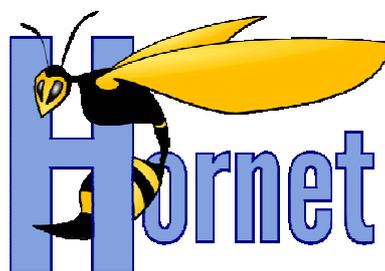


[Hornet] Charte de méthodologie



Charte de méthodologie Hornet

Cette création est mise à disposition selon le Contrat Paternité - Pas d'Utilisation Commerciale - Partage des Conditions Initiales à l'Identique disponible en ligne <http://creativecommons.org/licenses/by-nc-sa/2.0/fr/> ou par courrier postal à Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA

SUIVI DES MODIFICATIONS

Version	Auteur	Description	Vérification	Date
1.0	S.Percier	Initialisation du document		21/12/2012
1.1	S.Percier	Modifications mineures	S.Heurtematte	17/01/2013

SOMMAIRE

SUIVI DES MODIFICATIONS	2
SOMMAIRE.....	3
TABLEAUX	3
FIGURES.....	3
1 INTRODUCTION.....	5
2 METHODOLOGIE PROJET.....	6
2.1 PHASE D'ETUDE.....	7
2.1.1 Identifier les exigences.....	7
2.1.2 Analyse.....	7
2.1.3 Identification des risques inhérents au projet.....	7
2.1.4 Plan de projet.....	7
2.2 PHASE D'IMPLEMENTATION.....	7
2.2.1 Itérations de développement.....	7
2.2.2 Implémentation itération.....	7
2.2.3 Evaluation itération.....	8
2.2.4 Réévaluation des risques.....	8
2.3 PHASE DE RECETTE.....	8
2.3.1 Phase de recette technique.....	8
2.3.2 Phase de recette fonctionnelle.....	8
2.4 PHASE DE DEPLOIEMENT	8
3 METHODOLOGIE D'ANALYSE, DE MODELISATION ET DE MAQUETTAGE	9
3.1 ANALYSE TECHNIQUE	9
3.2 ANALYSE FONCTIONNELLE.....	10
3.3 ANALYSE SGBD	10
3.4 MODELISATION	10
3.5 MAQUETTAGE	10
4 METHODOLOGIE DE TEST.....	11
4.1 OPERATIONS DE TEST	11
4.1.1 Tests durant la phase d'implémentation	11
4.1.2 Tests durant la phase de recette.....	12
4.2 OPERATIONS DE VERIFICATION DANS LE CADRE D'UN TIERS DEVELOPPEMENT	13
4.2.1 Vérifications liées à la recette fonctionnelle.....	13
4.2.2 Vérifications liées à la filière Hornet.....	13
4.2.3 Vérifications documentaires.....	14
5 EQUIPE PROJET.....	15
5.1 COMPETENCES DSI	15
5.2 COMPETENCES TIERS DEVELOPPEUR	16
5.3 COMITE DE PROJET	16
5.4 COMITE DE PILOTAGE.....	17

TABLEAUX

Tableau 1 : Méthodologie de test déclinée pour la phase d'implémentation	12
Tableau 2 : Méthodologie de test déclinée pour la phase de recette	12

FIGURES

Figure 1 : Méthodologie projet (UP)	6
Figure 2 : Méthodologie d'analyse, modélisation et maquettage	9

Figure 3 : Méthodologie de test.....	11
Figure 4 : Equipe projet – compétences nécessaires	15

1 Introduction

La méthodologie de projet à employer dans le cadre de la filière de développement Hornet est basée sur les principes itératifs (Unified Process) pour pouvoir :

- Paralléliser les tâches associées à un lotissement fonctionnel et technique.
- Présenter une version applicative enrichie au fur et à mesure de l'avancement des travaux pour multiplier les points de contrôle et itérer la documentation correspondante.
- Démarrer au plus tôt la phase de recette dans le cycle de vie du projet.
- Identifier et limiter les risques inhérents au projet.

2 Méthodologie projet

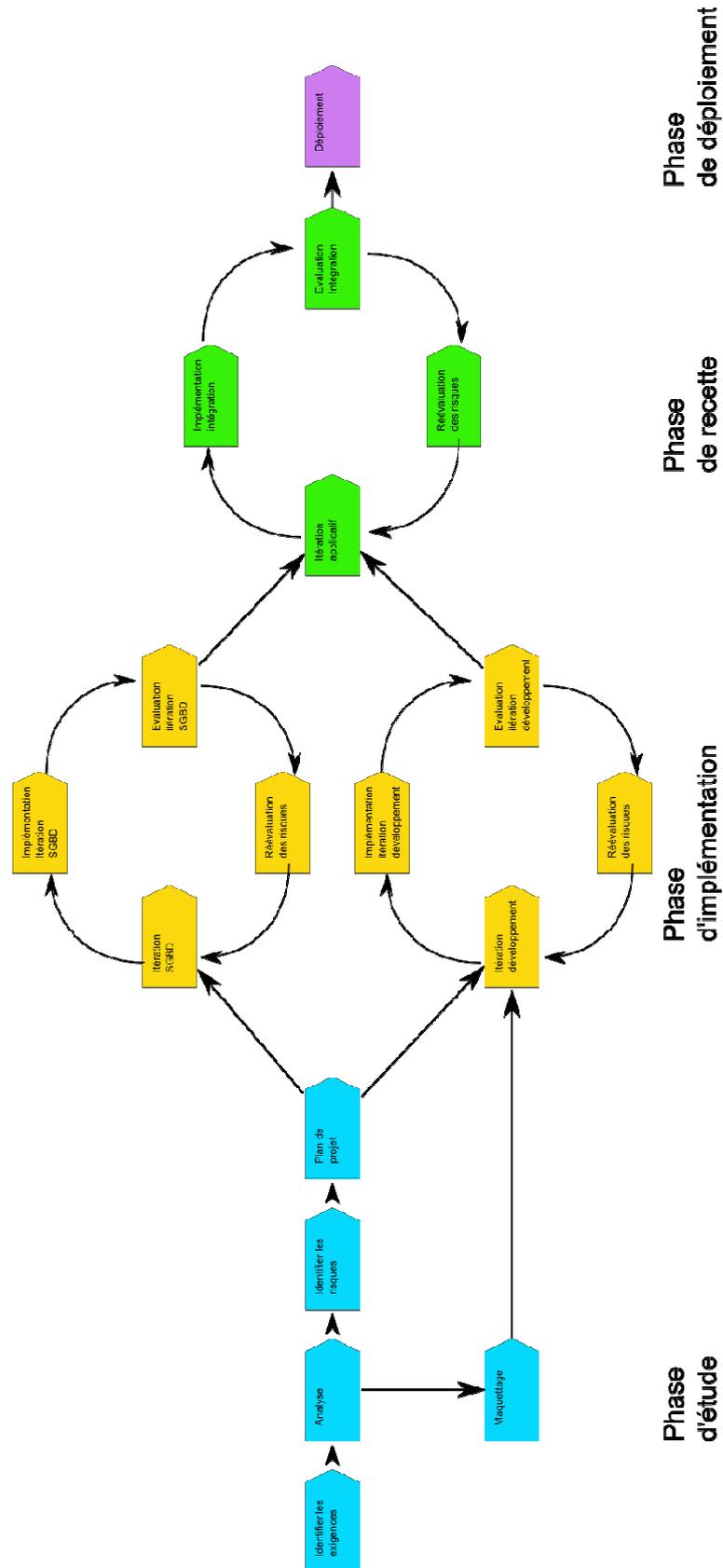


Figure 1 : Méthodologie projet (UP)

Comme illustré dans cette figure, la mise en œuvre d'un applicatif avec la filière de développement Hornet doit se découper en quatre phases :

- Etude
- Implémentation
- Recette
- Déploiement

2.1 Phase d'étude

La phase d'étude consiste à analyser le besoin exprimé, à identifier les risques inhérents au projet et planifier le projet en suivant les principes itératifs (Plan management projet).

Voici les différentes tâches attendues lors de cette phase :

2.1.1 Identifier les exigences

Reprise du cahier des charges et des chartes/normes propres à la filière de développement pour identifier et numéroter les exigences et ainsi en garantir la traçabilité tout le long du cycle de vie du projet.

Ces exigences concernent à la fois les aspects fonctionnels et techniques propres au contexte du projet ou à la filière de développement.

2.1.2 Analyse

Détaillée plus bas (voir partie Analyse).

2.1.3 Identification des risques inhérents au projet

- Points critiques dans le planning,
- Problématiques techniques nouvelles à étudier,
- Problématiques fonctionnelles liées à une mauvaise définition du besoin...

2.1.4 Plan de projet

- Recensement des itérations fonctionnelles et techniques,
- Définition du périmètre de chaque itération,
- Plan de test prenant en compte les transitions entre chaque itération,
- Planning du projet général par phase,
- Planning du projet détaillé par itération.

2.2 Phase d'implémentation

La phase d'implémentation consiste à implémenter l'ensemble des composants constituant l'applicatif. Les développements sont parallélisés et compartimentés en fonction des compétences nécessaires et des couches de l'architecture multi-niveaux. Les itérations de chacun de ces développements sont itérées par l'ajout successif de fonctionnalités métiers comme déterminé préalablement dans le plan de projet.

Voici les différentes tâches attendues lors de cette phase dans l'environnement de développement.

2.2.1 Itérations de développement

- Côté SGBD : itérations par ajout successif des besoins associés aux fonctionnalités métiers associées aux itérations de développement Serveur et Client.
- Côté développement Serveur et Client : itérations par ajout successif des fonctionnalités métiers à réaliser.

2.2.2 Implémentation itération

- SGBD : elle consiste en l'ensemble des analyses nécessaires (voir §3 pour plus de détail) à la modélisation de la base de données. La réalisation des scripts de création de

la base de données et d'auto-alimentation en données de cette base pour les tests d'intégration. La mise en œuvre de l'ensemble des requêtes SQL nécessaires à l'itération en cours est effectuée.

- Développement Serveur et Client : elle consiste en l'ensemble des analyses nécessaires (voir partie Analyse) à la modélisation des traitements associés aux fonctionnalités métier liées à l'itération. Le développement en langage Java/J2EE/HTML/JSP/XML est réalisé pour l'ensemble des fonctionnalités métiers associés à l'itération. Développement Client : elle consiste à enrichir la maquette en développement d'IHMs liées aux fonctionnalités métiers associés à l'itération

2.2.3 Evaluation itération

Détaillée plus bas (voir partie Méthodologie).

2.2.4 Réévaluation des risques

La phase d'implémentation d'une itération de développement permet de réévaluer les risques ou de déterminer des nouveaux risques par rapport à la liste des risques déjà identifiés.

2.3 Phase de recette

La phase de recette est découpée en deux sous-phases suivant les principes itératifs décrits plus haut, non plus vis à vis de composants applicatifs dédiés à une couche mais vis à vis d'une version de l'applicatif en cours de développement.

2.3.1 Phase de recette technique

Elle s'effectue dans l'environnement d'intégration et a pour but d'intégrer l'ensemble des développements parallélisés comme décrits ci-dessus. L'itération consiste à intégrer successivement :

- les deux couches (métier et données) fournissant une version de l'applicatif susceptible d'être recettée fonctionnellement. Un audit de code sur cette version peut être réalisé par la MOE.

2.3.2 Phase de recette fonctionnelle

Elle s'effectue dans l'environnement de recette. Le résultat de l'itération de la recette technique a permis de fournir une version de l'applicatif dont le périmètre fonctionnel a été préétabli. Le bon résultat de cette recette fonctionnelle détermine le passage vers une nouvelle itération de l'applicatif ou le passage en production quand l'ensemble du périmètre fonctionnel attendu a été évalué.

2.4 Phase de déploiement

La phase de déploiement consiste à mettre en production la dernière version applicative recettée. Cette phase nécessite de préciser la configuration serveur nécessaire et de fournir les scripts de constitution des livrables (scripts SGBD pour la « couche données » et script(s) Ant pour les « couches métiers et IHMs »).

3 Méthodologie d'analyse, de modélisation et de maquettage

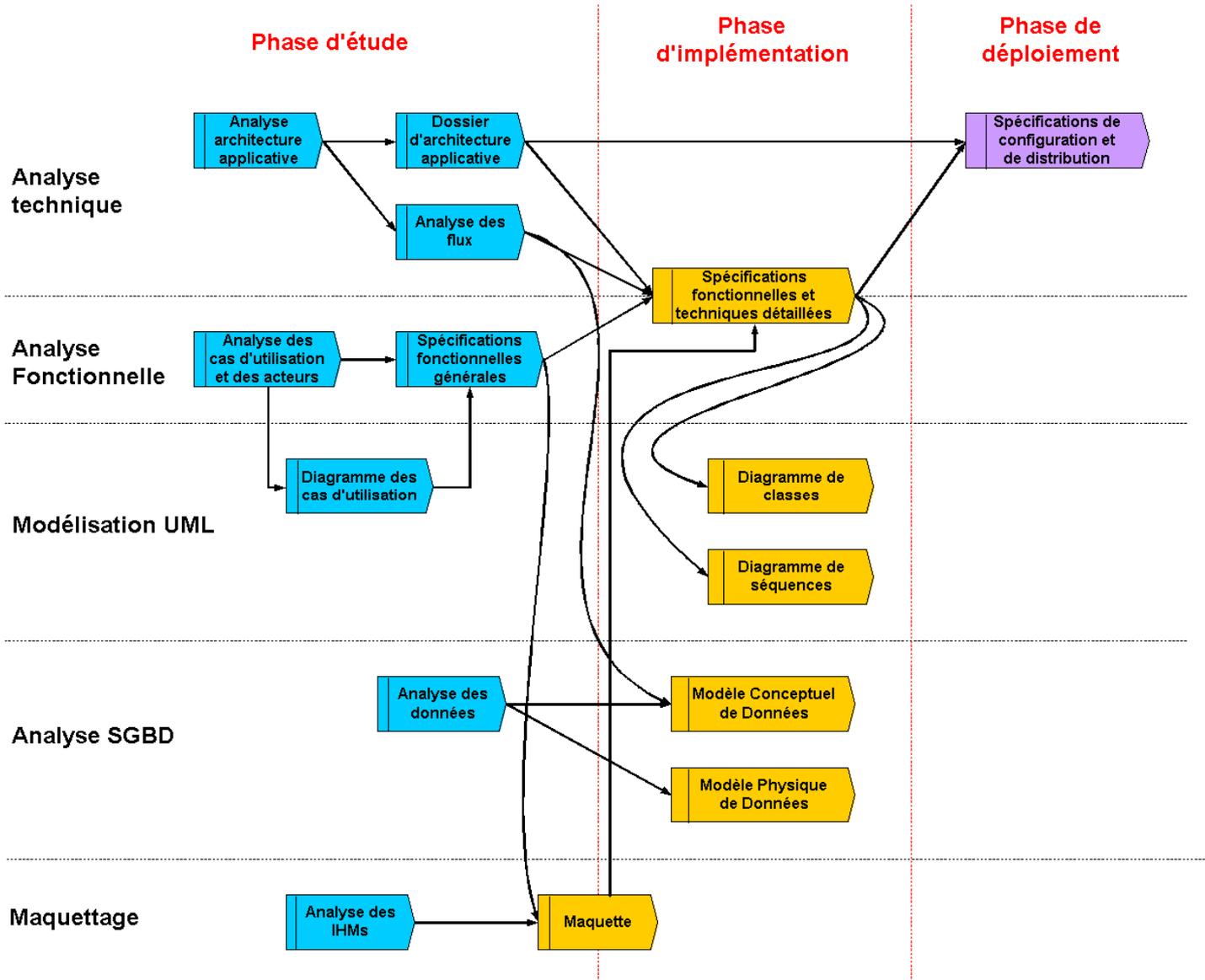


Figure 2 : Méthodologie d'analyse, modélisation et maquettage

Comme indiqué sur cette figure, la mise en œuvre d'un applicatif avec la filière de développement Hornet nécessite des analyses, modélisations et maquettage à travers l'ensemble de son cycle de vie.

3.1 Analyse technique

Cette analyse doit permettre de recenser l'ensemble des composants applicatifs à développer et des flux à générer. Elle doit aboutir à la production des documents suivants :

- le dossier d'architecture applicative présentant l'ensemble des solutions techniques retenues, les flux à générer et l'emplacement des composants à développer. Une étude de sécurité et d'intégration dans le SI (rationalisation et urbanisation) doit accompagner ce dossier pour vérifier la prise en compte de l'ensemble des contraintes de sécurité et d'urbanisation.
- les spécifications techniques détaillées reposent sur la documentation technique automatisée à partir des commentaires incorporés dans le code source et de templates descriptifs. Elles sont instanciées au fur et à mesure de l'avancement de la phase d'implémentation.

- les spécifications de configuration et de distribution contenant les informations nécessaires à la configuration des composants logiciels en vue de leur déploiement ainsi que des serveurs.

3.2 Analyse fonctionnelle

Cette analyse doit permettre de recenser l'ensemble des acteurs, habilitations et cas d'utilisation à mettre en œuvre. Les documents issus de cette analyse sont les suivants :

- les spécifications fonctionnelles générales présentant le résultat de ce recensement et le besoin en maquettage ;
- les spécifications fonctionnelles détaillées décrivant chaque cas d'utilisation et les règles de gestion associées.

3.3 Analyse SGBD

Elle consiste à analyser les besoins en flux, les données existantes et à créer pour ainsi modéliser la base de données nécessaire à l'applicatif attendu.

3.4 Modélisation

Toute analyse fonctionnelle et technique doit s'appuyer sur les modélisations suivantes :

- Modélisation UML fonctionnelle à partir d'un diagramme de l'ensemble des cas d'utilisation ;
- Modélisation UML technique à partir des diagrammes de l'ensemble des classes développées sur la « couche métier », de séquences pour les fonctionnalités métiers les plus complexes et d'états-transitions ou de collaboration pour décrire les mécanismes de workflow humain ou technique ;
- Modélisation Merise pour la « couche données » avec la réalisation du Modèle Conceptuel de Données (MCD) et du Modèle Physique de Données (MPD).

Les modélisations techniques sont incorporées dans la documentation technique automatisée en les intégrant sous forme d'images dans les templates descriptifs correspondants.

3.5 Maquettage

Pour valider la bonne compréhension du besoin fonctionnel et l'ergonomie définie dans l'analyse des IHMs, une maquette basée sur les principes du client riche est mise en œuvre pour lever toute ambiguïté avant la phase d'implémentation. La maquette sert ainsi en quelque sorte de modélisation de la « couche IHMs » pour les spécifications fonctionnelles et techniques détaillées. Des copies d'écran sont ainsi incorporées dans la documentation technique automatisée pour illustrer les exemples d'implémentation de la couche IHMs.

4 Méthodologie de test

4.1 Opérations de test

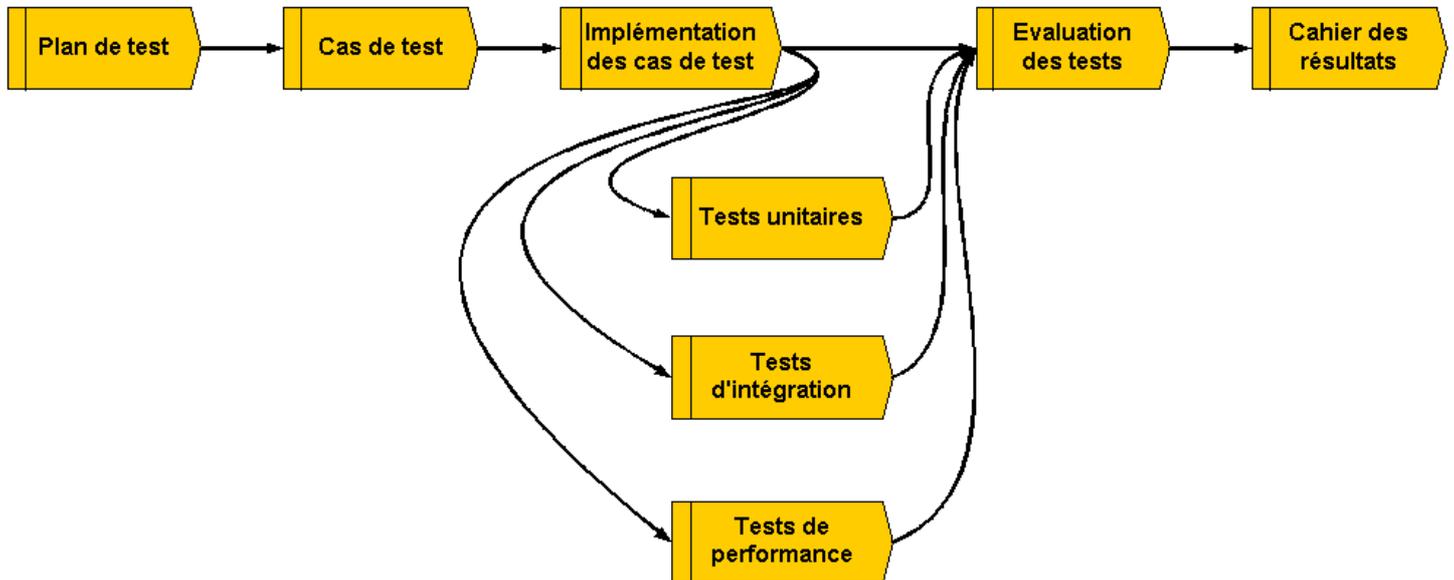


Figure 3 : Méthodologie de test

La méthodologie de test est appliquée lors de chaque évaluation d'itération indiquée dans la méthodologie projet. Lors de la phase d'étude, la rédaction du plan de projet s'accompagne de la rédaction du plan de test ayant pour objectif de recenser l'ensemble des cas de test en fonction des règles de gestion de la manière suivante :

- Cas passant : la règle de gestion associée à la fonctionnalité métier permet de continuer le processus métier.
- Cas d'avertissement : la règle de gestion associée à la fonctionnalité métier permet de continuer le processus métier mais un message d'avertissement est affiché à l'utilisateur.
- Cas bloquant : la règle de gestion associée à la fonctionnalité métier ne permet pas de continuer le processus métier et un message d'erreur est affiché à l'utilisateur.

Le résultat de l'évaluation des tests doit faire l'objet de la rédaction d'un cahier de résultats consignait la liste des tests et le résultat de leur exécution.

L'implémentation et l'évaluation des cas de test se déclinent différemment suivant le contexte de développement ou la phase projet.

4.1.1 Tests durant la phase d'implémentation

		Tests unitaires	Tests d'intégration	Tests de performance
Itération SGBD	Implémentation	Scripts d'alimentation SGBD développement pour jouer et rejouer tests unitaires	Scripts d'alimentation SGBD intégration pour jouer et rejouer tests d'intégration	Plan d'exécution SQL Positionnement des index
	Evaluation	Bonne exécution des requêtes SQL créés ou modifiées pour la nouvelle itération	Tests de non régression sur les tests unitaires des autres itérations	Détection des « Full Scan » Temps d'exécution des requêtes SQL
Itération Développement	Implémentation	« Value Objects » bouchonnés	« Value Objects » bouchonnés	Métriques temporels à ajouter dans les logs
				Positionnement des développements vis à vis de la persistance de données et cache navigateur
	Evaluation	Bonne exécution des développements Java/J2EE/HTML/XML pour la nouvelle itération	Tests de non régression sur les tests unitaires des autres itérations	Temps d'exécution des traitements pour chaque flux métier ou d'édition
		Bonne exécution des développements client riche pour la nouvelle itération	Tests de non régression sur les tests unitaires des autres itérations	Nombre et poids des flux échangés pour gérer un métier (en-tête HTTP et logs serveur Web)

Tableau 1 : Méthodologie de test déclinée pour la phase d'implémentation

4.1.2 Tests durant la phase de recette

		Tests unitaires	Tests d'intégration	Tests de performance
Recette technique	Implémentation	NA	Tests d'intégration définis dans l'analyse pour la phase de recette	Scénarios de test pour l'outil de stress et effectuer les tests de montée en charge
	Evaluation	NA	Tests d'intégration définis dans l'analyse pour la phase de recette	Temps de réponse résultant dans l'environnement de pré-production
Recette fonctionnelle	Implémentation	NA	NA	NA
	Evaluation	NA	NA	NA

Tableau 2 : Méthodologie de test déclinée pour la phase de recette

La recette fonctionnelle ne comporte pas d'implémentation ou d'évaluation spécifique des cas de test car toute version de l'applicatif doit avoir été préalablement entièrement testée avant de passer dans l'environnement de recette.

4.2 Opérations de vérification dans le cadre d'un tiers développement

D'une manière générale, les opérations de vérification effectuées dans le cadre d'un tiers développement portent non seulement sur le fonctionnement et les caractéristiques des produits (configuration) et des composants applicatifs (livrables), mais également sur la documentation.

Les opérations de vérifications portent sur la conformité de l'application vis à vis des attentes fonctionnelles, de la documentation et du respect des contraintes décrites dans le présent document.

4.2.1 Vérifications liées à la recette fonctionnelle

Elles nécessitent préalablement que l'ensemble des tests décrits dans la méthodologie de tests aient été réalisés et contrôlés par le tiers développeur avec consignation dans un cahier de résultats.

Dans le cas les premiers tests réalisés lors de la recette fonctionnelle montrent un taux d'anomalies, mineures ou pas, important, alors les défauts constatés seront considérés comme une anomalie majeure, bloquant le processus de validation.

Les tests et résultats des tests liés à la recette fonctionnelle sont consignés dans un cahier de résultats, comportant la liste des tests et le résultat de leur exécution.

Pour chaque test non satisfaisant, une ou plusieurs fiches d'anomalie sont établies. Dans chaque fiche, l'anomalie constatée est décrite et sa gravité est précisée. La référence du test est indiquée dans la fiche.

Dans la phase de validation, on distingue deux niveaux d'anomalie, qui sont considérés par rapport à la possibilité ou non de la mise en service :

- les anomalies majeures dont les deux types de défaut sont les suivants :
 - les défauts empêchant la poursuite correcte de la validation. Ce peut être une anomalie rendant impossible le fonctionnement de l'application (blocage ou arrêt), une fonctionnalité absente ou une fonction importante inexploitable,
 - les défauts n'empêchant pas de poursuivre la validation mais empêchant la mise en service. Ce sont les anomalies de caractère grave affectant de façon significative le fonctionnement de l'applicatif vis à vis du système ou de sa sécurité. Par exemple lorsque l'anomalie provient d'un composant applicatif réalisé par le tiers développeur :
 - perte d'information, en fonctionnement normal,
 - non concordance entre question et réponse lors de l'interrogation des flux,
 - impossibilité de traiter toutes les demandes d'interrogation...
- Les anomalies mineures n'empêchant pas la mise en service de l'applicatif dont les types de défaut sont les suivants :
 - des défauts de faible gravité,
 - des divergences de faible importance par rapport aux spécifications approuvées,
 - des défauts de cohérence ou de présentation n'affectant pas le résultat fourni (par exemple des fautes d'orthographe ou des problèmes de cadrage des données, un non-respect d'une charte graphique, ...)

L'applicatif est déclaré valide s'il ne subsiste aucune anomalie majeure à l'issue de la vérification.

Si des anomalies majeures sont constatées pendant le passage des tests, elles doivent toutes être corrigées pour que l'applicatif soit déclaré valide.

Les éventuelles anomalies mineures constatées doivent être corrigées dans une itération suivante de l'applicatif.

4.2.2 Vérifications liées à la filière Hornet

Un contrôle qualité du code source est effectué à chaque itération de la phase de recette.

Cette vérification porte sur les critères suivants :

- Documentation du code source.

- Respect de la charte et des normes de développement.
- Logique de programmation vis à vis des principes techniques fondateurs de l'architecture Hornet (voir charte d'architecture Hornet).
- Bonne utilisation des frameworks propres à la filière.
- Mutualisation des développements soit par évolution des frameworks propres à la filière soit par le développement de frameworks propres au projet.
- Compatibilité des navigateurs cibles.

4.2.3 Vérifications documentaires

Elles ont lieu pour les documents cités dans la partie méthodologie d'analyse, de modélisation et de maquettage.

L'absence d'un ou plusieurs documents empêche de déclarer l'applicatif valide.

5 Equipe projet

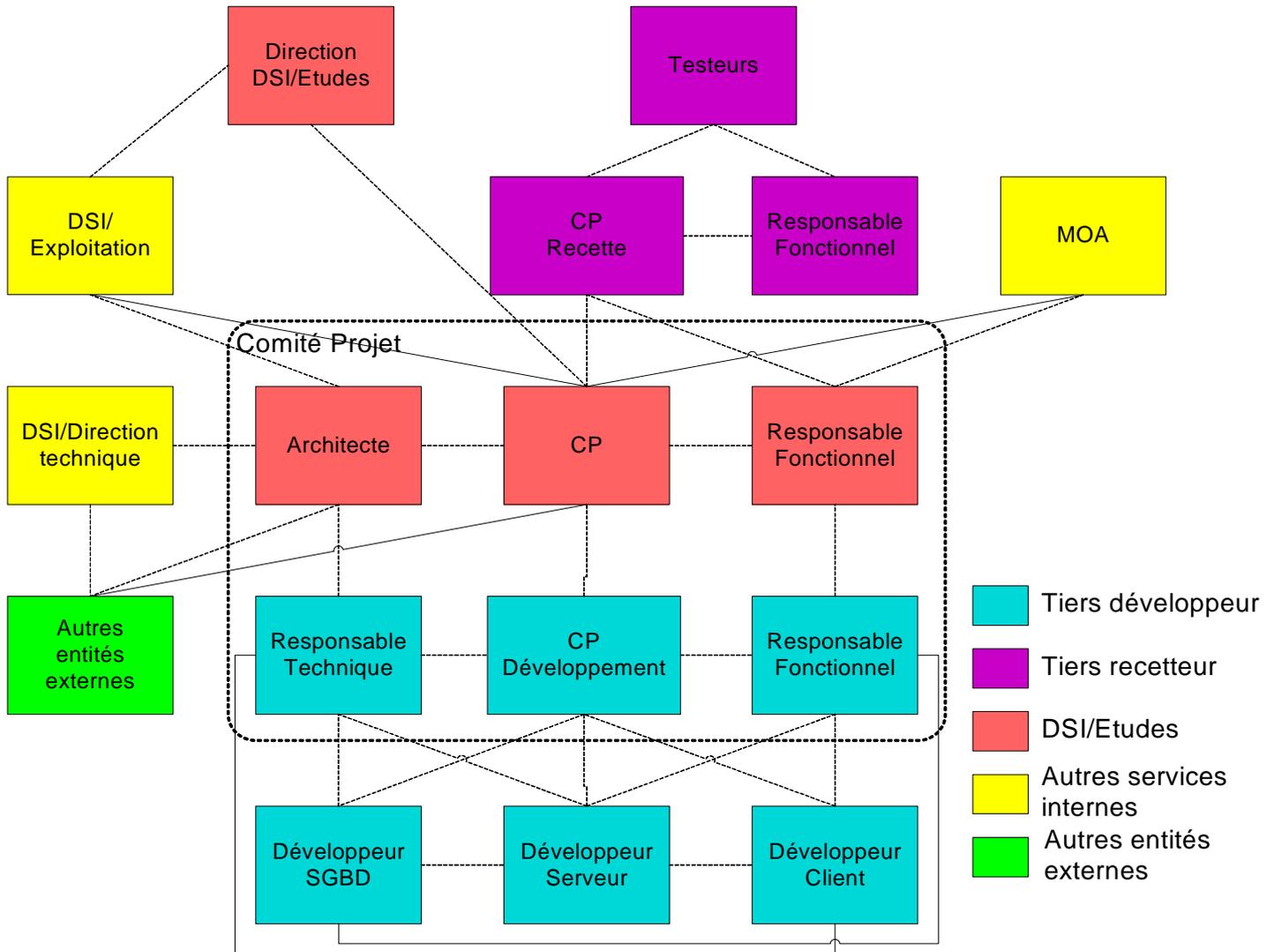


Figure 4 : Equipe projet – compétences nécessaires

Les compétences de la figure ci-dessus peuvent être réparties sur une ou plusieurs personnes physiques.

5.1 Compétences DSI

La DSI/Etudes met à disposition pour la mise en œuvre d'un projet Hornet les compétences suivantes :

- Architecte** : Compétence interne en charge de vérifier la cohérence des choix techniques vis à vis de l'architecture Hornet. Toute nouvelle solution technique est validée par son intermédiaire à des fins de mutualisation du besoin sur l'ensemble des projets développés par l'intermédiaire de la filière de développement. Elle assure aussi l'interface avec la DSI/Direction technique et/ou les éventuelles autres entités externes concernées en cas du choix de solution technique impactant des orientations stratégiques. Elle est aussi en charge de l'audit de code des développements pour vérifier leur conformité vis à vis des chartes, des normes et des principes fondateurs de l'architecture Hornet. Elle intervient dans la recette technique et les tests de performance de l'applicatif mis en œuvre.

- **Chef de projet** : Compétence interne en charge du planning projet, de la remonter des incidents et des risques à la maîtrise d'ouvrage et la direction de la DSI/Etudes et de la mise en place des procédures associées au cycle de vie du projet vis à vis de la DSI/Exploitation.
- **Responsable fonctionnel** : Compétence interne en charge de l'ensemble des problématiques fonctionnelles associées au projet. Elle fait l'interface avec le responsable fonctionnel du tiers développeur et la maîtrise d'ouvrage fonctionnelle exprimant un besoin. Elle accompagne particulièrement les phases d'analyse et de recette fonctionnelle.

5.2 Compétences tiers développeur

Le tiers développeur s'engage à fournir au minimum pour la mise en œuvre d'un projet Hornet les compétences suivantes :

- **Responsable technique** : compétence externe en charge de l'ensemble des problématiques techniques associées au projet. Elle joue le rôle d'interface entre le pôle d'architecture logicielle de DSI/Etudes et l'équipe de développement. Cette compétence permet de remonter tout nouveau besoin technique auquel la filière Hornet ne répond pas déjà et valider ainsi la solution proposée. Elle garantit la bonne application des chartes, normes et principes fondateurs de l'architecture Hornet et est donc l'interlocuteur privilégié du pôle lors de l'audit de code. Cette compétence garantit la cohérence des sources déposés dans l'outil de gestion de configuration (gestion des conflits, des livraisons, des branches...) et effectue la gestion des livraisons dans les différents environnements.
- **Chef de projet** : compétence externe en charge du planning projet, de la remonter des incidents et des risques, de l'affectation des ressources de développement sur les travaux à réaliser et de l'activité induite. Elle doit ainsi piloter le projet par les délais et par les risques. Elle est aussi en charge de rédiger l'ensemble des comptes rendus de réunion associés aux différents comités.
- **Responsable fonctionnel** : compétence externe en charge de l'ensemble des problématiques fonctionnelles associées au projet. Elle joue le rôle d'interface entre le responsable fonctionnel de DSI/Etudes et l'équipe de développement.
- **Expert SGBD** : compétence externe en charge de définir le modèle conceptuel de données et le modèle physique de données de la base de données destinée au projet. Elle met à disposition les scripts de création et/ou modification de base de données et d'alimentation pour jouer et rejouer les cas de test. Elle joue donc le rôle d'interface avec l'administrateur SGBD.
- **Développeur Serveur** : compétence externe en charge du développement des fonctionnalités attendues pour la partie serveur J2EE. Les compétences de programmation sous-jacentes sont liées aux langages XML, XSL, XSL-FO, JDOM et Java/J2EE.
- **Développeur Client** : compétence externe en charge du développement des IHMS basées sur les principes du client riche. Les compétences de programmation sous-jacentes sont liées aux langages XML, XHTML, CSS, DOM W3C, JavaScript et AJAX. Elle intervient particulièrement lors de la phase de maquettage et d'intégration.

5.3 Comité de projet

Un comité de projet rassemblant les compétences d'architecte, chef de projet et responsable fonctionnel de DSI/Etudes et les compétences de responsable technique, chef de projet et responsable fonctionnel du tiers développeur est organisé hebdomadairement.

5.4 Comité de pilotage

Un comité de pilotage rassemblant les mêmes compétences que celles du comité de projet, mais élargies aux autres divisions et services (DSI/Exploitation, MOA, Direction de DSI/Etudes...) est organisé mensuellement.