SÉANCE 1

MODÈLE DE DÉVELOPPEMENT DE LOGICIEL

SUJETS

Cycle de vie d'un projet logiciel

Processus de la «boîte noire»

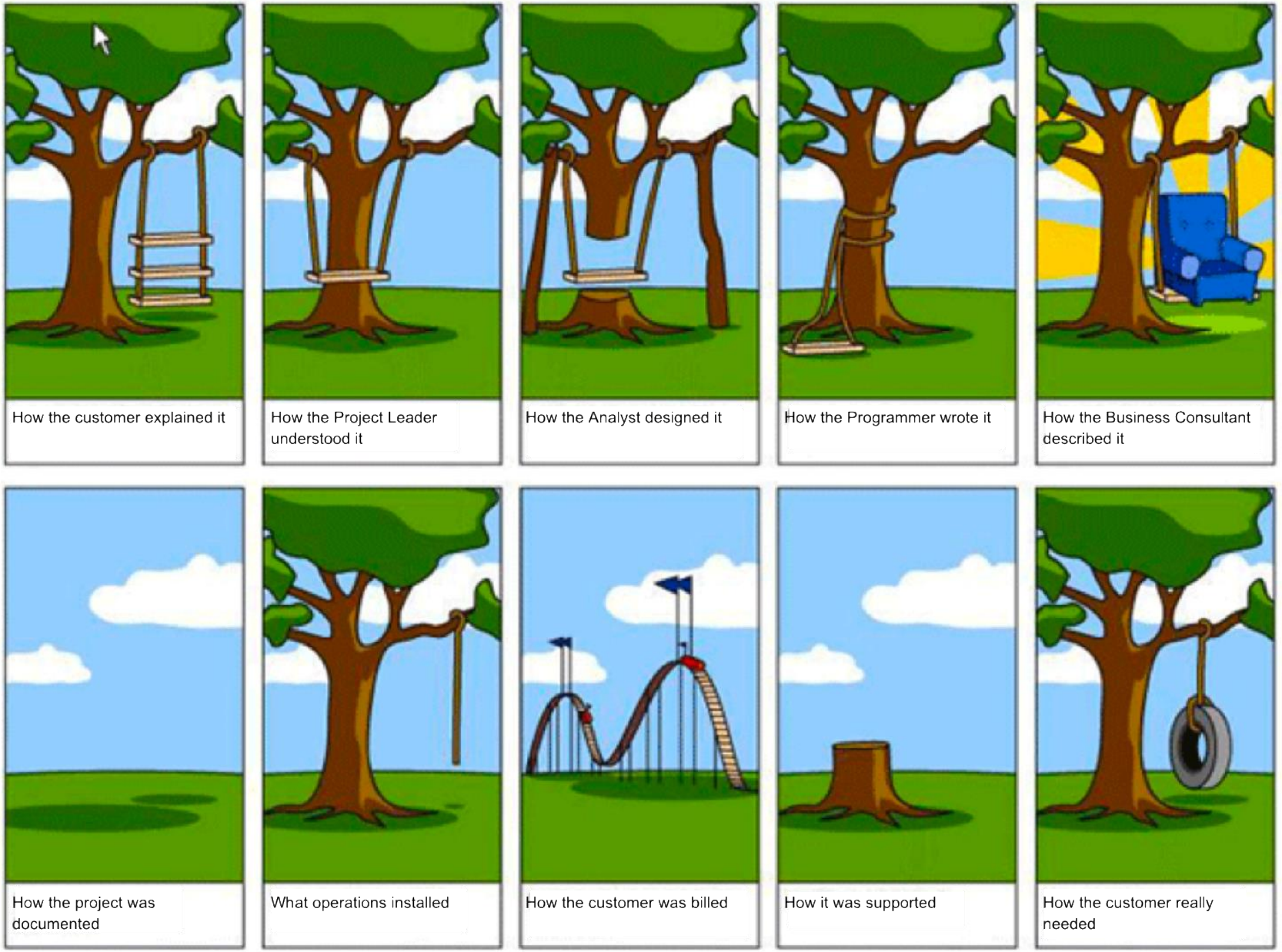
Processus de la «boîte blanche»

Modèle cascade

Modèle itératif

Modèles agile

- Scrum


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
How the customer really needed

AU DÉBUT...

Est-ce qu'il y a des problèmes avec cette approche?

Coder et Tester

Avantages	Problèmes
Obtenir rapidement le système desiré	Ajouter des fonctionnalités peut devenir de plus en plus difficile
Les bonnes équipes peuvent faire du bon travail sans processus	De nouvelles fonctionnalités peuvent faire des ravages sur votre architecture
Peu de frais généraux pour gérer la cérémonie de processus	Être occupé n'est pas la même chose que d'être productif
Problèmes rencontrés très tôt dans le processus	Traiter les problèmes, plus problématique comme le temps passe

BESOIN POUR DES MODÈLES POUR GÉNIE LOGICIEL

uOttawa
L'Université canadienne
Canada's university

Les symptômes d'insuffisance : la crise de logiciel

- temps prévu et coût dépassé
- attentes des utilisateurs n'est pas atteintes
- mauvaise qualité

La taille et valeur économique des applications logicielles nécessite des "modèles de processus"

CRISE LOGICIELLE (DÉFINIE)

- La crise des logiciels est un terme utilisé **dans les débuts de la science informatique** (première conférence de NATO sur l'ingénierie logicielle en **1968**) pour la difficulté d'écrire des programmes informatiques utiles et efficaces dans le temps requis.
- La crise des logiciels était due à l'augmentation rapide de la **puissance informatique** et à la **complexité des problèmes** qui ne pouvaient être résolus. Avec l'augmentation de la complexité du logiciel, de nombreux problèmes de logiciel sont apparus parce que **les méthodes existantes étaient insuffisantes**.

CRISE LOGICIELLE

- **Au milieu des années 90**, certaines études clés ont été réalisées sur le développement de logiciels (Defence Science Board '94, Standish Group '95, Jones '96). Ils ont tous atteint les mêmes **conclusions générales**:
  - Le développement de logiciels est **hautement imprévisible**; Seulement **10%** des projets sont livrés dans le **budget** et le **horaire** initiaux.
  - La **gestion** a **plus d'effet** sur le succès ou l'échec que les progrès technologiques.
  - Trop de **rebuts logiciels et de retouches**: le processus est immature.
- Yourdan rapporte que **25%** des grands projets **ne finissent jamais** et que le projet MIS moyen est e**n retard d'un an** et **dépasse de 100% le budget**.

CYCLE DE VIE

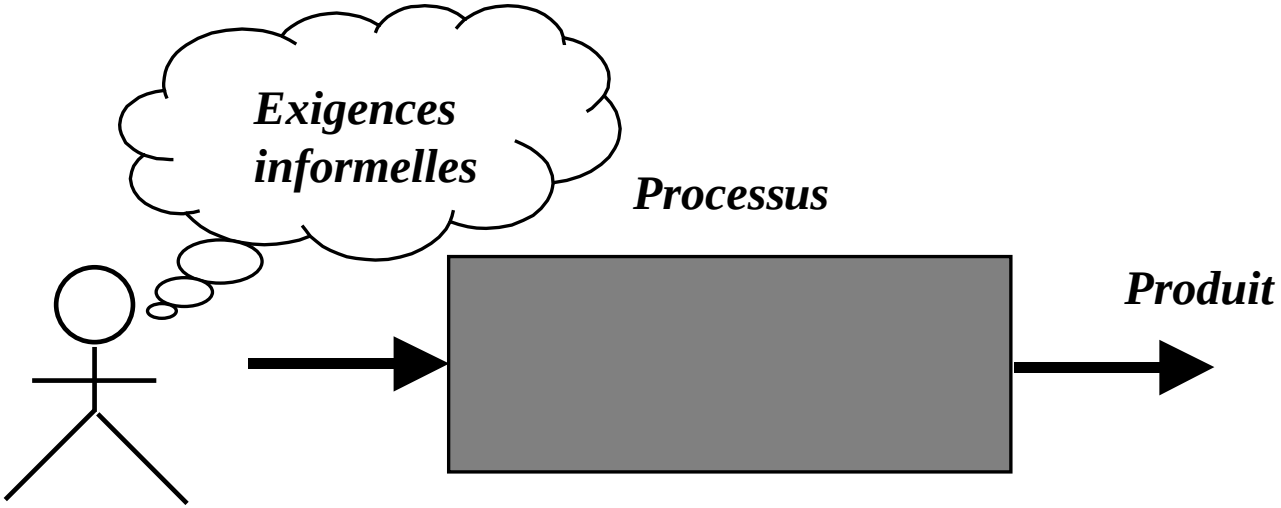
La cycle de vie d’un produit

- depuis la création d'une idée pour un produit à travers
  - analyse du domaine
  - collecte et modélisation des exigences
  - conception de l'architecture et spécification
  - programmation et vérification
  - livraison et le déploiement
  - maintenance and évolution
  - retraite

CYCLE DE LA VIE DES LOGICIELS (SDLC)

- Le cycle de vie comprend la vie entière d'un logiciel en tant que **séquence d'activités**.
  - Très **dépendante du processus** de développement.
  - La gestion de projet logiciel (SPM) dépend fortement du processus de développement (DP).
- Vous **ne pouvez pas gérer** quelque chose si vous ne savez **pas comment cela fonctionne**!

PROCESSUS DE LA «BOÎTE NOIRE»



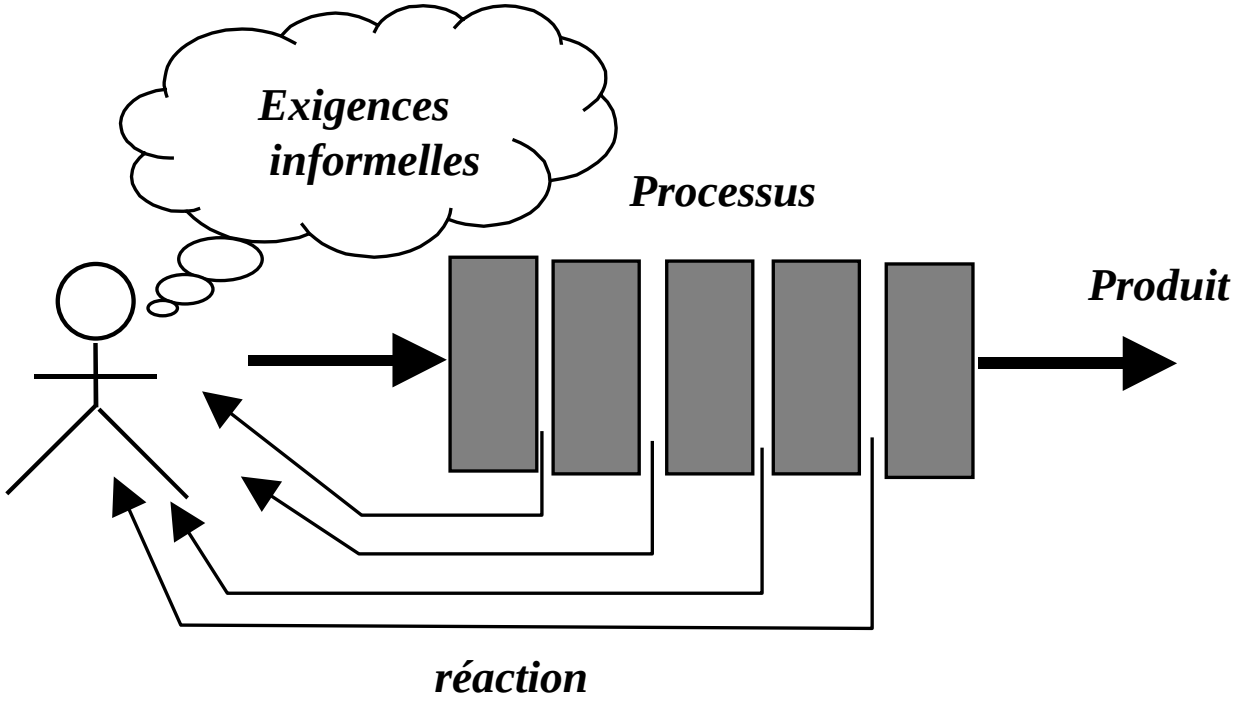
PROBLÈMES

L'hypothèse est que les exigences peuvent être pleinement compris avant le développement

Malheureusement l'hypothèse presque jamais valide

Interaction avec le client ne se produit qu'au début (exigences) et la fin (après la livraison)

PROCESSUS DE LA «BOÎTE BLANCHE»



AVANTAGES

Réduire les risques en améliorant la visibilité

Autoriser les changements de projet pendant que le projet avance

- Basés sur la réaction du client

LES ACTIVITÉS PRINCIPALES

uOttawa
L'Université canadienne
Canada's university

Elles doivent être réalisées indépendamment du modèle

Le modèle affecte simplement le flux entre les activités

MODÈLE CASCADE

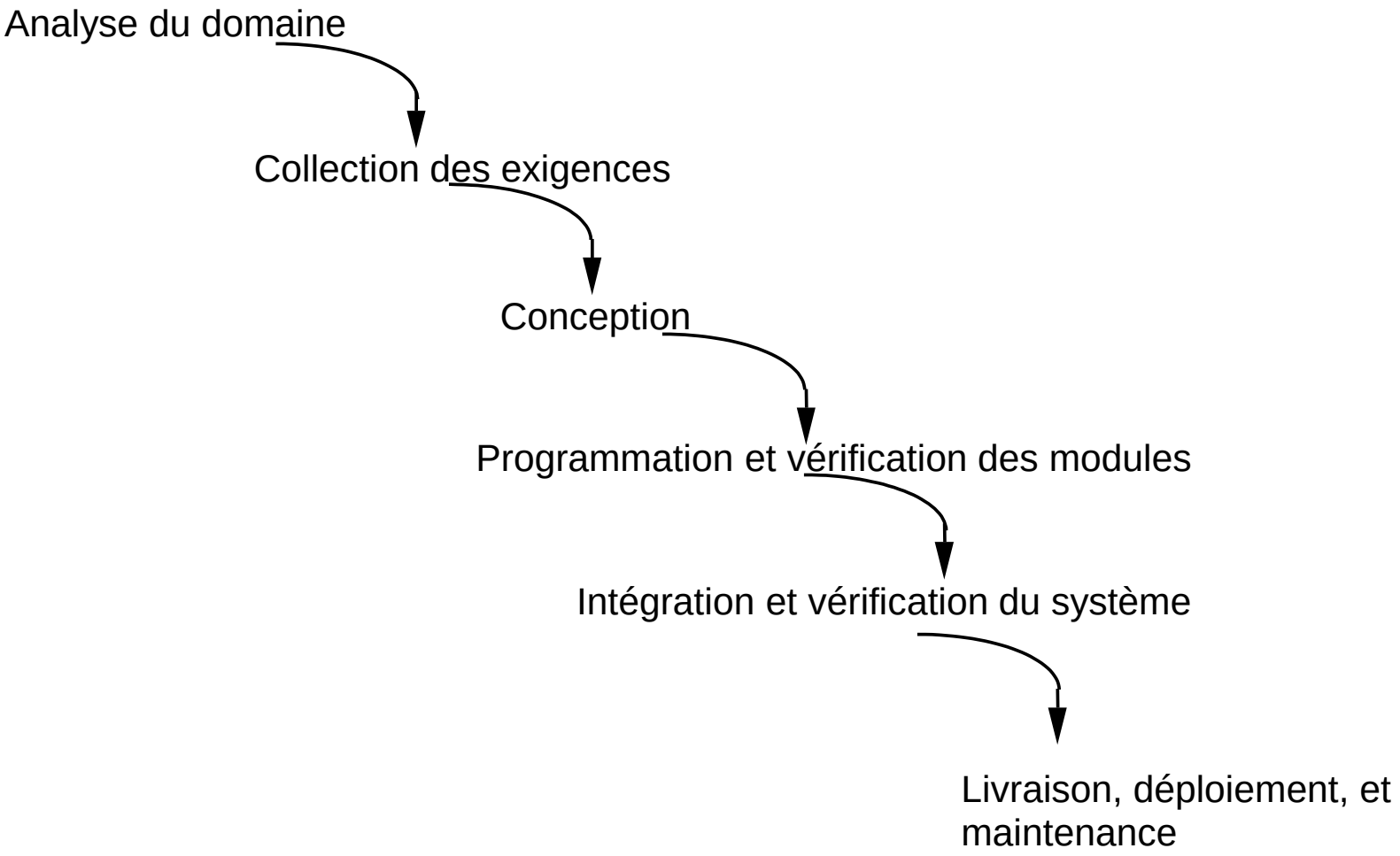
Inventé dans les années 1950 pour les grands systèmes de défense aérienne, popularisé dans les années 1970

Organise les activités dans un flux séquentiel

- On standardise les sorties des différentes activités

Existe en de nombreuses variantes, tous partageant le style de flux séquentiel

MODÈLE CASCADE



POINTS FORTS DU CASCADE

Facile à comprendre, facile à utiliser

Fournit la structure pour les personnels inexpérimenté

Les étapes sont bien comprises

Crée des exigences stables

FAIBLESSES DU CASCADE

uOttawa
L'Université canadienne
Canada's university

Toutes les exigences doivent être connus d'avance

Livrables de chaque phase sont considérés statique- inhibe la flexibilité

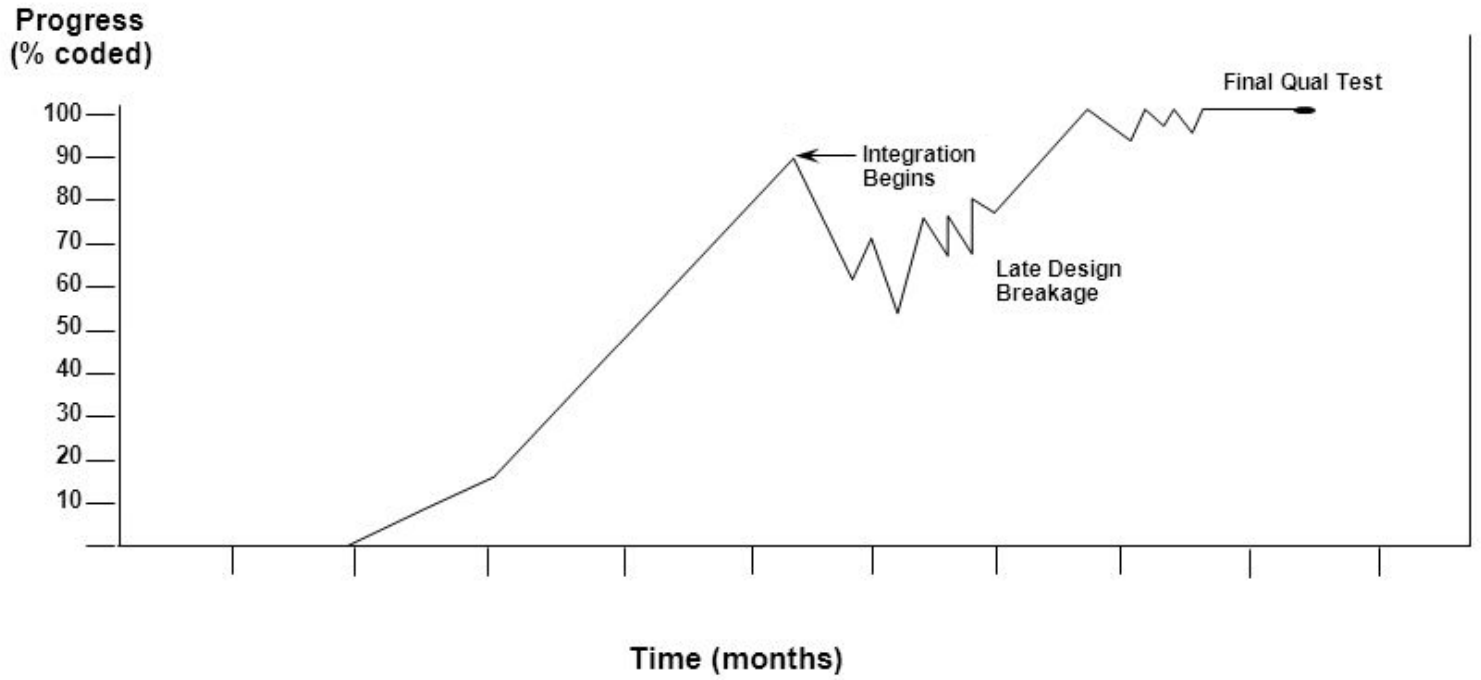
Peut donner une fausse impression de progrès

Ne reflète pas la vrais nature de développent de logiciel: résolution des problèmes d'une façon itératif

L'intégration est un big bang à la fin

Peu de chances pour que le client voie et vérifie le système (avant qu'il soit trop tard)

RUPTURE DE CONCEPTION TARDIVE



QUAND UTILISER CASCADE

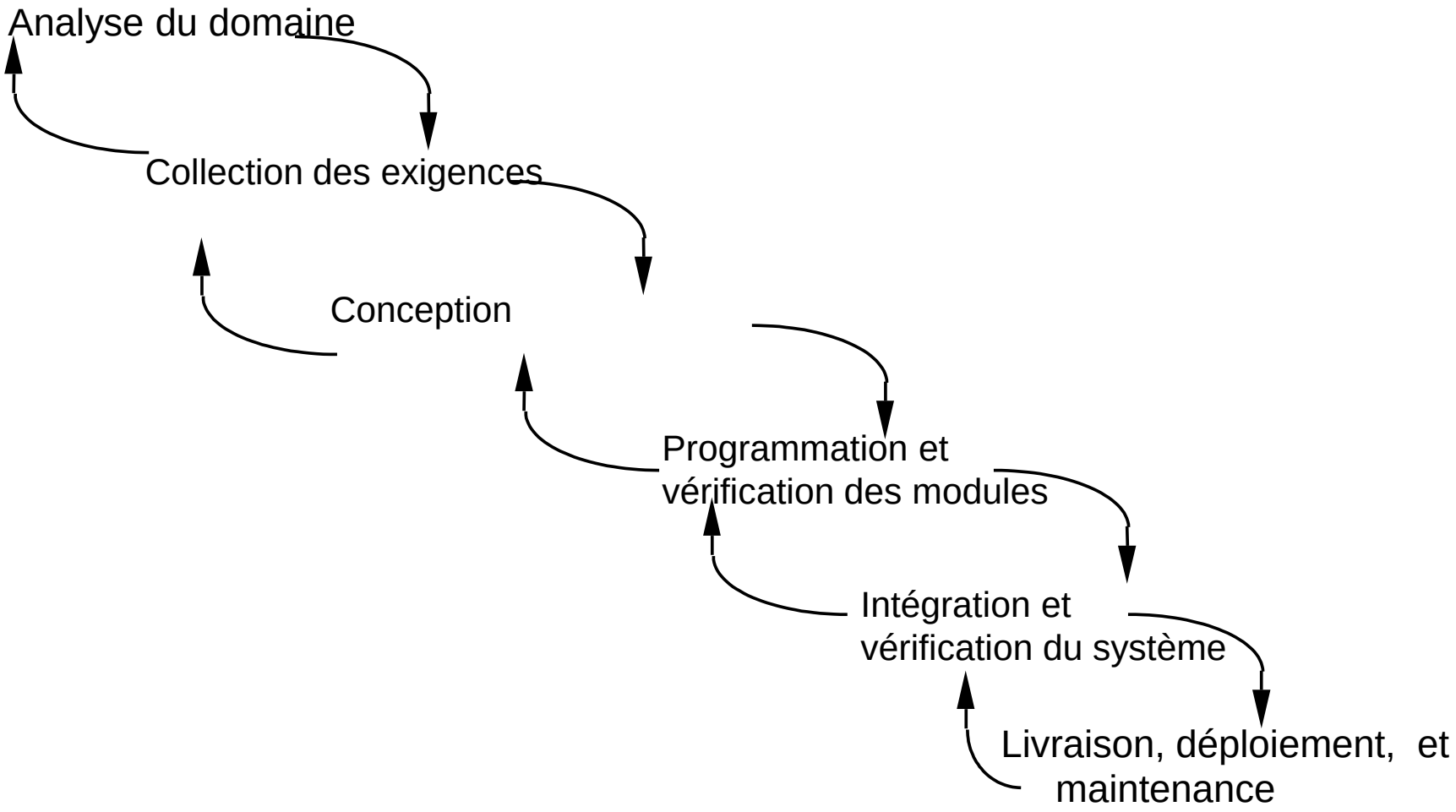
uOttawa
L'Université canadienne
Canada's university

Aujourd'hui, presque jamais!!

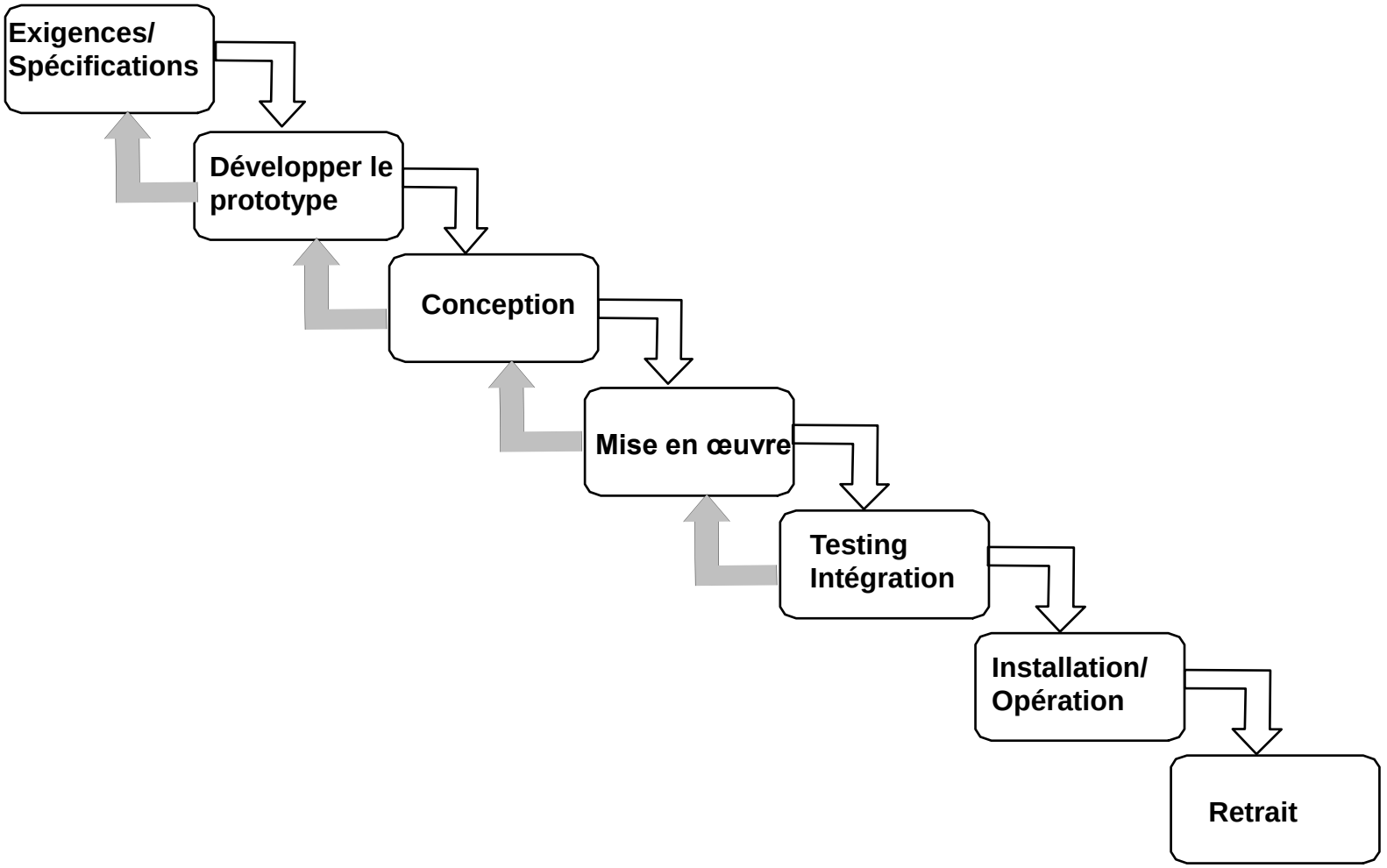
Mais, rarement, lorsque:

- Les exigences sont très bien connus
- Définition du produit est stable
- La technologie est ***très bien*** comprise
- Nouvelle version d'un produit existant *(peut-être!)*
- Portage d'un produit existant à une nouvelle plate-forme

CASCADE – AVEC RÉACTION



CASCADE AVEC PROTOTYPAGE RAPIDE



MODÈLE DE DÉVELOPPEMENT ITÉRATIF

uOttawa
L'Université canadienne
Canada's university

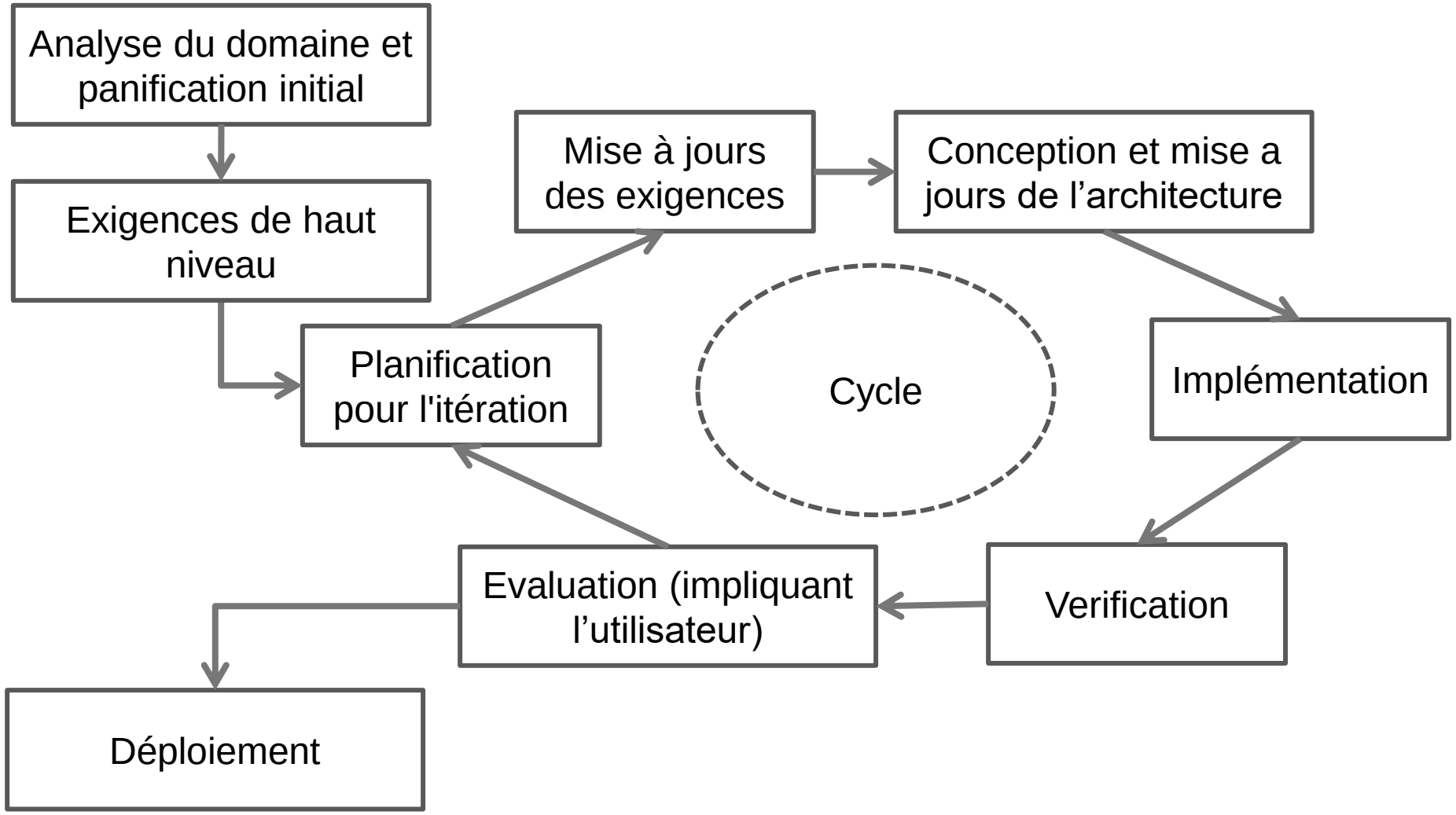
Développer le système par cycle répété (itérations)

- Chaque cycle est responsable du développement d'une petite portion de la solution (tranche de fonctionnalité)

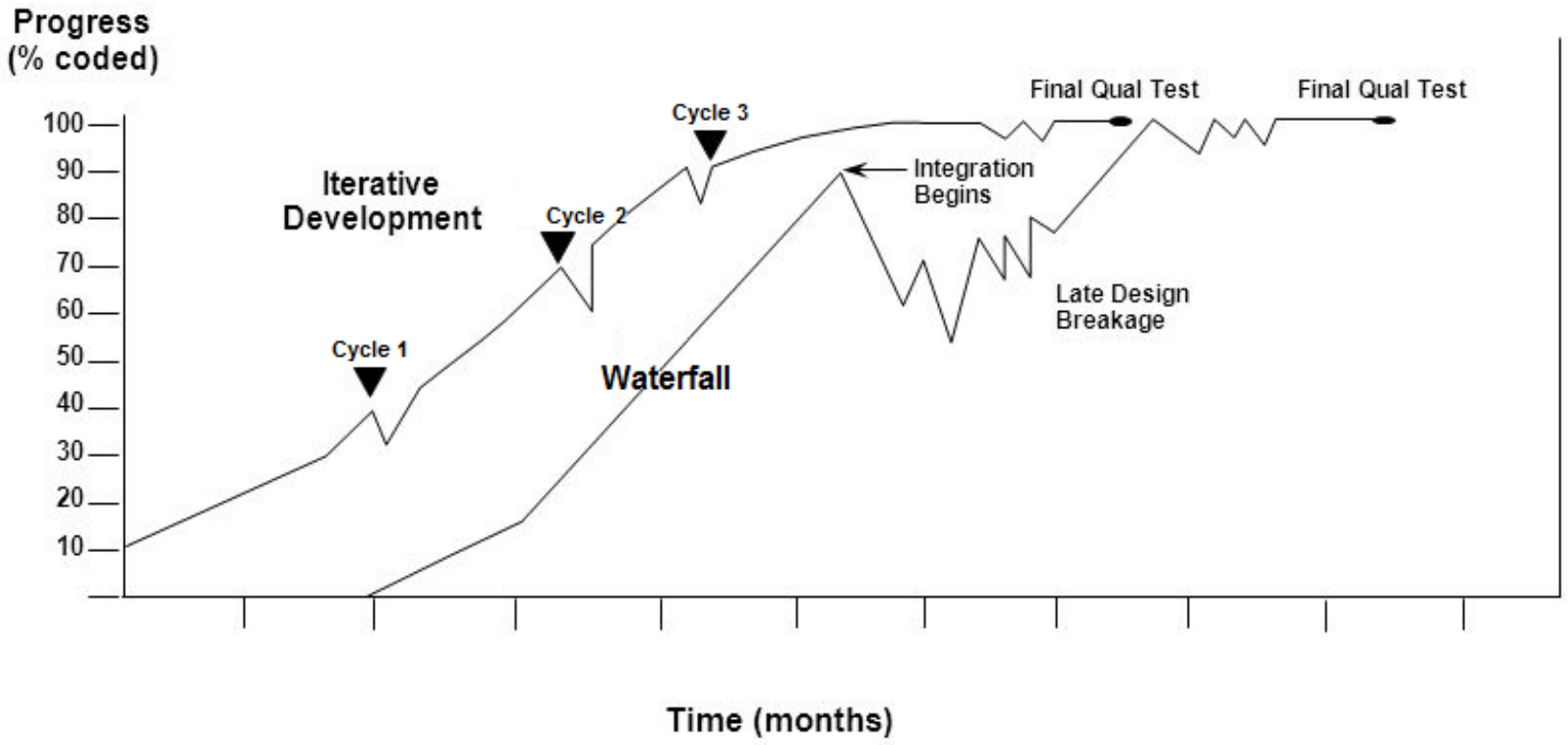
Contraste avec cascade:

- Cascade est un processus itératif particulier avec un seul cycle

PROCESSUS DE DÉVELOPPEMENT ITÉRATIF



INTÉGRATION CONTINUE



PROCESSUS AGILE

Insatisfaction avec les processus de développement des années 1980 et 1990 (après des faillites successive) a résulté dans la création de méthodes agiles

Ces méthodes:

- Concentre sur le code plutôt que la conception
- Sont basés sur une approche itérative pour le développement de logiciels
- Sont supposé d'aider a produire des logiciels rapidement
- Ces logiciels doivent être capable d'évoluer rapidement avec les changements dans les exigences

L'objectif de ces méthodes est de réduire les frais généraux (overhead) dans le processus de développement (par exemple limiter la documentation) et à être en mesure de répondre rapidement à l'évolution des exigences

MANIFESTE AGILE

Nous découvrons des meilleures façons pour développement de logiciels. Grâce à ce travail, nous sommes rendu-compte que:

Plus utile	Moins utile
Interaction entre les individus	Processus et outils
Logiciel qui fonctionne	Documentation compréhensive
Collaboration avec le client	Négociations prolongées sur les contrats
Réagir au changement	Suivre un plan

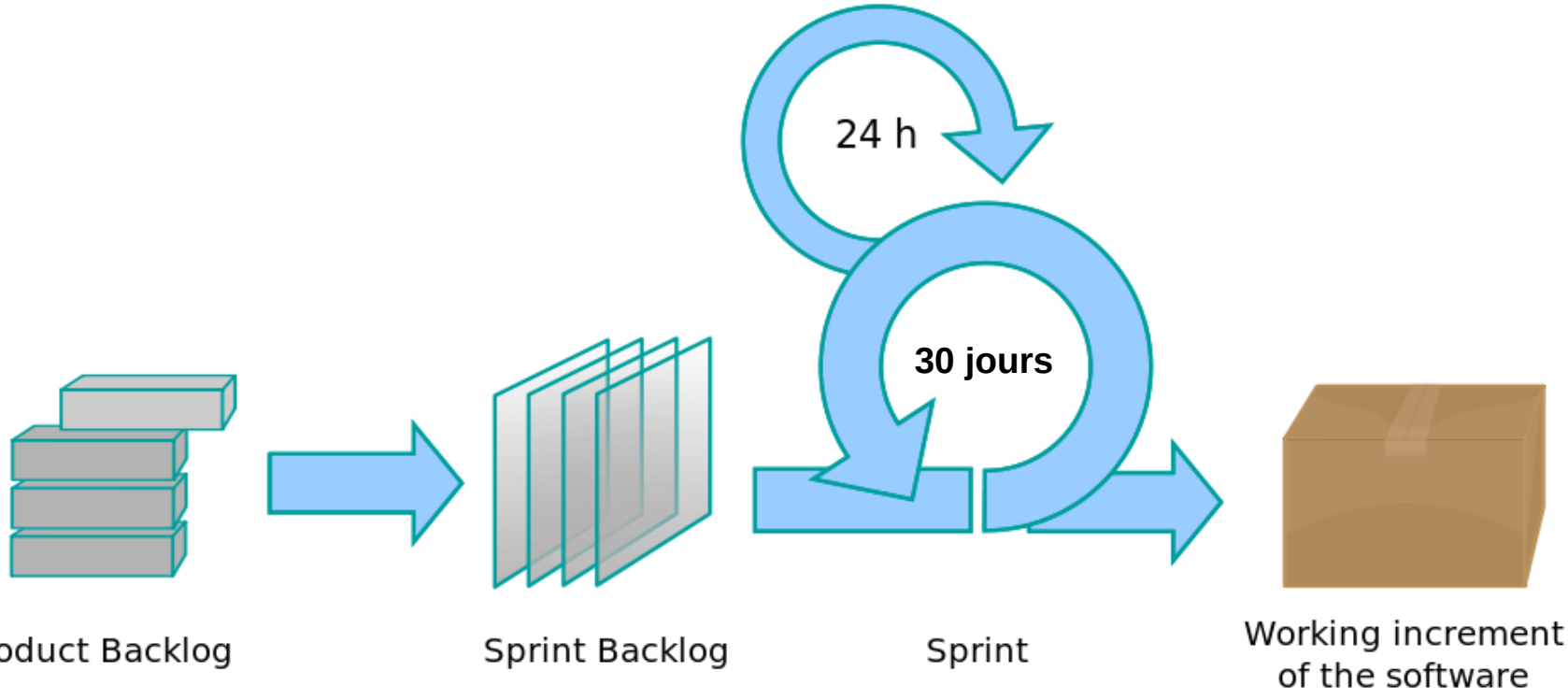
Cependant, les items sur la droite sont utiles, main nous accordant plus d'importance sur les items à gauche

LES PRINCIPES DES MÉTHODES AGILES

uOttawa
L'Université canadienne
Canada's university

Règle	Description
Participation du client	Les clients devraient être impliqués tout au long du processus de développement. Leur rôle est de fournir et de prioriser les exigences et d'évaluer les itérations déjà complétées.
Livraison incrémentielle	Le logiciel est développé par incréments avec le client spécifiant les exigences à inclure dans chaque incrément.
Individus et non processus	Les compétences de l'équipe de développement doivent être reconnus et exploités. Membres de l'équipe avoir la liberté de développer leurs propres méthodes de travail sans processus normatifs.
Acceptez les changement	Supposez que les exigences du système vont changer et concevez alors le système pour accommoder ces changements futurs.
Maintenez la simplicité	Concentrer sur la simplicité dans l'architecture du logiciel que vous développez et dans le processus de développement. Lorsque c'est possible, éliminer les complexités de votre système.

PROCESSUS SCRUM



(Backlog du Produit)

(Backlog d'un Sprint)

(Portion du logiciel déjà compléter)

PROBLÈMES AVEC LES MÉTHODES AGILES

Il peut être difficile de maintenir l'intérêt des clients qui sont impliqués dans le processus

Membres de l'équipe peuvent être inadaptées à l'intensité qui caractérise les méthodes agiles

Le minimisation de documentation: presque rien n'est capturé, le code est la seule autorité

MERCI!

QUESTIONS?