Développement Agile

Rappel sur les cycles de vie de développement logiciel

Mohamed Elyaakoubi

cycle de vie

- On entend, par un cycle de vie, toutes les phases de développement du logiciel, de l'établissement des besoins du client jusqu'à l'achèvement du logiciel en tant que produit commercial

- La notion de cycle de vie n’est pas spécifique au logiciel, mais le cas du logiciel est assez particulier. On distingue deux types de cycle de vie

  - Le cycle de vie des produits s'applique à tous les types de produits, et peut être considéré comme un outil de gestion.

  - Le cycle de développement des logiciels s'insère dans le précédent, on l'appelle souvent abusivement cycle de vie des logiciels

De main en main

- Ce qu’il fallait
- Ce que l’utilisateur demande
- Ce qui est écrit dans le cahier de charge
- Ce que l’analyste a compris
- Ce que le programmeur a réalisé
- Après la mise au point

Le grand risque

- Les défauts apparaissent lors de l'exploitation du logiciel
- la découverte d’une erreur lors des phases avancées de développement
- Le logiciel, certes bien, fait quelque chose, mais peut-être pas, ce qu’il fallait !!!!!
  - régression devient très coûteuse
  - coût de correction élevé

Le grand souci

- Réduire la distance entre les premières phases de développement et la phase de validation (vérification externe) qui fait intervenir le client/utilisateur

Validation et Verification (V & V)

Cycle de vie et assurance qualité sont fortement liés.
il faudra donc en permanence assurer:

la **validation**: sommes nous en train de faire le bon produit?
(par rapport aux exigences de l'utilisateur final)

la **vérification**: est ce que nous faisons le produit correctement
(confirmation des exigences spécifiées[ISO 9000])

Objectifs d’un cycle de vie

- Maîtriser les risques,
- Maîtriser au mieux les délais et les coûts,
- Obtenir une qualité conforme aux exigences.

Facteurs transcendants

Facteur humain

- Quelque soit le cycle de développement choisi, La réussite d'un projet dépend en grande partie de la qualité des personnes qui travaillent.
- Laisser l'équipe se constituer : des gens qui se connaissent et s'apprécient travaillent mieux ensemble
- Mettre en place des primes ou des compensations
- L'architecte logiciel et le testeur doivent programmer : pour garder leur crédibilité
- Faite tourner la responsabilité : augmenter la cohésion, améliorer la communication, équilibrer les charges.

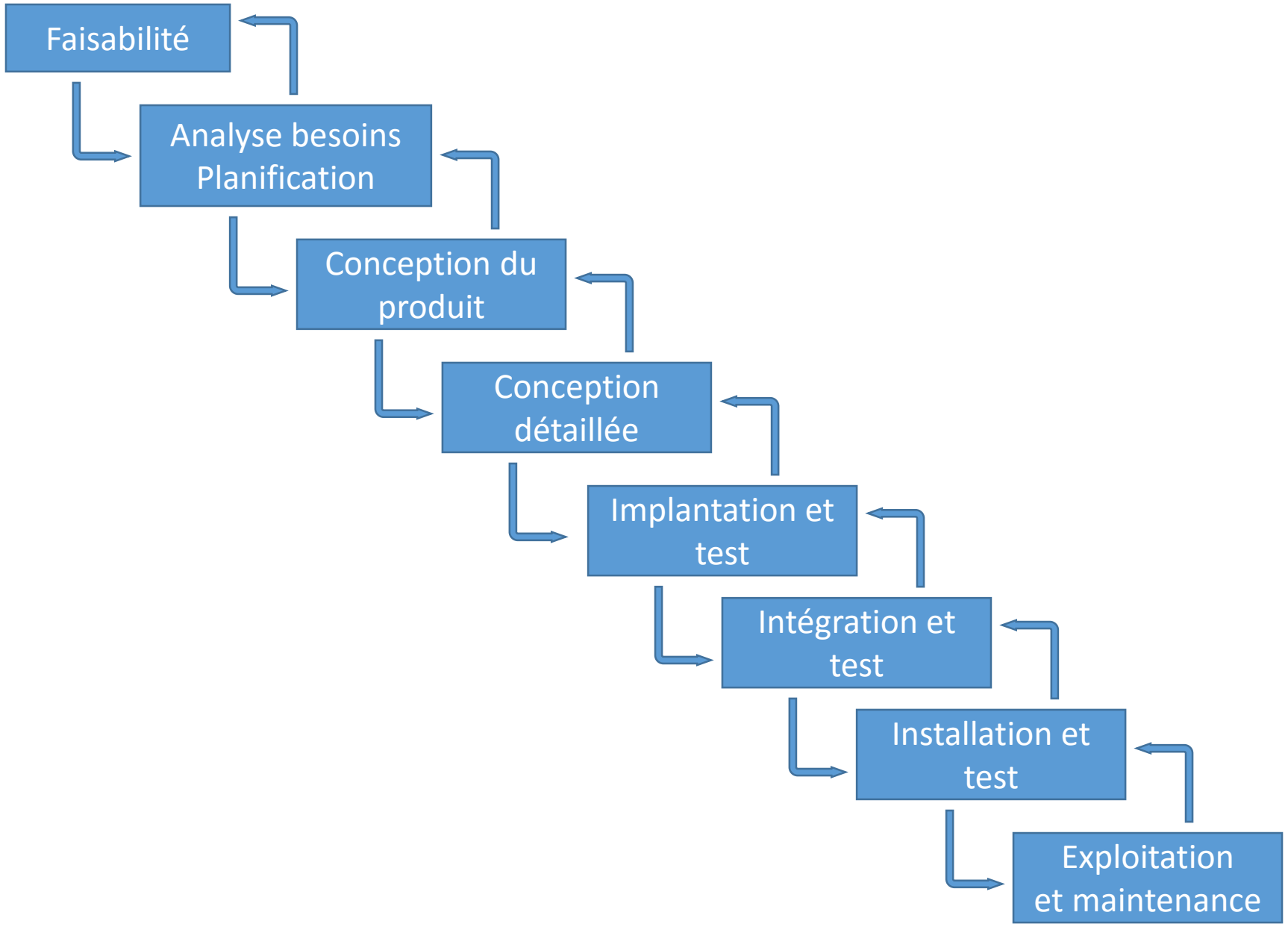
Facteurs transcendants

Estimation et planification

- PERT : converger sur des dates au plus tôt et au plus tard
- Oracle : estimer par consensus, s'adresser à des experts
- Analogie : comparer avec des projets antérieurs
- Planning pocker : tous les membres d'équipe y participent

Le modèle en cascade

- Le modèle en cascade défini des étapes (ou phases) durant lesquelles les activités de développement se déroulent
- Une étape doit se terminer à une date donnée par la production de certains livrables, logiciels ou documents
- L'étape suivante n'est abordée que si les résultats sont jugés satisfaisants
- Les résultats de l'étape sont soumis à une étude approfondie


Faisabilité
Analyse besoins
Planification
Conception du
produit
Conception
détaillée
Implantation et
test
Intégration et
test
Installation et
test
Exploitation
et maintenance

Faisabilité (pourquoi ?)

Répondre aux questions

- Pourquoi faut-il réaliser ce logiciel ?
- Y a-t-il de meilleures alternatives ?
- Benchmark, étude orientée vers la comparaison de performances
- Le logiciel sera-t-il satisfaisant pour les utilisateurs ?
- Y a-t-il un marché pour le logiciel ?
- A-t-on le budget, le personnel, le matériel nécessaires ?

Analyse des besoins (quoi ?)

Analyse des besoins (quoi ?)

- Définir précisément les fonctions que le logiciels doit réaliser/fournir
- Le résultat de cette phase est le cahier des charges du logiciel

Conception (comment ?)

- Définir la structure du logiciel
- Les résultats comprennent l'architecture du logiciel (décomposition en modules) et la spécification des interfaces des modules
- La définition des algorithmes de chacune des procédures des modules est appelée la conception détaillée du logiciel

Implantation et test (comment ?)

- Implantation et test (comment ?)
  - Implanter les procédures des modules
  - Tests unitaires
- Intégration et test
  - Intégrer les différents modules
  - Valider / vérifier l'adéquation de l'implantation, de la conception et de l'architecture avec le cahier des charges (acceptation)

Installation et test

- Installation et test
  - Déploiement du logiciel chez le client et tests avec un sous ensemble d’usager choisi
- Exploitation et maintenance
  - Utilisation en situation réelle, retour d’information des usagers, des administrateurs, des gestionnaires…
  - Maintenance corrective, perfective et adaptative

Problèmes du modèle en cascade

- Découpage rigide du projet en étapes distinctes
- difficile de s’adapter aux changements des besoins utilisateurs
- Modèle bien adapté si les spécifications peuvent être précises dès le début
- Toutefois, il est rare d’avoir des spécifications stables
- Les tests sont prévus tardivement

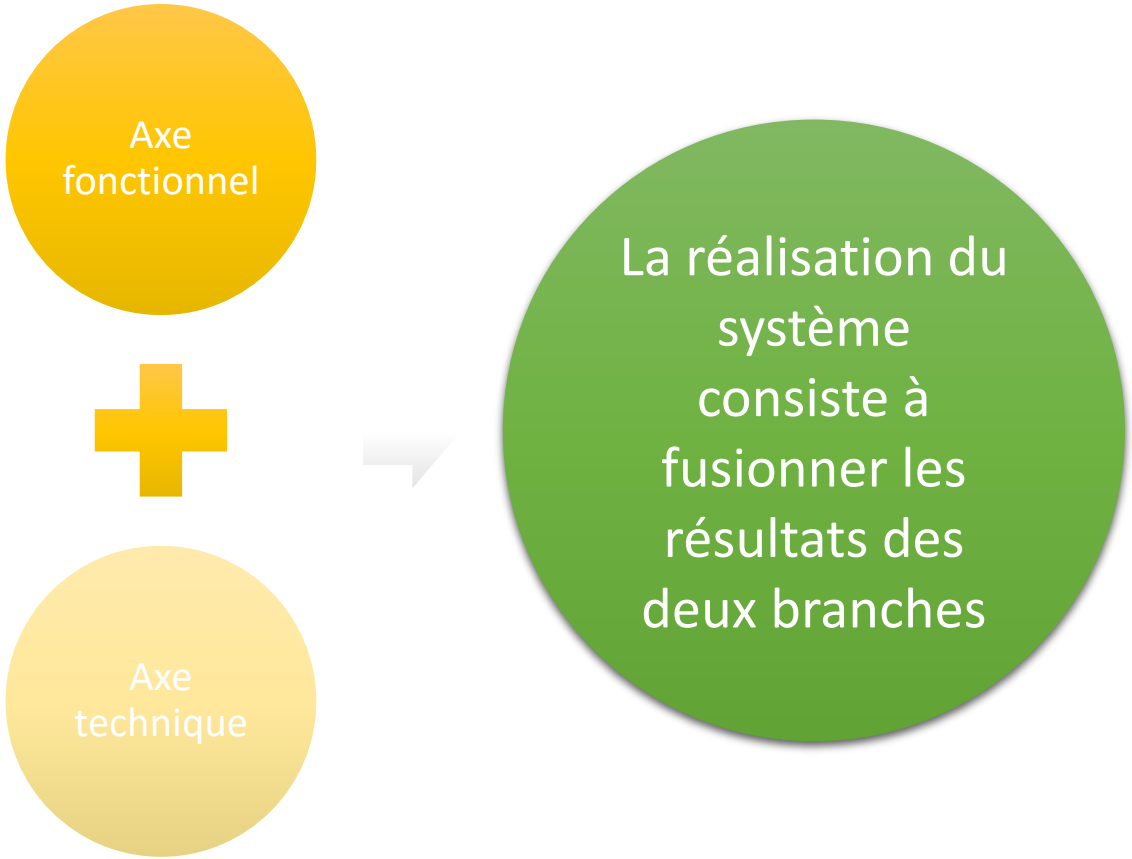
Quels sont les avantages ?

- Simple
- Logique
- Contrôle facile
- Accent sur la documentation et la structure
- Idéal pour les projets logiciels stables

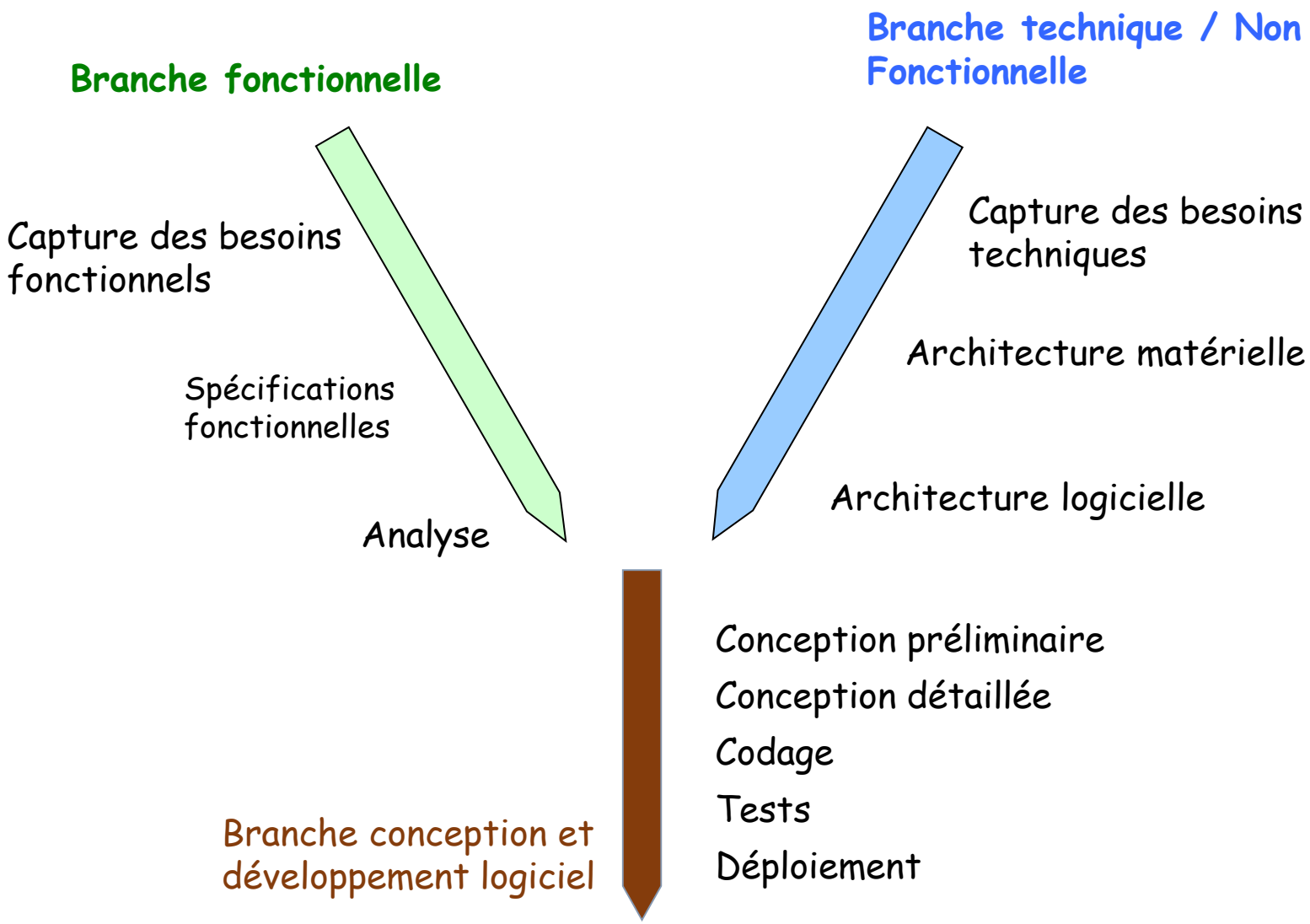
Les inconvénients du modèle

- Absence de flexibilité
- Incapacité de revenir en arrière
- Nécessite une phase de conception parfaite
- De moins en moins de droit à l'erreur avec l'avancement du projet, notamment lors des tests
- Les projets utilisant ce modèle ont tendance à être abandonnés

Modèle en Y


Axe fonctionnel
Axe technique
La réalisation du système consiste à fusionner les résultats des deux branches

Cycle de vie en Y



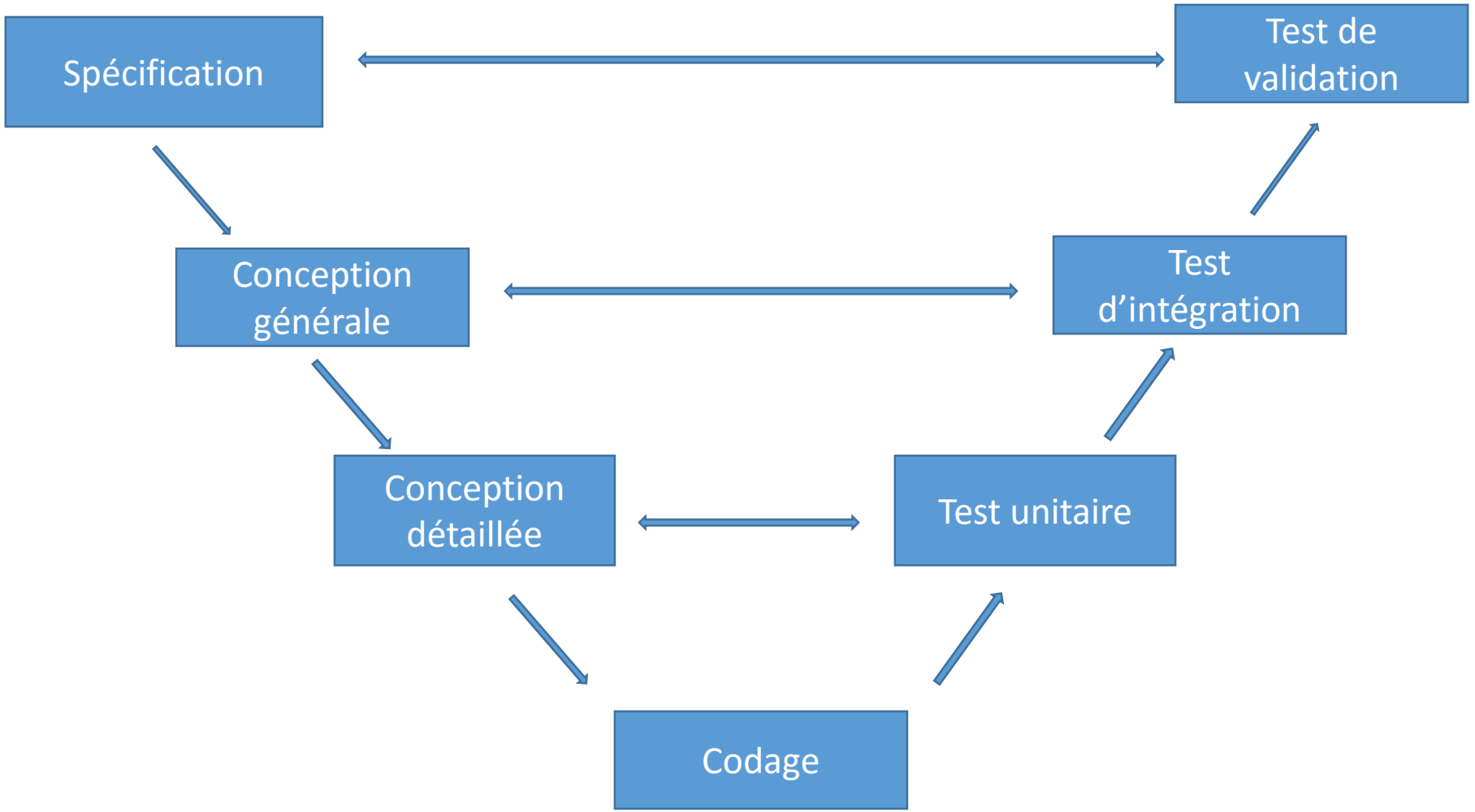
2TUP (2 tracks unified process)

Le 2TUP propose un cycle de développement en Y itératif.

Le projet est découpé en itérations et chaque itération est un cycle en Y

Cycle en V

- Le modèle du cycle en V est un modèle de gestion de projet imaginé suite au problème du modèle en cascade.
- Son apport : Il permet de limiter les retours aux étapes précédentes.
- Mettre en aval, certaines tâches tardives : intégration, validation


Spécification
Test de validation
Conception générale
Test d'intégration
Conception détaillée
Test unitaire
Codage

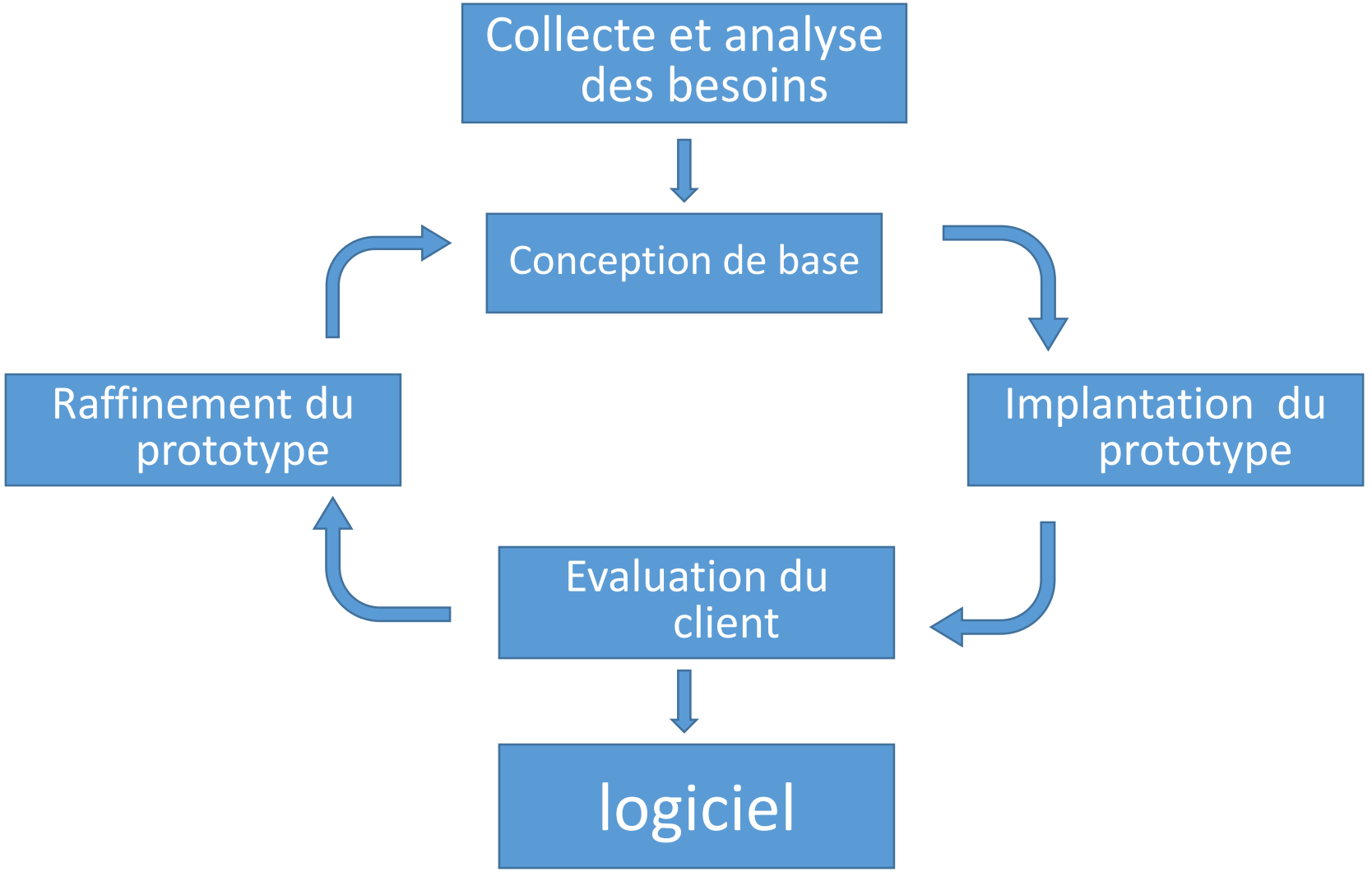
- Les cas de tests sont élaborés (plans de test) lors de la partie descendante d'un cycle en V, en parallèle des phases de spécification, conception, et de codage. L'exécution des tests se fera dans chacune des phases de la partie remontante du cycle, sur la base des cas de tests élaborés dans la phase jumelle de la partie descendante. On parle alors de test unitaire, de test d'intégration, de test système (ou de test de conformité).

- Tâches effectuées en parallèle
- Horizontalement : préparation de la vérification
- Ex. : dès que la spécification fonctionnelle est faite :
  - ✓plan de tests de validation
  - ✓plan d'évaluation des performances

Le modèle par prototypage

- Le modèle en cascade suppose que les besoins sont clairs, arrêtés et bien définis
- Le modèle par prototypage est intéressant
  - Besoins pas clairement définis
  - Besoins changeant au cours du temps
- Le prototypage permet le développement rapide d'une ébauche du futur logiciel
  - Prototype jetable
  - Prototype évolutif

Le modèle par prototypage



Prototypage

- Prototypage
  - Horizontal
  - Vertical

Avantage du modèle

- Pas d’effet tunnel
- Meilleure adaptation aux changements
- Mise en œuvre d’une analyse de risques
- Ajustement des choix techniques et fonctionnels tôt dans le processus

RUP (Rational Unified Process)

- Piloté par les cas d'utilisation
- Piloté par les risques
- **Centré sur l'architecture**
- Développement itératif et incrémentiel

Centré sur l'architecture …

- Tout système complexe doit être décomposé en parties modulaires afin de garantir une maintenance et une évolution facilitées.
- Cette architecture (fonctionnelle, logique, matérielle etc …) doit être modélisée en UML et pas seulement documentée sous forme textuelle.

Centré sur l’architecture …

- Très simplement, on peut dire que l’architecture est la structure d’un système.
- La description de l’architecture n’est pas monolithique, elle est constituée de l’agrégation cohérente de différents points de vue.
- RUP préconise d’utiliser le modèle des 4+1 vues ci-contre pour guider l’élaboration de l’architecture.

le modèle des 4+1

Représentation du SI sur les données:
Diag. de classes, séquence …

Traduction du Si en modules:
Diag. Composants

Vue Logique

Vue Composants

Vue Utilisateur

Vue Processus

Vue Déploiement

Aspect fonctionnel
Diag d'états transition
Diag d'activités

Guide l'analyse des besoins,
cimente les vues:
cas d'utilisation, scénarios

Projection des composants sur le matériel:
Diag. de déploiement

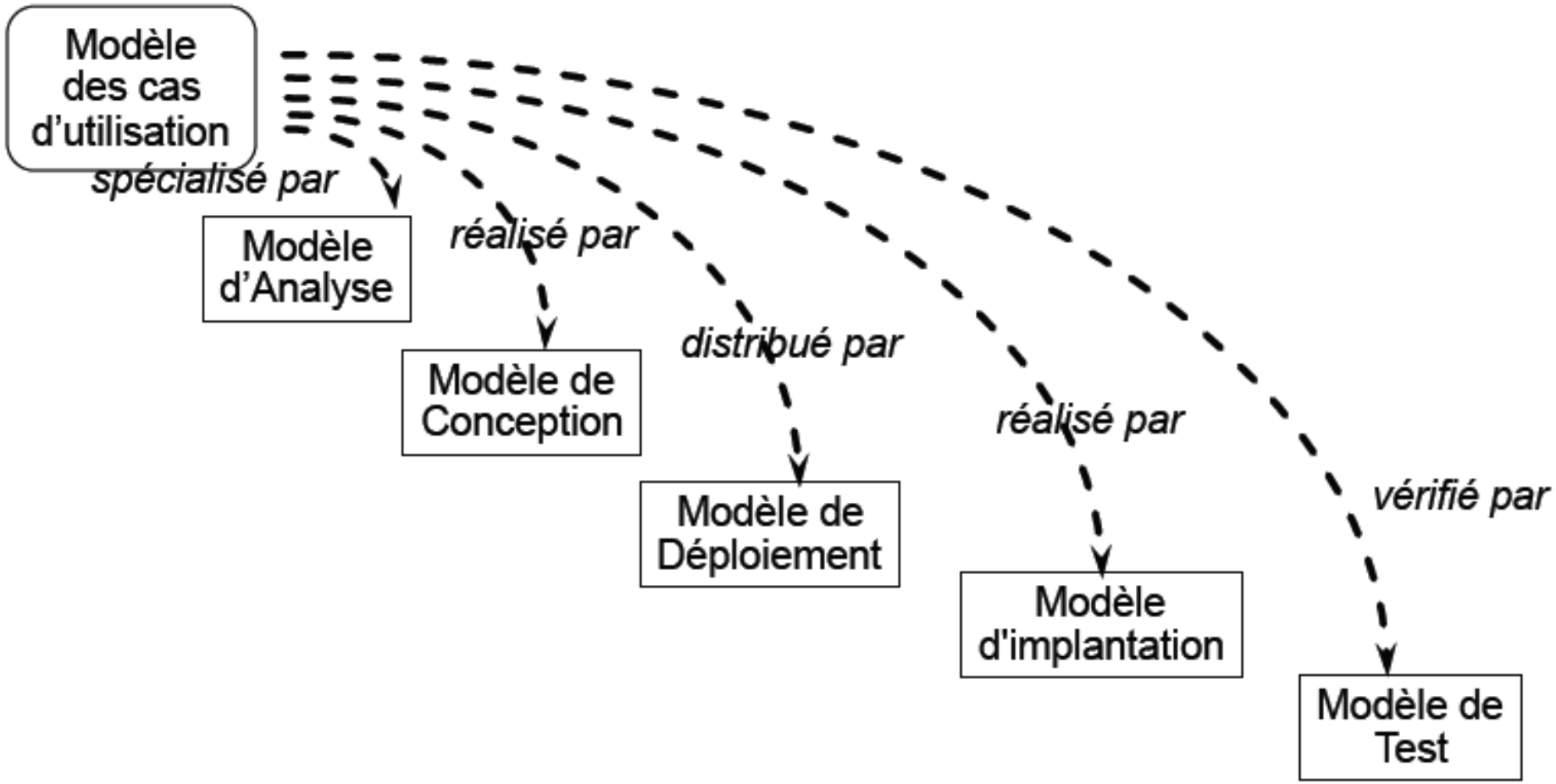
Piloté par les risques …

- Les risques majeurs du projet doivent être identifiés au plus tôt, mais surtout levés les plus rapidement possible.
- Les mesures à prendre dans ce cadre déterminent l’ordre des itérations.

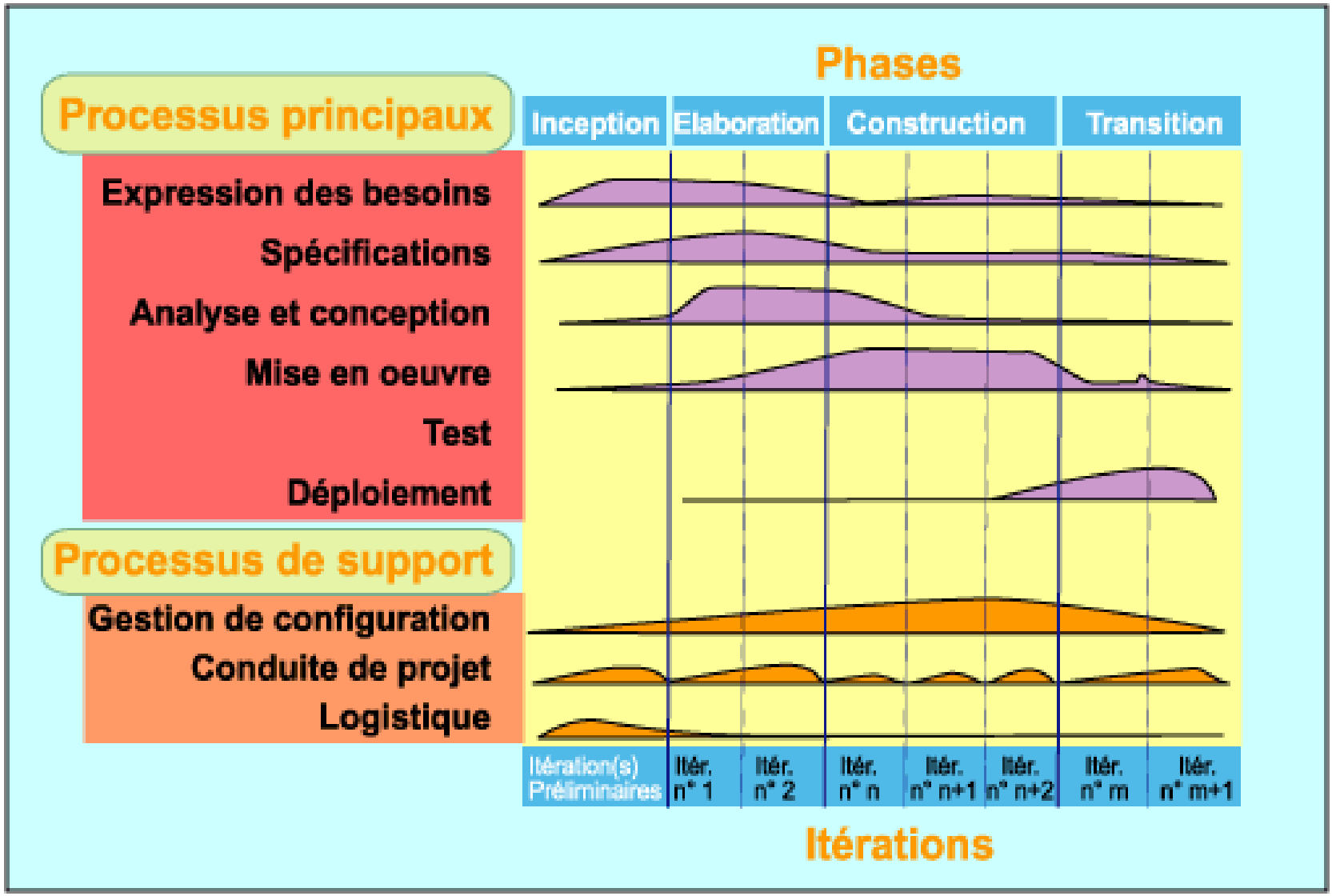
Piloté par les cas d’utilisation

- Le projet est mené en tenant compte des besoins et des exigences des utilisateurs.
- Les cas d’utilisation du futur système sont identifiés, décrits avec précision et priorisés.

RUP (Rational Unified Process)



RUP (Rational Unified Process)



Phase RUP

Création (Inception)

- Vision approximative, définition de l'étendue du projet, estimés vagues
- Le projet est-il réalisable?
- étude de faisabilté

Élaboration

- Vision raffinée
- Développement itératif de l'architecture de base
- Résolution des risques les plus élevés
- Identification de la plupart des besoins
- Estimés plus réalistes

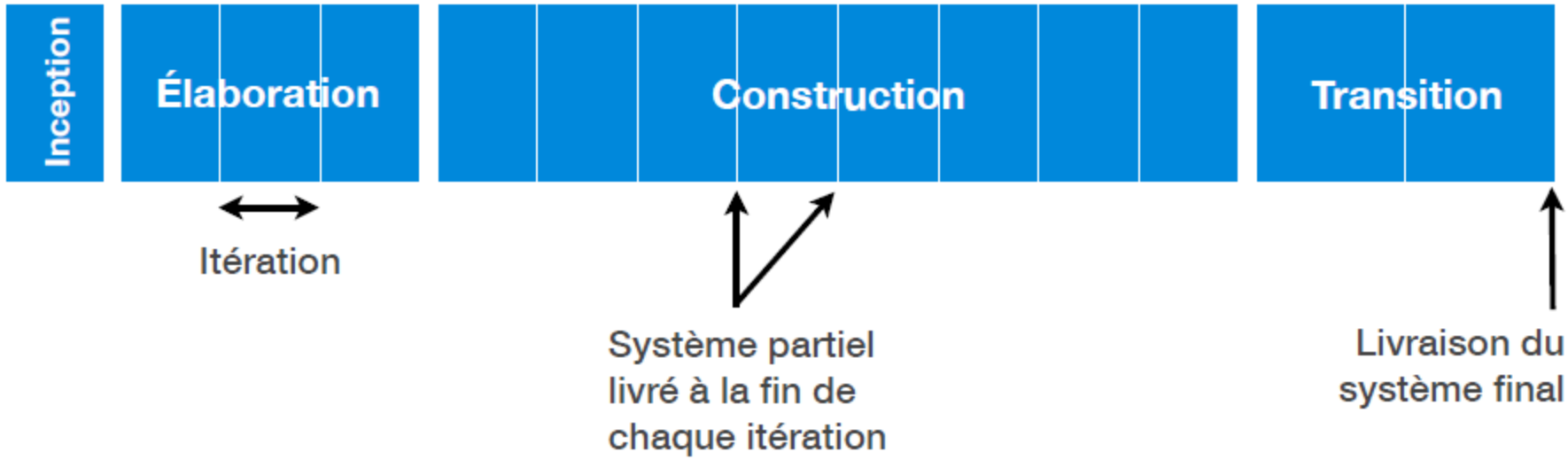
Construction

- Implémentation itérative des éléments plus simples / à plus faible risque
- Préparation pour le déploiement

Transition

- Implantation du système dans un environnement de production

Phase RUP


Inception
Élaboration
Construction
Transition
Itération
Système partiel livré à la fin de chaque itération
Livraison du système final

Développement Agile
Scrum

En génie logiciel, beaucoup de méthodes dites agiles sont apparues ces dernières années : Scrum, Lean, eXtreme Programming...

Contrairement aux méthodes traditionnelles prédictives, elles permettent d'éviter une définition trop précoce et figée des besoins grâce à leur souplesse et leur adaptabilité.

Parmi ces méthodes, Scrum s'est peu à peu imposée comme la méthode agile la plus populaire, bien loin maintenant devant les autres.

Les 12 principes du manifeste agile

Prioriser la satisfaction du client

Accepter les changements

Livrer en permanence des versions opérationnelles de l'application

Assurer le plus souvent possible une coopération entre l'équipe du projet et les gens du métier

Construire les projets autour de personnes motivées

Favoriser le dialogue direct

Mesurer l'avancement du projet en fonction de l'opérationnalité du produit

Adopter un rythme constant et soutenable par tous les intervenants du projet

Contrôler continuellement l'excellence de la conception et la bonne qualité technique

Privilégier la simplicité en évitant le travail inutile

Auto-organiser et responsabiliser les équipes

Améliorer régulièrement l'efficacité de l'équipe en ajustant son comportement

Les trois piliers de la méthode SCRUM sont les suivants :

- La transparence : les aspects importants du processus doivent être visibles à tous. C'est pour cela que SCRUM insiste sur le fait de créer un langage commun entre les membres de l'équipe et le management, ce qui permettra une compréhension commune du projet.

- L'inspection : un bilan régulier sur les résultats produits est réalisé afin de détecter les écarts indésirables. Il est important que ces inspections soient faites par un inspecteur bien formé et cela de manière adaptée car cela pourrait nuire à l'avancement du projet.

- L'adaptation : lorsqu'un écart est constaté pendant l'inspection, le processus devra être adapté grâce à des actions visant à améliorer la situation

Agiles vs classiques

Approche prédictive des méthodes classiques qui se basent sur une expression figée du besoin. Elle aboutit à la création d'un logiciel développé d'un seul jet. Comme son nom l'indique, tout est prévu, rien n'est laissé au hasard

Approche adaptative des méthodes agiles qui acceptent l'évolution du besoin en cours de développement et propose la mise en place de l'application par briques incrémentales et itératives.

Agiles vs classiques

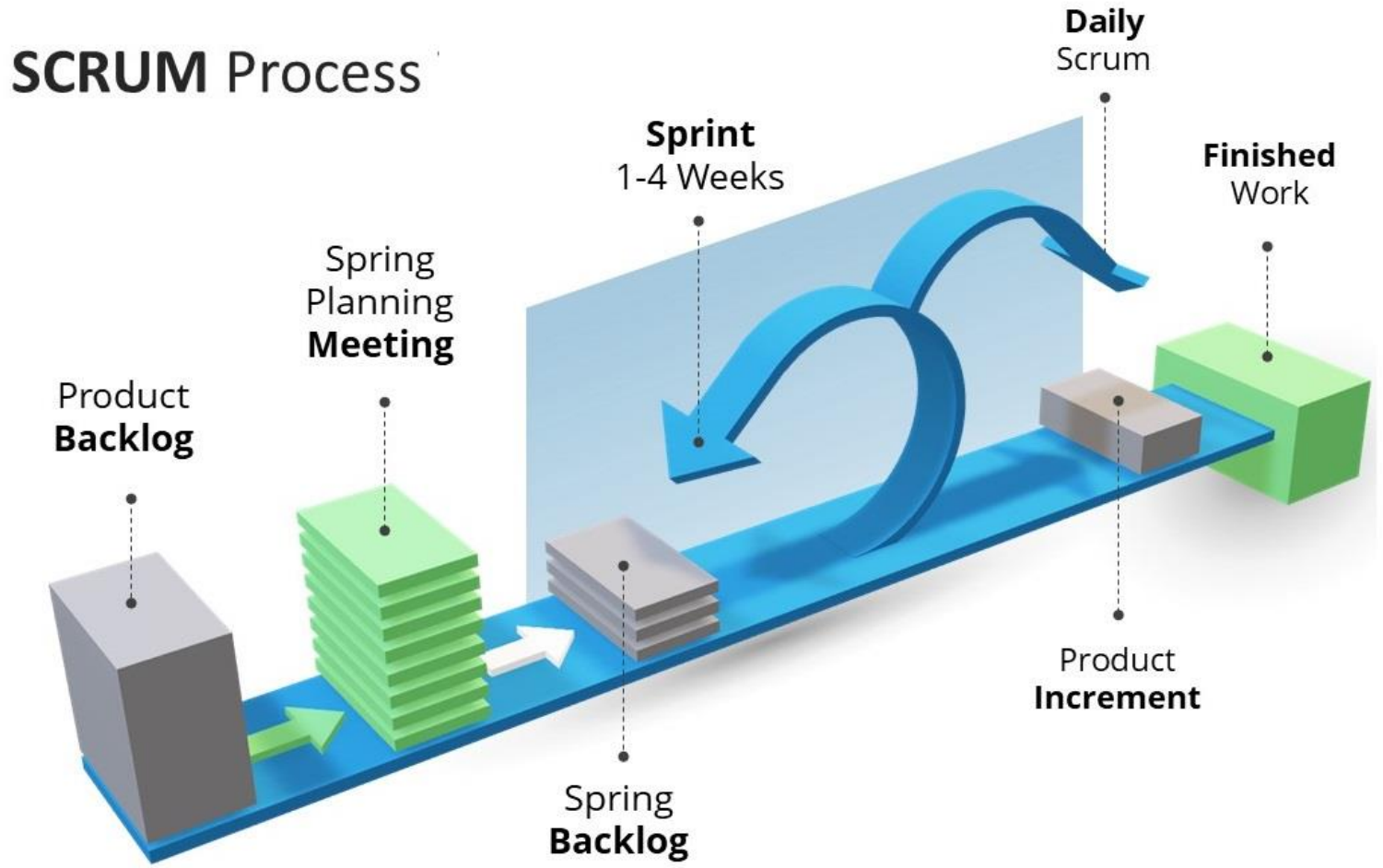
	MÉTHODES AGILES	MÉTHODES CLASSIQUES
Objectif	Satisfaire l'utilisateur	Respecter le besoin initial et les engagements
Changement	Accepter le changement	Opposé au changement ou, en tout cas, moins enclin à l'accepter compte tenu des livraisons tardives et des processus de gestion lourds
Livraison	Livrer fréquemment	Livrer en une seule fois une application « finalisée »
Équipe	Travailler en synergie	Travailler de façon segmentée (chacun voit sa partie du travail)
Moteur	Stimuler la motivation	Stimuler la productivité
Communication	Communiquer en direct avec les opérationnels	Communiquer de façon verticale en passant par des relais hiérarchiques (par exemple, le chef de projet MOE est relais entre la MOA et ses développeurs)
Indicateurs	Un seul indicateur : les fonctionnalités implémentées	Justifier par les indicateurs (sans livraison intermédiaire, les indicateurs sont les seuls justificatifs de l'avancement, des écarts, etc.)

	MÉTHODES AGILES	MÉTHODES CLASSIQUES
Rythme	Bannir les *rushs* de production	Adapter la production aux contraintes projets
Qualité	*Que ce soit en méthode agile ou classique, l'excellence technique est essentielle. Ce neuvième principe sert principalement à justifier le fait que les méthodes agiles prônent la qualité des livrables techniques (contrairement à ce que peuvent en dire un certain nombre de détracteurs)*	
Livrables	Rester concentré sur l'essentiel : la production	Une documentation précise est essentielle pour assurer les échanges et la validation autours des besoins client
Autonomie	Favoriser une certaine autonomie des équipes	Encadrer scrupuleusement le travail des équipes
Amélioration	Intégrer la notion d'amélioration continue tout au long du projet	Introspection possible mais uniquement en fin de projet

Au final, ce qui est le plus fréquemment reproché à l'utilisation des méthodes classiques tient en deux mots : effet tunnel.

Les projets qui suivent la méthode agile SCRUM sont divisés en plusieurs cycles de travail relativement courts que l'on appelle « sprints ».

Les sprints peuvent durer d'une à quatre semaines. Ils permettent également de réajuster ou réorienter la direction prise par le projet si besoin.

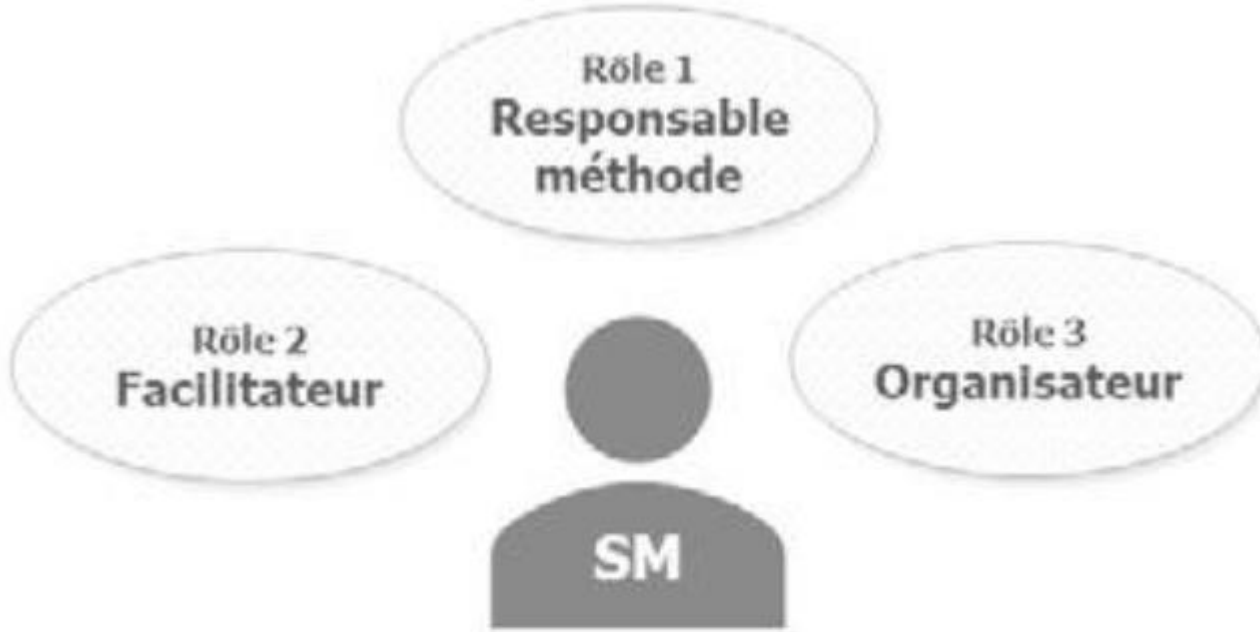


Scrum Master

Son positionnement au niveau de l'équipe peut être vu comme assez similaire à celui du chef de projet. Il est central dans l'organisation et intervient comme référent. Cependant, il existe deux différences essentielles :

Sur la notion de hiérarchie : le SM n'est pas au-dessus mais au même niveau que l'équipe de développement.

Sur son périmètre d'action : le chef de projet comme un leader dans la partie organisationnelle et opérationnelle. Le SM, de son côté, est sur un périmètre qui se « limite » aux aspects méthodologiques (application de Scrum) et organisationnels (implication de l'équipe, résolution des conflits, etc.).



Scrum Master

COMPÉTENCES

CONNAISSANCES
Scrum et agile

CAPACITÉ
Transmettre et évaluer les acquis

INTELLIGENCE
Relationnelle

Product Owner

Le Product Owner ou PO est le responsable de la définition et de la conception d’un produit. Là où le Scrum Master porte la « vision méthodologie » sur le projet, le Product Owner porte quant à lui la « vision produit ».

De façon plus spécifique à Scrum, sa mission s'articule autour des trois rôles suivants :

- Responsable du produit : le PO va définir le produit et garantir sa cohérence au travers de l'artefact central de Scrum, le « carnet de produit ». Ce catalogue de fonctionnalités est à la charge du PO qui le gère afin d'apporter continuellement de la valeur au produit
- Validateur : étant le seul responsable du carnet de produit, le PO va jouer un rôle de validateur. Il tranche en respectant les apports et les propositions de l'équipe,
- Traducteur de besoins : si le contenu du carnet de produit est de la responsabilité du Product Owner, ce dernier n'est pas le seul à consulter ce document. Ce carnet est central et utilisé par tous les membres de l'équipe. Il doit donc être accessible et surtout compréhensible par tous. L'ensemble des fonctionnalités doit être traduit et présenté de manière claire.

Les Membres de l’équipe :

Dans la méthode SCRUM, Ce sont les personnes chargées de la réalisation du sprint et d’un produit utilisable en fin de sprint. Il peut s’agir de développeurs, architectes, personnes chargées de faire des tests fonctionnels. L’équipe s’adresse directement au client.

Le Sprint

Le cœur de Scrum est le Sprint, qui a une boîte de temps (time-box), une durée, d'un mois ou moins au cours de laquelle un Incrément Produit « Fini » fonctionnel et potentiellement publiable est créé.

Les sprints ont une durée cohérente durant la phase de développement. Un nouveau Sprint commence immédiatement après la conclusion du Sprint précédent.

Les Sprints contiennent et consistent en

- une planification du Sprint (Sprint Planning),
- des mêlées quotidiennes (Daily Scrums),
- des activités de développement,
- une revue de sprint (Sprint Review)
- et une rétrospective de Sprint (Sprint Retrospective).

Planification du Sprint

Le travail à effectuer durant un Sprint est défini durant la réunion de Planification du Sprint (Sprint Planning). Ce plan est créé de manière collaborative par tous les membres de l'équipe Scrum. Le Scrum Master veille à ce que l'événement ait lieu et que les participants en comprennent le but. Le Scrum Master apprend à l'équipe Scrum à le garder dans la boîte de temps (time-box).

Daily Scrum

- La mêlée quotidienne (Daily Scrum) est un événement de 15 minutes (time-boxé) destiné à l'équipe de développement. La mêlée quotidienne est tenue tous les jours du Sprint.

- L’équipe de développement planifie le travail pour les prochaines 24 heures. Cela optimise la collaboration et la performance de l'équipe tout en inspectant le travail depuis la dernière mêlée quotidienne et envisageant le travail restant durant le Sprint.

- La mêlée quotidienne est tenue à la même heure et au même lieu chaque jour pour réduire la complexité.

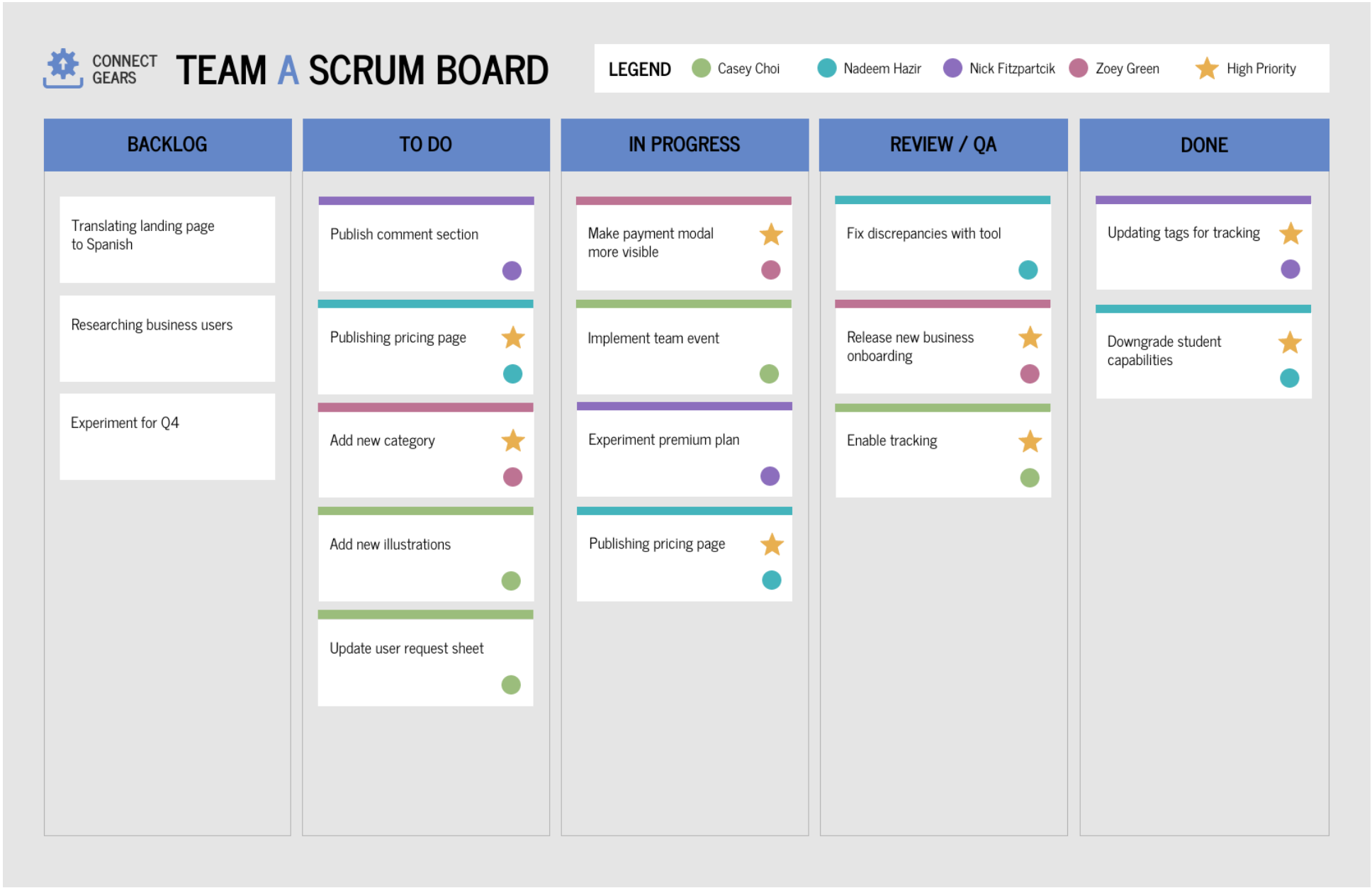
Daily Scrum

La structure de la réunion est définie par l'équipe de développement et peut être conduite de différentes manières si elle se concentre sur la progression vers l'objectif du Sprint. Certaines équipes de développement utiliseront les questions, d'autres seront plus basées sur la discussion. Voici un exemple de ce qui pourrait être utilisé :

- Qu'est-ce que j'ai fait hier qui a aidé l'équipe de développement à atteindre l'objectif du Sprint ?
- Que ferai-je aujourd'hui pour aider l'équipe de développement à atteindre l'objectif du Sprint ?
- Est-ce que je vois des obstacles qui m'empêchent ou empêchent l'équipe de développement de respecter l'objectif du Sprint ?

La Daily Scrum se tient un scrum board, qui donne la visibilité sur l’avancement du sprint. Cette mélé peut mettre à jour la scrum board.

Exemple de Scrum Board



backlog

Le backlog est une liste de tâches priorisées définissant les caractéristiques d'un produit. Il est un des éléments fondamentaux de la méthodologie Scrum. Il s'agit de l'outil de travail principal du Product Owner qui se charge de recueillir les besoins auprès des parties prenantes et de les transformer en liste de fonctionnalités prêtes à être développées par l'équipe de développement.

Communément on peut formaliser ce items sous forme de Users Stories (US). A ces US peuvent également s'ajouter des évolutions et des anomalies recensées sur le produit.

Au lancement de chaque incrément, à l'issue d'un ou plusieurs sprints de développement, le Product Owner enrichit le backlog avec :

- les feedbacks remontés par les utilisateurs finaux
- l'émergence de nouveaux besoins

Cahier des charges

- Élaboré en début de projet
- Périmètre figé
- Difficilement maintenable
- Budget défini sur un ensemble de fonctionnalités

Backlog

- Élaboré au fil du temps
- Périmètre évolutif
- Affiné régulièrement
- Budget défini sur un ensemble de ressources mobilisées

Epic et user story

Epic

Une epic est une grosse user story non encore affinée qui n'est pas envisagée pour les prochains sprints. Elle sera découpée en User Stories le moment venu.

User story

Une User story est une exigence du système à développer, formulée en une ou deux phrases dans le langage de l'utilisateur. Les User Stories émergent au cours d'ateliers de travail menés avec le Métier, le Client ou les Utilisateurs.

Quelques Exemples

« En tant que recruteur, je peux déposer des offres d'emploi »;
« En tant qu'utilisateur, je peux réserver une chambre d'hôtel »;
« En tant utilisateur, je peux annuler une réservation »;
« En tant que jeune diplômé, je peux créer un CV »;
« En tant que jeune diplômé je peux supprimer un CV »; « En tant que jeune diplômé je peux modifier un CV »…


Wishlist
Theme
As a customer, I want to be able to have wishlists so that I can come back to buy products later
Epic
As a customer, I want to be able to save a product in my wishlist so that I can view it again later
As a customer, I want to be able to view my wishlist so that I can buy items from it
Stories
Put 'Add to wishlist' button on each product page
Create new db to store wishlist items
Create page to display user's wishlist
Add 'View wishlist' link to homepage
Tasks

ID	Thème	User Story	Priorité	Story point
1	Afficher produit	En tant que visiteur, je peux consulter tous les produits par : catégorie, coopérative, origine ou matière première.	Must	5
2	Ajouter au Panier.	En tant que visiteur, je peux ajouter des produits au panier.	Must	8
3	Effectuer la commande.	En tant que visiteur, je peux effectuer une commande après avoir authentifier.	Must	8
4	Authentification.	En tant que visiteur, je dois m'authentification pour envoyer la commande.	Must	8
5	Valider la commande.	En tant que administrateur, je peux valider une commande envoyé par client.	Must	5
6	Générer la facture	En tant que client, je peux générer ma facture.	Must	5
7	Gérer la livraison client.	En tant que administrateur, je peux modifier l'état de la commande client en fonction de son état courant.	Must	2

8	**Gérer la livraison fournisseur.**	**En tant que administrateur, je peux modifier l'état de la commande fournisseur en fonction de son état courant.**	**Must**	**2**
9	Gérer produit	En tant que administrateur, je peux ajouter un nouveau produit d'une coopérative existante.	Must	2
10	Gérer produit	En tant que administrateur, je peux supprimer un produit.	Must	2
11	Gérer produit	En tant que administrateur, je peux modifier un produit.	Must	2
12	Gérer produit	En tant que administrateur, je peux afficher la liste des produits.	Must	5
13	Gérer coopérative	En tant que administrateur, je peux ajouter une nouvelle coopérative.	Must	2
14	Gérer coopérative	En tant que administrateur, je peux supprimer une coopérative existante.	Must	2
15	Gérer coopérative	En tant que administrateur, je peux modifier une coopérative existante.	Must	2

Sprint Backlog

Le Sprint Backlog représente l'ensemble des tâches que l'équipe de développement s'engage à produire au cours du sprint.

D'abord, le responsable du produit (Product Owner) identifie toutes les fonctionnalités nécessaires pour créer le produit souhaité. Puis, en fonction de l'objectif et de la durée du sprint, choisissez en équipe les items que vous allez embarquer pour le prochain sprint.

Enfin, décomposez ces items en de multiples petites tâches techniques pour vous aider à y voir plus clair. Cette liste de petites tâches spécifiques s'appelle le Sprint Backlog.

Exemple

User Story Sprint baacklog

		Tâche		Estimation par Jour
1	consulter les produits par : catégorie, coopérative, origine ou matière première.	Développer l'interface de sélection de produit par matière premières et par origine (Front)	Combo box, liste déroulant pour la sélection	0,5
		Développer le traitement de la sélection Back-end	Développer le endpoint rest de selection	0,5
		….	….	
		….	….	