Chapitre 1

Génie logiciel et processus

Introduction au génie logiciel

1. Pourquoi un génie logiciel ?
2. Défis actuels en logiciel
3. Utilité d'une approche génie
4. Processus de développement

À partir de matériel produit par Mathieu Lavallée, Bram Adams, Michel Gagnon et Nikolay Radoev

Pourquoi un *génie* logiciel ?

- Grande variété dans les types de logiciels :
  - Logiciels de base,
  - Systèmes d'exploitation,
  - Applications Web,
  - Jeux vidéos (surtout à Montréal),
  - Logiciels embarqués (sur circuit électronique),
  - Modules d'intelligence artificielle,
  - Etc.
- Chaque logiciel est développé en utilisant une approche différente, qui demande **réflexion** !

Pourquoi un *génie* logiciel ?

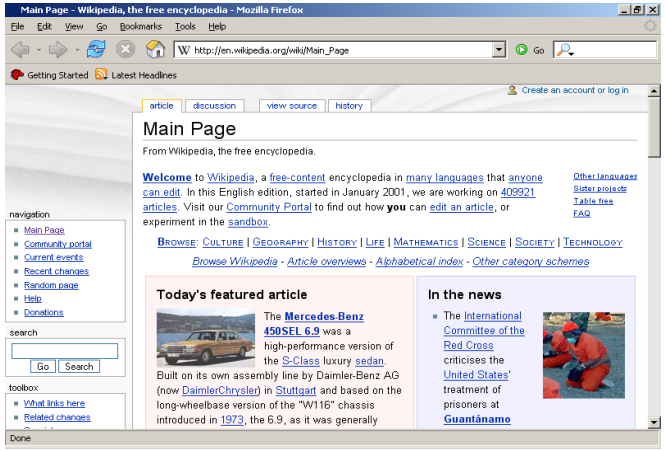
Exemple : On ne programme pas de la même manière un jeu vidéo et un électrocardiogramme.

Livraison : Pression du marché (ex.: sortir pour Noël, pour une plateforme, pour devancer la compétition).
Impact d'un bogue : On fera un correctif (« *patch* »).

Livraison : Préférable d'être en retard que de livrer un produit dangereux.
Impact d'un bogue : Peut être très grave.

Pourquoi un *génie* logiciel ?

Exemple : On ne développe pas un logiciel de la même manière qu'on le maintient.



Firefox version 1.0 (2004)

Firefox version 91.0.2 (2022)

Priorités du développement
Découvrir les besoins des clients, trouver des solutions, implémenter, et valider que ça correspond aux besoins, etc.

Priorités de la maintenance
Corriger les bogues, fermer les failles de sécurité, identifier des besoins émergents, assurer que l'architecture reste saine, etc.

Défis actuels en génie logiciel

- **Équipes globalisées :**
  - Airbus 380 : 16 sites dans 4 pays.
  - Problème : Utilisation de CATIA v5 en France et de CATIA v4 en Allemagne qui provoque une erreur de calcul → 530km de fils à changer ...
- **Équipes nombreuses :**
  - Entre 400 et 500 personnes pour un jeu AAA,
  - Environ 4000 personnes ont travaillé sur Windows 7.
- **Équipes multidisciplinaires :**
  - Programmeur *backend*, Programmeur *frontend*, architecte logiciel, expert sécurité, artiste visuel, artiste sonore, analyste des besoins, expert du domaine, testeur, gestionnaire de projet, etc.

Défis actuels en génie logiciel

La loi de Brooks :

- "Ajouter des développeurs à un projet en retard fera qu'il sera encore plus en retard."
  - Fred Brooks dans son livre *Mythical Man Month, 1975*
- Pourquoi ?
  - Besoin de former la nouvelle personne (souvent fait par les développeurs originaux),
  - La nouvelle personne doit lire toute la documentation du projet (s'il y en a une)
  - La nouvelle personne doit comprendre comment l'équipe fonctionne (dynamique souvent implicite),
- Conséquence :
  - La nouvelle personne est laissée souvent à elle-même et ne sait pas quoi faire.
  - Ce que la nouvelle personne produit n'est pas cohérent avec le reste et est donc rejeté.

Défis actuels en génie logiciel

Le marché demande plus de flexibilité :

- Accélérer le déploiement de correctifs.
  - Métriques: *MTTR* (*mean time to repair/recovery).*
- Raccourcir le temps entre l'énoncé du besoin et le déploiement de la nouvelle fonctionnalité.
  - Solutions: approches agiles, déploiement en parallel.
- Raccourcir le temps entre les versions pour donner une impression d'innovation continue.
  - Ex.: Chrome/Firefox « *browser wars* ».

Source : shoze.blogspot.com

Défis actuels en génie logiciel

Plusieurs processus différents dans l'industrie :

- **Firefox:**
  - *RapidRelease* → Six semaines.
- **Facebook**, **Netflix**, **Amazon** :
  - Implémentation en multiples micro-projets (*micro-services*),
  - Lorsqu'une micro-projet est prêt, il est mis sur le pipeline de déploiement,
  - Plusieurs déploiements par jour.
- **Eve Online** (jeu vidéo, processus hybride) :
  - *Frontend*, amélioration de l'interface : Six semaines.
  - *Backend*, changement majeurs dans l'architecture : Six mois.

Défis actuels en génie logiciel

Perception qu'on ne peut évaluer la qualité que d'un produit fonctionnel :

- Faux : La qualité doit être planifiée, évaluée durant la totalité du projet.
  - Qualité des exigences, qualité de la conception, qualité du code, qualité des tests eux-mêmes ...
- Il faut donc **réfléchir** aux activités de qualité nécessaires pour atteindre nos objectifs.

Objectifs typiques:

Produire un produit de **qualité**,
Avant la **date limite**,
Que ça ne **coûte pas trop cher**,
Et que ça se passe **agréablement**.

Défis actuels en génie logiciel

- Utiliser le logiciel ne l'use pas.
- Modifier le logiciel peut cependant causer de l' « usure » :
  - Appelé *software aging* ou *technical debt*.
  - Ex.: ajouter des fonctionnalités que l'architecture initiale ne supporte pas bien → problèmes de performance.
  - Ex.: faire des changements sans être pleinement conscients des conséquences (*ignorant surgery*) → introduction de bogues.
  - Ex.: ne pas supporter de nouvelles fonctionnalités qui deviennent populaires → dégradation de l'expérience utilisateur.
  - Etc.

On retire rarement des logiciels → le bagage logiciel des entreprises ne cesse d'augmenter ...

Exemples d'échecs majeurs

Therac-25 (1985-1987) : Machine de radiothérapie canadienne avec défauts logiciels :

- Réutilisation de code déficient → Code fonctionnel dans l'ancienne machine à cause de blocages matériels → Code réutilisé non-testé parce que l'ancienne machine n'avait jamais eu d'accident.
- Problème rare difficile à reproduire + Suivi inadéquat des plaintes = Plus de deux ans pour découvrir qu'il y avait un problème.
- Résultat → Application de 100x la dose prévue → Six incidents connus résultants en plusieurs morts.

Problèmes:

- Mauvaise approche de réutilisation du code,
- Mauvais suivi des plaintes,
- Mauvais enseignement : Même problème au Panama (2001) et avec la machine « Varian SRS » (2010).

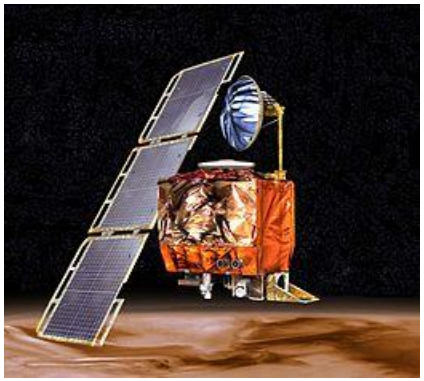
Exemples d'échecs majeurs

Mars Climate Orbiter (1999) : Mélange entre Newton-secondes (SI) et Livre-secondes (Impérial) :

- Exigences initiales envoyées au sous-contractant en impérial → Client s'attendait finalement à des données métriques.
- Problème de trajectoire détecté lors du vol entre la Terre et Mars → Les navigateurs et navigatrices n'ont pas été écoutés.
- Résultat → Perte d'une sonde de 330M$!

Problèmes :

- Mauvaise gestion des exigences : les documents des utilisateurs (NASA) et des développeurs (Lockheed) étaient différents.
- Manque de tests durant l'utilisation.
- Consultation inadéquate des experts.



Défis actuels en logiciel - résumé

Complexité :

- Tout élément logiciel est unique. Il n'y a pas de redondance, mais de la réutilisation.
- Le génie traditionnel se base sur des modèles simplifiés de réalités complexes. En logiciel, c'est le contraire.

Conformité :

- Pas de loi fondamentale comme en physique (ex.: E=mc2)
- Le logiciel doit se conformer à des règles arbitraires parfois illogiques (ex.: architecture von Neumann, TCP et QoS).

Flexibilité :

- Un bon logiciel aura des mises-à-jours importantes, qui peuvent toucher n'importe quelle partie de celui-ci.
- Les bâtiments changent aussi, mais il est plus facile de convaincre et de mesurer le coût et l'impact d'un changement ayant une réalité physique.

Avancement rapide :

- Les outils de développement (librairies, matériel, plateformes, etc.) évoluent très rapidement.

Qu'est-ce que l'approche d'ingénierie ?

- "The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software."
- « L'application d'une approche **systématique, disciplinée, quantifiable** au développement, l'opération et la maintenance du logiciel ; soit l'application de l'ingénierie au logiciel. »
    - IEEE (Institute of Electrical and Electronics Engineers)
- **Systématique** : pas limitée au code, mais appliqué à toutes les étapes.
- **Disciplinée** : basée sur une méthodologie structurée.
- **Quantifiable** : basée sur des données objectives.

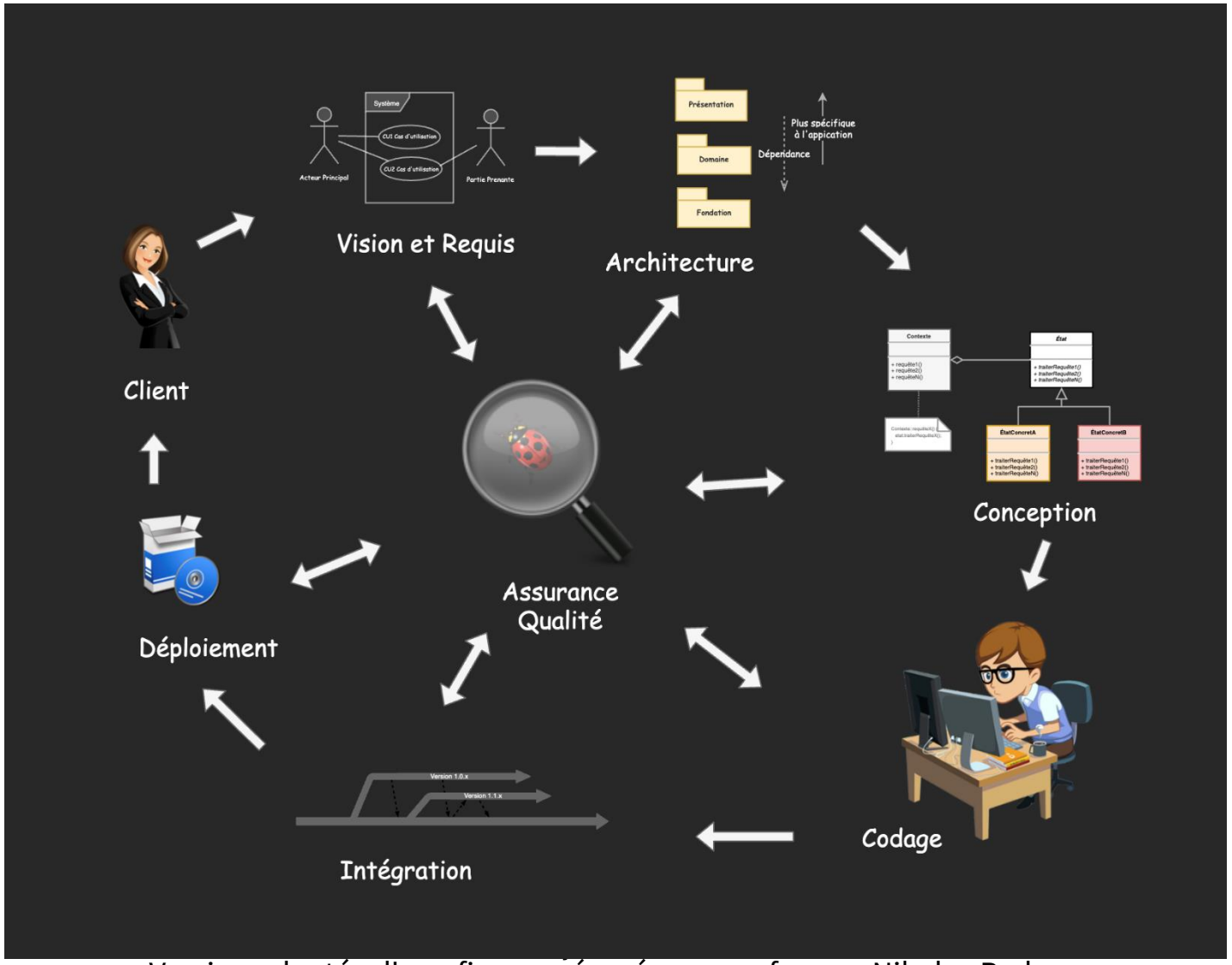
Qu'est-ce que l'approche d'ingénierie ?

- Exemple d'une donnée objective provenant de la recherche :
  - Dans un projet typique, entre 60% et 80% de l'effort consacré au logiciel a lieu après la livraison → correction de bogues et maintenance.
  - Conséquence → le développement doit produire un logiciel maintenable
  - « Le seul logiciel qui n'est pas modifié est celui qui n'est jamais utilisé. » - David Parnas (1994)

Qu'est-ce que l'approche d'ingénierie ?

Discipline	Exemple d'activités
Exigences	• Définition du problème • Définition des exigences (ou spécifications)
Conception	• Conception architecturale • Conception détaillée
Implémentation	• Codage et débogage • Tests unitaires • Intégration et tests d'intégration
Tests	• Tests boîte blanche (vérification) • Tests boîte noire (validation)
Déploiement	• Période de rodage (ajustements post-livraison) • Maintenance corrective (correction de bogues) • Maintenance perfective (ajout de fonctionalités)

Qu'est-ce que l'approche d'ingénierie ?



Version adaptée d'une figure préparée par professeur Nikolay Radoev

C'est quoi un ingénieur logiciel?

Voici ce qui fait un bon ingénieur logiciel, selon des recherches faites chez Microsoft (2015) :

- Être prêt à s'améliorer, à s'adapter,
- Bien connaître les gens et l'organisation,
- Être capable de communiquer avec les autres,
- Produire du matériel simple et intuitif.

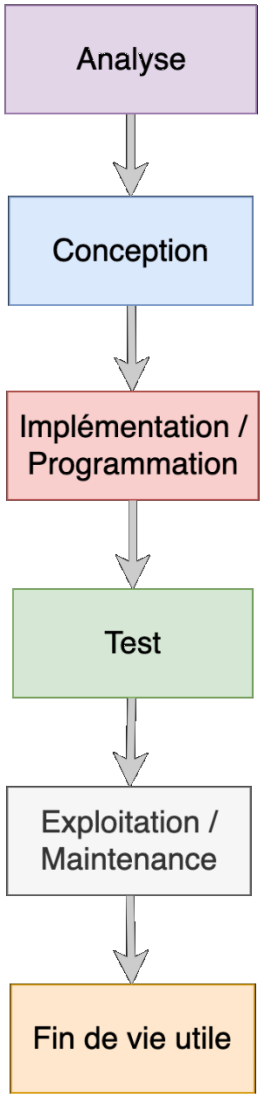
Où sont les compétences techniques ?

- Être bon programmeur se trouve plus loin derrière.

Une équipe de programmeurs ordinaires réussit typiquement mieux que des cracks qui travaillent de manière isolée.

- Un peu comme une équipe de hockey, de football.

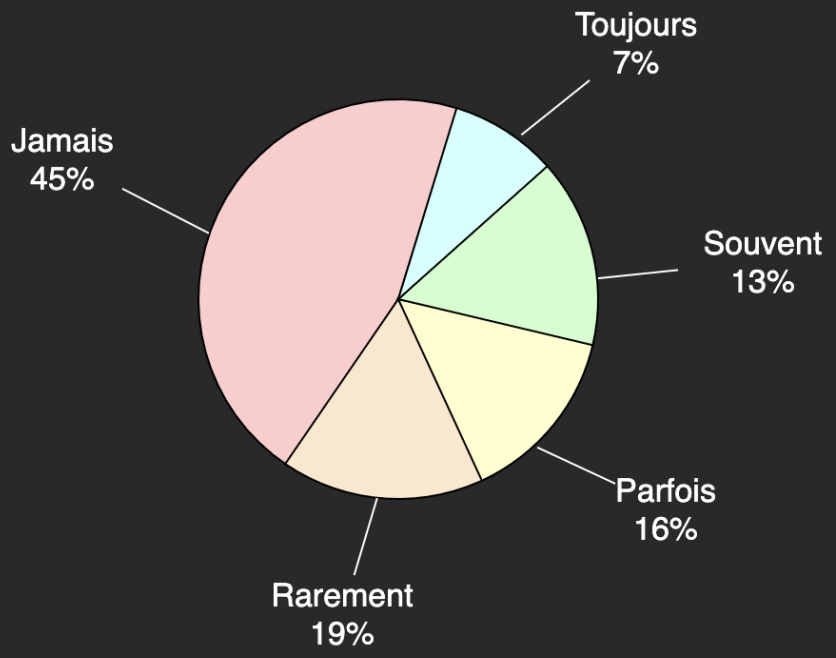
Processus de développement en cascade



Version adaptée d'une figure préparée par M. Éric Demers

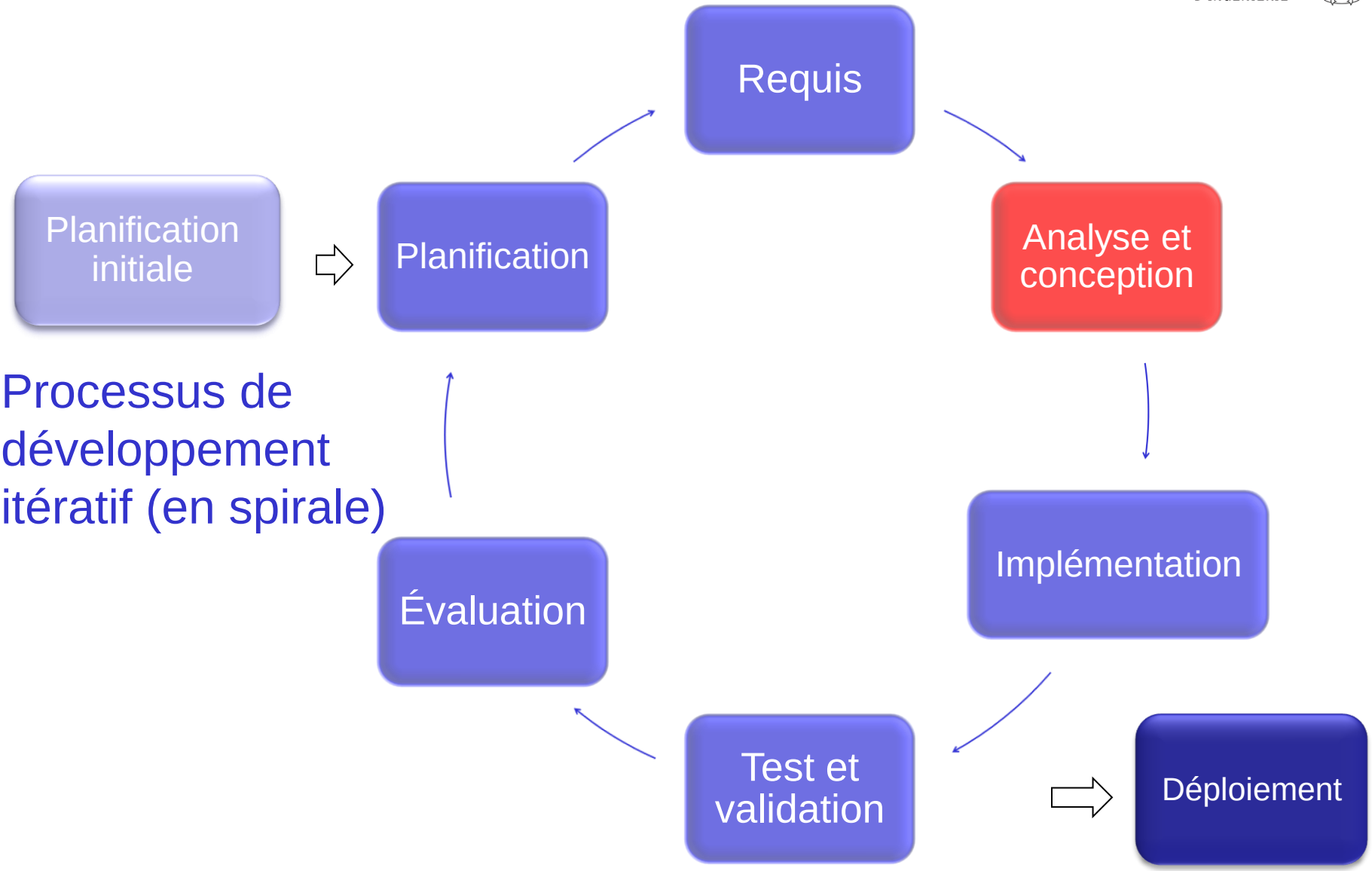
Approche en cascade

Combien de fonctionnalités spécifiées prématurément sont-elles réellement utiles dans le produit final?

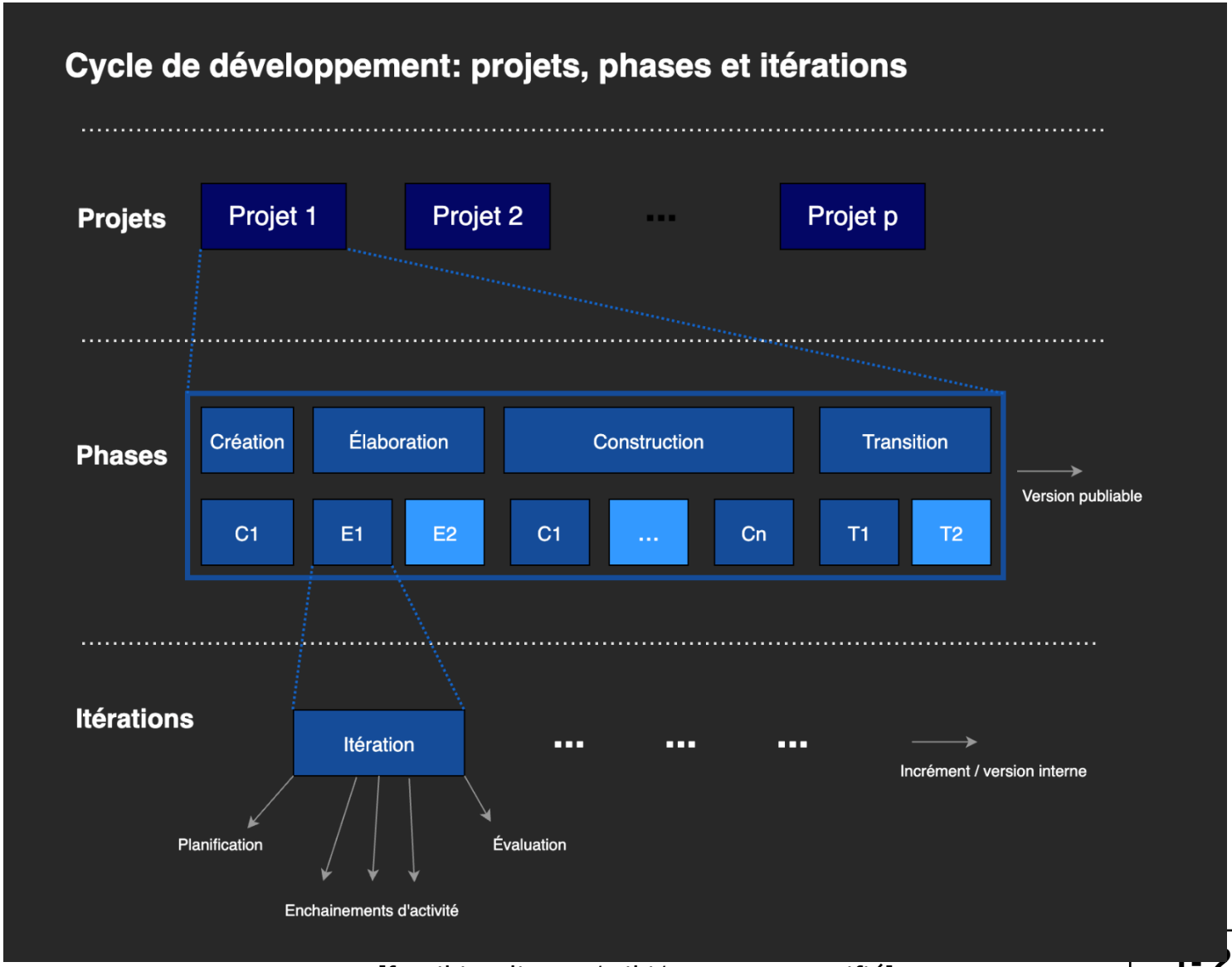


Source: Johnson, J. (2002). ROI: It's your job. *Proceedings of the Third International Conference on Extreme Programming and Flexible Processes in Software Engineering (XP 2002), Alghero, Sardinia, Italy.*

Processus de développement itératif (en spirale)



Processus unifié



[fr.wikipedia.org/wiki/Processus_unifié]

Enchaînements d'activités et la relation entre les disciplines et les phases