Cours 2 – Méthodologies de Développement

MODULE INTRODUCTION AU GÉNIE LOGICIEL

MOSTEFAI MOHAMMED AMINE – BATATA SOFIANE

Objectifs du Cours

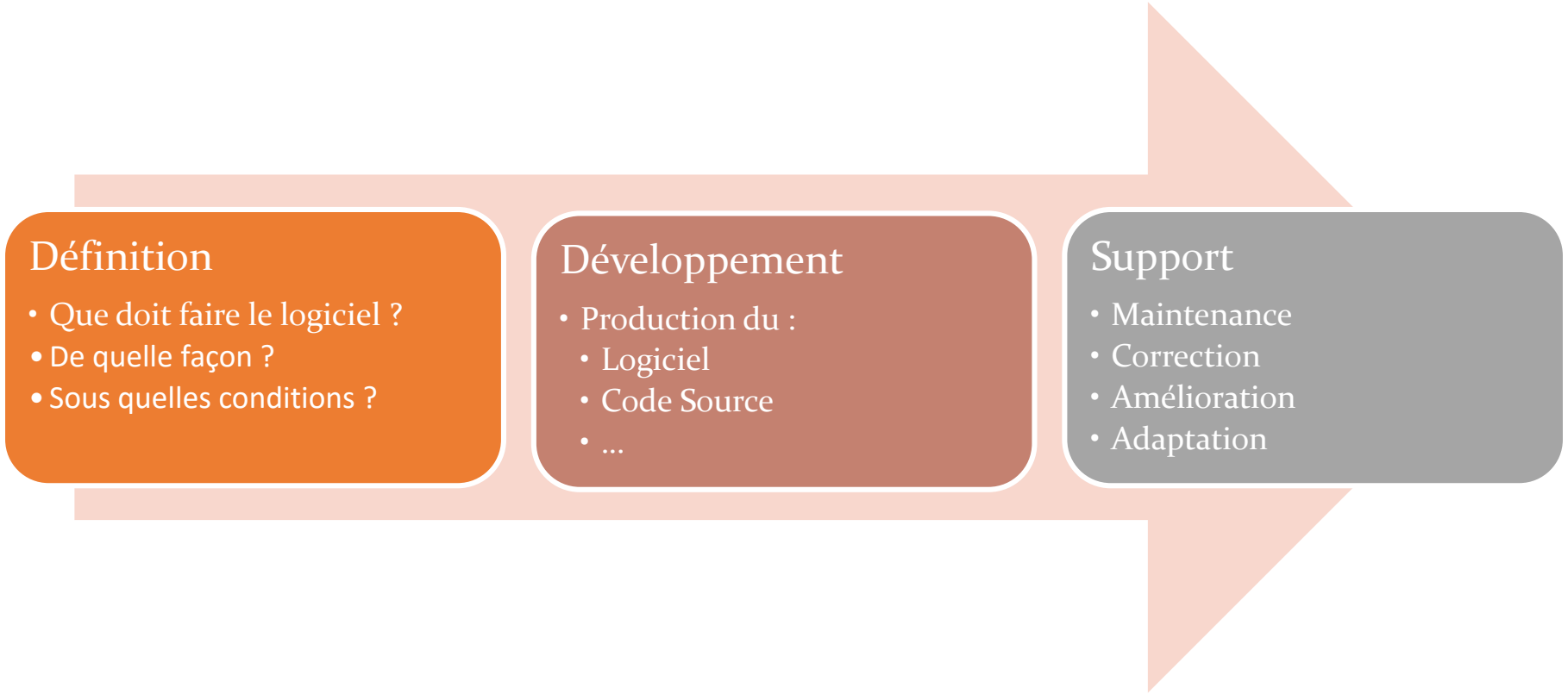
- Découvrir les principales activités de développement de logiciels
- Connaître les méthodologies et leur philosophies
- Connaître les méthodologies classiques et les méthodes agiles
- Pourvoir choisir une méthodologie sur la base des données concernant un projet de développement
- Prise de contact avec la méthodologie UP
- Découvrir les outils de support (CASE)

Plan du Cours

Activités de Développement

SECTION 1

Etapes de développement



Activités

- Chaque projet de développement est composé de plusieurs ***activités***
- Chaque activité est conduite et réalisée par plusieurs ***acteurs***
- Une activité a des entrées et des sorties. Les livrables font partie des sorties des activités,
- Les livrables sont des produits ou des documents produits par une activités et utilisé par les activités qui en dépendent
- Par exemple : document, planning, code source sont tous des livrables

Principales activités

Analyse de besoins

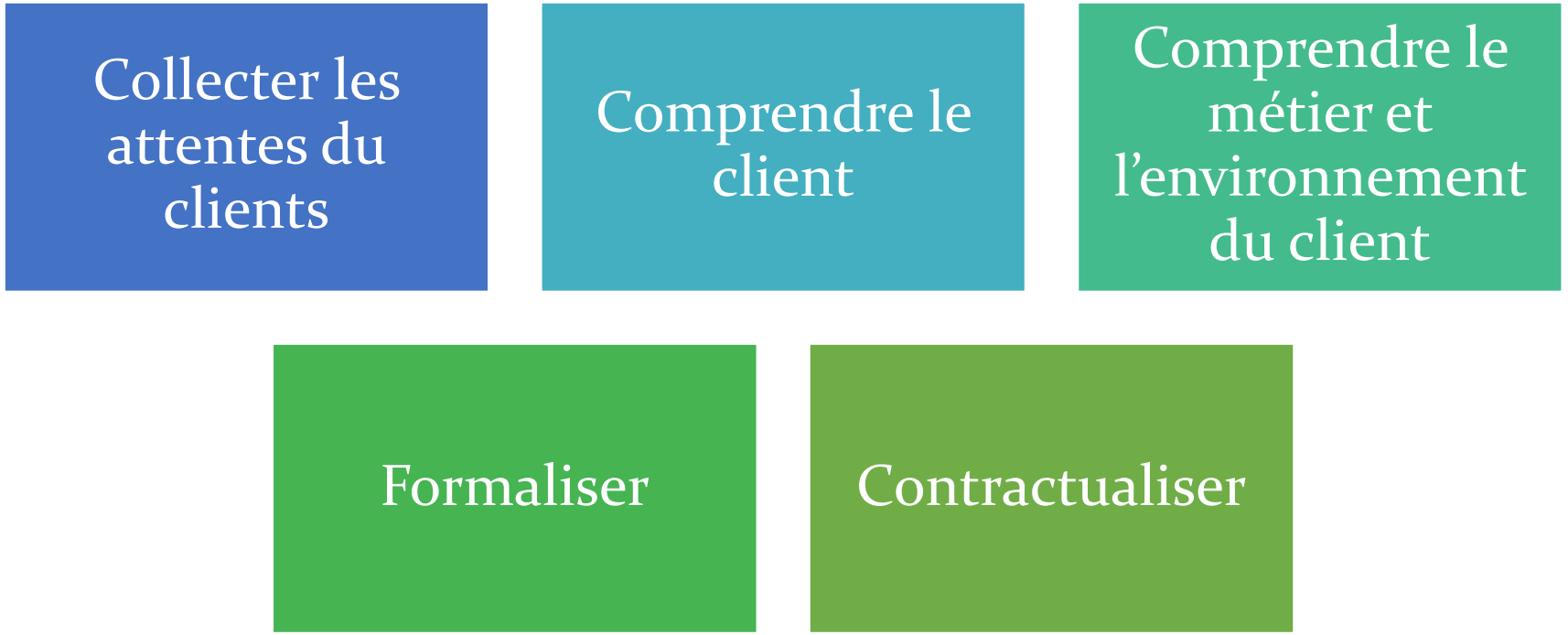
Conception

Codage

Tests

Maintenance

Analyse de besoins



Analyse de besoins → Difficultés

Le client parle un autre langage

Incompréhensions, oublis, complexités,…

Difficultés d'estimation

Changement des besoins en cours de projet

Conception

Choix des solutions techniques permettant de répondre aux attentes

Aide à l'établissement d'un planning de la solution

Élaboration de ***l'architecture*** de la solution

Maquettes, prototypes,…

Connaissances techniques mises en oeuvre

Conception → Difficultés

Dépendance forte envers du résultat de l'analyse des besoins

Plusieurs solutions possibles et choix de la meilleure solution

Nécessite une compétence technique accrue

Evolution très rapide de la technologie

Codage

Transformation des solutions proposées lors de la conception en un ***code opérationnel***

Basé sur les langages de programmation

La plus grande part du travail

L'équipe la plus grande

Utilisation d'un référentiel unique du code source (emplacement, modèles,…)

Codage → Difficultés

- Gestion de projets pour équipes nombreuses
- Intégration du code source
- Uniformisation de la compréhension du projet
- Uniformisation des méthodes de travail
- Mobilité des développeurs
- Différence de niveau technique entre développeurs

Tests

Détermination de la qualité du logiciel

Conformité du le logiciel par rapport aux spécifications

Plusieurs types de test dont deux principaux : tests unitaires et tests fonctionnels

Tests en boîte blanche : avec accès au code source

Tests en boîte noire : sans accès au code source

Tests → Difficultés

Nécessite concentration

Souvent lassant

Optimisation : trouver le maximum de défaillances en un moins de temps

Difficilement automatisable

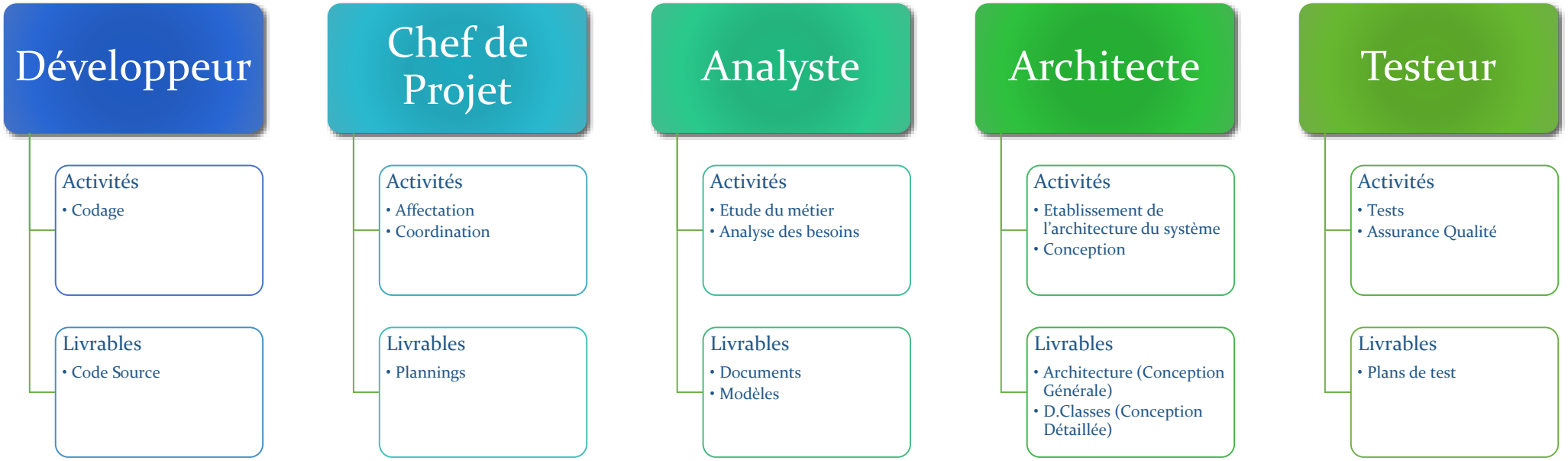
Activités de Développement

SECTION 1 – DÉBAT (05 MNS)

Outils et Métiers

SECTION 2

Principaux Métiers de Développement



Métiers et Activités

	Expression de besoins	Analyse	Conception	Implémentation	Tests
Développeur					
Analyste					
Architecte					
Testeur					
Chef de Projet					

Outils CASE

- CASE est un nom donné aux ***logiciels*** utilisés dans les différentes activités de GL (besoins, conception,…)
- Exemples : compilateurs, éditeurs, débogueurs, …etc.
- Le but des outils CASE est d'automatiser les tâches et / ou gérer le projet de développement

Classification des CASE

Les outils CASE peuvent être classés :

- D'un point de vue fonctionnel : selon la fonction de l'outil.
- D'un point de vue activité : selon les activités dans lesquelles intervient l'outil

Classification fonctionnelle

Outil	Exemples	Références	Métier
Outils de planification	Outils PERT, outils d'estimation, tableurs	Microsoft Project, Excel, GanttProject, DotProject	CDP
Editeurs	Editeurs de texte, éditeurs d'image, éditeurs de diagrammes	vi, bloc notes, GIMP, Photoshop, Visio	Tous
Gestion de configuration	Gestion de versions, gestion de builds	SVN, CVS, Team Foundation Server, ClearCase	CDP, Développeur, Architecte
Outils de support de procédé	Générateurs de code, outils d'assistance, IDE	Team Foundation Server, Accurev, Enterprise Architect	Tous
Outils de traitement de langage	Compilateurs, interpréteurs, débogueurs		Développeur, Architecte

Classification fonctionnelle - Suite

Outil	Exemples	Références	Métier
Outils de test	Environnements de tests, outils de tests unitaires	Junit, Nunit, TestWorks, Bugzilla	Testeur, Développeur
Outils de documentation	Documents de projet, documents de code	Word, Open Office, Sandcastle, Doxygen, javadoc	Développeur

Outils et Métiers

SECTION 2, DÉBAT 05 MNS

Méthodologies de Développement

SECTION 3

Qu'est-ce qu'une méthodologie ?

- Une ***méthodologie de développement***, appelée aussi ***procédé logiciel*** (software process) est un ensemble d'activités conduisant à la production d'un logiciel
- Les méthodologies sont aussi appelés cycle de vie d'un logiciel (SDLC)
- Une méthodologie définit les ***étapes*** qui composent un projet de développement ainsi que leur enchaînement
- La méthodologie définit comment les activités de développement sont ***affectées*** aux développeurs
- Les Méthodologies de développement sont ***complexes*** et dépendent fortement des acteurs qui dirigent les activités
- Les activités ne peuvent être automatisées mais il y a des outils de support, appelés outils CASE (Computer-Aided Software Engineering)

Motivations

Maîtriser les gros projets

Découper le projet et affecter correctement les tâches

Anticiper et gérer les *risques*

Réduire la complexité

Générations de méthodologies

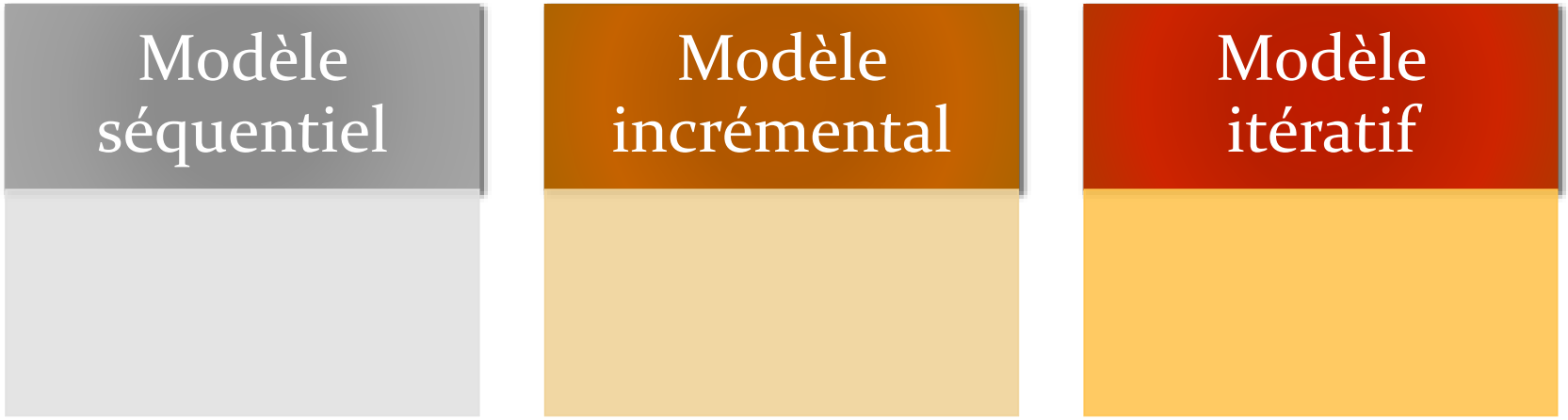
Méthodologies classiques

- Modèles stricts
- Etapes très clairement définies
- Documentation très fournie
- Fonctionne bien avec les gros projets et les projets gouvernementaux

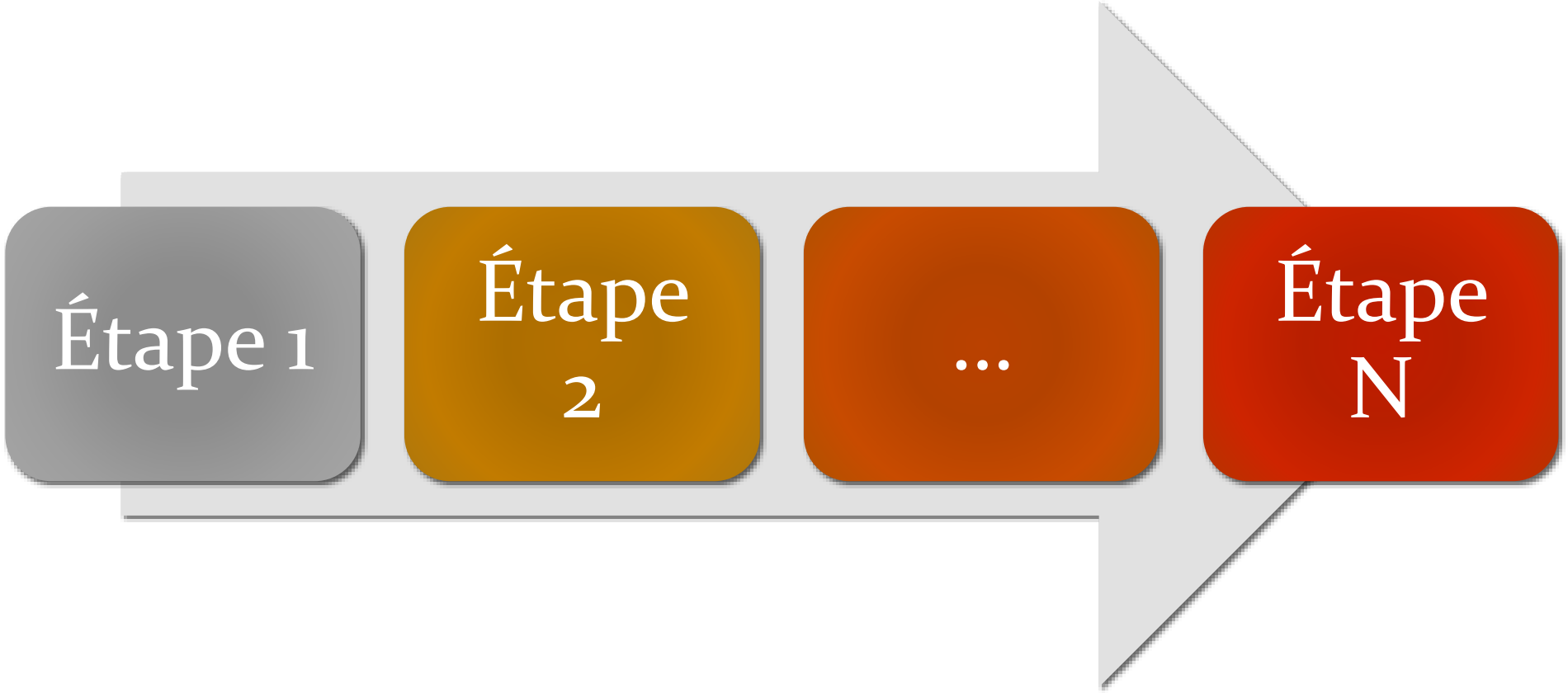
Méthodes agiles

- Modèles incrémentaux et itératifs
- Petites et fréquentes livraisons
- Accent sur le code et moins sur la documentation
- Convient aux projets de petite et moyenne taille

Typologie



Modèle séquentiel



Modèle incrémental

1 

2 

3 

Modèle itératif

1

2

3

Quand Utiliser Une Méthodologie X ?

- Nature du Projet
- Taille du Projet
- Nature du client
- Exigences du contrat
- Compétences de l'équipe

Méthodologies de Développement

SECTION 3, DÉBAT (10 MNS)

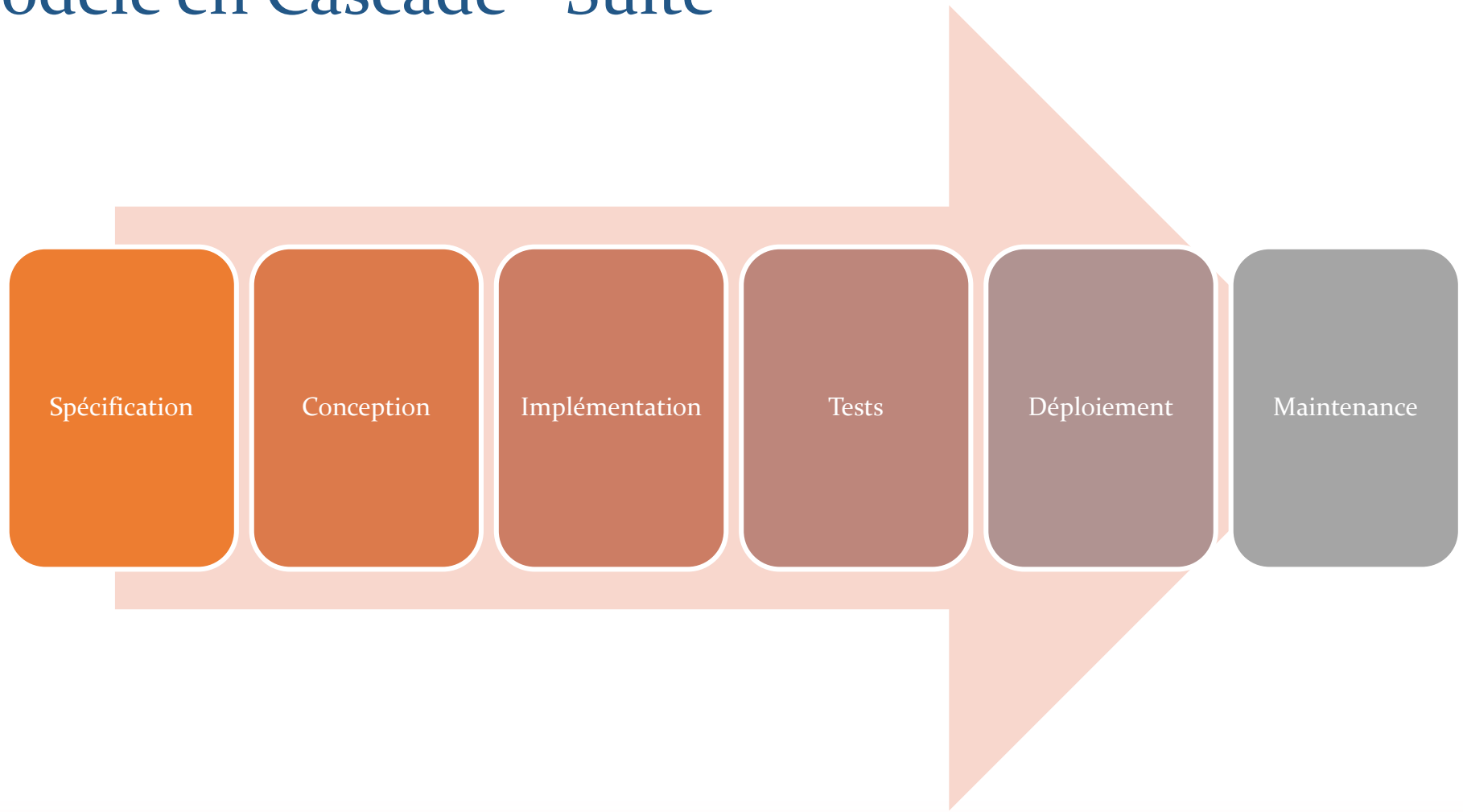
Méthodologies de Développement Classiques

SECTION 4

Modèle en cascade

- L'un des premiers modèles proposés, inspiré du modèle de Royce (1970)
- Aussi appelé modèle ***linéaire***
- Le résultat de chaque phase est un ensemble de livrables,
- Une phase ne peut démarrer que si ***la précédente est finie***
- Le modèle ***académique*** par excellence

Modèle en Cascade - Suite



Avantages du Modèle en Cascade

Facile à utiliser et à comprendre

Un procédé structuré pour une équipe inexpérimentée

Idéal pour la gestion et le suivi de projets

Fonctionne très bien quand la qualité est plus importante que les coûts et les délais

Inconvénients du Modèle en Cascade

Les besoins des clients sont très rarement stables et clairement définis

Sensibilité aux nouveaux besoins : refaire tout le procédé

Une phase ne peut démarrer que si l'étape précédente est finie

Le produit n'est visible qu'à la fin

Les risques se décalent vers la fin

Très faible implication du client

Modèle en Cascade – Quand l'Utiliser (Indications) ?

Quand les besoins sont connus et stables

Quand la technologie à utiliser est maîtrisée

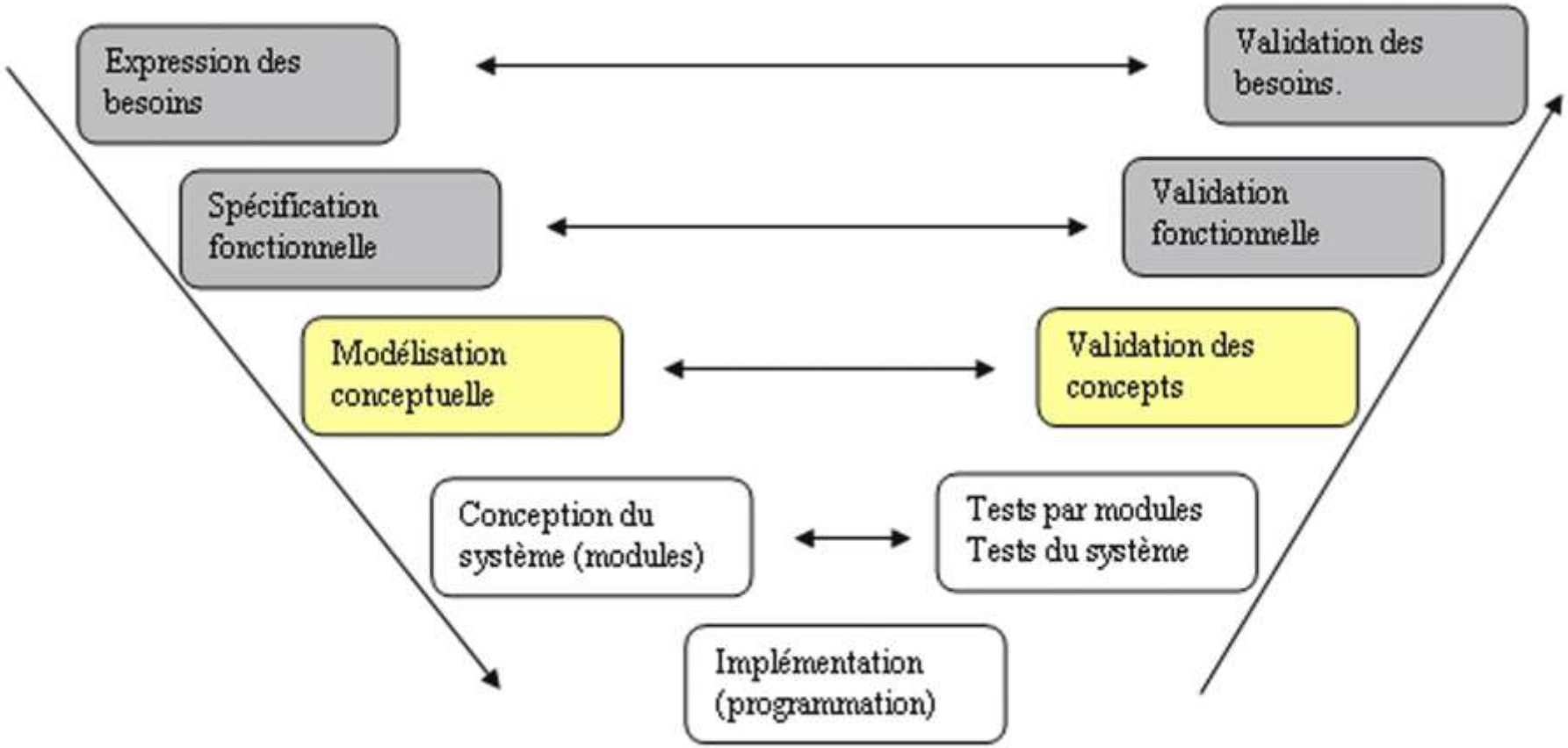
Lors de la création d'une nouvelle version d'un produit existant

Lors du portage d'un produit sur une autre plateforme

Modèle en V

- Variante du modèle en cascade qui fait l'accent sur la ***vérification*** et la ***validation***
- Le test du produit se fait en ***parallèle*** aux autres activités

Modèle en V - Suite



Avantages du Modèle en V

Met l'accent sur lest tests et la validation et par conséquent, ça accroît la qualité du logiciel

Chaque livrable doit être testable

Facile à planifier dans une gestion de projets

Facile à utiliser

Inconvénients du Modèle en V

Ne gère pas les activités parallèles

Ne gère pas explicitement les changements des spécifications

Ne contient pas d'activités d'analyse de risque

Modèle en V – Quand l'Utiliser ? (Indication)

Quand le produit à développer à de très hautes exigences de qualité

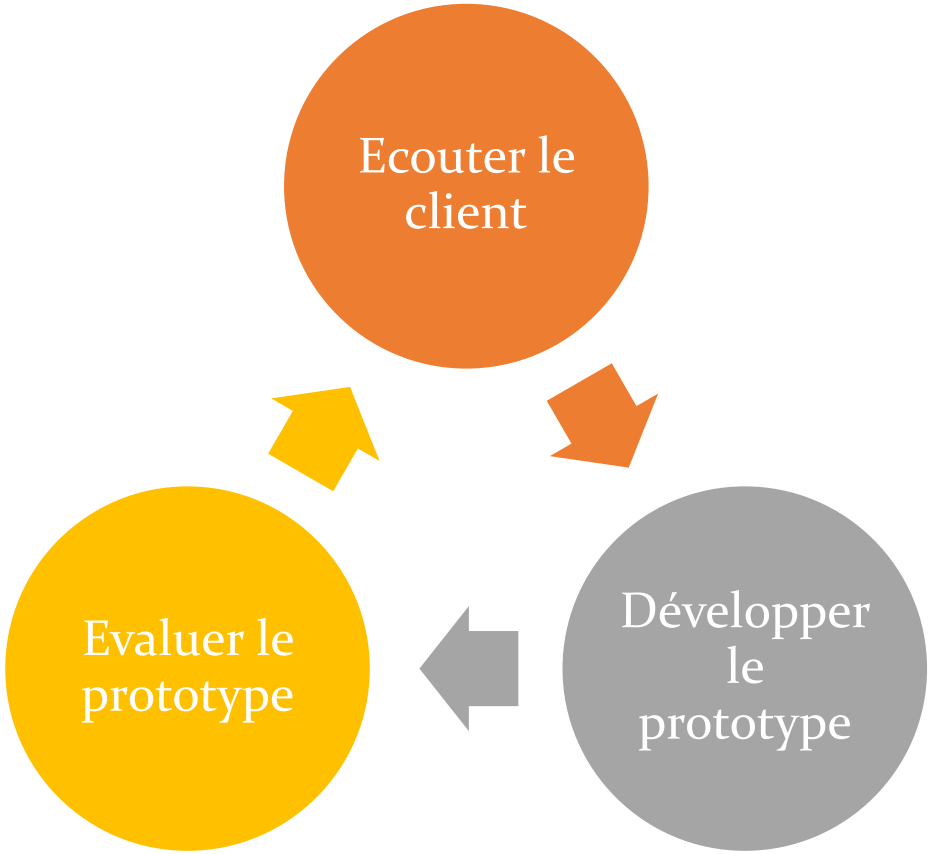
Quand les besoins sont connus à l'avance

Les technologies à utiliser sont connues à l'avance

Prototypage

- Le projet se fait sur plusieurs ***itérations***
- Les développeurs construisent un ***prototype*** selon les ***attentes*** du client
- Le prototype est ***évalué*** par le client
- Le client donne son ***feedback***
- Les développeurs adaptent le prototype selon les ***feedbacks*** et les ***nouvelles exigences*** client
- Quand le prototype satisfait le client, le code est normalisé selon les standards et les bonnes pratiques

Prototypage - Suite



Avantages du Prototypage

Implication active du client

Le développeur apprend directement du client

S'adapte rapidement aux changements des besoins

Progrès constant et visible

Une grande interaction avec le produit

Inconvénients du Prototypage

Le prototypage implique un code faiblement structuré

Degré très faible de maintenabilité

Le processus peut ne jamais s'arrêter

Très difficile d'établir un planning

Prototypage – Quand l'Utiliser ?

Pour de très petits projets impliquant très peu de personnes

Quand les besoins sont instables et/ou nécessitent des clarifications

Peut être utilisé avec le modèle en cascade pour la clarification des besoins

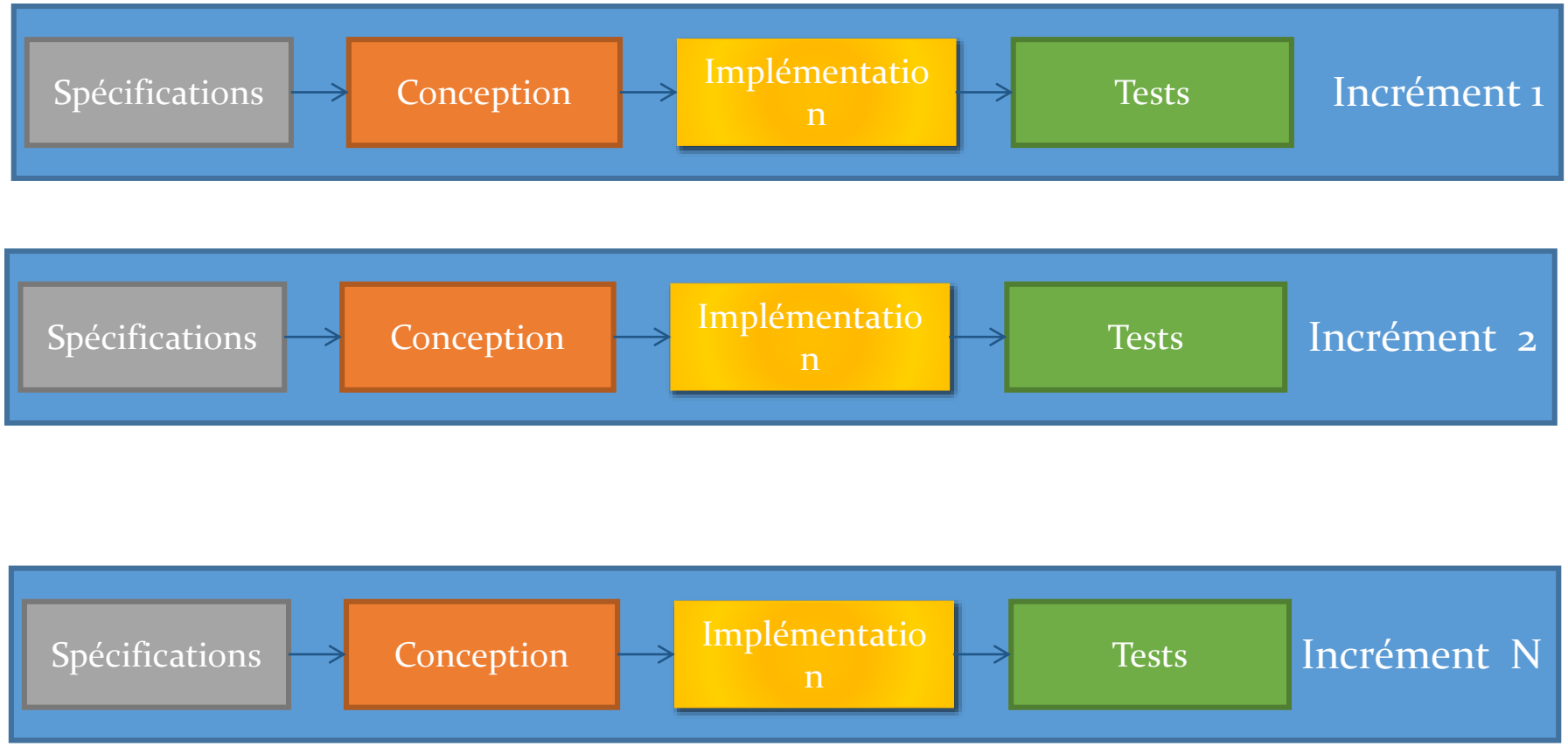
Quand des livraisons rapides sont exigées

Très déconseillé en situations professionnelles

Modèle Incrémental

- Chaque incrément est une ***construction partielle*** du logiciel
- Trie les spécifications par ***priorité*** et les regroupent dans des ***groupes de spécifications***
- Chaque incrément implémente un ou plusieurs groupes jusqu'à ce que la totalité du produit soit finie

Modèle Incrémental - Suite



Avantages du Modèle Incrémental

Développement de fonctionnalités à risque en premier

Chaque incrément donne un produit fonctionnel

Le client intervient à la fin de chaque incrément

Utiliser l'approche « diviser pour régner »

Le client entre en relation avec le produit très tôt

Inconvénients du Modèle Incrémental

Exige une bonne planification et une bonne conception

Exige une vision sur le produit fini pour pouvoir le diviser en incréments

Le coût total du système peut être cher

Modèle Incrémental – Quand l'Utiliser ? (Indications)

Quand la plupart des spécifications sont connues à l'avances et vont être sujettes à de faibles évolutions

Quand on veut rapidement un produit fonctionnel

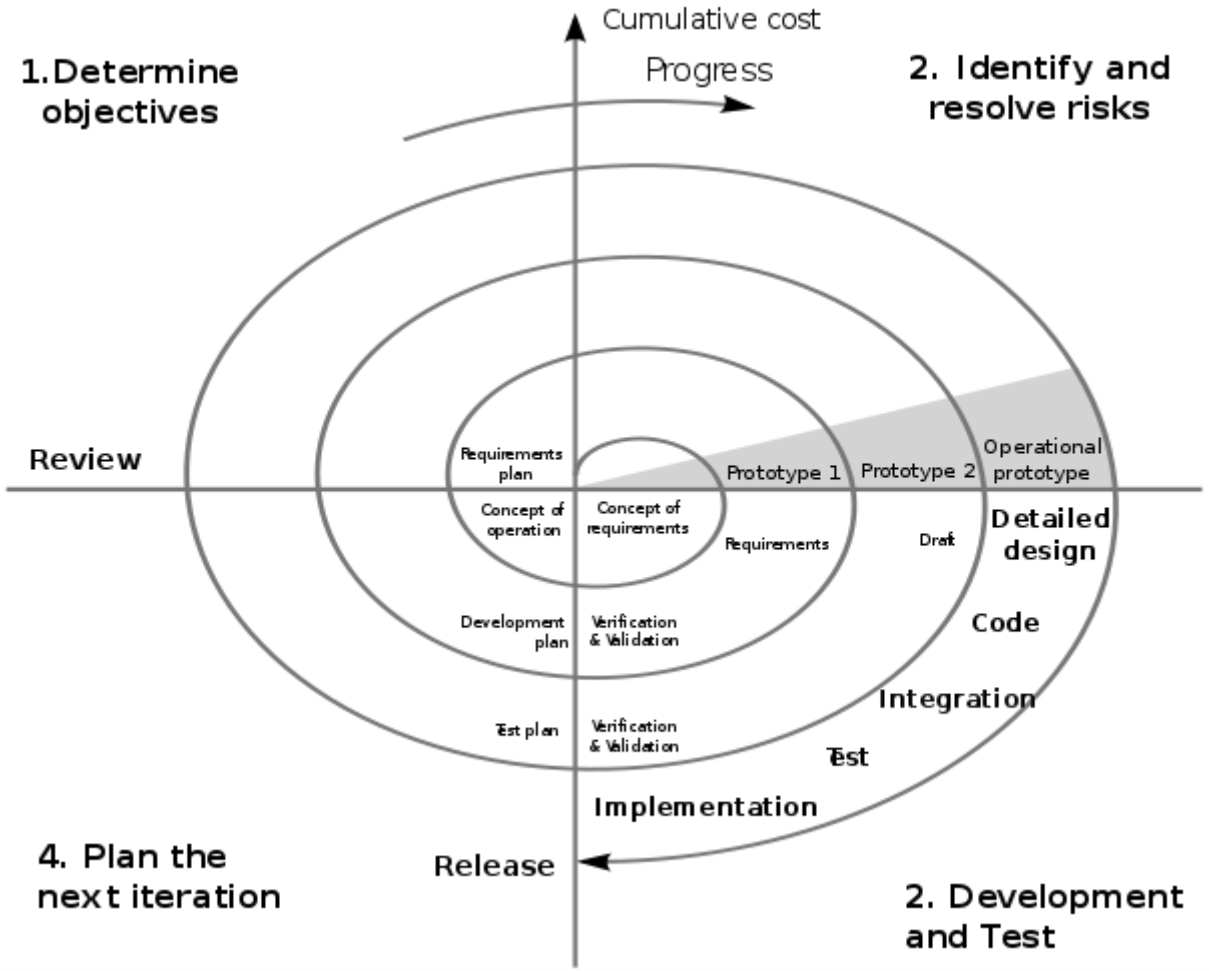
Pour des projets de longues durées

Pour des projets impliquant de nouvelles technologies

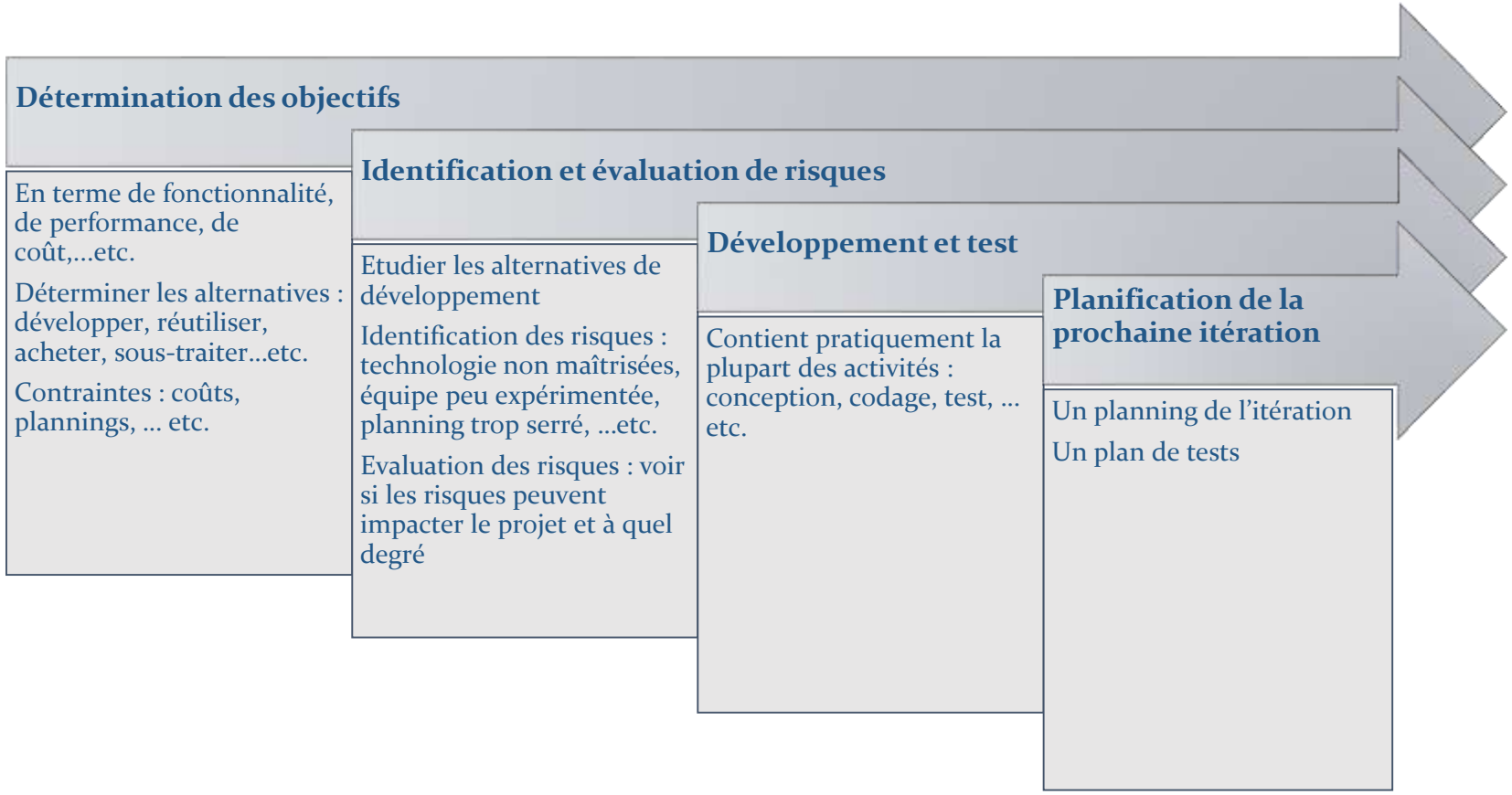
Modèle en spirale

- Modèle ***itératif***
- Des incréments sous forme de cycle
- À la fin de chaque cycle on détermine les ***objectifs*** du cycle suivant
- Chaque cycle est composé des même activités que du modèle en cascade
- Inclut l'analyse de risque et le prototypage

Modèle en Spirale – Suite



Modèle en Spirale – Une Itération



Avantages du Modèle en Spirale

Identification rapide des risques

Impacts minimaux des risques sur le projet

Fonctions critiques développées en premier

Feedback rapide du client

Une évaluation continue du procédé

Inconvénients du Modèle en Spirale

L'évaluation des risques peut prendre beaucoup de temps

Le modèle est très complexe

La spirale peut s'éterniser

Les développeurs doivent être réaffectés pendant les phases de non-développement

Les objectifs ne sont pas souvent faciles à formuler

Modèle en Spirale – Quand l'Utiliser ? (Indication)

Quand le prototypage est exigé

Quand le risque du projet est considérable

Quand les spécifications ne sont pas stables

Pour les nouveaux produits

Quand le projet implique de la recherche et de l'investigation

Méthodologies de Développement Classiques

SECTION 4 – DÉBAT (10 MNS)

Méthodes Agiles

SECTION 5

Historique

- Au milieu des années 90, un groupe d'expert en gestion de projet de développement voulaient proposer de nouvelles méthodologies
- Les nouvelles méthodologies sont plus « légères » : moins de documentation et moins de contrôle sur le procédé
- Ces méthodologies s'adressent à des projets de petite ou moyenne taille avec une équipe réduite
- Ces méthodologies permettent de ***s'ajuster*** rapidement aux changements des spécifications tout en garantissant des livraisons fréquentes
- Ces méthodologies sont qualifiés de « méthodes agiles »

Principes agiles

Individus et interactions au lieu de processus et outils

Logiciel fonctionnel au lieu de documentation massive

Collaboration du client au lieu de négociation de contrats

Réagir au changements au lieu de suivre le plan

Principe 1 - INDIVIDUS ET INTERACTIONS AU LIEU DE PROCESSUS ET OUTILS

- Les ***collaborateurs*** sont la clé du succès
- Les « seniors » échoueront s'ils ne collaborent pas ***en tant qu'équipe***
- Un bon collaborateur n'est pas un forcément un bon programmeur. ***C'est quelqu'un qui travaille bien en équipe***.
- Une surabondance d'outils est aussi mauvaise que le manque d'outils.
- Démarrer petit et investir peu au démarrage.
- Construire l'équipe c'est ***plus important*** que construire l'environnement.

Principe 2 - LOGICIEL FONCTIONNEL AU LIEU DE DOCUMENTATION MASSIVE

- Un code sans documentation est un désastre
- Trop de documents est ***pire*** que pas de documents
- Difficulté à produire et à synchroniser avec le code
- Souvent les documents sont des « mensonges » formels
- Le ***code ne ment jamais sur lui-même***
- Produire toujours des documents aussi courts que possible

Principe 3 - COLLABORATION DU CLIENT AU LIEU DE LA NÉGOCIATION DE CONTRATS

- Très difficile de ***décrire la totalité du logiciel*** depuis le début
- Les projets réussis impliquent les clients d'une manière fréquente et régulière
- Le client doit avoir un ***contact direct*** avec l'équipe de développement

Principe 4 - RÉAGIR AUX CHANGEMENTS AU LIEU DE SUIVRE UN PLAN

- Un logiciel ne peut pas être planifié ***très loin*** dans le futur
- ***Tout change*** : technologie, environnement et surtout les besoins
- Les chefs de projets classiques fonctionnent sur la base de GANTT, PERT et le système de tâches
- Avec le temps, les diagrammes se dégradent car des tâches s'ajoutent et d'autres deviennent non nécessaires
- Une meilleure stratégie est de planifier très court (02 semaines à 01 mois)
- Plannings détaillés pour la semaine à venir, rigoureux pour les trois mois et très vagues au-delà

LES DOUZE PRINCIPES AGILES

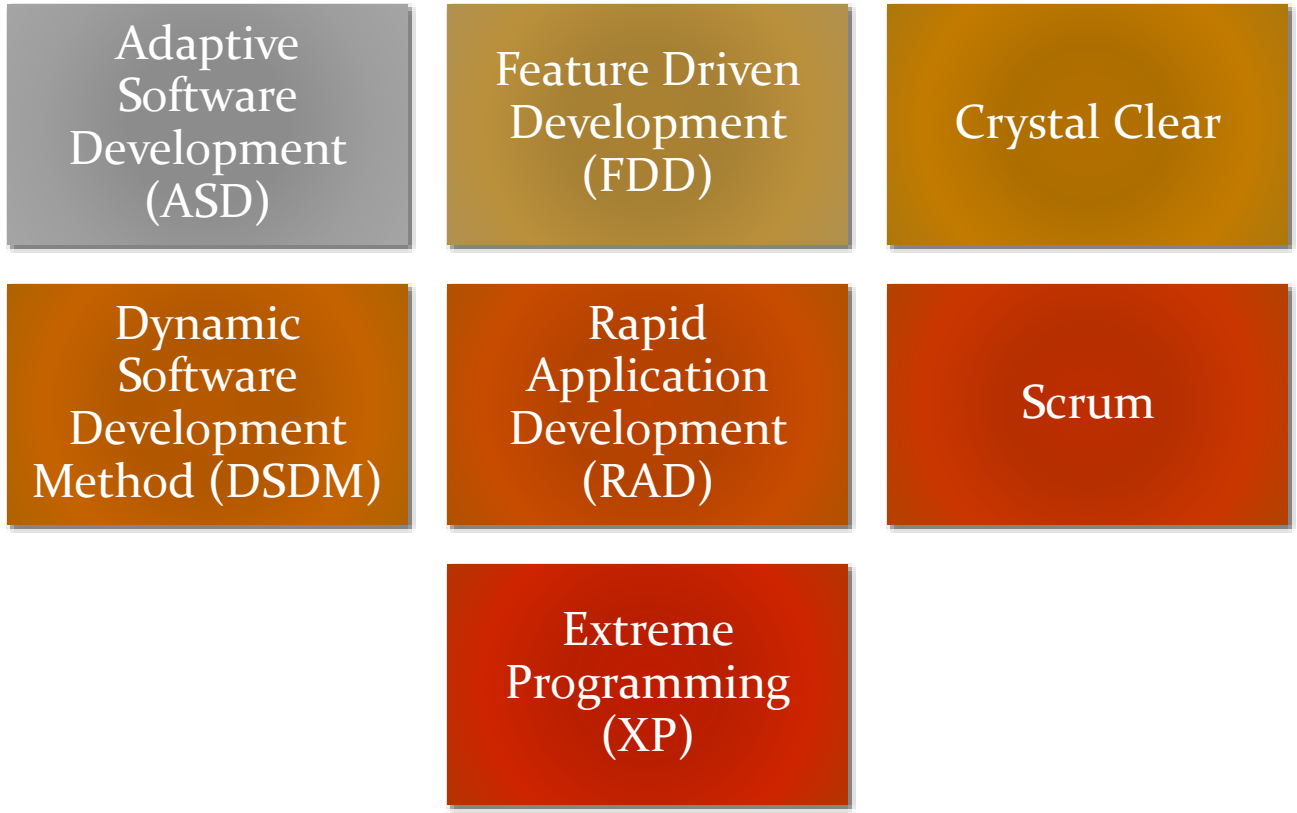
1. Toujours satisfaire le client à travers des livraisons rapides et continues
2. Bien accueillir tous les changements même les tardifs
3. Livrer fréquemment un système fonctionnel
4. Les clients et les développeurs doivent collaborer
5. Conduire le projet autour d'équipes motivées
6. La meilleure méthode de faire circuler l'information c'est le contact direct entre collaborateurs
7. La première mesure d'avancement c'est un logiciel fonctionnel

LES DOUZE PRINCIPES AGILES - Suite

LES DOUZE PRINCIPES AGILES

8. Le développement doit être durable et à un rythme constant
9. La bonne conception et l'excellence technique augmentent l'agilité
10. Simplifier au maximum
11. Les meilleurs architectures, besoins et conceptions proviennent d'équipes qui s'organisent d'elles-mêmes
12. L'équipe s'améliore d'une manière autonome et régulière

Principales Méthodes Agiles



Méthodologie XP

- eXtreme Programming
- Créée en 1995 Kent Beck and Ward Cunningham
- XP est un moyen léger, efficace, à bas risques, flexible, scientifique et amusant de développer des logiciels
- Destinée à des équipes de moyenne taille avec des spécifications incomplètes et / ou vagues
- Le codage est le ***noyau*** de XP

XP - Fondamentaux

Implication massive du client

Test unitaire continu (TDD)

Programmation par paires

Itérations courtes et livraisons fréquentes

Méthodologie XP – Principales activités

Codage (noyau de XP)

Tests (pendant le codage)

Ecoute (le client ou le partenaire)

Conception (encore basée sur le codage)

Pratiques XP

Organisation

- Programmation par binômes
- Travail énergisé
- Espace de travail informatif
- Analyse de cause racine
- Rétrospective

Collaboration

- Confiance
- S'assoir ensemble
- Implication du client
- Langage universel
- Réunion debout
- Standard de codage
- Démo d'itération
- Rapports

Livraison

- Fait Fait
- Pas de bugs
- Contrôle de version
- Génération de 10 minutes
- Intégration Continue
- Propriété Collective
- Documentation

Planification

- Vision
- Plan de livraison
- Jeu de planning
- Gestion des risques
- Planification de l'itération
- Relâchement
- Estimation

Développement

- Exigences Incrémentales
- Tests d'acceptation
- TDD
- Refactoring
- Conception Simple
- Conception et Architecture Incrémentales
- Solutions de Pointe
- Optimisation de Performance
- Tests Exploratoire

Méthodologie XP – Inconvénients

Demande une certaine maturité des développeurs

La programmation par paires n'est toujours pas applicable

Difficulté de planifier et de budgétiser un projet

Stress dû aux devoir de l'intégration continue et des livraisons fréquentes

La faible documentation peut nuire en cas de départ des développeurs

Méthodologie Scrum

- Inspiré par une approche développée en 1986 par H. Takeuchii et II.. Nonaka, le terme « Scrum » utilisé dans « Wiicked Problems, Rightteous Solutions » par DeGrace et Stahl en 1991

- Utilisé comme méthodologie dans le livre : « Agile Software Developmentt with SCRUM" par K. Schwaber et M.. Beedlle en 2001

Méthodologie Scrum - Principes

Simple

- Peut être combiné avec d'autres méthodes
- Compatible avec les best practices

Empirique

- Itérations courtes (sprints)
- Feedback continu

Techniques simples

- Sprints de 2 à 4 semaines
- Besoins capturés en tant que user stories

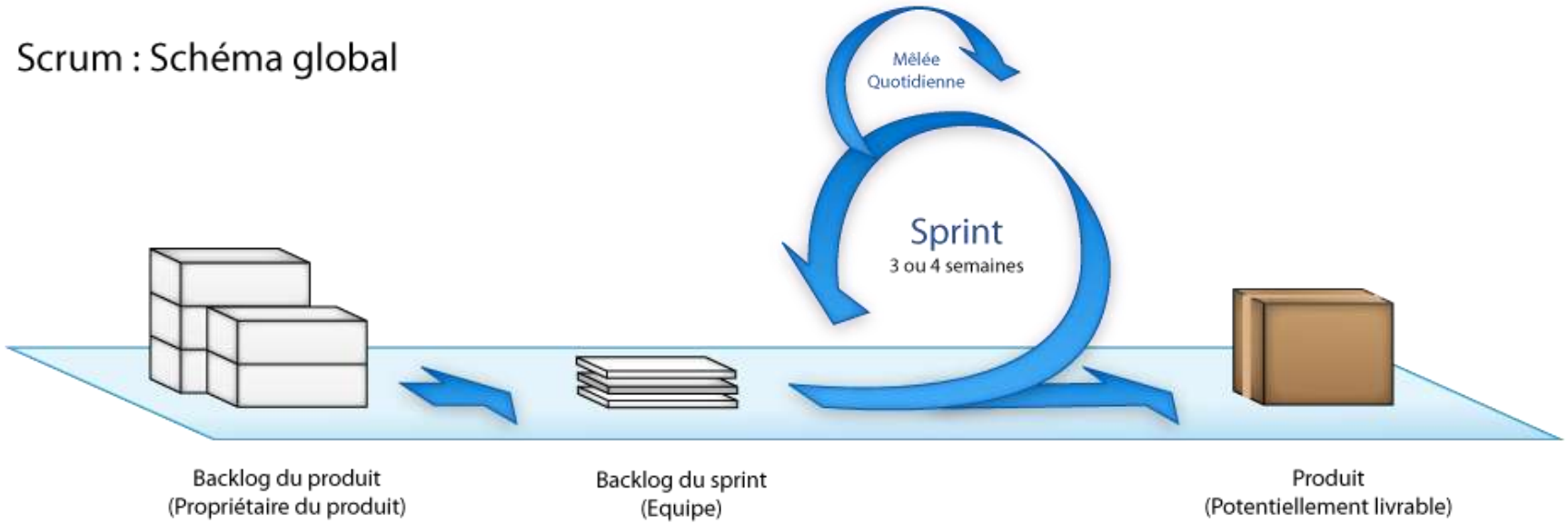
Equipes

- Auto-organisation
- Multi-compétences

Optimisation

- Détection rapide des anomalies
- Organisation simple
- Requiert l'ouverture et la visibilité

Méthodologie Scrum - Déroulement



Méthodologie Scrum – Principes et Concepts

Processus
- Sprint

Backlog
- Product Backlog
- Sprint Backlog

Equipe
- Srum Master
- Product Owner
- Team
- Users and stakeholders

Réunions
- Scrum quotidien
- Planning
- Revue
- Rétrospective

Suivi
- Rapports
- Vélocité

Méthodologie Scrum - Avantages

Méthode très simple et très efficace

Adoptée par les géants du marché : Microsoft, Google, Nokia..

Orientée projet contrairement à XP qui est orientée développement

Peut inclure d'autres activité venant d'autres méthodologies (surtout XP)

Méthodologie Scrum - Inconvénients

N’est pas 100% spécifique au GL

Difficulté de budgétiser un projet

Méthodes Agiles

SECTION 5 – DÉBAT 10 MNS

Processus Unifié (UP)

SECTION 6

UP - Introduction

- UP est une méthodologie très populaire
- UP est ***incrémental*** et ***itératif***
- UP a plusieurs ***implémentation*** et / ou variation dont la plus célèbre est RUP (Rational Unified Process)

UP – Implémentations

RUP (Rational Unified Process)

Agile Unified Process (AUP) par Scott W. Ambler

Basic Unified Process (BUP) par IBM

Enterprise Unified Process (EUP), une extension de RUP

Essential Unified Process (EssUP), une légère variation

Open Unified Process (OpenUP) par Eclipse

Oracle Unified Method (OUM),

UP – Principes

Processus itératif et incrémental

Basé sur les cas d'utilisation

Centré sur l'architecture

Accent sur les risques

Utilisation de Modèles UML

UP – Principes

PROCESSUS ITÉRATIF ET INCRÉMENTAL

- UP est composé de quatre phase ***: l'analyse de besoins*** (inception), ***l'élaboration***, la ***construction*** et la ***transition***
- Chaque phase peut être décomposée en plusieurs ***itérations***
- Chaque itération produit un ***incrément*** du produit

PROCESSUS BASÉ SUR LES CAS D'UTILISATION

- Les cas d'utilisation formalisent les ***spécifications fonctionnelle*** du produit
- Chaque itération prend un ensemble de cas d'utilisation et les traite selon plusieurs workflows : modélisation métier, analyse de besoins, analyse et conception, implémentation, tests et déploiement

PROCESSUS CENTRÉ SUR L'ARCHITECTURE

- UP supporte plusieurs architectures logicielles
- La phase d'élaboration fournit l'architecture de l'exécutable
- Cette architecture est une ***implémentation partielle*** qui sert de fondation aux développements futurs

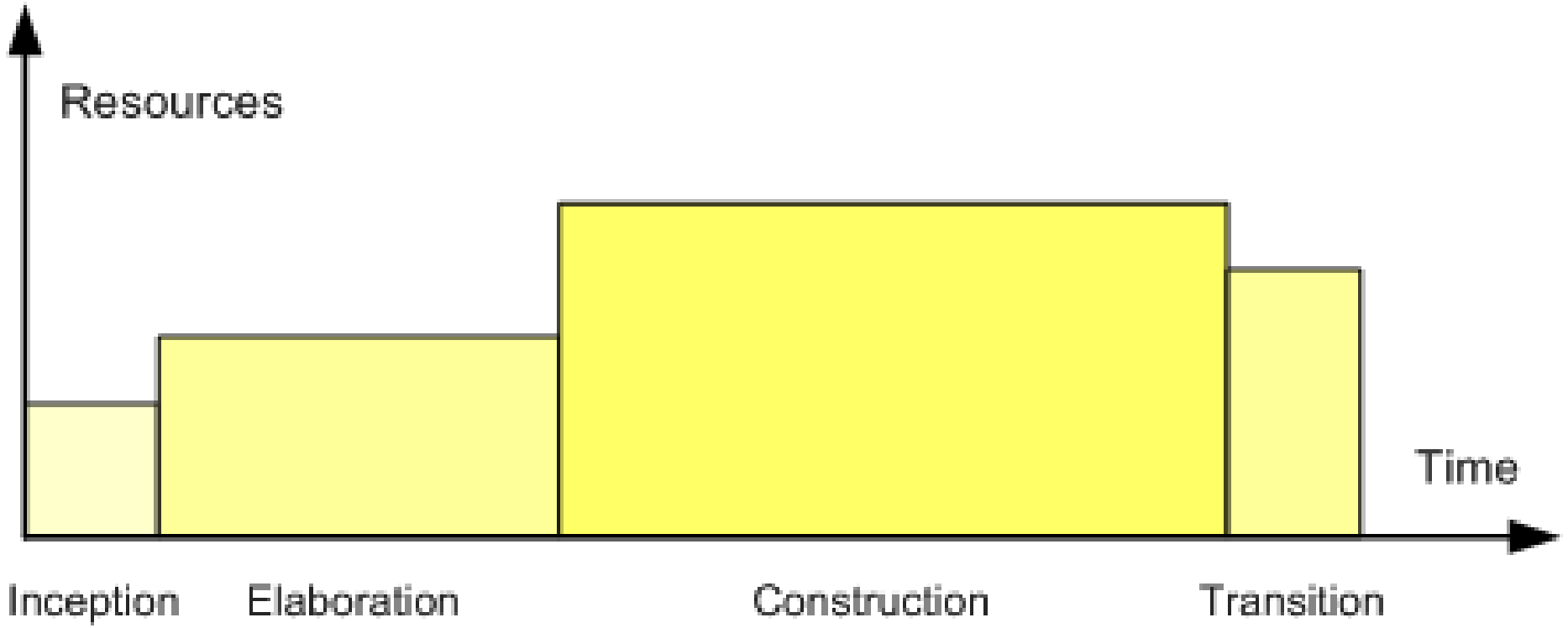
PROCESSUS QUI MET L'ACCENT SUR LES RISQUES

- ***Identification rapide*** des risques
- Traitement des risques dans les premières phases du projet

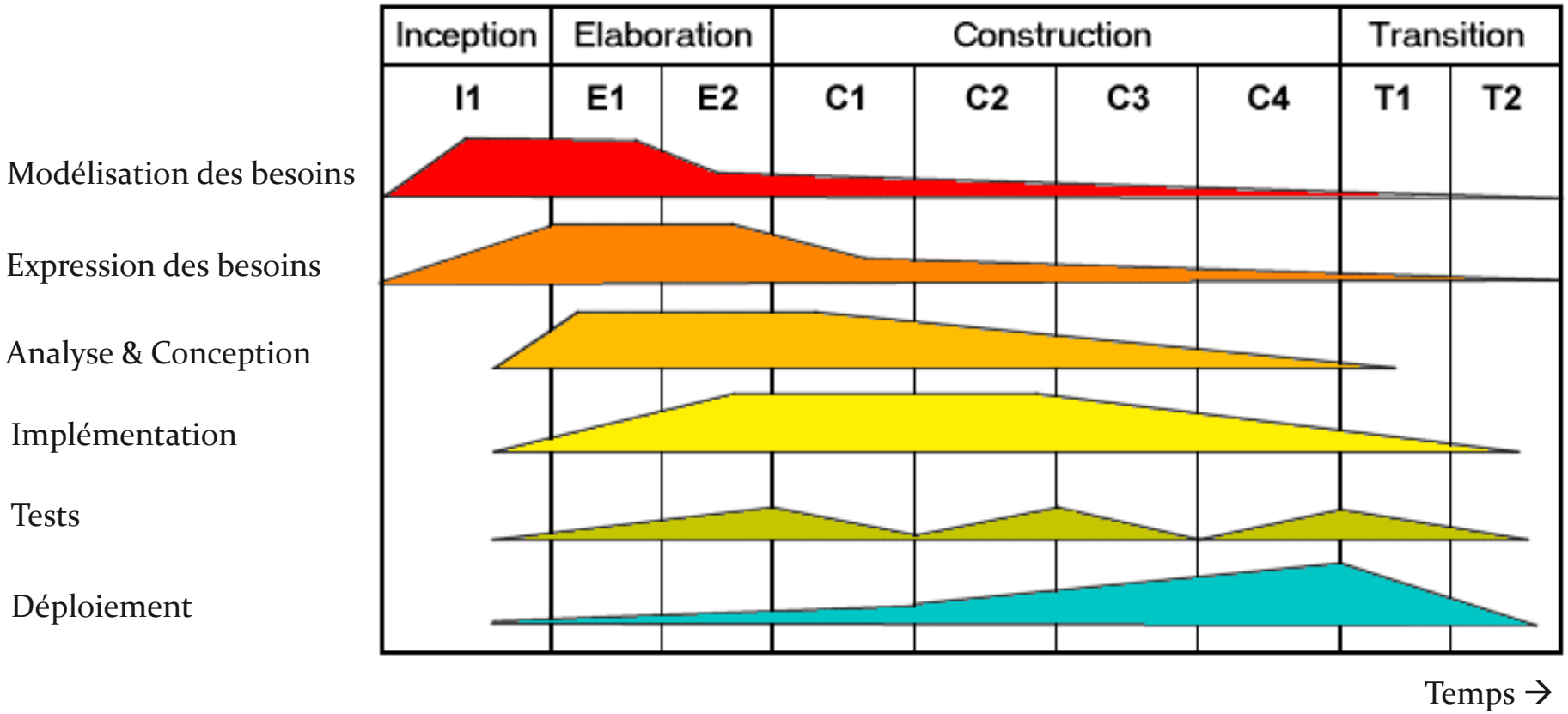
Utilisation de Modèles UML

- Utilisation des diagrammes UML dans les activités

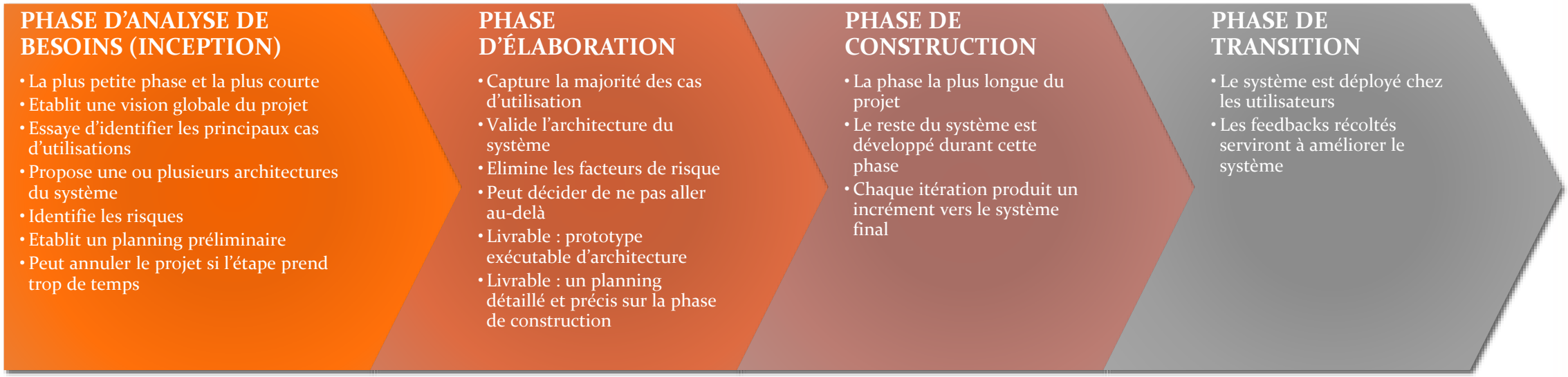
UP – Cycle de vie



UP – Méthodologie



Phases UP



UP - Activités

Expression de besoins	• Recenser les besoins fonctionnels et non fonctionnels du système • Le diagramme UML de cas d'utilisation est utilisé pour cette phase
Analyse	• Détaille les besoins en spécifications détaillées • Une ébauche de la conception
Conception	• Décide comment sera construit le système durant l'implémentation • Définit les sous-systèmes et les composants • Crée des abstractions de la solution
Implémentation	• Traduire les résultats de la conception en un système opérationnel • Livrables : code source, exécutables, ...etc.
Tests	• Vérification et validation des résultats de l'implémentation

Avantages de UP

- Méthodologie complète
- Identification rapide des risques
- Largement adoptée en entreprise
- Intégration avec UML
- Séparation concise des phases et des livrables
- Des formations / livres / tutoriaux disponibles

Inconvénients de UP

Complexe

Plusieurs implémentations d'où une source de confusion ou d'éloignement du concept d'origine

Processus Unifié (UP)

SECTION 6, DÉBAT 05 MNS

Bibliographie

- Software Engineering Right Edition, Ian Sommerville, Addison Wesley, 2007
- Software Development and Professional Practice, John Dooley, APress, 2010
- Software Development Life Cycle (SDLC), Togi Berra, course session 2
- Rational Unified Process - Best Practices for Software
- Development Teams, IBM / Rational, 1998