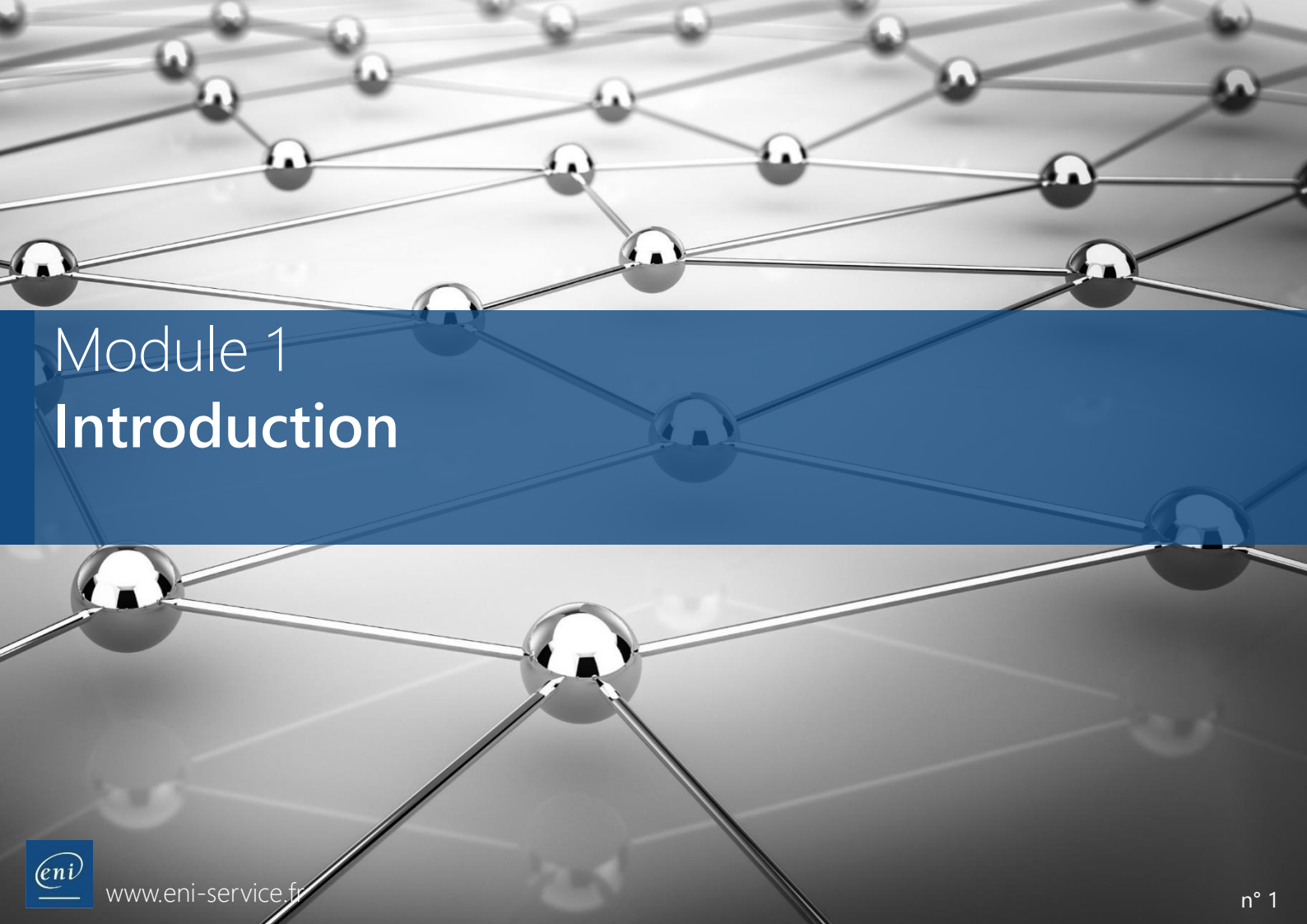
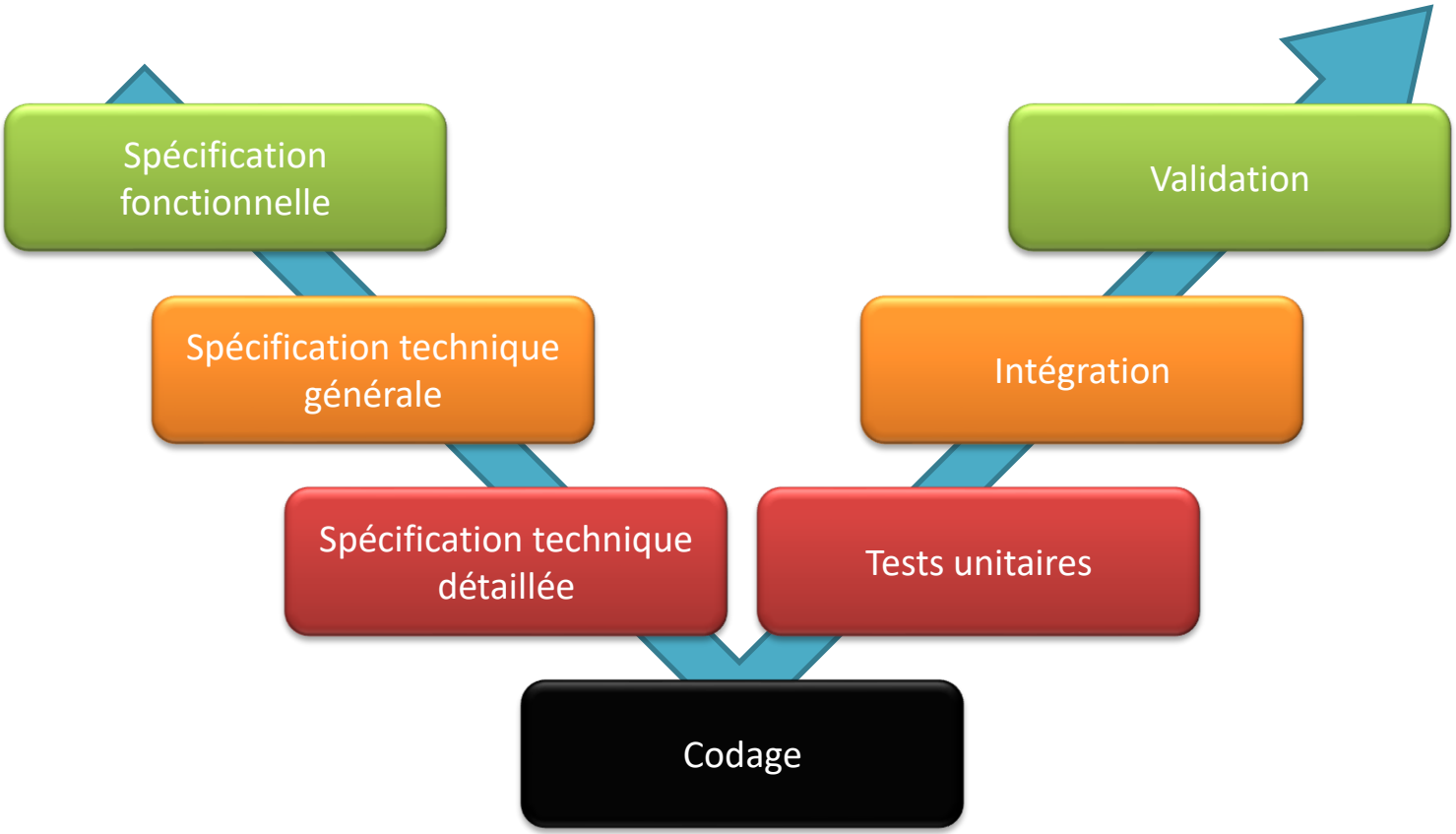


Module 1
Introduction

Contenu du module

- Tests et qualité logicielle
- La chaine d'intégration continue
- Principes de mise en œuvre intégration continue
  - Le serveur d'Intégration continue
- Le positionnement des différents types de tests
- Historique de Jenkins
- Extensibilité de Jenkins par les plugins
  - Les plugins les plus populaires

Cycle en V

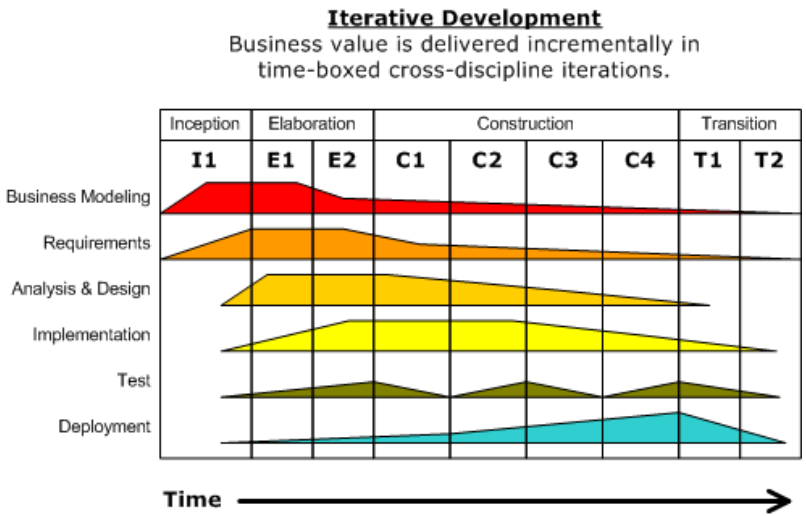

Spécification fonctionnelle
Spécification technique générale
Spécification technique détaillée
Codage
Tests unitaires
Intégration
Validation

Cycle en V

- Risque
  - Taille du projet : quelles bornes ?
  - Effet tunnel
    - Tant qu'on n'a pas fait la validation, pas d'idée de la conformité
    - Difficulté d'appréhender la bonne marche
  - Les tests sont d'autant plus cruciaux
    - Les méthodes associées au cycle en V les intègrent : plan qualité
    - Notion de « campagne de test »

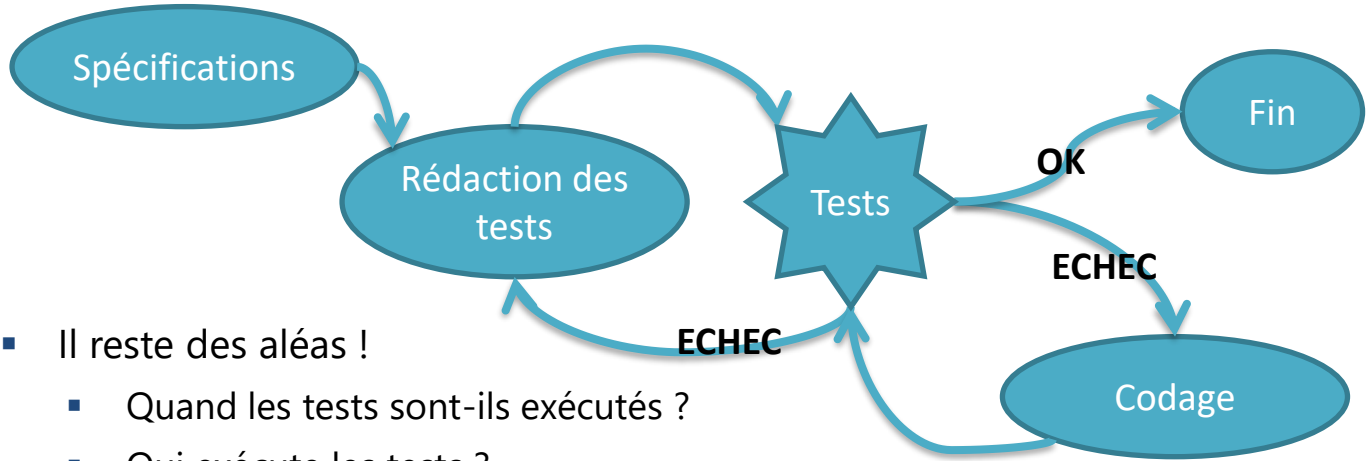
RUP : Rational Unified Process

- Ancêtre des méthodologies Agiles
- But raccourcir les cycles de développements
  - Itérations qui bornent la complexité
  - Maximum 3 mois
  - Parallélisation possible/souhaitable
- Risque :
  - Avoir des régressions d'une itération sur l'autre
  - Multiplier le coût des campagnes de tests



Frameworks de tests

- Evolution capitale
- Idée : programmer les tests des programmes
- Démarches associées : Peer Programming, Test Driven Development



- Il reste des aléas !
  - Quand les tests sont-ils exécutés ?
  - Qui exécute les tests ?
  - Que se passe-t-il en cas d'échec ?
    - Techniquement qui est prévenu, quel est le workflow ?
    - Humainement : le fautif est identifiable (SCM). Comment est-ce perçu et par qui ?

Les frameworks de tests

- Un programme de tests
  - Doit mettre en œuvre un aspect du code
  - Vérifie ses comportements attendus :
    - En cas de fonctionnement nominal
    - En cas de fonctionnement dans un environnement dégradé
  - Emet un résultat binaire : Succès/Echec
- Des frameworks
  - Pour éviter de réécrire ces concepts
  - Pour disposer d'un « lanceur » qui exécute les tests et affiche le verdict
- xUnit
  - Une famille de frameworks, déclinés selon les langages de programmation
  - Sous l'impulsion de Kent Beck pour Smalltalk, puis Erich Gamma, Martin Fowler pour Java
  - Les plus connus : JUnit, CppUnit, NUnit, PHPUnit, PyUnit, …

Les types de tests

- Tests unitaires
  - Test d'une fonctionnalité unique
  - Au niveau des fonctions/méthodes
  - Simples à écrire mais très nombreux
- Tests d'intégration
  - Test de composants
  - Vérification du comportement dans l'environnement
    - Serveur Web, Base de données, …
  - Relativement simples à écrire (si outillés)
    - Mais l'environnement de test peut être complexe à mettre en place et à réinitialiser entre chaque tests
- Tests fonctionnels
  - Test de scénario utilisateur
  - Impliquent la rédaction de l'ensemble des scénarios et leur jeu (et rejeu)
  - Mettent en œuvre l'IHM
    - Donc plus ou moins facilement programmable
  - Longs en rédaction/enregistrement et ajustements
- Tests de charge
  - Comportement en charge réelle (idéalement)
  - Les scénarios sont écrits ! (Tests fonctionnels)
  - Impliquent un environnement proche de celui de production

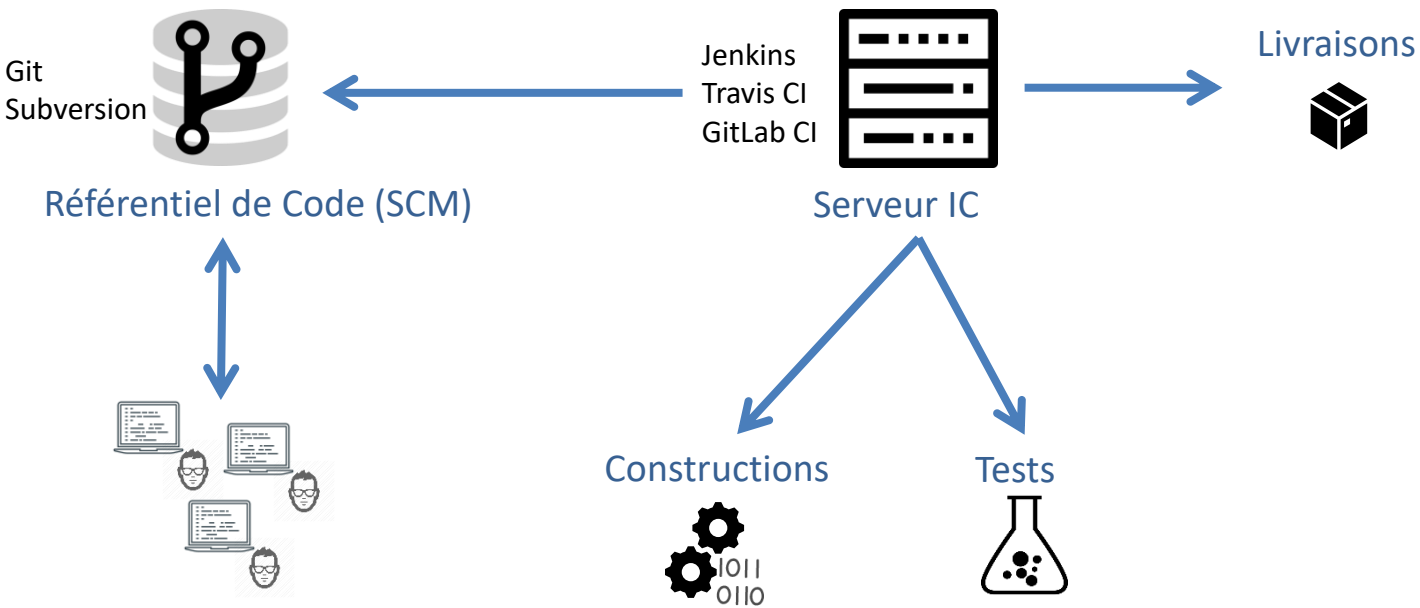
Intégration continue

- Les principes
  - Introduire des bugs est inévitable donc, pas grave
  - Ce qui est grave c'est :
    - De ne pas s'en rendre compte ou (trop) tard
    - De ne pas corriger
- L'outil : un ordonnanceur
  - Lancer les tests programmés
    - De façon automatique
      - Ce n'est pas un responsable hiérarchique qui apporte la « mauvaise nouvelle »
    - Très fréquemment
      - Détection des problèmes au plus tôt
    - Pas d'effet tunnel
      - Suivi du niveau de conformité en permanence
      - Permet de corriger au plus tôt
      - Les tests sont la spécification : on sait si on converge vers la solution

L'intégration continue

- **L'intégration continue consiste à déléguer à un automate la construction périodique de tous les projets**
  - Et donc l'exécution des tests
- L'utilisation d'un serveur d'intégration continue est essentielle pour exécuter ces tests.
  - Jenkins, GitLab-CI, Travis-CI, …
- Le serveur peut alerter si les tests ne fonctionnent pas
  - Evite d'updater un projet si un collègue a publié des bugs
- Bonne pratique
  - Toujours faire passer les tests unitaires avant de valider les modifications dans le SCM !
    - Renforce l'argument que ces tests doivent être rapides
    - Certains IDE proposent cette fonctionnalité (ils refusent de valider vers le SCM s'il y a des erreurs)

Intégration Continue



L'approche DevOps

- Un ensemble de pratiques et de principes qui visent à rapprocher les développeurs et les exploitants, à fluidifier les développements et ainsi créer de la valeur plus rapidement pour l'entreprise.

> Mouvement en ingénierie informatique et une pratique technique visant à l'unification du développement logiciel (*dev*) et de l'administration des infrastructures informatiques
>
> https://fr.wikipedia.org/wiki/Devops

- Un ensemble de bonnes pratiques associé à des outils

#COLLABORER #OPTIMISER #LIVRER

Présentation de Jenkins

- Serveur d'intégration continue développé en Java
  - Mais il sait construire tous types de projets !
- Licence Open Source
- Extensible par un mécanisme de plugins
  - Un certain nombre de plugins sont installés par défaut avec Jenkins
  - D'autres plugins peuvent être ajoutés pour prendre en charge l'intégration de technologies (langage de développement par exemple) ou d'outils (Maven, MSBuild, …)
- https://jenkins.io