de qualité de services
 - performances, coût,
 - robustesse, sécurité ...
- End to End Testing



Validation d'une release

- Test d'usage avec le client, des utilisateurs
 - Démonstration du produit
 - Bref manuel utilisateur expliquant comment installer et utiliser l'application



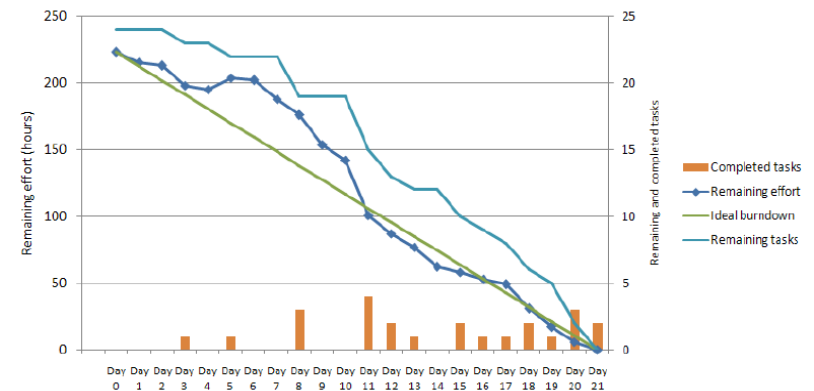
Do the product **right**
vs.
Do the **right** product

7 - Suivi de projet Scrum

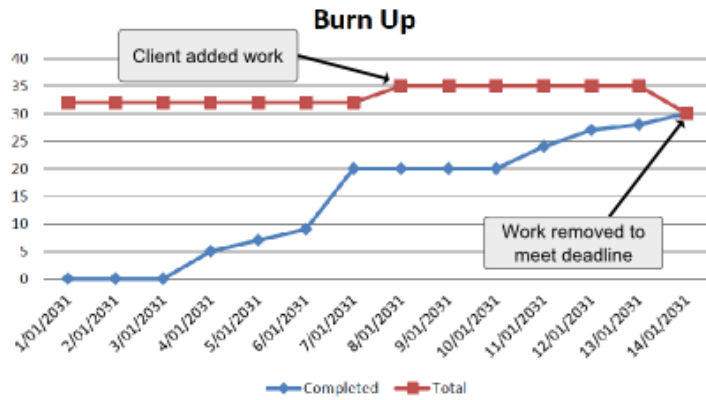
Indicateurs

- Sprint
 - *Burndown* de *sprint* (orienté reste à faire)
 - *Burnup* de *sprint* (orienté ce qui a été déjà fait)
- Release
 - *Burndown* de *release*
 - *Burnup* de *release*
- Equipe
 - Vitesse (capacité de l'équipe)
 - Suivi des obstacles
 - Perturbations exogènes ou endogènes qui perturbent le bon déroulement du sprint
 - Peut de générer de nouvelles tâches (dette) et/ou leur réorganisation

Burndown graphe



Burnup graphe



ISTIC- L2GEN

61

Vélocité

- Estime la capacité de l'équipe en nombre de points de stories par sprint
- Utilisé pour la planification de la release
- Affinée à la fin de chaque sprint
- Tendance à la stabilité

ISTIC- L2GEN

62

8 – Revue de projet Scrum

Revue d'un *sprint*

- Démonstration de chaque story finie
- Collecte du feedback
- Evaluation du niveau de réalisation de l'objectif
- Evaluation de l'impact du travail réalisé et décision d'une release (livraison) ou pas

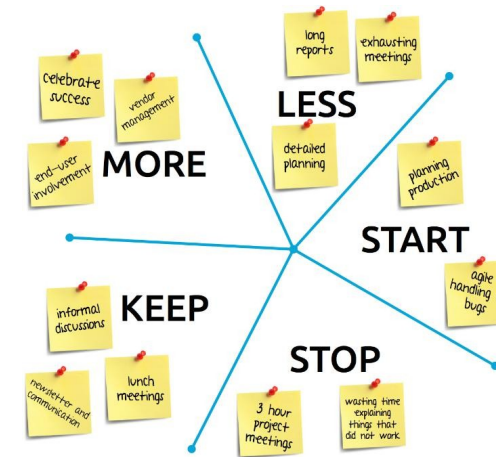
ISTIC- L2GEN

64

Rétrospective d'un *sprint*

- Collecter les information sur le sprint passé
 - Ce qui c'est bien passé, moins bien passé
- Identifier les choses à améliorer
- Décider d'améliorer certaines choses
- Combinée à la revue, de courte durée

Rétrospective de sprint : « Starfish »



9 – Retour sur les LLM

Large Language Models

ChatGPT, Copilot...

(section basée sur les slides de O. Barais)

ChatGPT ?

- ChatGPT =
 - ChatBot (mais qui marche) produit par la compagnie OpenAI
 - 30 novembre 2022
 - 1 million d'utilisateurs en 5 jours
 - 100 millions fin janvier 2023
 - 1.16 milliard en mars 2023
 - Fondé sur un LLM = *Large Model Language*
 - Tuné sur des dialogues humains
 - Interactions fluides
 - « safe »
 - Son job de chaque instant : **prédire le prochain mot à écrire** via des stats

Votre question

+

Le début de sa réponse



Quel prochain mot ???

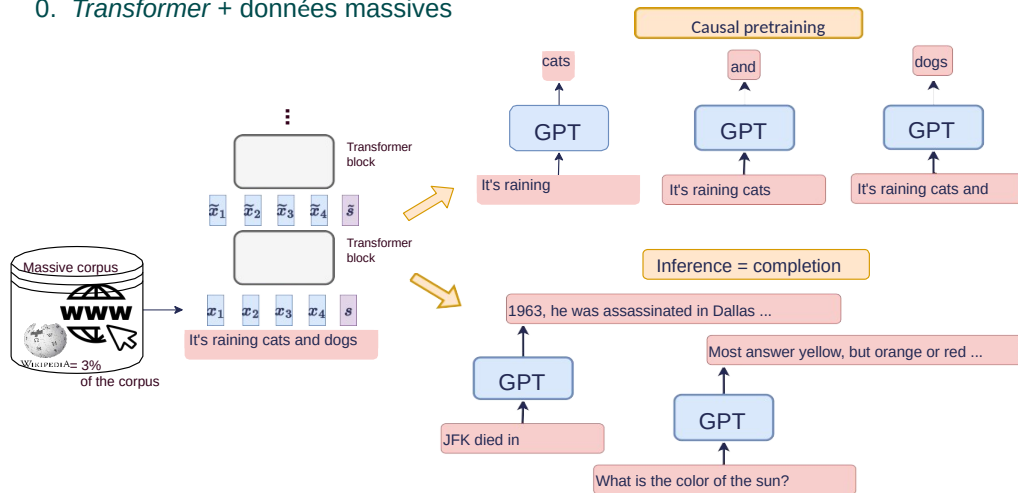
Détails gore sur comment ça marche ?

Improving Language Understanding by Generative Pre-Training

Alec Radford OpenAI, Karthik Narasimhan OpenAI, Tim Salazar OpenAI, Bryce Stender OpenAI

<https://www.youtube.com/watch?v=7ell8KEbhJo&t=1s>

0. Transformer + données massives

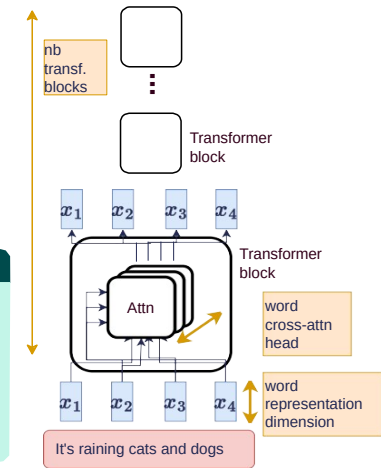


1. Toujours plus!

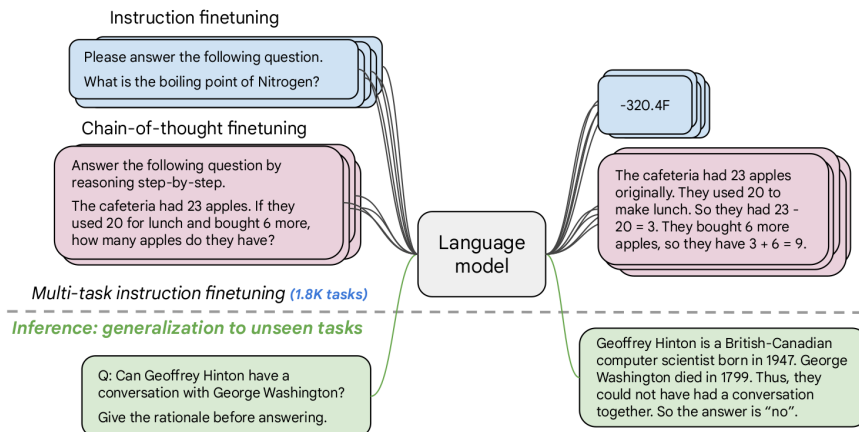
- + de mots en entrée [500 → 2k, 32k]
- + de dimensions dans l'espace des mots [500-2k → 12k]
- + de têtes d'attention [12 → 96 (dim 128)]
- + de blocks/couches [5-12 → 96]

175 Milliards de paramètres... Ça fait

- ▶ $1.75 \cdot 10^{11} \Rightarrow 300 \text{ Go} + 100 \text{ Go}$ (stockage des données en inférence) $\approx 400\text{Go}$
- ▶ GPU NVidia A100 = 80Go de mémoire (=20k€)
- ▶ Coût pour (1) entraînement : 4.6 Millions d'€



2. L'affinage sur différentes tâches de raisonnement (±) complexe

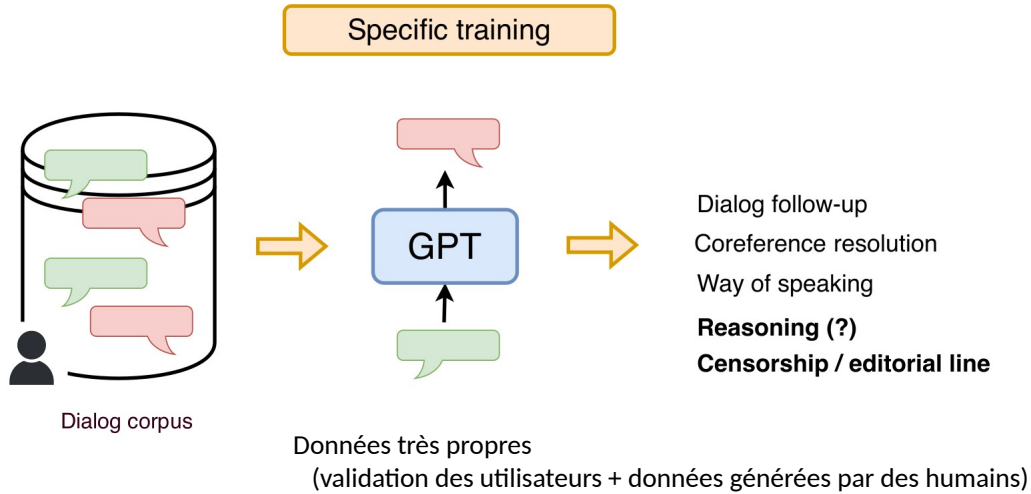


2. L'affinage sur différentes tâches de raisonnement (±) complexe

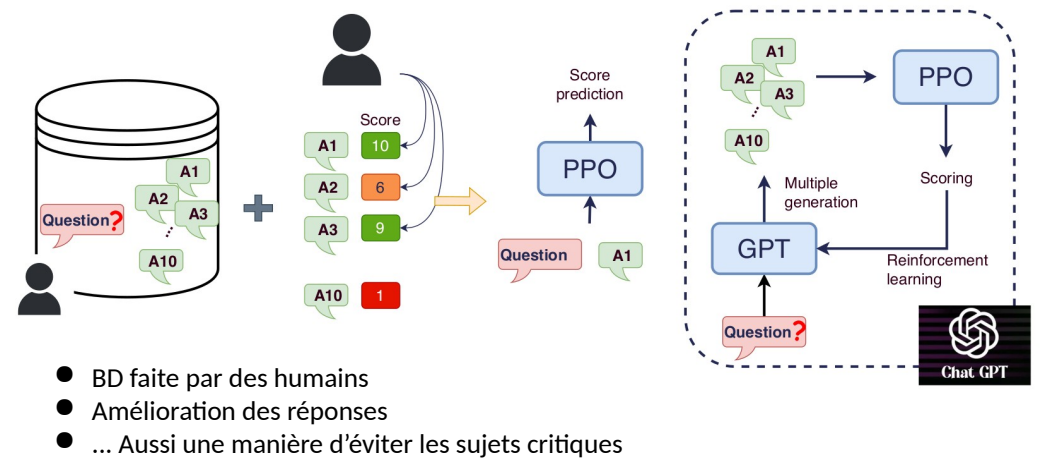
<p>TO-SF</p> <p>Commonsense reasoning Question generation Closed-book QA Adversarial QA Extractive QA Title/context generation Topic classification Struct-to-text ...</p> <p>55 Datasets, 14 Categories, 193 Tasks</p>	<p>Muffin</p> <p>Natural language inference Code instruction gen. Program synthesis Dialog context generation</p> <p>Closed-book QA Conversational QA Code repair ...</p> <p>69 Datasets, 27 Categories, 80 Tasks</p>	<p>Natural Instructions v2</p> <p>Cause effect classification Commonsense reasoning Named entity recognition Toxic language detection Question answering Program execution Text categorization ...</p> <p>372 Datasets, 108 Categories, 1554 Tasks</p>		
<p>CoT (Reasoning)</p> <p>Arithmetic reasoning Commonsense Reasoning Implicit reasoning ...</p> <p>Explanation generation Sentence composition ...</p> <p>9 Datasets, 1 Category, 9 Tasks</p>	<p>MMLU</p> <p>Abstract algebra College medicine Professional law Sociology Philosophy ...</p> <p>57 tasks</p>	<p>BBH</p> <p>Boolean expressions Tracking shuffled objects Dyck languages Navigate Word sorting ...</p> <p>27 tasks</p>	<p>TyDiQA</p> <p>Information seeking QA</p> <p>8 languages</p>	<p>MGSM</p> <p>Grade school math problems</p> <p>10 languages</p>

Mots-clés = tâches => apprendre à structurer

3. Suivi de dialogue



4. Suivi de dialogue & amélioration des réponses



- BD faite par des humains
- Amélioration des réponses
- ... Aussi une manière d'éviter les sujets critiques

Pffff ça marche pas ce truc...Il hallucine...

Question : Que peux-tu me dire d'Alexandre Termier ?

Réponse : Alexandre Termier est un chercheur français en informatique et en intelligence artificielle. Il est notamment connu pour ses travaux dans le domaine de l'intelligence artificielle appliquée aux jeux, en particulier aux échecs. Termier a obtenu son doctorat en informatique à l'Université Joseph Fourier à Grenoble, en France. **FAUX**

2 problèmes ici:

- Demander des faits à ChatGPT = ERREUR. Ce n'est pas son job. Utilisez Google !!!
- Utiliser ChatGPT gratuit (modèle 3.5) = il hallucine plus

Que peux-tu me dire d'Alexandre Termier ?

Finished browsing

Alexandre Termier est professeur à l'Université de Rennes 1, au sein du laboratoire d'informatique IRISA. Il est également le chef du groupe de recherche LACODAM de l'INRIA à l'IRISA, qui se concentre sur le Data Mining et l'Intelligence Artificielle. Termier est également membre associé du Laboratoire d'Informatique de Grenoble (LIG), au sein du groupe Scalable Information Discovery and Exploitation¹. [Lien vers SA page](#)

Au fil de sa carrière, Alexandre Termier a participé à de nombreux projets de recherche. Quelques exemples de son travail incluent:

1. "Anomaly Detection in Streams with Extreme Value Theory" (Détection d'anomalies dans les flux avec la théorie des valeurs extrêmes), en collaboration avec Alban Siffer, Pierre-Alain Fouque et Christine Largouët, publié à KDD 2017². [Lien vers DBLP](#)

En activant le plugin « Bing » : il peut lire et résumer sa page web. Demander des faits devient envisageable.

aucune
ible que
na dernière
nations
, n'hésitez

VRAI ;!)

Quelques utilisations pour des informaticiens

- Répondre à des questions de prog plus vite que StackOverflow
- Trouver et expliquer des erreurs dans des codes
 - *Et parfois en rajouter ;-)*
- Écrire du code à partir d'une spécification plus ou moins précise
- Traduire du code d'un langage à un autre
- **Préf Assistant pédagogique d'info interactif, patient, toujours dispo**

Création de code

OP Tu es un expert en C#/WPF avec plus de 10 ans d'expérience. Ta mission est d'ajouter un composant graphique WPF à une application de gestion d'emploi du temps. Le rôle de ce composant est d'afficher un emploi du temps. Il est basé sur un Canvas. Sa donnée d'entrée est un emploi du temps pour une semaine type.

On a d'abord une List<Cours>, qui est la liste des cours.

Chaque Cours a une propriété Intitule qui donne une chaîne courte (entre 3 et 4 caractères).

Et la liste des créneaux de la semaine pour ce cours, représentée par la propriété suivante :

Dictionary<DayOfWeek, List<CreneauHoraire>> SemaineType

Où CreneauHoraire a les propriétés suivantes :

```
public TimeSpan HeureDebut { get; set; }
public TimeSpan HeureFin { get; set; }
```

Création de code suite

Le dictionnaire donne pour chaque jour une liste des créneaux affectés à ce jour. Et les heures de début et fin du créneau sont données par l'objet CreneauHoraire.

Le Canvas qui représente l'emploi du temps a la spécification suivante :

- son arrière plan est blanc
- il y a un grille gris pâle, où les lignes verticales séparent les jours ouvrés de la semaine (du lundi au vendredi), et les lignes horizontales représentent les heures, une par heure, entre 7h et 19h. L'heure est écrite en petit à gauche au dessus de chaque ligne, le jour est écrit en haut au milieu de chaque colonne.
- un créneau de cours est un rectangle coloré qui prend toute la largeur de la colonne, et l'espace vertical correspondant à sa durée.
- il est placé dans la colonne correspondant à son jour, et verticalement de son heure de début à son heure de fin. Attention les créneaux ont des minutes (ex: 18h15), qui doivent être prises en compte pour son placement correct.
- l'intitulé du cours est écrit au milieu du rectangle coloré.
- une couleur arbitraire est assigné à chaque cours

Création de code suite



Pour réaliser ce composant WPF, on va utiliser principalement un contrôle `Canvas` et les contrôles `Rectangle` et `TextBlock` pour afficher les créneaux.

Commençons par définir la classe Cours et la classe CreneauHoraire.

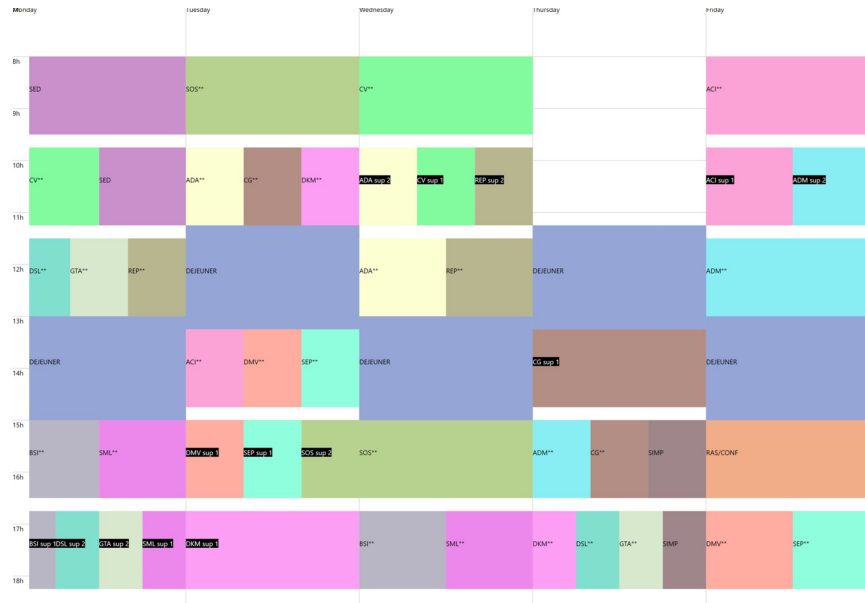
```
csharp Copy code
public class CreneauHoraire
{
    public TimeSpan HeureDebut { get; set; }
    public TimeSpan HeureFin { get; set; }
}

public class Cours
{
    public string Intitule { get; set; }
    public Dictionary<DayOfWeek, List<CreneauHoraire>> SemaineType { get; set; }
}
```

On va utiliser ensuite un `UserController` pour représenter le composant d'emploi du temps.

.....blablabla...

Création de code Résultat :



Université de Rennes Réécriture de code

OP J'ai cette structure de données :

Dictionary<TDemiSemestre, Dictionary<DaysOfWeek, List<CreneauHoraire>>>

Que je parcoure avec cette boucle :

```
foreach (DaysOfWeek jour in Sem_SemaineType[DemiSemestre].Keys)
{
    foreach (CreneauHoraire creneau in Sem_SemaineType[DemiSemestre][jour])
    {
        if (creneau.Flottant) return true;
    }
}
```

pour vérifier si l'un des CreneauHoraire a la propriété Flottant à true.
Pourrais tu me transformer cette boucle en construction linq plus courte ?

Université de Rennes Réécriture de code

Bien sûr, vous pouvez utiliser LINQ pour simplifier cette boucle en une seule ligne. Voici comment vous pouvez le faire :

```
csharp
bool flottantExists = Sem_SemaineType[DemiSemestre]
    .SelectMany(pair => pair.Value)
    .Any(creneau => creneau.Flottant);
```

Dans cet exemple, `SelectMany` est utilisé pour "aplatir" les listes de `CreneauHoraire` dans une seule séquence, puis `Any` est utilisé pour vérifier si au moins un `CreneauHoraire` a la propriété `Flottant` à `true`.

Notez que cette version renvoie `false` si `Sem_SemaineType[DemiSemestre]` est vide ou si aucun `CreneauHoraire` n'a la propriété `Flottant` à `true`, tout comme votre version avec les boucles.

Université de Rennes

Tout n'est pas rose au pays de ChatGPT

- Hallucinations -> on a vu, et ça arrivera toujours!
- Longueur de contexte limitée
 - « mémoire » de 32k tokens max (disons 20k mots) avec GPT-4
 - -> si conversation longue oublie le début, donc la consigne !
 - Très problématique quand écriture collaborative d'un code
 - On donne une spec, il donne un code
 - On trouve une erreur ou un cas nouveau à gérer -> question -> correction
 - On itère une ou deux fois => Ses réponses se mettent à oublier des corrections précédemment effectuées, voire la consigne initiale !
- Manque d'explicabilité/interprétabilité
- Manque de stabilité/prédictibilité
- Manque de transparence
- Manque de confidentialité : interdit dans certaines entreprises/projets

Bottes de 7 lieues si on comprend ce qu'on fait
Déconnecteur de cerveau sinon

Je vous encourage et je m'attends à ce que vous **utilisiez** l'IA (ChatGPT et/ou Copilot) dans ce cours
Apprendre à utiliser l'IA est une compétence émergente et il est important de savoir l'utiliser en génie logiciel

Soyez toutefois conscient des **limites** de ChatGPT/Copilot/... : <https://youtu.be/R2fjRbc9Sa0>

- Si vous fournissez des questions (*prompts*) en fournissant peu d'effort, vous obtiendrez des résultats de faible qualité. Affinez vos *prompts* afin d'obtenir de bons résultats. **Cela demande du travail.**
- Ne vous fiez jamais à ce qu'il dit. **Gardez un esprit critique. Vous serez responsable de toute erreur** ou omission fournie par l'outil. Il vaut mieux utiliser l'IA sur des sujets que vous comprenez.
- L'IA est un outil que vous **devez reconnaître avoir utilisé**. Veuillez utiliser les commentaires de votre code pour expliquer à quelles fins vous avez utilisé l'IA et quels prompts vous avez utilisés pour obtenir les résultats. Ne pas le faire constitue une violation des politiques d'honnêteté académique.
- Vous devez **comprendre** le code que propose l'IA. Vous pouvez d'ailleurs lui demander de vous l'expliquer. Lors de votre évaluation individuelle **tout code que vous n'êtes pas capable d'expliquer entrainera une note de 0.**
- Réfléchissez au moment où cet outil est utile. Ne l'utilisez pas s'il n'est pas approprié au contexte du travail à fournir, **au minimum ayez un usage raisonné en lien avec l'impact environnemental associé.**