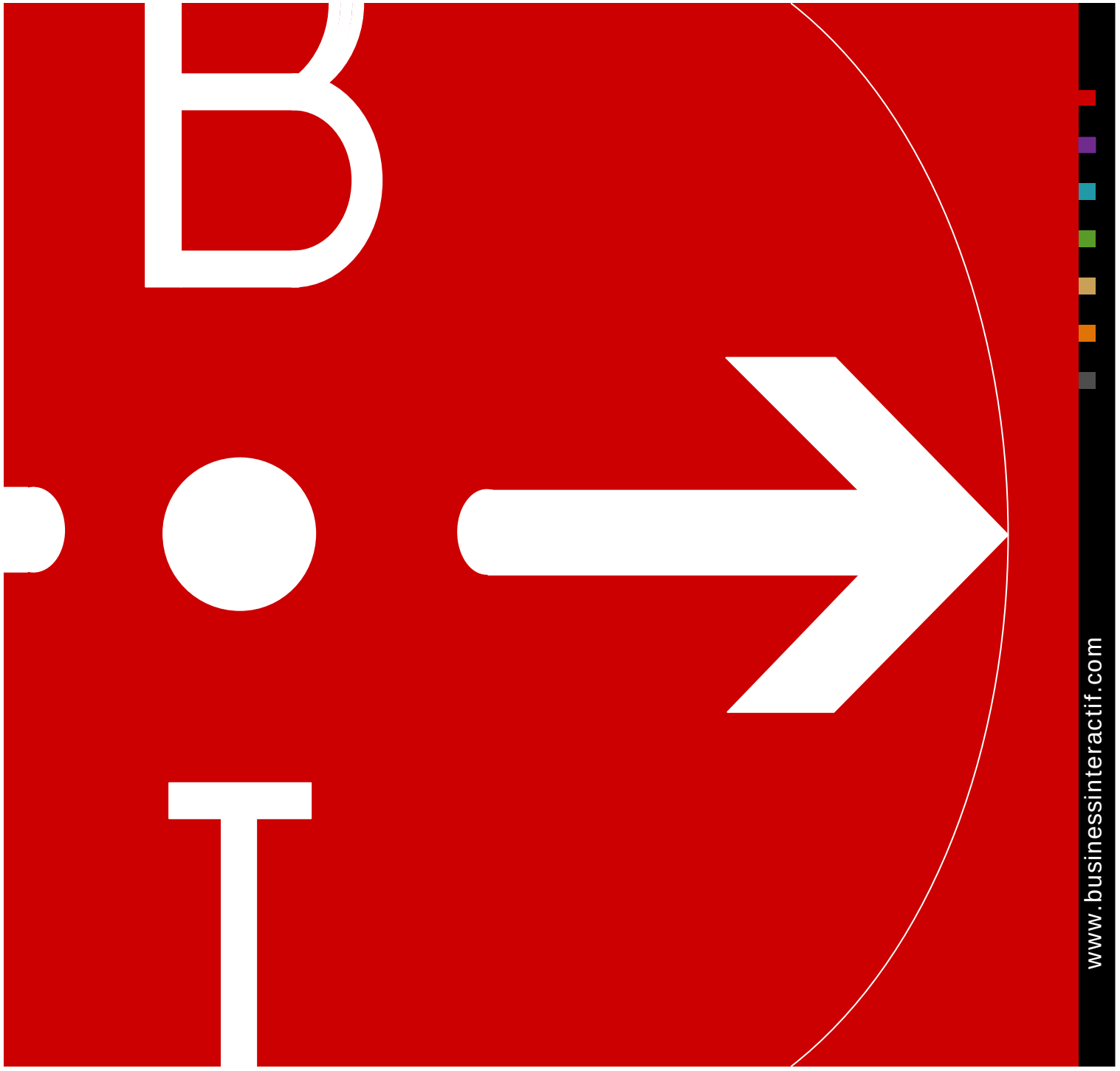
Méthodes agiles



Business Interactif

Jean-Louis Bénard
jlb@businessinteractif.fr

CONSEIL
& DÉVELOPPEMENT
DE SOLUTIONS
E-BUSINESS

20 mai 2002

Sommaire

- Méthodes agiles : une réponse à un malaise ?
- Caractéristiques des méthodes agiles
- Panorama
  - RAD, UP
  - eXtreme Programming
  - Dynamic Software Development Method
  - Adaptive Software Development
  - Crystal Clear
  - SCRUM
  - Feature Driven development
- Synthèse
- Retours d'expérience

Méthodes agiles : une réponse à un malaise ?

- Quelques chiffres…
- Selon le standish group (1998)
  - 25% des projets respectent les délais et les budgets
  - 45% dépassent le budget ou sont en retard
  - 28% sont abandonnés ou voient leur périmètre largement restreint
  - Combien correspondent à des besoins utilisateurs réels ?
- 30% des projet e-CRM sont abandonnés (Gartner Group)
- En cause : l'opportunité des projets, la conformité aux besoins, les technologies…mais aussi les méthodes

Méthodes agiles : une réponse à un malaise ?

- Le contexte économique difficile met une pression forte
- Rapports parfois difficiles entre la maîtrise d'ouvrage et la maîtrise d'œuvre
- Confusion entre des besoins estimés et des besoins réels
- Les guerres contractuelles font oublier l'objectif initial
- Chacun s'abrite derrière les modalités d'un forfait qui fige dans le marbre des spécifications incomplètes
- Exemples

Méthodes agiles : une réponse à un malaise ?

- Les méthodes portent souvent le chapeau
  - Merise = has-been
  - RAD = pour le client-serveur (donc has been)
  - Unified Process = UML = trop lourd = has-been
- Attention : les méthodes ne sont pas forcément mauvaises mais souvent mal utilisées !!!
  - Formation négligée
  - Peu adaptées au contexte
- L'utilisation d'une méthode outillée et de documentations ne garantit pas le succès
- Le super-AGL bi-directionnel avec rétro-conception intégré : le mythe

Méthodes agiles : une réponse à un malaise ?

Facteurs clé d'échec :

- Le manque de communication à tout niveau
- Une mauvaise compréhension des besoins
- L'insuffisance de l'architecture
- L'absence de maturité des outils utilisés
- La mauvaise formation des personnes
- Le cadre contractuel inadapté
- L'insuffisance des tests
- L'absence d'une démarche Thing Big – Start Small
- Les effets Tunnel

L'un des principaux facteurs d'échec est l'absence de gestion du risque

Quelque soit la méthode utilisée les risques doivent être identifiés et surveillés en permanence

Dépendance de quatre facteurs

Coût

Qualité

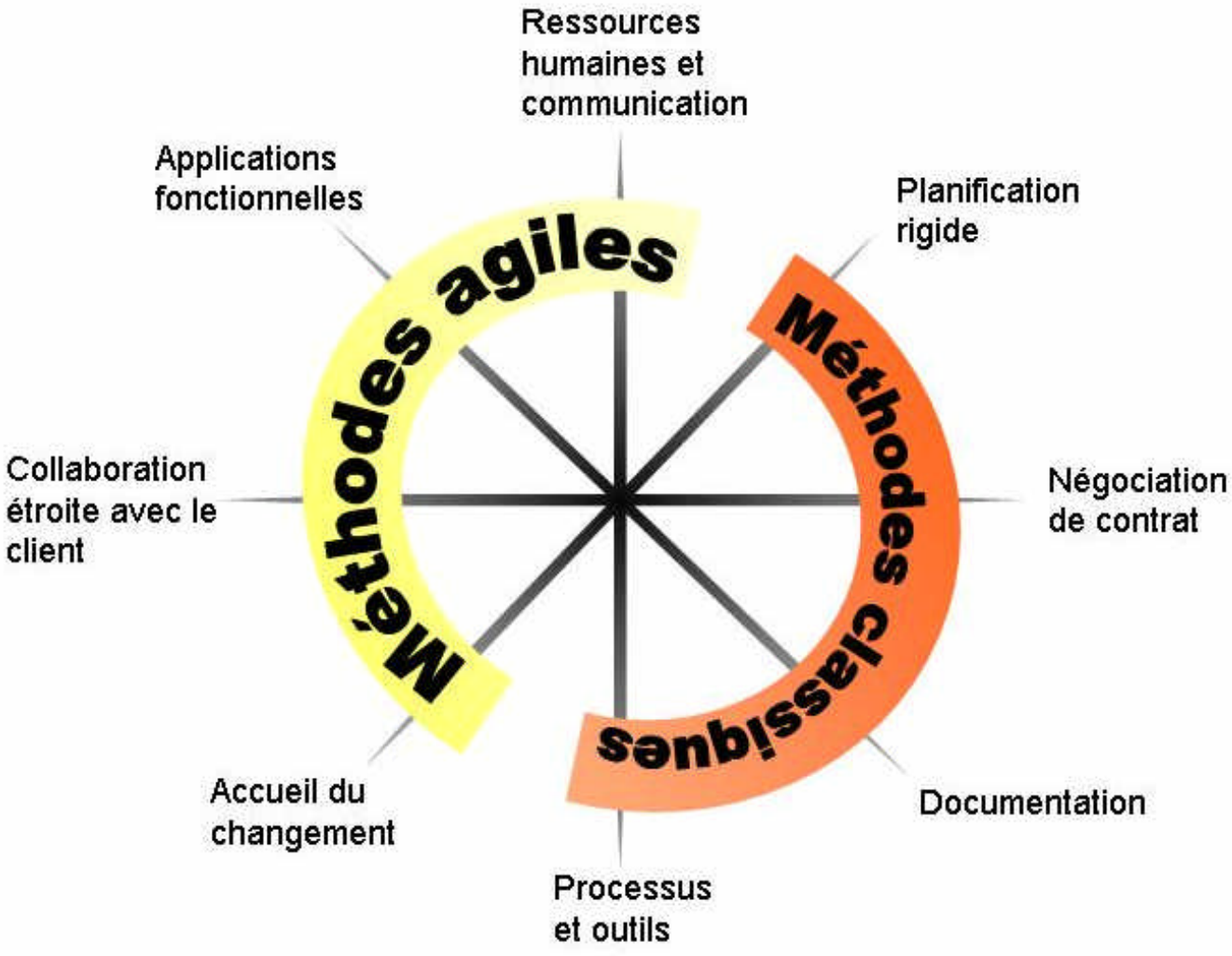
Durée

Périmètre fonctionnel

Caractéristiques des méthodes agiles : le manifeste

- Priorité des personnes et des interactions sur les procédures et les outils
- Priorité d'applications opérationnelles sur une documentation exhaustive
- Priorité de la collaboration avec le client sur la négociation de contrat
- Priorité de l'acceptation du changement sur la planification

Caractéristiques des méthodes agiles : le manifeste



Caractéristiques des méthodes agiles : les principes

- Délivrer rapidement et très fréquemment des versions opérationnelles, pour favoriser un feed-back client permanent
- Accueillir favorablement le changement
- Assurer une coopération forte entre client et développeurs
- Garder un haut niveau de motivation
- Le fonctionnement de l'application est le premier indicateur du projet

Caractéristiques des méthodes agiles : les principes

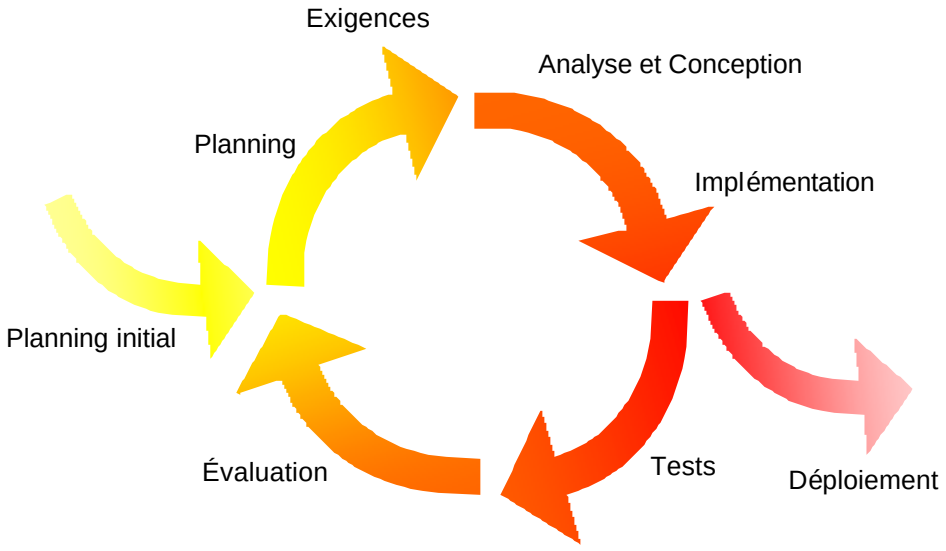
- Garder un rythme soutenable
- Viser l'excellence technique et la simplicité
- Se remettre en cause régulièrement

Méthodes agiles : panorama

- eXtreme Programming
- Dynamic Software Development Method
- Adaptive Software Development
- Crystal Clear
- SCRUM
- Feature Driven development
- [RAD, UP ?]

UP du point de vue de l'agilité

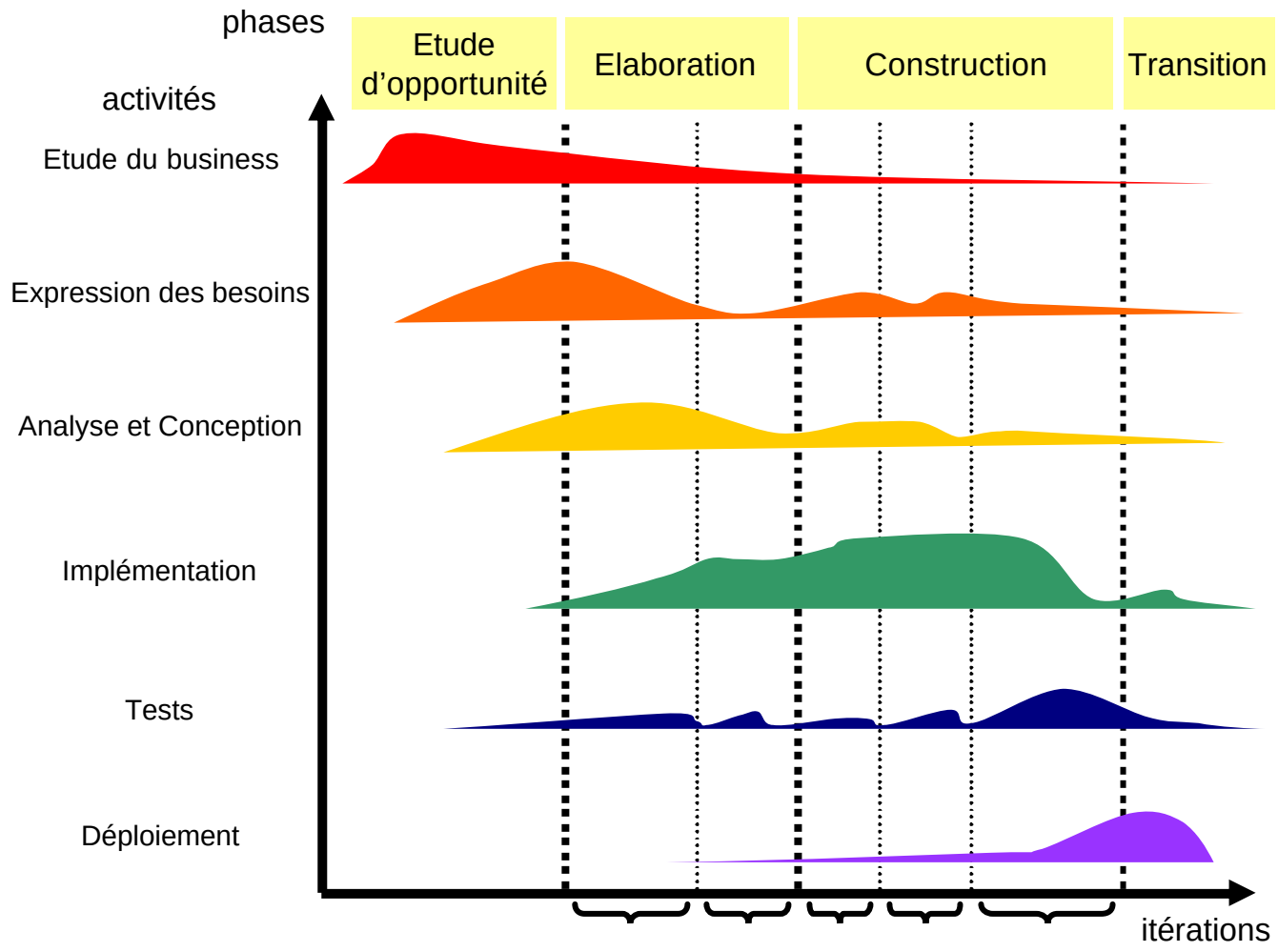
- Les fondamentaux d'UP
  - Pilotage par les cas d'utilisation
  - UP est centré sur l'architecture
  - UP est itérative et incrémentale
  - UP gère les besoins et les exigences
  - UP est fondée sur la production de composants
  - UP pratique la modélisation visuelle
  - UP surveille la qualité et les risques



UP du point de vue de l'agilité

- Les activités dans UP
  - Expression des besoins
  - Analyse (expression des besoins du pt de vue développeur)
  - Conception (définition de l'architecture, incrémentale)
  - Implémentation
  - Tests
- Les phases du cycle de vie
  - Etude d'opportunité (faisabilité, analyse des risques, vision globale des exigences…10% des uses cases cartographiés)
  - Elaboration (précision de l'architecture, 80% des use cases cartographiés)
  - Construction (fourniture d'une version beta)
  - Transition (correction de bugs, préparation de l'itération suivante)

UP du point de vue de l'agilité



UP du point de vue de l'agilité

- UP n'est pas opposé à l'agilité (privilégie les cycles courts itératifs, etc.)
- Mais souvent considéré comme trop lourd (en dépit des outils mis à disposition par Rational)

eXtreme Programming

- Pères de la méthode : Ward Cunningham et Kent Beck (1996). 4 valeurs clés :

- Communication
  - **rendre la communication omniprésente entre tous les intervenants**
- Simplicité
  - **il coûte moins cher d'aller au plus simple et de rajouter des fonctionnalités par la suite plutôt que de concevoir dès le départ un système très compliqué dont on risque de n'avoir jamais l'utilité**
- Feedback
  - **indispensable pour que le projet puisse accueillir le changement**
- Courage
  - **concerne aussi bien les développeurs que le client**

eXtreme Programming : 12 principes

- Feedback rapide
- Assumer la simplicité
- Changements incrémentaux
- Accueillir le changement à bras ouverts
- Un travail de qualité
- Apprendre à apprendre
- Faible investissement au départ
- Jouer pour gagner
- Des expériences concrètes
- Communication ouverte et honnête
- Responsabilités acceptées
- Adaptation aux conditions locales
- Voyager léger
- Mesures honnêtes

- Les principes sont déclinés en pratiques

le club des utilisateurs JAVA

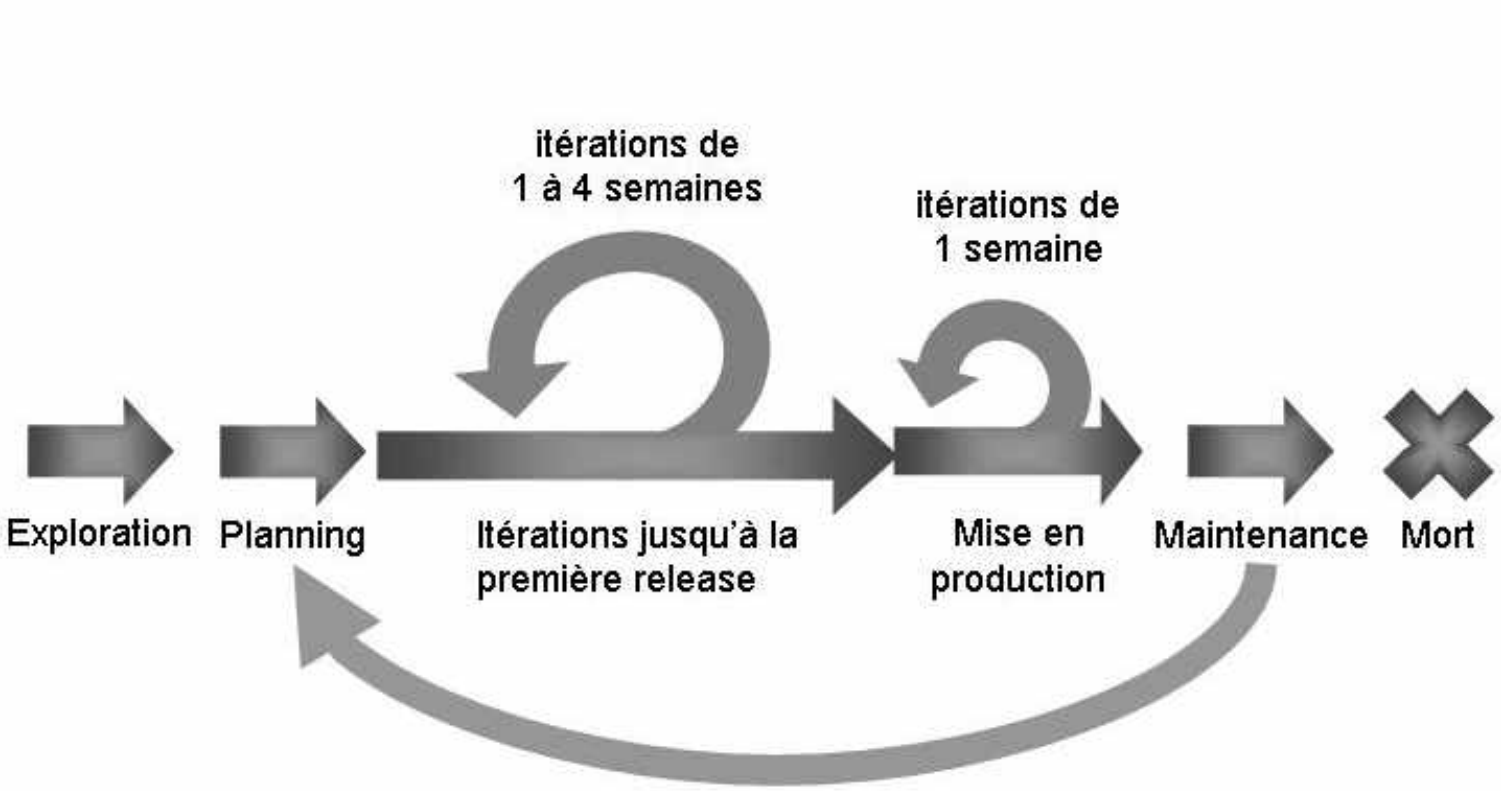
Pratiques eXtreme Programming

- Planning game
  - **phase d'exploration : écriture des besoins sous forme de "user stories" et estimation de leur durée**
  - **phase d'engagement : classement des user stories par ordre de priorité. phase de direction : mise à jour du planning**
- Petites releases
- Utilisation de métaphores pour décrire l'architecture du système
- Conception simple : toujours développer le solution la plus simple possible
- Tests (unitaires et fonctionnels)

Pratiques eXtreme Programming

- Refactoring du code : retravailler le code pour le rendre plus lisible et plus robuste
- Programmation en binôme
- Propriété collective du code
- Intégration continue (plusieurs fois par jour)
- Pas de surcharge de travail : ne pas dépasser 40 heures de travail par semaine
- Client sur site : présence sur site d'une personne minimum à temps plein pendant toute la durée du projet
- Standards de code (normes de nommage et de programmation)

Cycle de vie d'un projet XP



maintenance = ajouter des fonctionnalités → nouvelles releases
utilise le même processus que pour la release 1

Rôles dans XP

- Développeur
  - **travaille en binôme, communique**
  - **doit être autonome**
  - **a une double compétence : développeur – concepteur**
- Client
  - **doit apprendre à exprimer ses besoins sous forme de user stories**
  - **a à la fois le profil de l'utilisateur et une vision plus élevée sur le problème et l'environnement du business**
  - **doit apprendre à écrire les cas de tests fonctionnels**
- Testeur
  - **a pour rôle d'aider le client à choisir et à écrire ses tests fonctionnels**

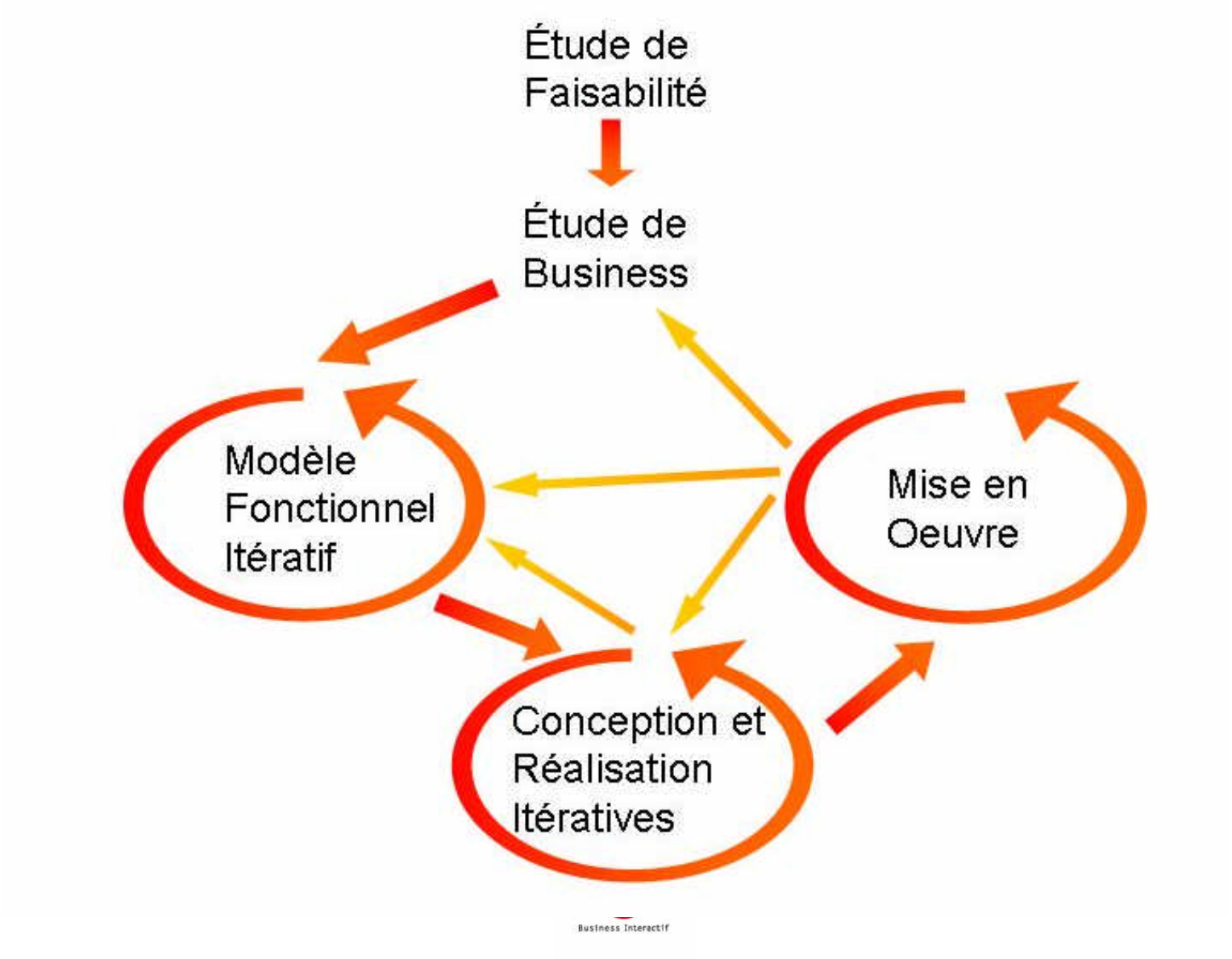
Rôles dans XP

- Tracker
  - aide l'équipe à mieux estimer le temps nécessaire à l'implémentation de chaque user story
  - contrôle la conformité de l'avancement au planning
- Coach
  - recadre le projet
  - ajuster les procédures
  - doit intervenir de la manière la moins intrusive possible
- Consultant
  - n'apporte pas de solution toute faite
  - apporte à l'équipe les connaissances nécessaires pour qu'elle résolve elle-même les problèmes
- Big Boss
  - apporte à l'équipe courage et confiance

Dynamic Software Development Method (DSDM) : Principes

- implication active des utilisateurs
- équipes autorisées à prendre des décisions
- produit rendu tangible aussi souvent que possible
- L'adéquation au besoin métier est le critère essentiel pour l'acceptation des fournitures
- Un développement itératif et incrémental permet de converger vers une solution appropriée
- Toute modification pendant la réalisation est réversible
- besoins définis à un niveau de synthèse
- tests intégrés pendant tout le cycle de vie
- esprit de coopération entre tous

DSDM : Cycle de vie



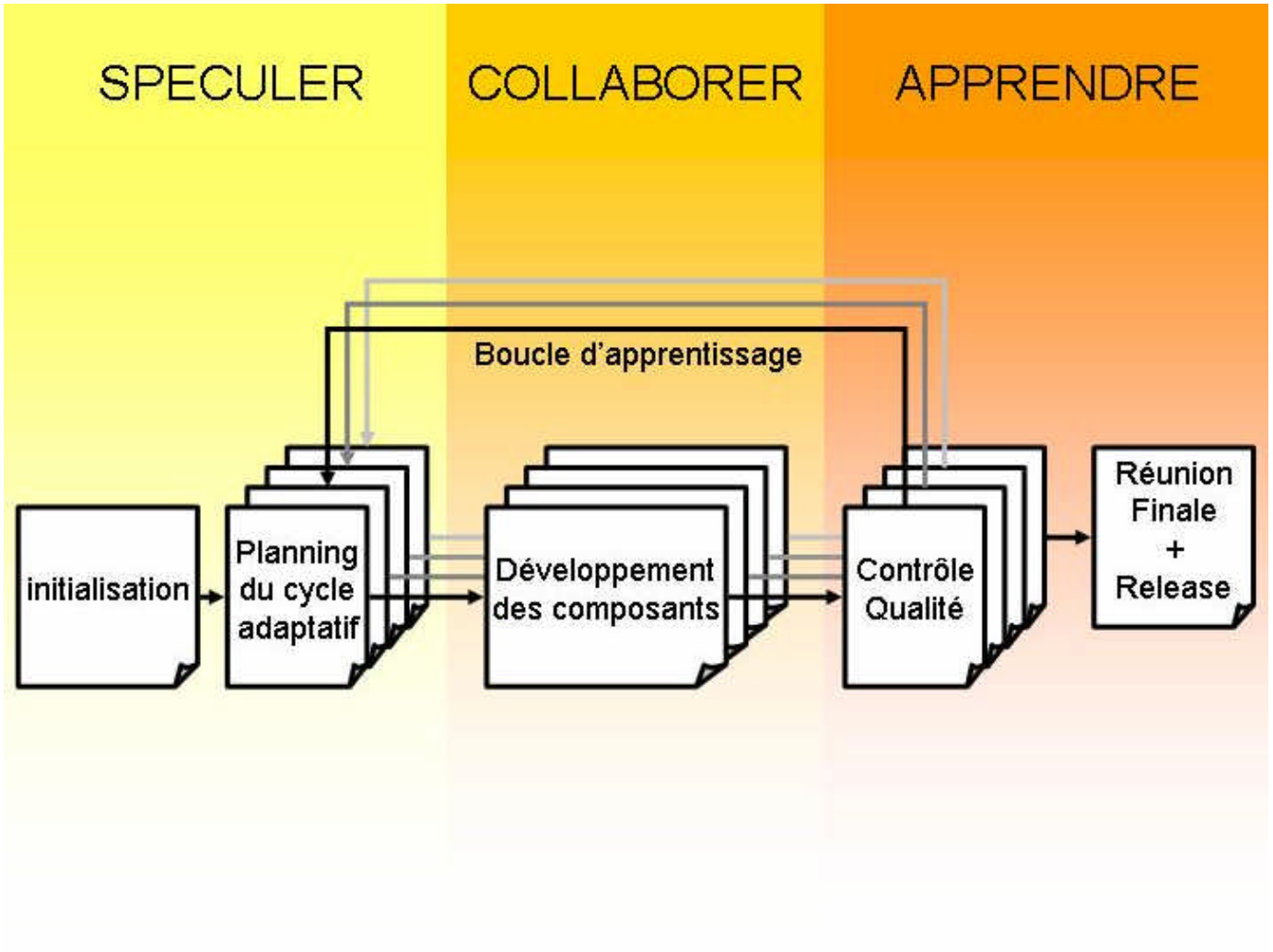
DSDM : Rôles

- Sponsor exécutif
- Visionnaire
- Utilisateur ambassadeur
- Utilisateur conseiller
- Chef de projet
- Coordinateur technique
- Chef d'équipe
- Développeur
- Facilitateur
- Rapporteur

Adaptive Software Development

- Méthode crée par Jim Highsmith
- 6 Caractéristiques principales
  - Focaliser sur une mission ("mission focused")
  - Se baser sur des composants ("component-based")
  - Itérer
  - Découper le temps et fixer des deadlines ("timeboxing")
  - Gérer le risque (projet risk-driven)
  - Tolérer le changement
- 3 phases clé
  - Spéculer
  - Collaborer
  - Apprendre

ASD : Cycle de vie



ASD : Cycle de vie

Spéculation

- Initier le projet (mission, contraintes, collaborateurs, expression des exigences, identification risques…)
- Déterminer la date butoir du projet
- Définir le nombre d'itérations et les dates associées (4 à 8 semaines par itération)
- Donner une mission à chaque itération
- Affecter les composants de base aux itérations
- Affecter les technologies aux itérations
- Développer une liste de tâches à réaliser

Collaboration

- Délivrance des composants
- Communication forte et assez informelle
- Possibilité d'appliquer des pratiques agiles (de type XP)

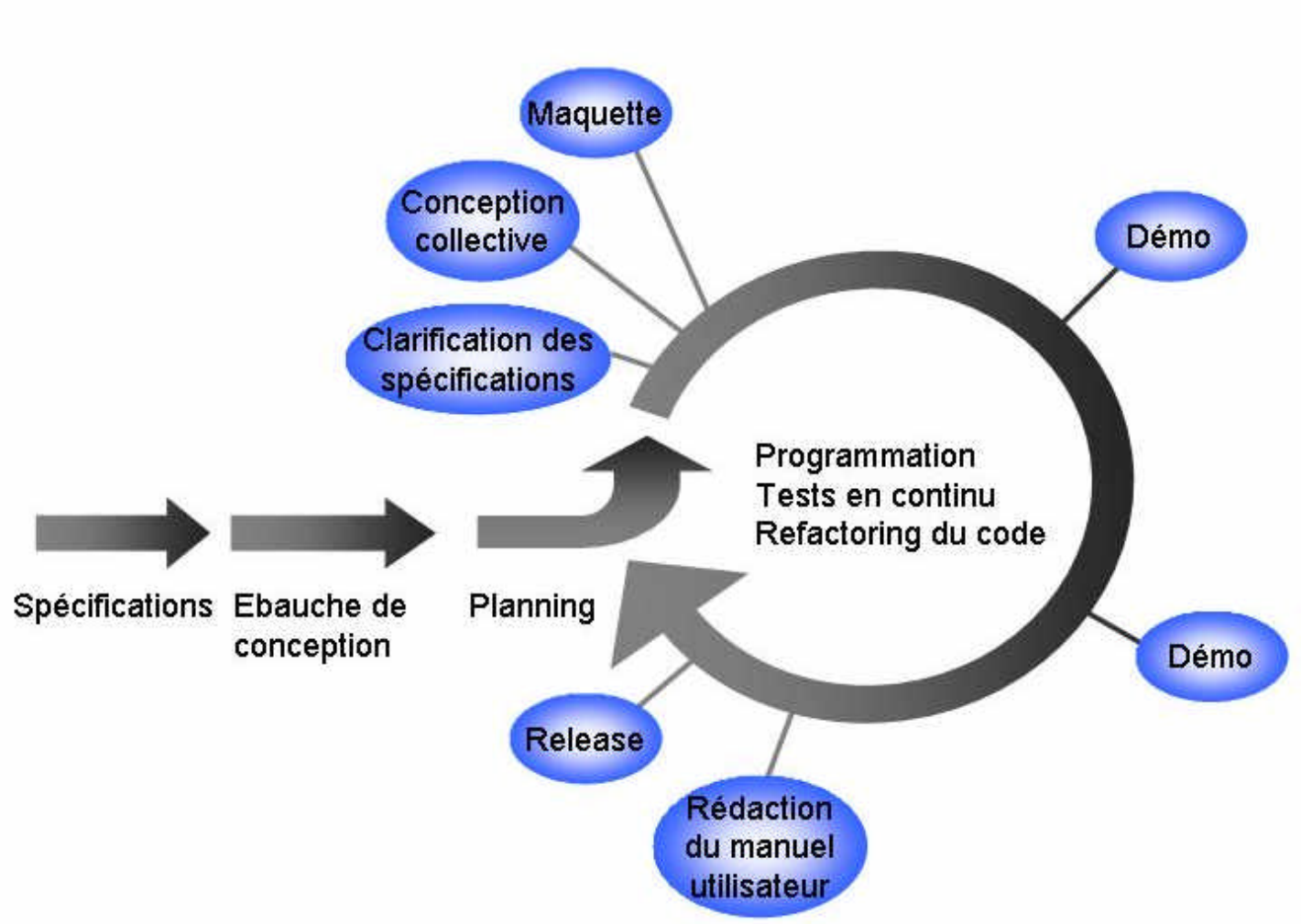
ASD : Cycle de vie

- Apprentissage
  - Contrôle qualité « utilisateur / client »
  - Contrôle qualité technique
  - Suivi de la performance
  - Bilan sur l'état d'avancement
  - Communication forte et assez informelle
  - Possibilité d'appliquer des pratiques agiles (de type XP)

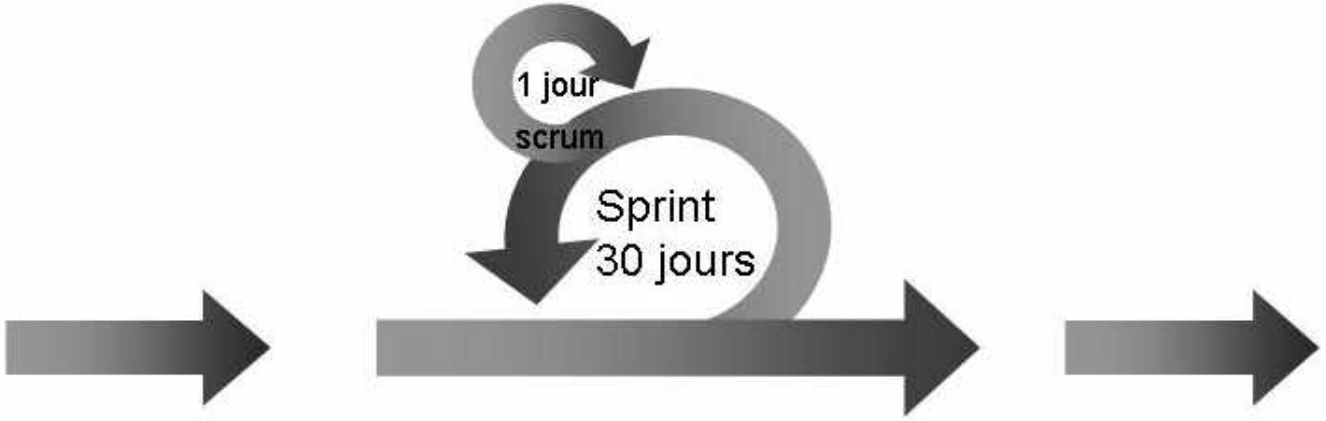
Crystal

- Méthode créée par Alistair Cockburn
- Importance de la Communication et du feed-back client
- Releases fréquentes
- Deux grandes phases
  - Conception et planning
  - Itérations
- Adapté à des équipes de 6 personnes maximum (pas de leadership clairement exprimé)

Crystal : Cycle de vie



- Méthode créée par ADM et WMARK software (spécialisées dans le développement objet)



Planning & Architecture

- phases linéaires
- processus bien définis
- backlog
- conceptualisation
- analyse - conception
- architecture système

Sprints de développement

- phase non linéaire
- processus empirique
- flexibilité
- accueil du changement
- contrôles permanents pour éviter le chaos, notamment management du risque

Clôture

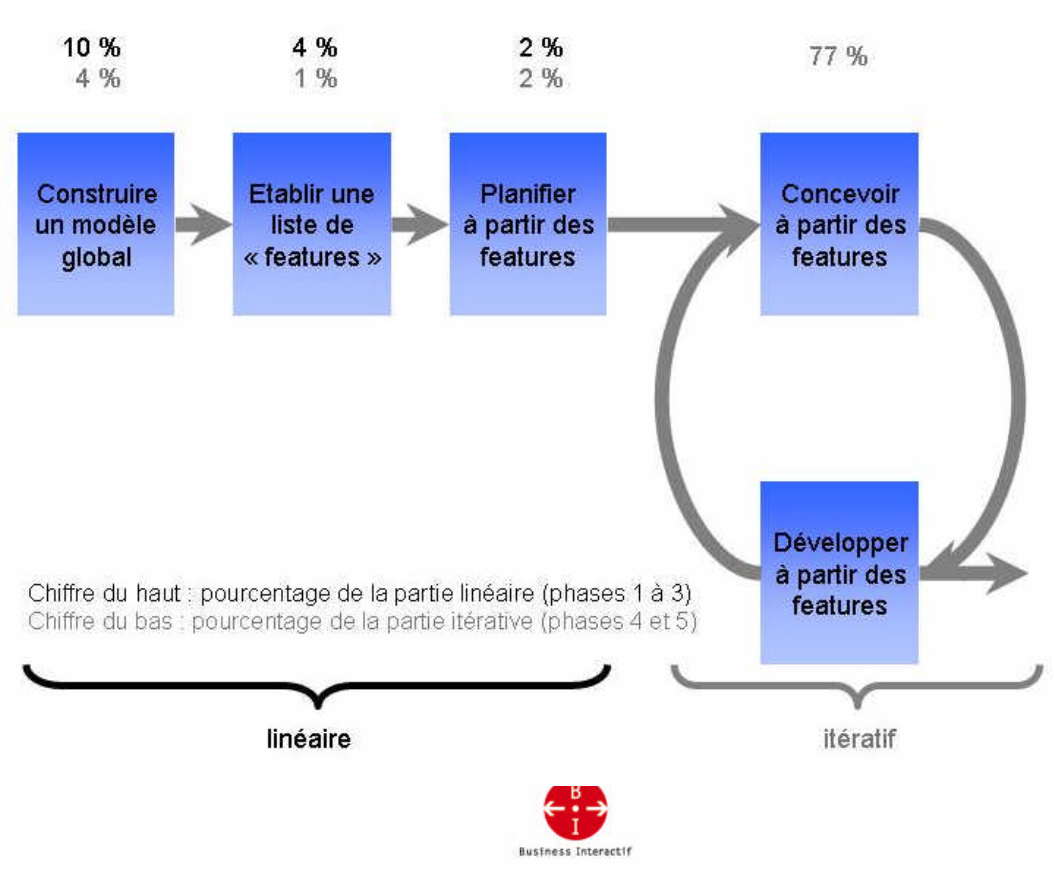
- préparation de la release
- documentation finale
- derniers tests
- livraison

Scrum : Cycle de vie

- Phase initiale
  - Planning
  - Mise en place d'un backlog (liste de tâches à effectuer)
  - Définition de l'équipe
  - Analyse des risques - Budget
- Sprints
  - 30 jours isolés de toute influence extérieure
  - Un sprint backlog est suivi et réalisé
  - Réunion quotidienne Scrum
  - Réunion post-sprint de présentation des résultats
- Clôture
  - documentation finale
  - Livraison…

Feature Driven Development

- Méthode créée par Jeff de Luca et Peter Coad
  - Itérations très courtes
  - Spécifications découpables en features, regroupables en features set
  - Le feature est une fonctionnalité porteuse de valeur pour le client



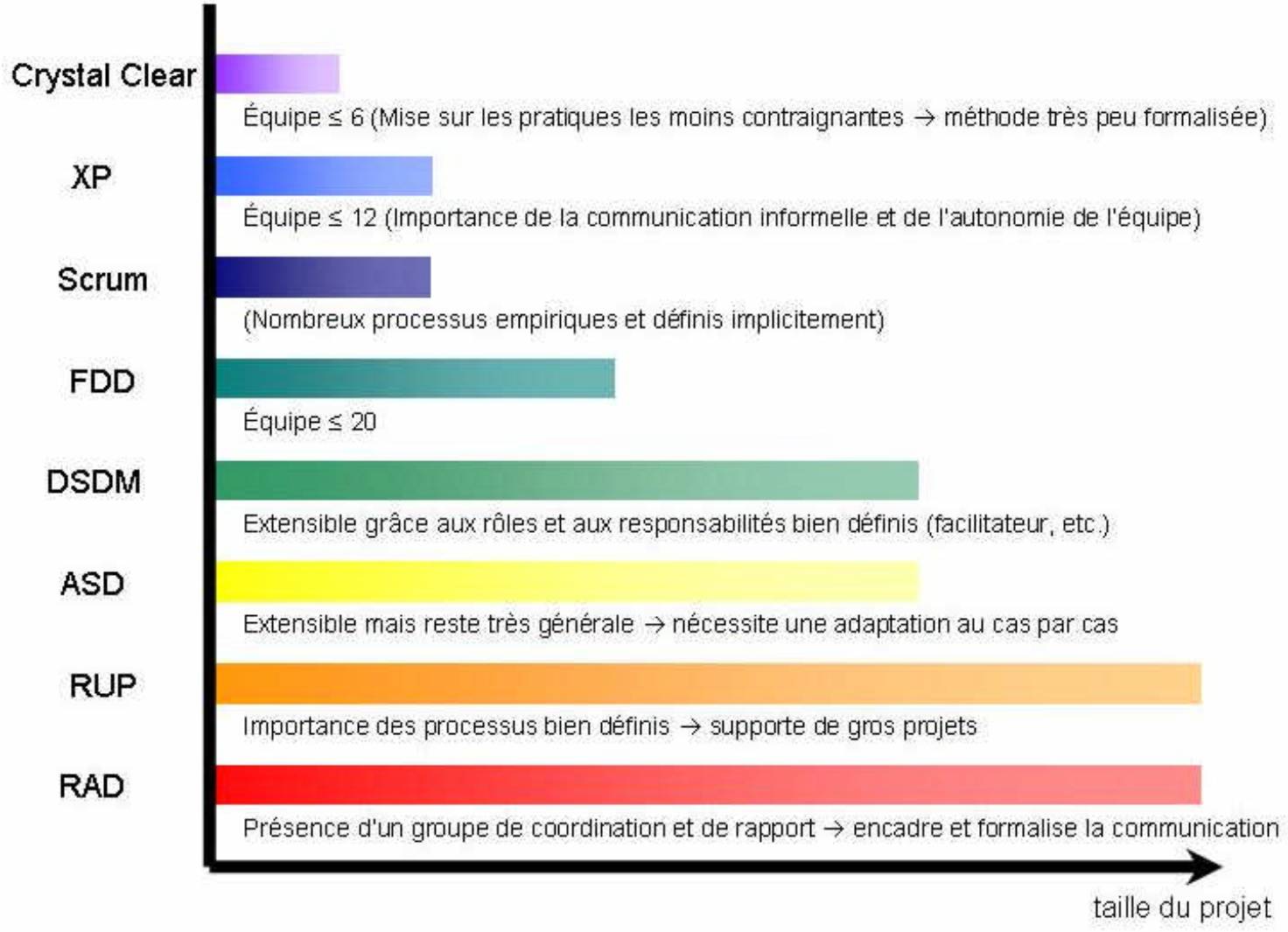
Feature Driven Development

- Phase initiale : développer un modèle global
  - Constitution l'équipe de modélisation
  - Etude du modèle (formation des développeurs)
  - Etude documentaire
  - Elaboration d'une liste informelle de features
  - Modélisation de classes en petits groupes puis consolidation
- Etablissement de la liste détaillée de features
  - A partir de la base informelle, listing détaillé des features
  - Les features trop complexes sont éclatés
  - Les features sont triés par ordre de priorité
  - L'ensemble est validé
- Planification à partir des features
  - Constitution de l'équipe planning
  - Séquencement des features
  - Affectation des classes
  - Affectation des features aux développeurs seniors

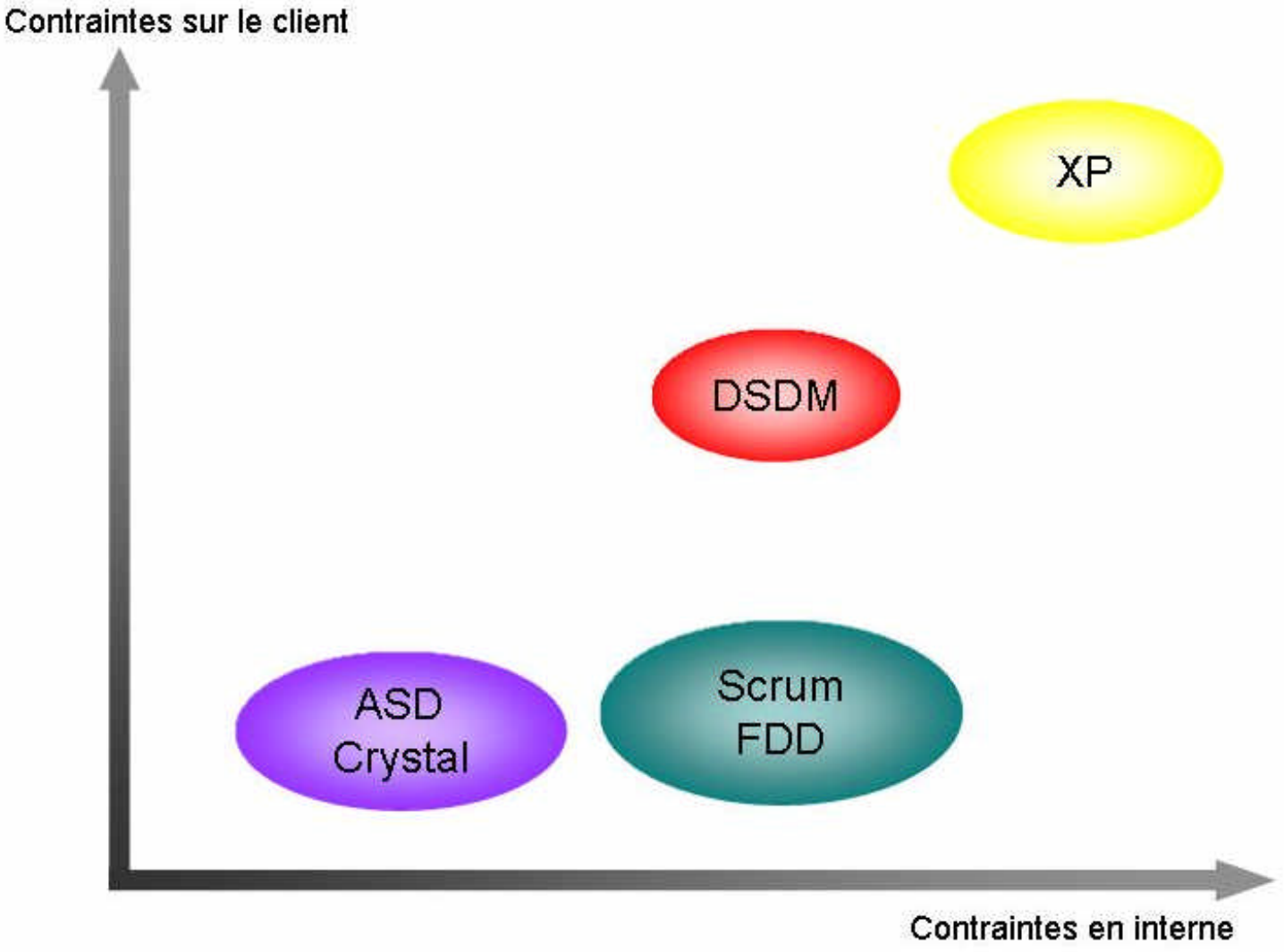
Feature Driven Development

- Conception à partir des features
  - Formation de groupes pour la réalisation de chaque feature
  - Etude des documents de référence associés au feature
  - Construction des diagrammes de séquence
  - Conception détaillée des classes
  - Inspection de l'ensemble
- Construction à partir des features
  - Implémentation des classes / méthodes
  - Inspection du code + tests unitaires
  - Préparation pour l'intégration

Méthodes agiles : synthèse



Méthodes agiles : synthèse



Méthodes agiles : synthèse

- Globalement proximité des différentes méthodes, notamment sur les aspects suivants :
  - Importance des itérations
  - Proximité du client
- Difficile d'évaluer l'efficacité de chacune de ces méthodes
- eXtreme Programming est actuellement la méthode qui monte en force. Pourquoi ?
- Les méthodes agiles ne doivent pas être opposées à RAD et Unified Process
- Il n'existe stricto sensu ni bonne ni mauvaise méthode
- La réussite d'un projet dépend avant tout de l'adaptation de la méthode au contexte

Retours d'expérience : Points clé

- Petites équipes et outils véloces
- Polyvalence des équipes
- Feedback client et itérations
- Gestion des risques
- Collaboration MOA et MOE
- Tester toujours et encore
- Les valeurs clés
  - Humilité
  - Pragmatisme
  - Soucis de la qualité
  - Partage
  - Courage

Merci de votre attention

- Jean-Louis Bénard – Business Interactif
- jlb@businessinteractif.fr