Gestion de projets

ou

Comment ne pas se fâcher avec tous ses amis

Marek BUCKI (UGA / twinsight-medical.com)
marek.bucki@univ-grenoble-alpes.fr

Jan Steen
1665

Gestion de projets

ou

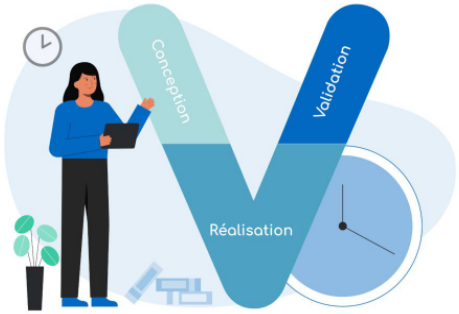
Comment ne pas se fâcher avec tous ses amis

Marek BUCKI (UGA / twinsight-medical.com)
marek.bucki@univ-grenoble-alpes.fr

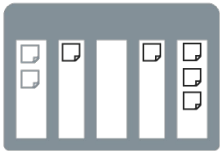
Jan Steen
1665

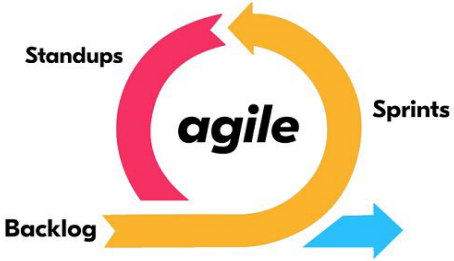
Objectifs

- Enjeux et Phases de la vie d'un projet
- Méthodes de conduite d'un projet
- Outils et Conseils pour réussir votre projet
- Bonus slides



https://bubbleplan.net/pedagogie-projet/cycle-en-v


Kanban



Trello

Overleaf

Google Chat

slack

Google

GitHub

Scrum.org

Enjeux

Projet

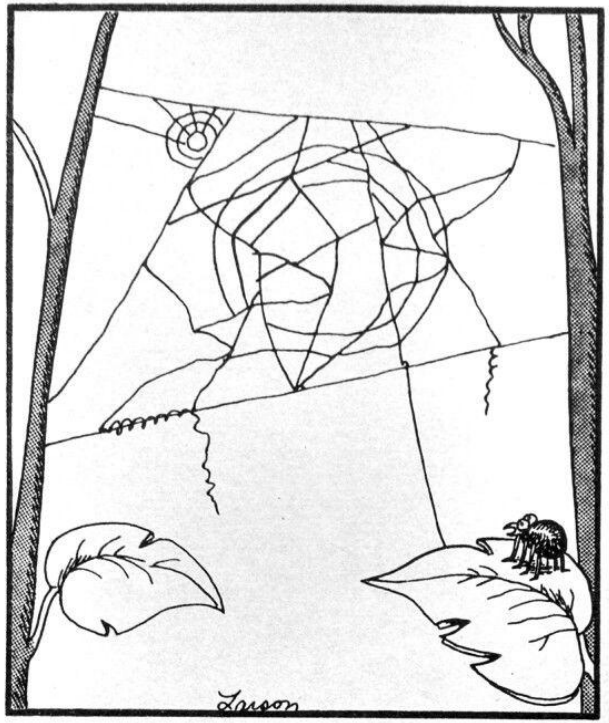
- Objectif : *“Quel service / fonction est attendu(e) par le client ?”*
- Dates : début et fin
- Moyens et contraintes

Enjeu professionnel

- Répondre aux attentes du client !

Enjeux pédagogiques

- Développer vos compétences, travailler en équipe, mener un projet
- Produire un rapport (*livrable*) qui démontre les compétences acquises
- Obtenir une bonne note !



“Whoa! . . . That *can't* be right!”

→ Pour y arriver : il faut une **méthode** et des **outils** !

Phases d'un projet – 1/4

La Roue de Deming
Plan → Do → Check → Act → …

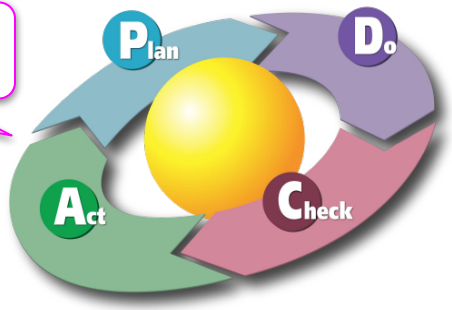


Diagram by Karn G. Bulsuk
(http://www.bulsuk.com)

Phases

- 1 – Démarrage : *"Que doit-on faire ?"*
- 2 – Planification : *"Comment va-t-on s'organiser pour le faire ?"*
- 3 – Exécution : *"On le fait, et on démontre qu'on avance."*
- 4 – Recette : *"Voici ce qu'on a fait, et voici ce qui a marché ou pas."*

Image by macrovector on Freepik

1 – Démarrage

- Client : formule le besoin (sujet)
- Equipe : doit s'approprier le sujet, et le raffiner si besoin avec l'aide du Client
- A l'issue de cette phase : le livrable est le **Cahier des charges**
  - **Spécifications fonctionnelles** = description de haut niveau du service attendu que souhaite le client
  - **Dates** de début et de fin du projet
  - **Moyens** : données, articles, accès à des machines, accès à des personnes, …
  - **Contraintes** = technologique (OS/langage/…), réglementaires (documentation), confidentialité (non-divulgation), …

Phases d'un projet – 2/4

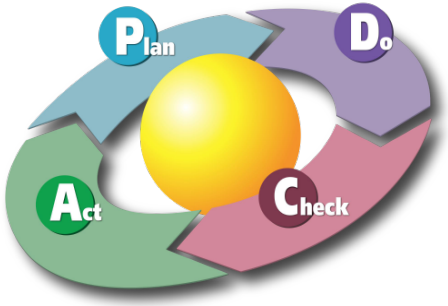


Diagram by Karn G. Bulsuk
(http://www.bulsuk.com)

Phases

- 1 – Démarrage : *"Que doit-on faire ?"*
- 2 – Planification : *"Comment va-t-on s'organiser pour le faire ?"*
- 3 – Exécution : *"On le fait, et on démontre qu'on avance."*
- 4 – Recette : *"Voici ce qu'on a fait, et voici ce qui a marché ou pas."*

Image by Freepik

2 – Planification

- Equipe : découpe le projet en tâches, estime leur difficulté (durée), définit les jalons et livrables intermédiaires
- A l'issue de cette phase : l'équipe dispose d'une **Feuille de route** (partagée avec le Client)
  - **Liste de tâches** = réalisations (techniques ou non), avec une date de début et de fin, et un responsable
  - **Planning** = enchaînement temporel des tâches + graphe d'interdépendance entre les tâches
  - **Jalons et livrables** = points de rendez vous entre l'Equipe et le Client pour **montrer l'avancement** et **confirmer les choix techniques**
- Client : valide la feuille de route proposée par l'équipe

Phases d’un projet – 3/4

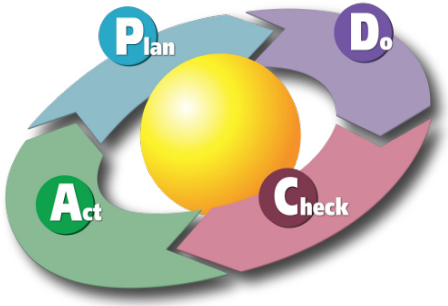


Diagram by Karn G. Bulsuk
(http://www.bulsuk.com)

Phases

- 1 – Démarrage : *“Que doit-on faire ?”*
- 2 – Planification : *“Comment va-t-on s’organiser pour le faire ?”*
- 3 – Exécution : *“On le fait, et on démontre qu’on avance.”*
- 4 – Recette : *“Voici ce qu’on a fait, et voici ce qui a marché ou pas.”*

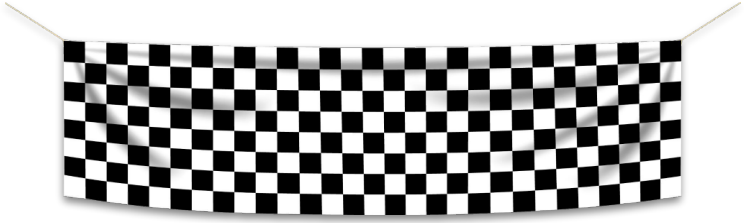


Image by macrovector on Freepik

3 – Exécution

- Equipe : déroule la feuille de route et assure le suivi du projet, en **ajustant la marche en fonction des aléas**
- L’Equipe contacte le Client à chaque jalon pour **fournir les livrables** qui **montrent l’avancement** du projet
- Lors de ces réunions, l’Equipe peut également fournir des éléments de documentation intermediaires
- Les points à aborder en priorité lors des réunions d’avancement sont les problèmes que rencontre l’Equipe
- A l’issue de cette phase : la feuille de route a été **déroulée**, et toutes les **tâches du projet ont été réalisées**

Phases d'un projet – 4/4

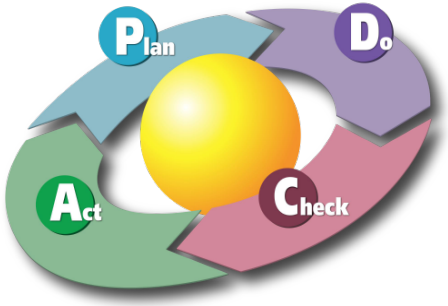


Diagram by Karn G. Bulsuk
(http://www.bulsuk.com)

Phases

- 1 – Démarrage : *"Que doit-on faire ?"*
- 2 – Planification : *"Comment va-t-on s'organiser pour le faire ?"*
- 3 – Exécution : *"On le fait, et on démontre qu'on avance."*
- 4 – Recette : *"Voici ce qu'on a fait, et voici ce qui a marché ou pas."*

Image by macrovector on Freepik

4 – Recette

- Equipe : organise une dernière réunion avec le Client afin de lui fournir les livrables finaux
  - **Livrable final** : executable, codes sources, résultats des calculs, …
  - **Documentation** : spécifications techniques (description des briques techniques), rapport, …
- Cette réunion est l'occasion de faire une **rétrospective** (succès et échecs) du projet afin d'en **tirer des leçons** (Equipe et Client)
- A l'issue de cette phase : le projet est **Fini**.
- L'Equipe présentera le projet plus tard, au cours d'une soutenance

Méthodes de conduite d'un projet – 1/2

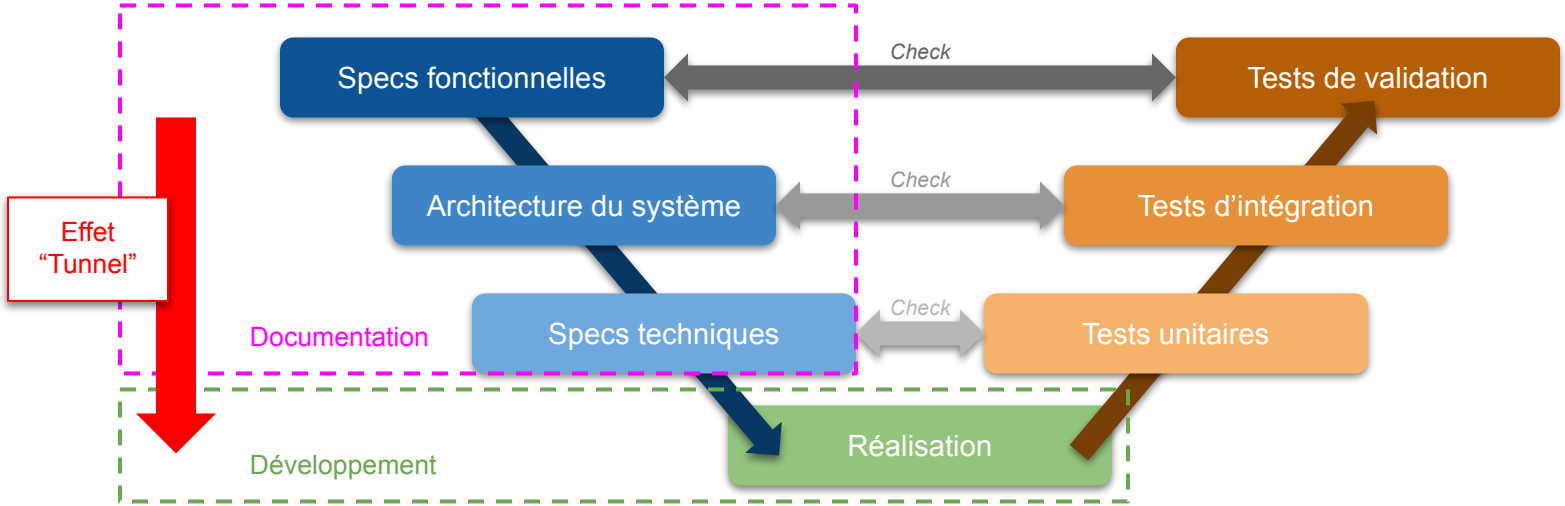
Il existe essentiellement 2 types d'approches

- Traditionnelles : *Cycle en V, Waterfall*
- Agiles : *Scrum, Kanban*

Que cela ne vous décourage pas d'essayer le **Cycle en V** si vous privilégiez la **réflexion** par rapport à l'**action**

Méthodes traditionnelles : Cycle en V, Waterfall

- Méthodes **éprouvées**, adaptées à des projets avec **peu d'éléments inconnus**
- **Principe** : *"Si on y réfléchit bien on peut écrire des spécs quasi parfaites du premier coup."*
- Utilisées dans les **grandes entreprises (cadre réglementé)** mais aussi dans certaines **start-ups innovantes**
- Peu adaptées à un développement itératif, où l'on **revient sur des choix techniques** faits plus tôt
- En pratique le **Cycle en V** se transforme souvent en **Cycle en VVV** …



Méthodes de conduite d'un projet – 2/2

Le **Kanban** est adapté pour un premier essai sur un projet relativement court

Il existe essentiellement 2 types d'approches

- Traditionnelles : *Cycle en V, Waterfall*
- Agiles : *Scrum, Kanban, …*

Méthodes Agiles

- Méthodes itératives, flexibles, relativement récentes (circa 2001) mais de plus en plus répandues
- **Principe** : *"Plutôt que de réfléchir à des problèmes qu'on n'a pas encore découverts, avançons sur le projet et on verra bien."*
- **Attention** : cela n'exclut pas un minimum de réflexion préliminaire mais inutile de figer toutes les spécs pour commencer
- Scrum, Lean, Kanban : variantes autour de ce principe

Workflow SCRUM

Rôles et cérémonies subtiles à mettre en oeuvre …

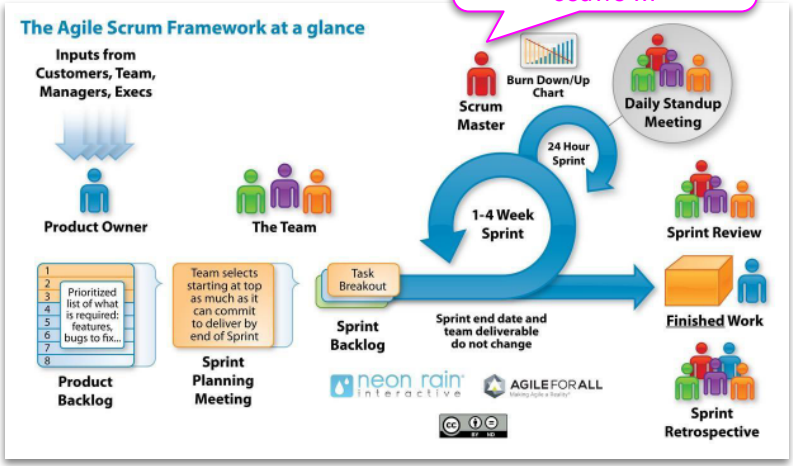
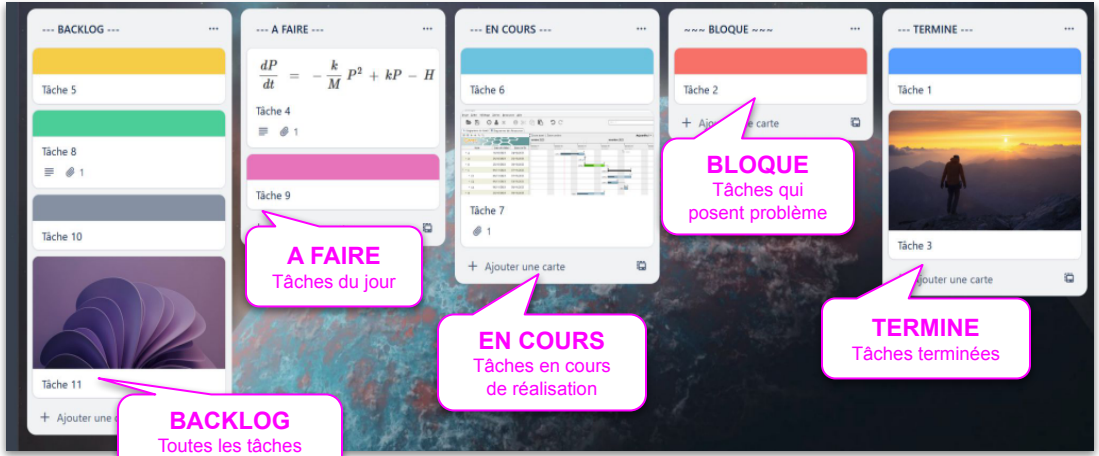


Tableau KANBAN



Outils – 1/5

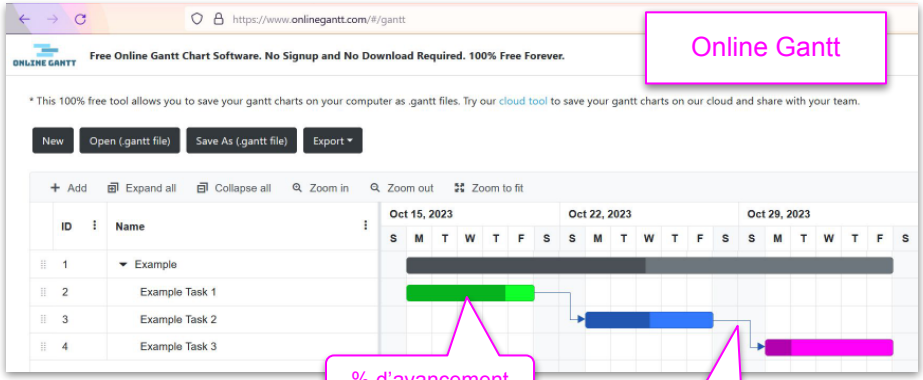
Les outils décrits ici sont **utilisés intensivement** dans l'industrie et la recherche

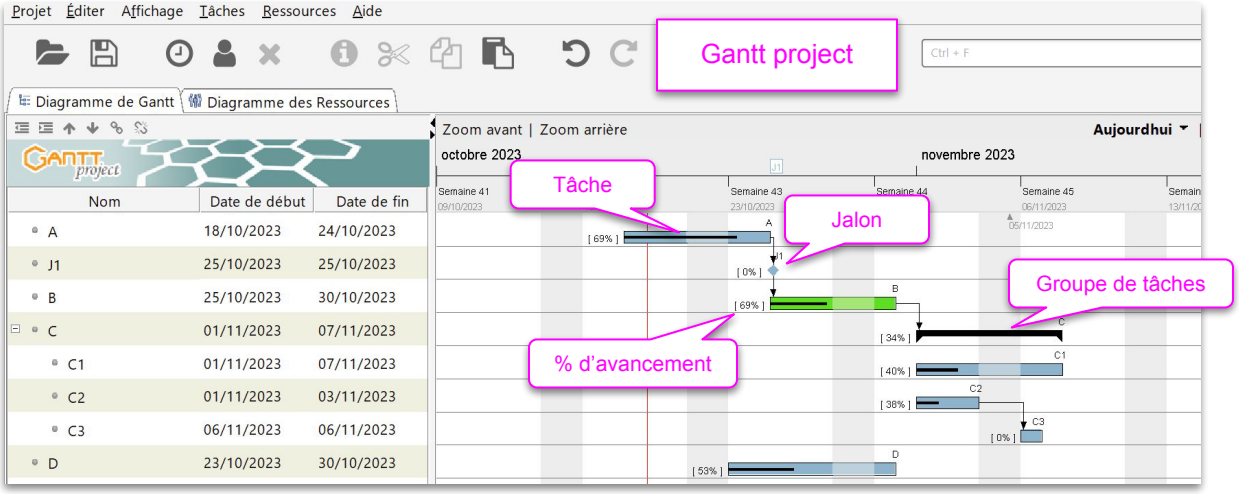
Outils libres qui facilitent la collaboration

- Planification
- Gestion de l'avancement
- Partage des codes sources
- Communication dans l'équipe
- Documentation

Planification des tâches

- OFFLINE → Gantt project sourceforge.net/projects/ganttproject
- ONLINE → Online Gantt www.onlinegantt.com/#/gantt





Outils – 2/5

PLANNING : suivi au long cours du projet

AVANCEMENT : focus sur les tâches en cours

Outils libres qui facilitent la collaboration

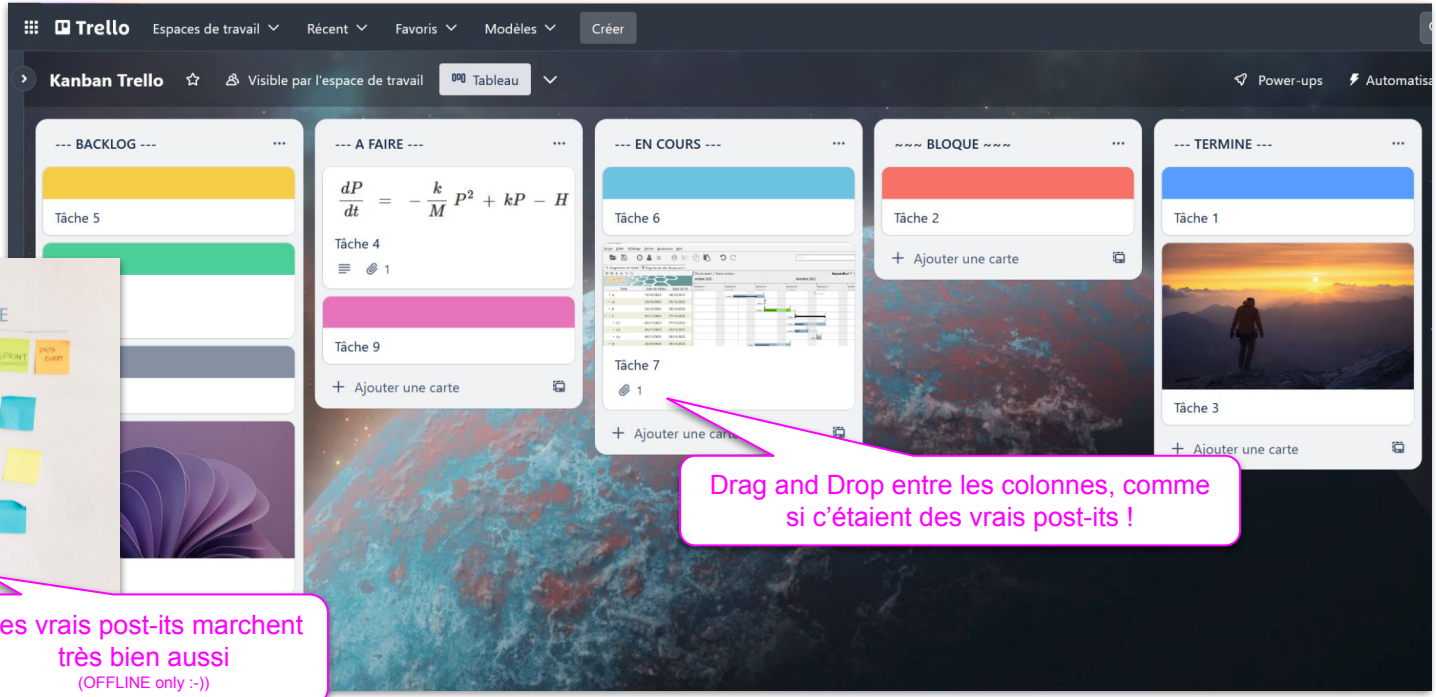
- Planification
- Gestion de l'avancement
- Partage des codes sources
- Communication dans l'équipe
- Documentation

Tableau Kanban

- 1 carte = 1 tâche = 1 "*user story*"
- 1 carte = Titre + Description + Responsable

Trello

trello.com



Drag and Drop entre les colonnes, comme si c'étaient des vrais post-its !

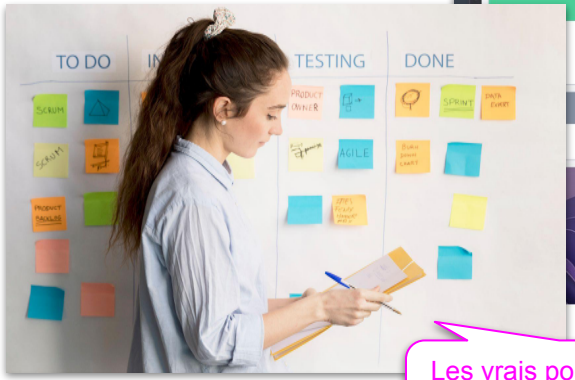


Image by Freepik

Les vrais post-its marchent très bien aussi
(OFFLINE only :-))

Outils – 3/5

Outils libres qui facilitent la collaboration

- Planification
- Gestion de l'avancement
- Partage des codes sources
- Communication dans l'équipe
- Documentation

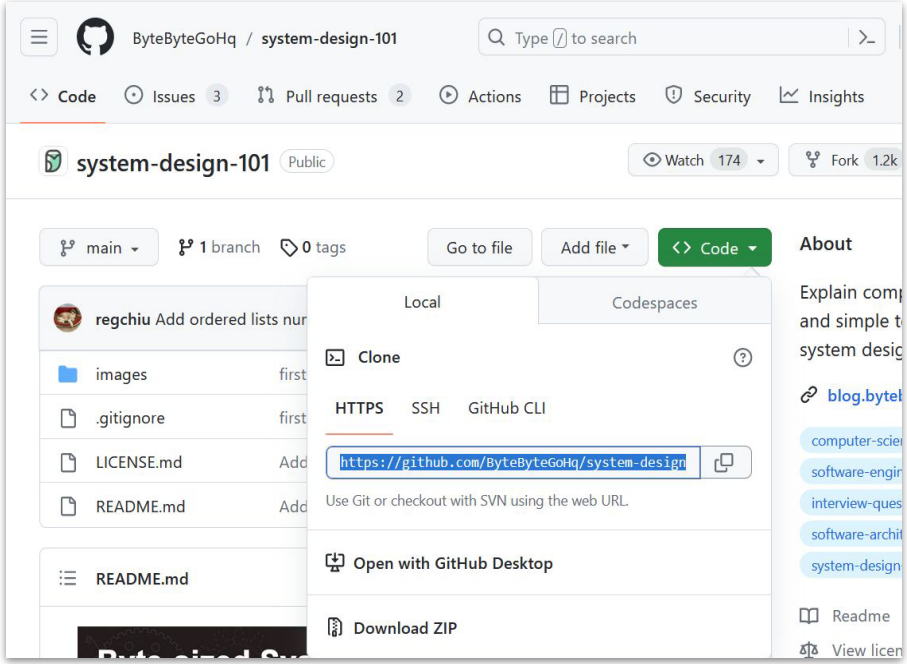
Outil **complexe** mais peut être utilisé **facilement** et **rapidement** de manière **simple**

Voir slides bonus

repo, stage, commit, push, pull, merge, …

GitHub

GitLab



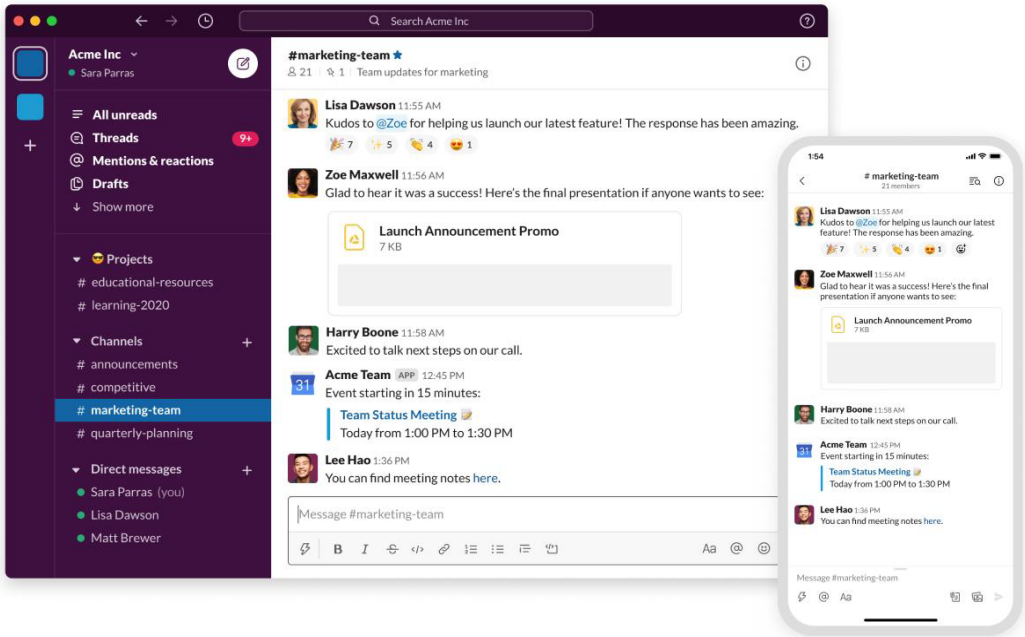
Outils – 4/5

Outils libres qui facilitent la collaboration

- Planification
- Gestion de l'avancement
- Partage des codes sources
- Communication dans l'équipe
- Documentation

Nombreux « tuyaux »

- Google Chat, Slack, Teams, Discord, Zoom, …
- Possibilité d'un **canal fourni par le client** ?
- Privilégier outils qui permettent de **trier les conversations par sujet** □ éviter le mail…



Discord

discord.com

Google Chat

mail.google.com/chat

slack

slack.com

Microsoft Teams

www.microsoft.com/fr/microsoft-teams

Outils – 5/5

Outils libres qui facilitent la collaboration

- Planification
- Gestion de l'avancement
- Partage des codes sources
- Communication dans l'équipe
- Documentation

www.overleaf.com

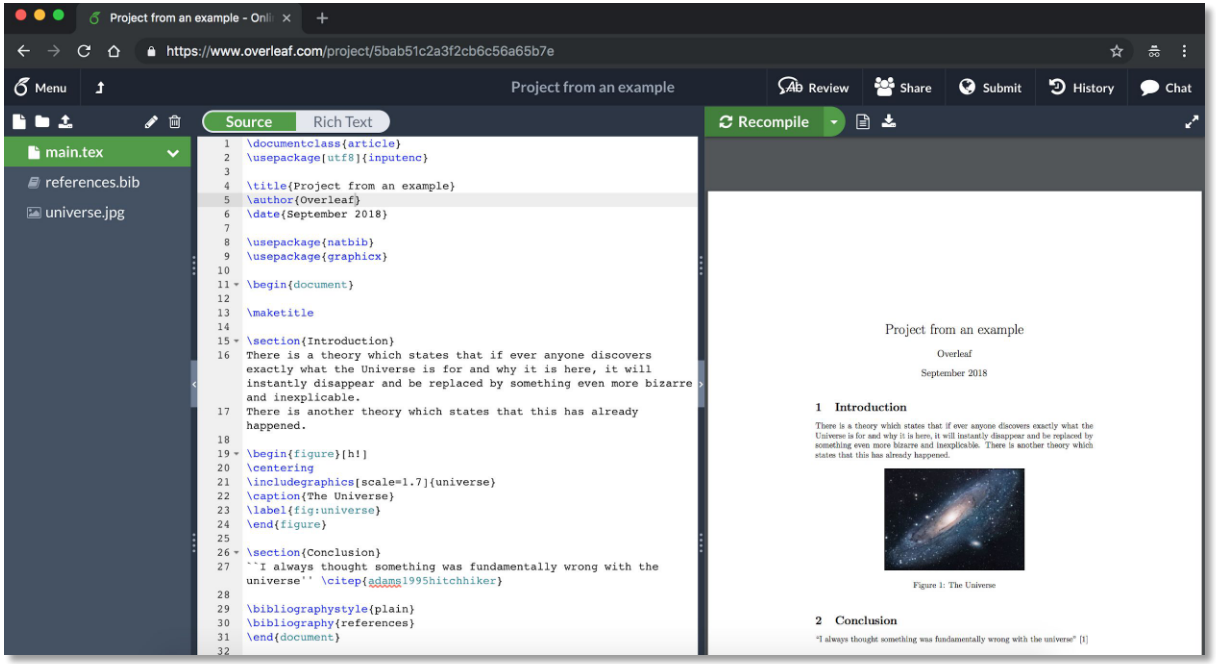
Formats classiques

- DOCx, PPTx, XLSx, … → Google Docs
- LaTeX → Overleaf

Autres

- Notebook Jupyter … ?
- Markdown to PDF … ?

www.google.com/docs/about

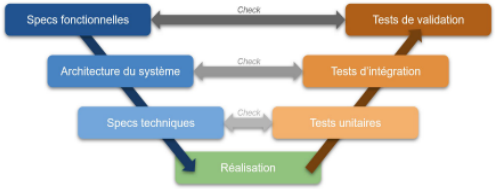


Conseils

- La gestion de projet ne permet pas de prédire (ou de contrôler) l'avenir …
- Les **dérives sont possibles** (probables)
  - Vous devez apprendre à **ajuster le planning** pour **affiner vos prédictions** : charges, dates des jalons, date de clôture du projet, …
- **Identifiez**, **estimez** et **planifiez les tâches avant d'agir**
- Réfléchissez (crayon + papier) **avant de coder** (même en Kanban !)
- Anglais ou français ? En fonction de **votre aisance**, en fonction des **besoins de votre client** … pratiquez l'anglais "technique" !
- Projet = **versant technique** + **versant gestion** : mettez **les 2 en valeur** dans votre rapport
- Adoptez certaines **"cérémonies"** de l'**Agile** (→ slide *Bonus : Cérémonies de l'Agile*)
- **Communiquez** avec le client et les encadrants … et entre vous (bonnes ou mauvaises nouvelles)
- **Ne restez pas bloqués !**

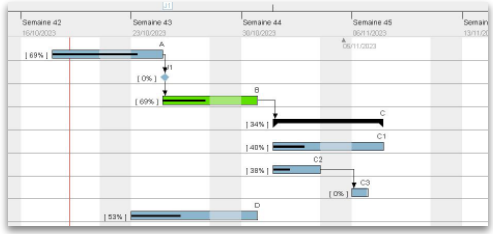
Conclusion

Avec la méthode de votre choix
… mais **avec méthode**

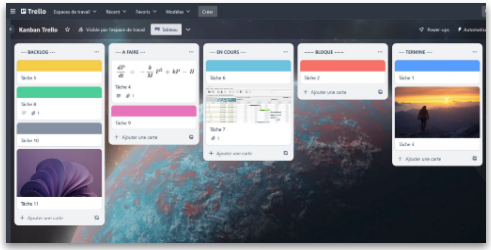


Ce qui est attendu des étudiants

- Dérouler les phases du projet
  - **Cahier de charges** et **Planning** en début de projet
  - **Exécution** et **Suivi** courant avril
  - **Recette** et rapport et présentation mi juin



- Tout au long du projet : **alerter en cas de difficultés !**



- **Travail d'équipe** : tout le monde s'implique (les notes peuvent être différenciées !)
- **Horaires de travail** : 9h → 17h, lundi à vendredi, à l'IM2AG
- **Organiser les réunions** en tenant compte des contraintes de vos interlocuteurs (UGA / entreprises)
- **Préparer les réunions** d'avancement … **à l'avance**

Et maintenant …

- Lecture détaillée de **tous les sujets**
- Classer par ordre de préférence **3 sujets**
- RdV cet après-midi pour **former les groupes**
- Lundi matin, **1er contact par email** avec chaque client

Image by vectorjuice on Freepik

Livrables DECEMBRE 2024

- Presentation de **15 ~ 20 minutes** + **5 minutes** de questions
- Rapport préliminaire de **5 ~ 20 pages**
- Sujets abordés
  - **Contexte du projet** – Entreprise, produit, activité au sein de l'entreprise, …
  - **Objectif du projet** – La fonction à développer = la valeur ajoutée à apporter au client
  - **Ressources du projet** – Moyens disponibles: encadrants, données, machines, langage, rapports, articles, …
  - **Mode de pilotage du projet** – Mode de gestion V, Agile, …
  - **Tâches, planning, roadmap** – List des tâches à réaliser pour atteindre l'objectif, Gantt, planning prévisionnel

Bonus : Chiffrage, planning, chemin critique

Et non pas
"jours / (par) personne"

La complexité (en temps de développement) ou **charge** d'une tâche se chiffre en **jours x hommes** (* genre neutre), ou **jours x personnes**

Exemple : si **Tâche T1** nécessite **4 jours x personnes** alors :

- **1x4** : **1 personne à temps plein** mettra probablement **4 jours** pour la réaliser
- **2x2** : avec **2 personnes**, la tâche ne prendra plus que **2 jours**
- **4x1** : avec **4 personnes**, la tâche peut être bouclée en **1 journée**

On sent qu'il y a une limite de "scalabilité" à ce raisonnement …

Unité : **jxh, j.h, jxp, j.p** … ou tout simplement **"jours"** (dès lors qu'on sait qu'on parle de jours de travail pouvant **être répartis entre plusieurs personnes**)
Mais attention : une **charge** n'est pas une grandeur additive car en réalité : toute **collaboration nécessite synchronisation**, ce qui prend du temps !

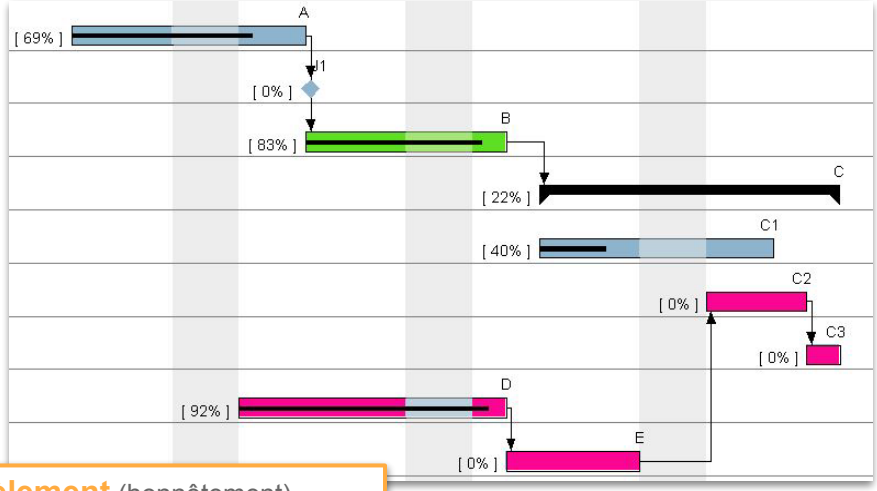
Chemin critique

Enchaînement de tâche (dépendantes) qui conditionne la **fin du projet**

Pour **rapprocher la date de fin**, on peut :

- Démarrer les tâches critiques **plus tôt**
- Les **terminer plus vite** en affectant **plus de personnes** à leur réalisation
- Réévaluer / réorganiser les dépendances
  → *"Ne peut-on pas commencer **E** un peu avant la fin de **D** ?"*

Les outils comme Gantt Project permettent d'**évaluer les scénarios** en début de projet et surtout de **mettre à jour le planning** en cours de route en fonction des difficultés rencontrées ou des verrous techniques levés.



Le planning doit **refléter fidèlement** (honnêtement) l'**avancement** du projet et sa **projection** dans le futur

Bonus : Cérémonies de l'Agile

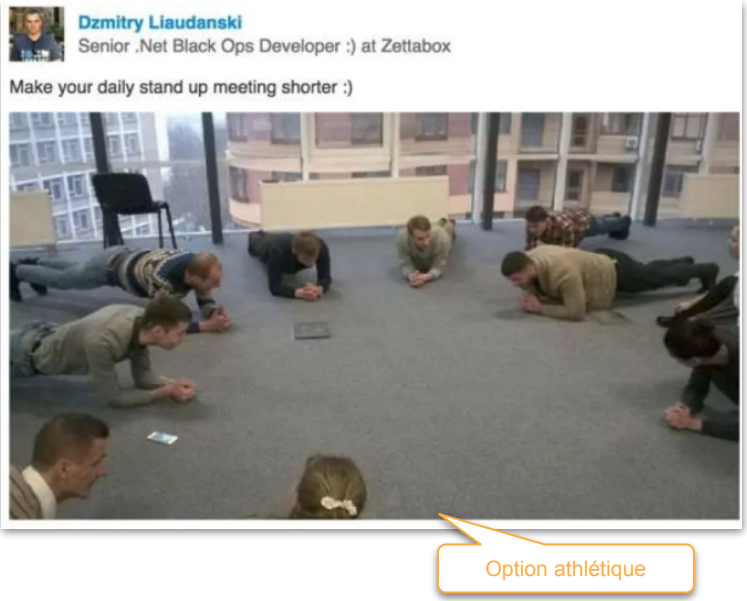
- **Daily** ou **Stand-up**
  - Courte réunion quotidienne (5 ~ 10 min le matin) pour faire le point sur les tâches de la journée

- **Weekly**
  - Réunion en fin de semaine (20 ~ 30 min) pour discuter de la semaine écoulée

- **Règles**
  - **Tous les membres de l'équipe** s'expriment **à tour de rôle**

  - L'équipe fait **point sur l'avancement du travail**
    - tâches accomplies, verrous techniques levés
    - problèmes (techniques, orga, planning, resources, …) rencontrés

  - Pour être efficaces ces réunions **doivent être bien cadrées**
    - temps de parole bien réparti,
    - concision attendue de la part de chacun,
    - pas de dépassement du temps alloué à la réunion,
    - SUPER IMPORTANT : un daily ou un weekly est une réunion de gestion de projet et ne doit pas se transformer en réunion technique !



Bonus : Propriété Industrielle (PI) / Industrial Property (IP)

NDA

→ ***Non Disclosure Agreement*** ***(unilateral ou bilatéral)***

Accord de non diffusion / confidentialité. Certaines entreprises pourraient vous demander d'en signer un. En le signant vous vous engagez à ne pas révéler la teneur des discussions techniques, du code ou des documents que l'entreprise partagera avec vous. Je vous encourage à respecter cet engagement. Poursuites légales possibles si une fuite de PI parvient à l'entreprise et qu'ils déterminent qu'elle vient de vous.

Ne vous laissez pas stresser par un **NDA**. Les sujets soumis à **NDA** ont des chances d'être plus poussés techniquement (et donc plus interessants).

Brevet

→ ***Document qui permet d'interdire l'utilisation d'une idée / algorithme***

Vous ne devriez pas être concernés car un brevet est stratégique pour une entreprise et il y a peu de chances qu'on en confie le développement à des étudiants (du fait de l'importance du secret industriel qui entoure ce genre de choses).

Si votre projet concerne (de loin) un brevet, il est possible que l'entreprise vous demande de signer une **cession de droits d'auteur** des codes / idées que vous serez amenés à développer. C'est une procédure tout à fait normale donc rien d'alarmant là non plus.

Anonymisation de données

→ ***Dissimulation volontaire d'informations permettant d'identifier la source des données***

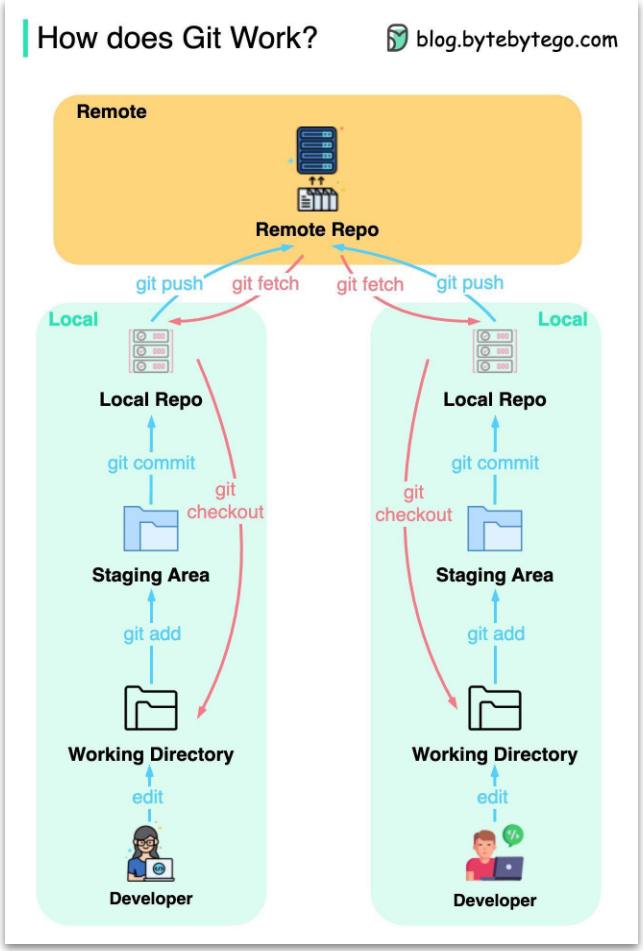
L'exemple typique ce sont les données personnelles (par exemple médicales) qui relèvent du **RGPD** (règlement général de protection des données). Ce règlement très strict protège les individus contre la diffusion (commerciale ou malveillante) de leurs données personnelles (ce qui est une très bonne chose !).

Pour ne pas s'exposer à des sanctions, si une entreprise vous fait travailler sur des données personnelles elle devra les anonymiser afin qu'il vous soit impossible d'identifier la personne concernée par ces données.

L'anonymisation peut concerner des éléments techniques sensibles (par exemple des spécifications électroniques) que l'entreprise ne souhaite pas révéler.

Bonus : Git

GitHub
github.com




```
git clone https://github.com/… .
```

Cloner un repo distant

```
<< CREER LE FICHIER my_file.py >>
```

Ajouter un nouveau fichier

```
git add my_file.py
```

“Stager” le nouveau fichier sur le repo local

```
git commit -m “Added my_file.py”
```

“Commiter” le nouveau fichier sur le repo local

```
git push
```

Pousser le nouveau fichier sur le repo distant

```
<< MODIFIER LE FICHIER my_file.py >>
```

Travailler sur le fichier en local

```
git add my_file.py
```

“Stager” les modifications faites sur le fichier

```
git commit -m “New function ABC”
```

Commit local

```
git pull
```

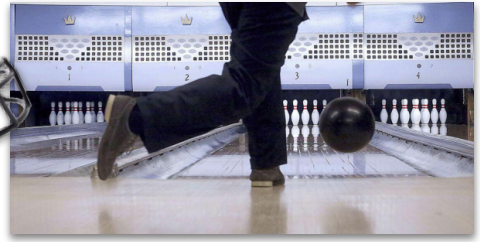
“Puller” (récuperer) les modifications des autres

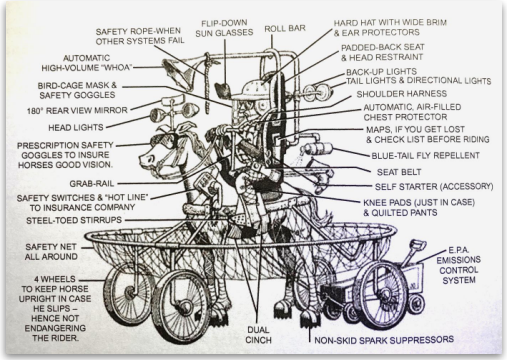
```
git push
```

Pousser ses propres modifs

Bonus : Analyse des Risques – 1/3

Dois-je porter un casque quand je joue au bowling ?





Overengineered cowboy

Safety Risk Management for Medical Devices
Bijan Elahi

Bonus : Analyse des Risques – 2/3

Vous avez fini votre projet un peu en avance ? Super. Vous allez pouvoir faire votre première … ***Analyse des Risques !***

Une Analyse des Risques consiste à mettre sa casquette de pessimiste (à la place de la casquette optimiste du concepteur) et à imaginer tout ce qui pourrait aller mal … en se posant la question suivante :

*Qu'est-**ce qui peut aller mal** avec le système que j'ai conçu, quelles peuvent en être les **conséquences**, et comment **y remédier** ?*

→ ***ce qui peut aller mal*** s'appelle le **danger** (bug dans le code, données mal acquises, mauvaise utilisation du logiciel, …)
→ les ***conséquences*** s'appellent les **dommages** (crash de l'application, calculs faux ou mal interprétés, perte de données, …)
→ ***y remédier*** c'est prendre une **mesure de réduction du risque** (relire le code, vérifier les calculs, bien documenter l'application, gèrer les cas exceptionnels, proposer des modes de calcul alternatifs p.ex. moins précis ou moins rapides mais plus robustes, …)

Dans la vraie vie, les **RAN** (*Risk Analysis*) sont faites de manière plus détaillée que cela encore (voir slide suivante, mais inutile d'aller jusque là !).

Ainsi, ce qu'on appelle **Risque** est le **produit** d'une **Probabilité** (fréquence d'occurrence) et d'une **Sévérité** (gravité des dommages).
Ces deux quantités peuvent:

- se combiner (grande Proba * grande Sévérité) ⇒ grand Risque, ou (petite Proba * petite Sévérité) ⇒ petit Risque; ou bien
- se compenser (grande Proba * petite Sévérité) ⇒ …?, ou (petite Proba * grande Sévérité) ⇒ …? et dans ce cas il faut faire une analyse plus fine.

Par exemple, le Risque de mourir foudroyé est faible car la **faible Probabilité** d'être frappé par la foudre contrebalance l'**extrême gravité des dommages** lorsqu'on les multiplie… Sauf si vous grimpez au sommet d'un sapin au cours d'un orage ! Dans ce cas le **Risque augmente** car la **Probabilité augmente**. La nouvelle situation dangereuse (en haut du sapin) vous expose à une plus grande Probabilité, mais la Sévérité quant à elle reste inchangée.

Bonus : Analyse des Risques – 3/3

Les notions de fréquence et de gravité sont en général définies **qualitativement** ou **quantitativement** (quand on peut…) par des “matrices” de ce genre :

Définition des niveaux de **Probabilité**

Qualitative probability level

Probability		
FREQ	Frequent	Can occur almost every time software is used during its lifetime (in normal use conditions)
PROB	Probable	Can occur some times in lifetime of software in normal use conditions
OCCA	Occasional	Can occur once in lifetime of software in normal use conditions, or multiple times in specific-customer use conditions
RARE	Rare	Can occur once in lifetime of software (in specific-customer use conditions)
IMPR	Improbable	Very unlikely occurrence with current knowledge. Especially in software, risks classified as improbable are not possible

Définition des niveaux de **Sévérité**

Qualitative severity level

Severity		Harm (patient, surgical team, environment)	Industrial impacts
CATA	Catastrophic	Potential of death or life-threatening injury.	Severe impact on financial revenues or company sustainability
CRIT	Critical	Permanent impairment of a body function or permanent damage to a body structure.	Major displease of the customer and/or high financial impacts, immobilized product
MAJ	Major	Injury or impairment requiring professional medical intervention.	Displease of the customer and/or financial impacts, Significant degradation of performances.
MIN	Minor	Temporary injury or impairment not requiring professional medical intervention.	Upset the customer or make him feel uncomfortable, does not involve significant degradations of performances.
NEGL	Negligible	Inconvenience or temporary discomfort.	The customer does not see the problem or is slightly embarrassed, minor disturbance of the function but no degradation of performances.

Et quand on fait le produit Proba * Sévérité,
on aboutit à la matrice suivantes des **niveaux de Risque** possibles :

Risk		Probability				
		FREQ	PROB	OCCA	RARE	IMPR
Severity	CATA	Into	Into	Into	Und	Tole
	CRIT	Into	Into	Und	Tole	Negl
	MAJ	Into	Und	Tole	Tole	Negl
	MIN	Und	Tole	Tole	Negl	Negl
	NEGL	Tole	Negl	Negl	Negl	Negl

La couleur de la case permet de déterminer s’il faut mettre en place une **RCM** (*Risk Control Measure*) ou si ça n’est pas la peine :

Il faut ABSOLUMENT réduire le Risque

On peut ne rien faire pour réduire le Risque

Initial Risk quotation

Risk acceptability criteria	Coding	Meaning
	Into	Intolerable Risk, Risk Control Measures are mandatory and shall be implemented to reduce the risk quotation
	Und	Undesirable Risk, Risk Control Measures are mandatory and shall be implemented to reduce the risk quotation
	Tole	Tolerable Risk, Risk Control Measures are not mandatory
	Negl	Negligeable Risk, Risk Control Measures are not mandatory

Bonus : Object Oriented Design – Data Oriented Design – 1/2

2 approches différentes du code … et de la machine.

Point de départ = ***"Il y a un problème à résoudre de manière informatique."***

Object Oriented Design = Approche **humaine**, **intuitive** du problème → Une solution facile à expliquer, reposant sur des **concepts**

"Quels sont les acteurs du problème ?"
"Comment interagissent-ils ? A travers quelles interfaces ?"

⇒ Structuration de la solution en **classes d'objets** avec une architecture faisant intervenir des objets (instances ces classes)

⇒ Solution portable car les concepts font **abstraction de l'implémentation**, un design OOD peut se porter du C++ au Python

Data Oriented Design = Approche "machine" → Optimise l'usage du hardware (CPU, caches, RAM)

"Comment organiser les données en mémoire pour que le CPU y accède le plus vite possible ?"

⇒ Structuration de la solution en **tableaux de données** cohérentes, regroupées en fonction de leur usage

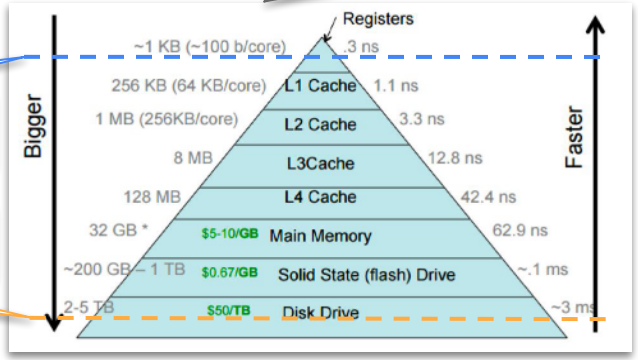
⇒ Accélération spectaculaire des calculs du fait de la diminution des "**cache misses**"

⇒ Solution fortement **liée au hardware**

Temps d'accès à différents types de mémoire : caches, RAM, disque

Un programme **DOD** fait beaucoup d'accès aux "caches" → **Faster**

Un programme **OOD** fait beaucoup d'accès à la "main memory" → **Slower**



Bonus : Object Oriented Design – Data Oriented Design – 2/2

Quelques liens pour aller plus loin… (et vous réconcilier avec le hardware ;))

OPTIMIZATION

Introduction to Data-Oriented Design

A shift from objects to data

Tamás Losonczi · Follow
Published in Wunderman ThompsonBudapest · 7 min read · Jan 24, 2020

Illustration avec un exemple simple

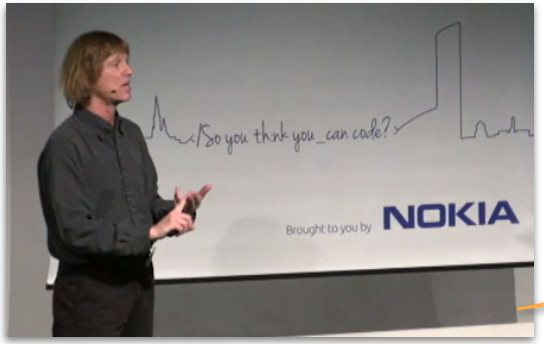
https://medium.com/mirum-budapest/introduction-to-data-oriented-programming-85b51b99572d

Mike ACTON

Style décontracté :)

https://www.youtube.com/watch?v=rX0ItVEVjHc

Scott MEYERS



Un peu austère mais très fouillé

https://www.youtube.com/watch?v=WDIkqP4JbkE

Bonus : Etiquette de l'email

Soignez la forme et le fond dans vos échanges par mail dans le milieu pro

- Fond : contenu clair et concis
- Forme : mise en page soignée (paragraphes logiques, ponctuation), ton courtois, registre soutenu
- Vérifier grammaire et orthographe → correcteur orthographique de Word, GMail, Google Doc, …

Attention

- **Reply all** malencontreux !
- **Fichiers attaché** oubliés
- **Ambiguïtés de l'écrit** : gérer les tensions (avec client et entre vous) **par oral**. *"Suite au malentendu lors des derniers échanges, pourrions-nous convenir d'un entretien pour clarifier la situation ?"*

Structure type

- **Objet** : *"[Projet L3-MI] Planning de notre première réunion / visio …"*
- **Salutations** : *"Bonjour | Bonsoir {Mr | Mme} Dupont"*
- **Introduction** : *"Nous sommes étudiants à …, et nous …"*
- **Contexte** : *"Nous vous contactons au sujet de … | Nous vous écrivons de la part de …"*
- **Attentes** : *"Nous aimerions convenir d'un entretien avec votre équipe afin de …"*
- **Conclusion** : *"Nous vous remercions par avance pour … Cordialement, Jenny, Steve & Bob"*

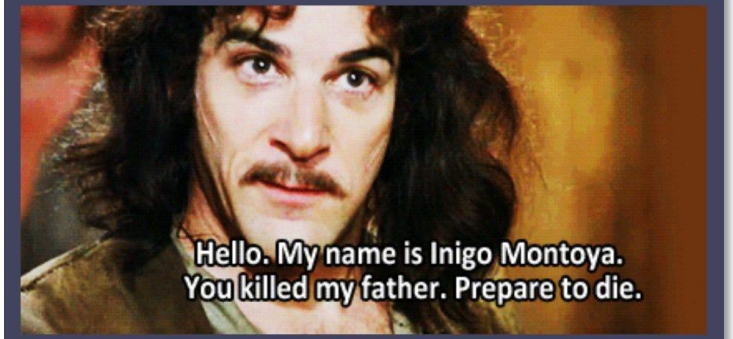
Préfixer l'objet par un **[…]** facilite le classement

Toujours répondre surtout si la personne vient de vous répondre ou vous aider

```
Merci {beaucoup} pour votre réponse {rapide | très claire | très complète}.
{Nous prenons note de .... | Merci pour ces précisions concernant ....}
{Au plaisir de vous rencontrer lors de la soutenance {prévue le ...}.}
Bien cordialement,
Jenny, Steve & Bob
```

Avant de conclure, **précisez votre attente de manière explicite** à la fin du mail

```
[...]
Pourriez-vous nous dire quelles seraient vos disponibilités la semaine prochaine pour ... ?
[...]
```



Remember Inigo Montoya:

1. Polite greeting
2. Name
3. Relevant personal link
4. Manage expectations

"You never get a second chance to make a good first impression."

"La forme c'est le fond qui remonte à la surface."
Victor Hugo

Bonus : Pour aller plus loin…

Agilité
→ https://agilemanifesto.org/iso/fr/manifesto.html

Roue de Deming
→ https://deming.org/wp-content/uploads/2020/06/PDSA_History_Ron_Moen.pdf

YouTube
→ **En anglais** : Getting Started in Project Management - 1. An Introduction - Rita Mulcahy (en anglais, et c'est … vintage!)
→ **En français** : Cours Fondamentaux de la gestion de projet, Rémi Bachelet (plutôt cycle en V)
et **gestiondeprojet.pm**/fondamentaux-de-la-gestion-de-projet/

Livres
→ **En anglais** : A Guide to the Project Management Body of Knowledge (PMBOK® Guide), Seventh Edition and The Standard for Project Management
→ **En français** : Le standard pour le management de projet et Guide du corpus des connaissances en management de projet
→ **En français** : Management de Projet, Jean-Claude Corbel (3eme édition, 2012)