Processus de développement Objet : Best Practices

Si Les nouvelles technologies font briller les yeux des developpeurs, le chef de projet se trouve quant a lui en proie a de nombreuses interrogations :

Mon processus de developpement est-il depasse ? Comment gerer la complexite technologique ? Comment organiser mes equipes ?

En nous appuyant sur nos retours d'experience, nous presenterons comment adapter son processus de developpement de façon a concilier qualite des realisations et complexite technologique.

Cet article s'adresse a des chefs de projets toutes experiences confondues.

Auteur

Stève SFARTZ est directeur du développement de la société IMPROVE. De formation Télécoms, ingénieur ENST de Bretagne, Stève réalise depuis 1996 des missions de développement et de conseil en architecture Objet, Java, Web, XML… Par ailleurs, Stève anime régulièrement des conférences et formations sur ces thèmes.

Mon processus de développement est-il dépassé ?

La construction du macro-planning est une étape clef du projet : le chef de projet définit le processus de développement (illustration figure 1), composé d'une succession de phases, et propose pour chacune d'entre elles, une durée, des moyens à mettre en œuvre, des documents à produire et des profils d'intervenants. Le succès du projet dépend dès lors, de l'adéquation du projet au processus de développement.

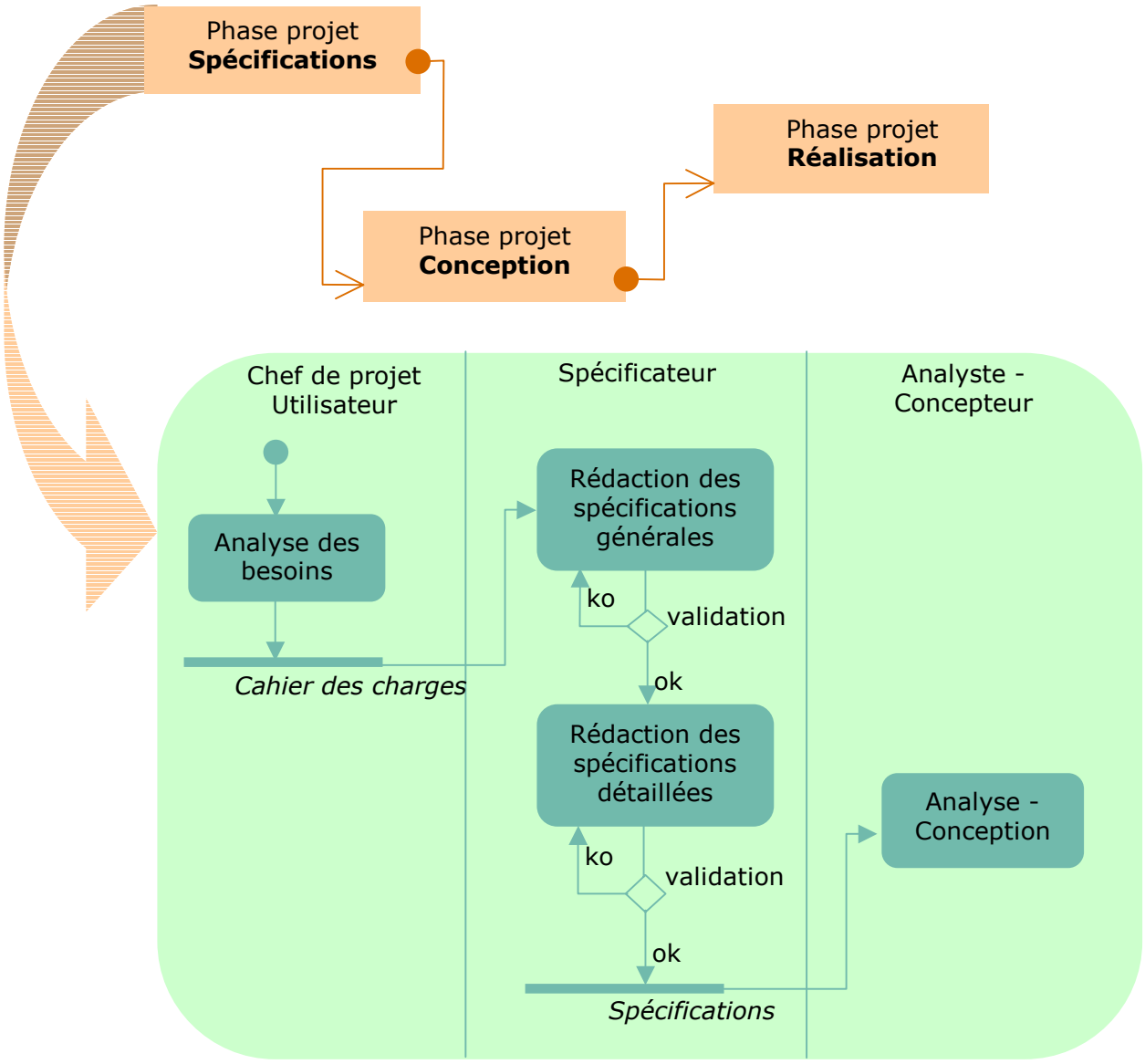

Figure 1 : Exemple de processus de développement, et détail d'une phase

Le chef de projet est libre de définir son propre processus projet, ou bien d'en choisir un parmi ceux proposés par les éditeurs. La figure 2 présente une synthèse des processus les plus en vogue dans la communauté Objet et Nouvelles Technologies.

	Description	Points forts	Points faibles
Cascade	• Propose de dérouler les phases projet de manière séquentielle • **Cité pour des raisons historiques**	• Distingue clairement les phases projet	• **Non itératif** • Ne propose pas de modèles de documents
RUP Rational Unified Process	• Promu par Rational. • **Le RUP est à la fois une méthodologie et un outil prêt à l'emploi** (documents types partagés dans un référentiel Web) • Cible des projets de plus de 10 personnes	• Itératif • **Spécifie le dialogue entre les différents intervenants du projet : les livrables, les plannings, les prototypes…** • Propose des modèles de documents, et des canevas pour des projets types	• **Coûteux** à personnaliser : batterie de consultants • Très axé processus, au détriment du développement : peu de place pour le code et la technologie
XP eXtreme Programming	• **Ensemble de « Bests Practices » de développement** (travail en équipes, transfert de compétences…) • Cible des projets de moins de 10 personnes	• Itératif • **Simple à mettre en œuvre** • Fait une large place aux aspects techniques : prototypes, règles de développement, tests… • **Innovant**: programmation en duo, kick-off matinal debout …	• Ne couvre pas les phases en amont et en aval au développement : capture des besoins, support, maintenance, tests d'intégration… • **Elude la phase d'analyse**, si bien qu'on peut dépenser son énergie à faire et défaire • **Assez flou dans sa mise en œuvre**: quels intervenants, quels livrables ?
2TUP Two Track Unified Process	• **S'articule autour de l'architecture** • **Propose un cycle de développement en Y** • Détaillé dans « UML en action » (voir références) • Cible des projets de toutes tailles	• Itératif • **Fait une large place à la technologie et à la gestion du risque** • Définit les profils des intervenants, les livrables, les plannings, les prototypes	• **Plutôt superficiel sur les phases situées en amont et en aval du développement** : capture des besoins, support, maintenance, gestion du changement… • Ne propose pas de documents types

Figure 2 : Synthèse des méthodologies utilisées dans le cadre de développement Objet et Nouvelles Technologies

Force est de constater que toutes ces méthodologies proposent de travailler de façon itérative, que ce soit au niveau des plannings, des spécifications, ou des développements…

Si l'itératif s'est imposé, c'est parce qu'il réduit la complexité de réalisation des phases, en travaillant par approches successives et incrémentales.
Il est alors possible de présenter rapidement aux utilisateurs des éléments de validation. De plus, l'itératif permet une gestion efficace des risques, en abordant dès les premières itérations, les points difficiles.
Par exemple, les premières itérations de la phase technique aborderont les aspects sécurité et transaction.

Best Practice

Définir pour chaque phase, le nombre d'itérations et leur contenu précis.

Exemple : Dans le cadre d'un projet Web qui comporterait un module de personnalisation, la réalisation de ce module pourrait être composé des itérations suivantes :

- Itération n°1 (longue / technique): création du moteur de personnalisation des interfaces Web (spécifications des besoins de personnalisation, élaboration de l'ergonomie, modélisation du moteur en adéquation avec l'architecture applicative, développement des composants logiciels, développement d'une maquette à des fins de validation)
- Itération n°2 (courte / technique): revue de l'ergonomie suite à la démonstration et intégration de nouveaux éléments à la maquette (mise à jour de l'ergonomie, mise à jour de la modélisation du moteur, mise à jour des composants logiciels, réalisation de la maquette à des fins de démonstration)
- Itération n°3 (courte / fonctionnelle): intégration d'éléments fonctionnels (définition d'une charte graphique et adaptation de la maquette à des fins de validation).

Au-delà de l'itératif, on notera que les méthodologies présentées en figure 2 mettent l'accent sur des phases projets différentes.
Le RUP couvre l'ensemble du processus en spécifiant les inter-actions entre chacune des phases, XP se concentre sur la phase de développement, tandis que 2TUP fait une large place à l'analyse et à l'architecture.
Aussi, ces processus peuvent se compléter plutôt qu'entrer en concurrence, comme l'illustre la figure 3.

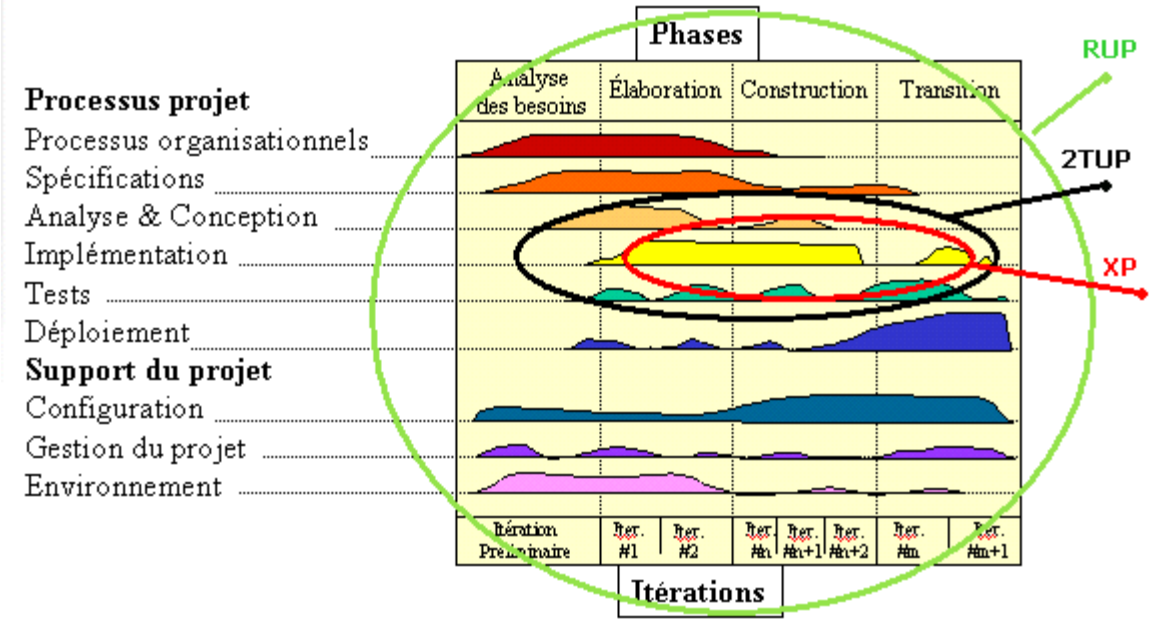

Figure 3 : Projection de XP et de 2TUP sur la matrice du RUP

Pour répondre aux exigences des nouvelles technologies, ne pas hésiter à compléter sa méthodologie de projet en s'inspirant des aspects des processus RUP, XP et 2TUP. Dans le cadre de cet article, nous vous proposons de nous attacher aux aspects :

- maîtrise de la technologie,
- et organisation des équipes.

Best Practice

Ne passez pas des mois à définir votre méthodologie de développement.
Prenez connaissance des processus les plus utilisés pour en intégrer un ou plusieurs à votre méthodologie de projet.

Exemple : Dans le cas d'un projet Nouvelles Technologies, on pourra intégrer :

- Les valeurs d'XP et quelques règles (communication, simplicité, feedback et énergie)
- les documents types du RUP et leur enchaînement
- la branche technique du 2TUP

Comment gérer la complexité technologique ?

La meilleure façon d'aborder un problème, c'est de s'y attaquer de front. C'est ce que propose le 2TUP en faisant une place à part entière à la technologie dans le processus de développement !

Le 2TUP propose un cycle de développement en Y, qui dissocie les aspects techniques des aspects fonctionnels.
Illustré en figure 4, le processus en Y s'articule autour de 3 phases,:

- une branche technique
- une branche fonctionnelle
- et une phase de réalisation

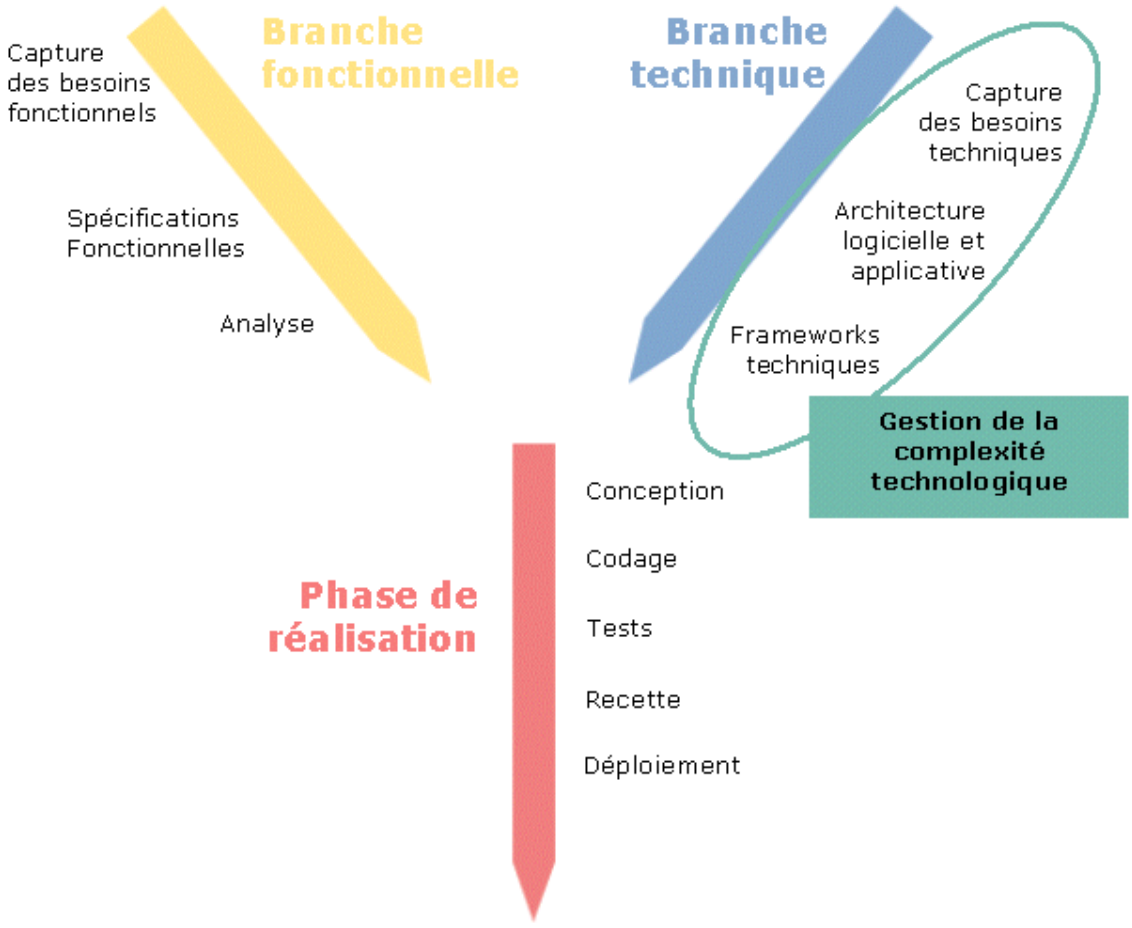

Figure 4 : Cycle de développement en Y

La branche technique est née de la constatation que les plate-formes de développement (C++ / Corba, J2EE, .Net …) ne proposent pas de modèle d'implémentation pour résoudre les problèmes de sécurité, de montée en charge, de gestion de la touche Back du navigateur…

L'objectif de la branche technique est donc de :

1. Rassembler les besoins techniques (sécurité, montée en charge, intégration à l'existant, …) dans un dossier,
2. Elaborer une architecture logicielle et applicative qui réponde aux contraintes présentées dans le dossier technique,
3. Identifier les besoins en frameworks techniques afin de palier aux manques de la technologie (gestion de la touche Back des navigateurs, formulaires de saisie inter-actifs, personnalisation de l'interface graphique, moteur de persistance Objet / Relationnel avec expressions SQL / Objet, …),
4. Proposer des règles de développement afin d'industrialiser l'implémentation (gestion de exceptions, règles de nommage, règles de codage, ...)

Cependant, la méthodologie ne peut à elle seule, réduire le risque technologique. Pour vous assurer que vos équipes ont une bonne maîtrise de la technologie, vous pourrez demander la réalisation de prototypes techniques ou fonctionnels.
Le prototype technique permet de s'assurer du respect des contraintes du dossier technique, en laissant de côté les aspects fonctionnels.
Quant au prototype fonctionnel, il a pour objectif de juger de la capacité des développeurs à intégrer l'architecture applicative, à monter en compétences sur les frameworks techniques, à comprendre la conception et à suivre les règles de développement. Nous vous recommandons par ailleurs de déployer le prototype fonctionnel, afin de vous heurter au plus tôt aux contraintes de production.
A travers le prototype fonctionnel, c'est donc l'ensemble du cycle de développement qui est validé !

Enfin, avant de démarrer l'itération suivante, on s'attardera sur le prototype fonctionnel pour :

- approfondir certaines règles de développement comme le traitement des erreurs ou la gestion des niveaux de logs,
- juger du niveau de maintenabilité de l'application
- et au vue de la productivité des ressources, ré-estimer le planning du projet, ou modifier l'organisation des équipes de développement.

Best Practice

Le processus de développement doit faire une place importante à la maîtrise des Nouvelles Technologies.
Sous-estimer cet aspect, c'est résoudre les problèmes techniques au fil des développements, avec des risques importants de remise en cause des réalisations (décalage de planning, démotivation des équipes, …).

Exemple : Faire reposer la phase de réalisation sur un processus en Y, en s'attachant à
intégrer les risques dès les premières itérations
isoler les solutions techniques dans des frameworks
valider l'avancement par des prototypes fonctionnels

Nous avons insisté sur la place à accorder à la technologie. Cependant, cela se traduit de façon différente selon le type de réalisation. Par exemple, dans le cas d'une application Intranet de gestion de Forum, on privilégiera les aspects montée en charge sur les aspects sécurité. De plus, on pourra remettre en question la nécessité

de développer des frameworks techniques ou de formaliser les règles de développement ?!
La figure 5 définit les objectifs de la branche technique du cycle de développement en Y en distinguant trois types d'applications : « One-Shot », « Départementale » et « Produit ».

	Exigences	Composantes de la branche technique		
		Equipe	Architecture	Règles de développement
Application « One-Shot »	• Maintenabilité • Robustesse • **Pas de réutilisation**	• **Développeurs seniors** sur les technologies retenues	• Définie par les développeurs	• Règles de codage et de nommage
Applications d'Entreprise	• Maintenabilité • Robustesse • **Réutilisation**	• Distinction des rôles : Chef de projet technique, Architecte, Développeur	• Définie par l'architecte • **Bibliothèques de composants fonctionnels**	• **Réalisation de prototypes** • Règles de codage, de nommage • **Stratégie de gestion des exceptions**
Produit	• Maintenabilité • Robustesse • **Configurabilité**	• Distinction des rôles : Chef de projet technique, Architecte, **Développeur technique,** **Développeur fonctionnel**	• Définie par l'architecte • **Frameworks techniques** • **Frameworks fonctionnels**	• **Réalisation de prototypes** • Règles de codage, de nommage • **Stratégie de gestion des exceptions et des logs**

Figure 5 : Objectifs et organisation de la phase technique selon le type d'application

Selon le type d'applications réalisées, on observe des exigences différentes en terme d'équipe, d'architecture et de règles de développement. C'est pourquoi, ce n'est qu'après la réalisation de ces exigences, que le chef de projet a une vision précise de la suite de son projet : c'est alors le moment de réactualiser le macro-planning à partir des éléments révélés par la phase technique, notamment la productivité des équipes.

Best Practice

En règle générale, on peut être confiant quant à la réussite du projet lorsque le prototype fonctionnel est validé et testé.
En effet, à cette étape, les équipes de développement sont productives (formées sur les frameworks techniques, bibliothèques de composants et règles de développement…).

Comment organiser mes équipes ?

Dans la plupart des projets Nouvelles Technologies, la gestion des ressources se révèle rapidement un véritable casse-tête :

- Les experts ne sont jamais disponibles,
- Les équipes de développement sont sur-dimensionnées sur certaines phases et sous-dimensionnées sur d'autres,
- Les transferts de compétences n'ont pas lieu si bien que les ressources critiques restent les mêmes tout au long du projet,

Nous présentons ci-dessous quelques règles pour éviter de tomber dans ces pièges :

Best Practice

Se concentrer sur la phase de développement en cours !

Ne pas essayer de nourrir ses équipes à tout prix : il est normal que certaines ressources ne soient pas affectées sur certaines phases du projet. Mieux vaut parfois les laisser vacantes, que de leur demander de lancer leur développement en dépit du manque d'informations, ou pire encore, les affecter à des tâches qui ne sont pas de leur ressort (car c'est autant d'énergie en gestion de ressources, transfert de compétences, qui ne vont pas directement au bénéfice du projet).

Exemple : Les spécifications ont pris du retard. Les experts techniques sont en train de travailler sur les problèmes de sécurité des sessions utilisateurs sur HTTP, et l'interfaçage des transactions CICS. Plusieurs développeurs n'ont pas de tâches affectées :

Ne pas leur demander de commencer à lire les spécifications et démarrer le développement !
Pourquoi ne pas laisser ces développeurs vacants (formation, revue de code existant) ?
Etudier s'il est possible de les affecter aux jeux de tests ou à la récupération de l'existant ?

Best Practice

Rechercher la simplicité !

Ne pas regretter de ne pas avoir que des experts techniques. En effet, les développeurs seniors sont une menace pour le projet. D'une part, ils risquent de produire du code non maintenable, car trop élaboré, trop Objet ! D'autre part, ils deviennent inéluctablement des ressources critiques.

1. identifier les besoins en expertise technique,
2. faire réaliser à des développeurs seniors des bibliothèques de composants ou des modèles de développement qui puissent être réutilisés par les équipes moins expérimentées.

Best Practice

Réactualiser son planning de façon itérative !

S'il faut prendre le temps de bien définir son processus de développement en début de projet, et proposer un macro-planning réaliste, il faut rester conscient que les charges estimées évoluent régulièrement jusqu'à la validation du prototype fonctionnel, moment à partir duquel on a une bonne idée de la productivité des équipes.

Aussi, plutôt que de mettre constamment à jour ses plannings, le revoir plutôt à la fin de certaines itérations.

Best Practice

Spécialiser ses équipes de développement !

Le temps d'apprentissage des nouvelles technologies peut se révéler interminable (Java + XML + HTML + JavaScript + JDBC + SQL + EJB + Objet Distribué + Suivi de session) sans parler de l'apprentissage du fonctionnel !
Aussi, nous conseillons de :

- ne pas chercher le mouton à 5 pattes !
- spécialiser ses équipes (Interfaces graphiques ou Code Métier, ce qui correspond assez bien aux aspirations de chacun) et libre ensuite à chacun de changer de spécialité,
- profiter des phases de spécifications et d'analyse pour former ses équipes de développement,
- placer des développeurs senior et junior en paire pendant quelques heures ou jours d'apprentissage se révèle souvent plus efficace que d'effectuer des relectures de code !

Best Practice

Démarrer tôt la phase d'architecture !

Exemple : Aussi longtemps que le travail d'architecture n'est pas terminé, les développements sont bloqués, en attente des éléments techniques (choix de produits tiers, frameworks, règles de développement).

- Pourquoi ne pas lancer en parallèle les phases de spécifications et d'architecture ?!

Enfin, la meilleure garantie de succès, c'est de s'assurer que chaque intervenant remplit au mieux le ou les rôles qui lui sont attribués, tel que l'illustre la figure 6.

	Fonction	Best Practices
Chef de projet	• Elabore le processus de développement et le macro-planning • Rend compte de l'avancement du projet	• Etre ferme face aux demandes des utilisateurs • **Trancher dans le sens de la simplification**
Chef de projet utilisateur	• Recueille le besoin des utilisateurs • Participe à l'élaboration des spécifications fonctionnelles	• **Etre sensible aux contraintes de réalisation**
Spécificateur	• Rédige les spécifications fonctionnelles générales et détaillées • Identifie les flux de données	• **Se plonger dans le modèle physique pour cerner le niveau d'adéquation avec l'existant** • Creuser les points fonctionnels plutôt qu' attendre les retours de l'analyse, et ce afin de minimiser le nombre d'itérations
Architecte	• Recense les besoins techniques dans le dossier technique • Elabore les architectures logicielle et applicative • Argumente ses choix technologiques • Identifie les besoins en produits tiers et frameworks techniques	• Expert Objet et Nouvelles Technologies • Assisté par les consultants des technologies tierces sélectionnées • Faire valider l'architecture par un consultant externe • Prendre connaissance du fonctionnel et assister le concepteur sur les premières modélisations
Concepteur	• Réalise les modèles de conception selon l'architecture applicative • Identifie les besoins d'évolutions de l'existant (modèle de données, interfaces, batchs…)	• Bonne connaissance de l'Objet • Bonne connaissance du fonctionnel et des données (SQL, Site central…) • A fait l'expérience d'un projet Objet et Nouvelles Technologies
Chef de projet technique	• Assure le suivi des plannings de développement • Responsable de la qualité des développements	• Expert Objet et Nouvelles Technologies • Valide les modèles de conception • Assistance technique aux équipes de développement • Relecture de code
Développeur technique	• Réalise les bibliothèques de composants et les frameworks techniques en collaboration avec l'architecte	• Expert Objet et Nouvelles Technologies • Forme les équipes fonctionnelles sur les réalisations techniques
Développeur fonctionnel	• Réalise les applications décrites dans les spécifications fonctionnelle, en tenant compte de l'architecture applicative, des frameworks techniques et des règles de développement	• Bonne connaissance du fonctionnel • Spécialisé sur un domaine : Interface Graphique ou Composants Métiers

Figure 6 : Rôles et Best Practices des intervenants du projet

Conclusion

Nous espérons vous avoir éclairé sur les processus de développement Objet et Nouvelles Technologies, et surtout sur la façon de maîtriser le risque, tout en travaillant de façon itérative.

A vous maintenant de doser ces exigences en fonction de votre type d'application (One-Shot, entreprise ou produit), et du cadre de sa réalisation (Régie, Forfait, ou régie forfaitée).

Bonne aventure…

Références

From Waterfall to Iterative Lifecycle - A tough transition for project manager
http://www.rational.com/media/whitepapers/TP173A.pdf

Completing the Unified Process With Process Patterns by Scott W. Ambler
http://www.ambysoft.com/unifiedProcess.html

A Comparison of RUP and XP
http://www.rational.com/media/whitepapers/TP167.pdf

Présentation du processus 2TUP
UML en action , Pascal Roques et Franck Vallée, Edition Erolles

Présentation de l'eXtreme Programming
http://www.extremeprogramming.org

Le mythe du mois-homme, Essais sur le génie logiciel
Frederick P. Brooks, Jr, Thomson Publishing