Implémentation, tests et le déploiement

I. Implémentation (construction)

Maintenant que tous les plans du système sont complétés (modèle de classes, diagrammes de séquences ou autre diagramme qui représente la dynamique du système), l'étape suivante consiste à l'écriture des programmes

Les activités de cette étape :

- Prise de décision définitive relative au type d'outil de développement à adopter
- Prise de décision relative à l'organisation physique de la base de données et à l'accès aux enregistrements des tables
- Prise de décision relative à l'organisation physique des classes
- Prise de décision relative au nombre de programmes différents qui composeront le système
- Test de programmes, de modules, de système et d'acceptation
- Préparation de documents, tels que des manuels d'utilisation et d'opération, de même qu'une documentation au sujet du système.

La programmation :

La programmation est une activité complexe, exigeant une grande minutie et consommant une partie importante du budget temps d'un projet de développement de systèmes. C'est pour cette raison que tant d'efforts sont faits pour développer et mettre en marché des outils de programmation qui réduiront l'ampleur de cette tâche. Dans le cas d'un système de grande taille, où plusieurs personnes effectuent la programmation, une coordination efficace **s'impose**. Sans elle, le risque serait grand de se retrouver avec des modules fonctionnant bien les uns indépendamment des autres, mais totalement incompatibles.

Comment le programmeur doit-il être face à son programme?

- ✓ Le programmeur doit produire un code lisible par les autres membres de l'équipe
- ✓ Le programmeur ne doit prétendre être expert alors qu'il ne l'est pas
- ✓ Admettre ses erreurs
- ✓ Comprendre les messages erreurs plutôt que de les ignorer
- ✓ Planifier, comprendre son programme plutôt que de le vérifier pour voir s'il fonctionne

Comment gérer la complexité d'un programme ?

- ✓ Diviser pour régner
- ✓ Vérifier et tester régulièrement
- ✓ Garder les programmes courts

Comment le programmeur doit-il se maintenir à jour?

- ✓ Prendre conscience de sa méthode de travail et l'améliorer
- ✓ Tester de nouvelles techniques avec des cas simples
- ✓ Comprendre le problème plutôt que de s'entêter à le résoudre sans toute fois le comprendre
- ✓ Réfléchir avant d'agir
- ✓ Lire les manuels, les aides et des revues pour se maintenir à jour
- ✓ Se documenter sur des projets qui ont réussi

Étapes de la phase de l'implémentation

Construire chaque programme

- ❑ Prendre connaissance des documents d'analyse et de conception
- ❑ Préparer les tests
- ❑ Implémenter une classe
- ❑ Implémenter un sous-système
- ❑ Effectuer les tests unitaires
- ❑ Intégrer le système.
- ❑ Finaliser la documentation du programme

Construire les utilitaires

- ❑ Accès aux fichiers et codes erreurs
- ❑ Les procédures de contrôles des accès

Effectuer les tests

Produire le manuel utilisateur

- ❑ Prendre connaissance des devis
- ❑ Établir la politique à suivre (procédures d'installation)
- ❑ Définir la séquence des gestes à poser
- ❑ Rédiger les procédures

II. Les tests

1. définitions :

Les Bogues : Viennent du mot anglais Bug,qui signifie insecte parasite. L'utilisation de ce mot pour désigner une anomalie vient de l'époque des premiers ordinateurs (à tubes à vides). Des cafards attirés par la chaleur dégagée par ces tubes venaient se réfugier dans ces machines et y mourraient. Leurs cadavres provoquaient des courts-circuits d'où des pannes qui ont été appelaient des Bug.

Une erreur : peut être commise par un programmeur

Un défaut ou bogue : est le résultat de l'erreur

Une panne : est la conséquence du défaut

Une panne, une **défaillance** ou une **anomalie** est un comportement anormal d'un programme.
Un logiciel est dit **correct** s'il ne contient aucun défaut
Un logiciel est dit **fiable** s'il s'exécute sans défaillance

2. Hiérarchie des tests :

- Tests unitaires ou programmes: (morceaux de codes, procédures) tests effectués par ceux qui développent. (Exemple : cas de valeurs limites, cas d'invalidité, cas de nullité)
- Test des composants (modules), ici rentrent en jeu les testeurs
- Tests d'intégration, visent à vérifier l'intégrité de données sur une chaîne complète de programmes. Tests réalisés par le chef de projet ou l'analyste (celui qui a plus connaissance du système)
- Tests d'acceptation ou de validation : tests réalisés chez le client.

3. Classification des tests

Tests de validation : Ces tests ont pour but de vérifier la conformité du produit logiciel développé (et complètement intégré) par rapport à sa spécification, document d'étude décrivant ce que le logiciel doit faire en terme de d'exigences fonctionnelles(aspect comportemental) et d'exigences non fonctionnelles (Performance et capacité). Cette vérification est effectuée chez le fournisseur sous entière responsabilité. Elle permet d'autoriser la livraison

Tests de qualification : Ces tests ont pour but de vérifier la conformité du logiciel livré au client par rapport au cahier de charge fonctionnel (Document décrivant les besoins du client). Cette vérification l'acceptation du produit de la part du client et qui entraînera le transfert de propriétés et soldes de règlements

Tests de fonctionnels : Ce type de test consiste à vérifier l'aptitude d'un produit (ou une partie du produit) à fonctionner correctement en ne s'intéressant qu'aux services rendus par le produit, c'est à dire à son comportement externe

D'une manière générale un système doit être testé sur les points suivants :

1. L'exactitude des outputs produits par le système;
2. Le temps de réponse;
3. La capacité de traiter le volume requis de données;
4. La capacité de recouvrement après un arrêt de fonctionnement;
5. La sécurité des données;
6. La facilité d'utilisation de la documentation et des manuels.

Les données utilisées pour effectuer les tests doivent être le plus près possible des données

réelles, et doivent représenter la plus grande variété de situations possibles. En général, ce sont les utilisateurs qui ont la responsabilité de préparer les données de tests. En effet, ce sont eux qui connaissent le mieux les diverses valeurs que peuvent prendre ces données, de même que les situations d'exception.

III. Le déploiement ou la phase de transition

Le chef de projet a jugé que le système est suffisamment fiable pour le faire fonctionner dans l'environnement de l'utilisateur. Il est possible que des problèmes ou des anomalies non détectés lors des tests systèmes fassent apparition dans l'environnement de l'utilisateur.il se peut également que les utilisateurs découvrent qu'ils ont besoins d'autres fonctionnalités pour le système. Si ces fonctions sont importantes et cohérentes avec le système le chef de projet peut décider de les rajouter, ces rajouts doivent cependant rester mineurs
Les objectifs de la phase de transition sont les suivants :

- Répondre aux besoins tels qu'ils ont été précisé dans les phases précédentes
- Gérer toutes les questions conditionnant le fonctionnement dans l'environnement des utilisateurs, y compris la correction des anomalies rapportées par les utilisateurs de la version béta ou par le responsable des tests d'acceptation.

Les tests d'acceptation peuvent être fait par le client ou les sous-traiter à une firme spécialisée dans ce type de test.

La phase de transition en bref :

La phase de transition s'attache à installer le produit dans l'environnement opérationnel. La manière de procéder dépend d'un produit à un autre.
L'équipe de projet observe les retours en provenance des sites opérationnels pour :

- Vérifier si le système offre véritablement les services exigés par les utilisateurs de l'entreprise.
- Découvrir les risques non perçus
- Prendre acte des problèmes non résolus
- Lever les ambigüités et combler les manques dans la documentation destinée aux utilisateurs
- Concentrer les efforts sur les secteurs où les utilisateurs semblent avoir plus de difficultés..

Planification de la phase de transition

L'équipe de projet ne peut s'attendre à planifier à l'avance la phase de transition aussi bien que les phases précédentes (élaboration, analyse, conception..). Les membres de l'équipe savent qu'ils vont livrer la version bêta pour évaluation à des utilisateurs bien sélectionnés. Cette phase : la sélection des utilisateurs (testeurs), la préparation des instructions de tests

peut être planifiée et constitue le fondement des premières planifications de la phase de transition.

En planifiant la phase de transition, le chef de projet s'attend à ce que la version opérationnelle issue de la phase de construction nécessite quelques ajustements mineurs. Dans la phase de transition, et pour un développement itératif, ces retouches ne dépassent pas 5%. Les chefs de projet doivent quand même s'attendre à un certain volume de retouches, il ya toujours des erreurs et des omissions qui passent à travers les différentes phases. Voici quelques raisons qui peuvent conduire une équipe de transition à sous-estimer la charge des retouches à prévoir. :

- Une pression trop forte sur les délais---travail vite fait, mal fait----
- L'absence de tests systèmes adaptés
- L'incapacité à se concentrer sur le travail considérable restant à fournir dans la phase de transition
- La tendance à considérer les retouches comme négative «aveu d'une incompétence de l'équipe de projet»

Que faut-il livrer ?

1. Le plan de projet de transition
2. Le logiciel exécutable lui-même
3. Tous les documents (Analyse, conception, programmes..)
4. Une description de l'architecture entretenue et actualisée
5. Un premier manuel utilisateur suffisamment détaillé pour guider les testeurs de la version béta
6. L'étude de rentabilité (un retour sur les investissements)
7. Les documents légaux (les contrats, licences, garanties ..)

Définir les critères d'évaluation

1. Les utilisateurs de la version bêta, ont-ils abordé les fonctions clés?
2. Le produit a-t-il passé avec succès les tests d'acceptation menés par le client?
3. La documentation destinée aux utilisateurs est-elle d'un niveau de qualité acceptable?
4. Les supports de formation sont-ils prêts?
5. Les clients semblent-ils satisfaits du produit?

Activité de la phase de la phase de transition :

1. Préparation de la véritable version bêta
2. Installation ou préparation de l'installation de cette version avec la migration des données si nécessaire ou conversion :
    a. Remplacer l'ancien système par le nouveau (fonctionnement en

parallèle ou non)

b. Transférer les données de l'ancien vers le nouveau

3. Prise en compte des retours des sites de tests (défaillances ou problème de plus grande envergure). Dans le cas d'une défaillance, l'équipe de projet doit poser les questions telles que :
   a. Cette anomalie est-elle susceptible d'être liée à d'autres anomalies non encore détectées?
   b. Peut-elle être corrigée sans qu'elle n'affecte l'architecture ou la conception?
   c. Sa correction n'a-t-elle introduit aucune autre anomalie?
4. Finalisation de toutes les phases du projet
5. Détermination de l'achèvement du projet. Quand le projet se termine t-il?
   La phase de transition prend fin lorsque le client est satisfait

IV. Formation des utilisateurs

De la même façon qu'une voiture, même neuve, pourra avoir des ratés si son conducteur n'a jamais appris à conduire, un système, même parfait du point de vue technique, ne pourra pas fonctionner adéquatement s'il est utilisé par des gens non formés. Loin de vouloir rendre les utilisateurs responsables des ratés que peut avoir un système, cet énoncé met l'accent sur l'importance de la formation à offrir aux futurs utilisateurs. Nombreux sont les exemples de systèmes techniquement corrects, dans lesquels les organisations avaient investi d'importantes sommes et les analystes mis beaucoup de temps, mais dont on avait négligé le côté formation des utilisateurs. Plusieurs raisons peuvent être données pour expliquer cette négligence. Dans certains cas, ce sera à cause d'un retard du projet par rapport aux échéances prévues; afin de ne pas créer de délais supplémentaires, on rogne sur les dernières activités, celles qui semblent les moins critiques. Dans le même ordre d'idées, il peut aussi y avoir des dépassements importants de budgets; on essaie alors de couper les coûts là où faire se peut. Dans le cas de systèmes d'envergure réduite, où il y a, par exemple, un seul analyste et quelques utilisateurs seulement, il peut arriver que l'analyste soit affecté à de nouvelles tâches avant que la formation n'ait pu être faite. Peu importe la raison, si elle explique que l'on ait négligé la formation, elle n'excuse pas cette négligence.

Le succès de la formation donnée aux utilisateurs dépend, bien évidemment, de la compétence de la personne qui en est responsable, mais aussi et surtout de l'adéquation entre le contenu de la formation offerte et l'utilisation que l'individu formé fera du système. En effet, le responsable de la formation devra avoir une connaissance complète du système afin d'être en mesure d'en expliquer le fonctionnement et de répondre aux questions des

utilisateurs. Il arrive parfois que l'on charge un utilisateur de la formation. Cette façon de faire a certains avantages, dont le principal est de fournir aux gens qui reçoivent la formation, un interlocuteur ayant la même culture organisationnelle qu'eux. Il faudra, idéalement, que cet utilisateur ait participé de façon intense au projet de développement; autrement, sa connaissance du système risquerait d'être inadéquate. Le responsable de la formation devra aussi être bon communicateur et savoir s'adapter aux besoins des diverses populations à former.

V. Évaluation post-implantation (post-mortem)

Tout au long du projet de développement d'un système d'information, l'analyste ou l'équipe d'analyse établit des prévisions au sujet du temps requis ou des frais à engager pour développer le système, des coûts d'exploitation ou encore des bénéfices escomptés. Il procède aussi à l'identification des objectifs à atteindre.
Une fois le système en place, on devra déterminer si ces prévisions étaient justes et si le système atteint les objectifs fixés; on procédera donc à l'évaluation du projet et du système.

L'évaluation du projet consiste principalement à comparer le temps réel requis pour le développement du système au temps qui avait été prévu, et à déterminer si le projet a respecté le budget fixé. Pourquoi procéder à une telle évaluation? Deux raisons s'imposent; la première procède de la gestion de personnel, la seconde est liée à la nécessité pour l'organisation d'accumuler de l'expérience dans le domaine de la gestion de projets d'informatisation.

La qualité du travail accompli par la personne chargée de la gestion de projet, qu'elle soit analyste ou chef de projet, doit être évaluée par ses supérieurs. Le premier critère qui permet d'évaluer la qualité de la gestion d'un projet est le respect des échéances et des budgets. Ceci ne signifie pas qu'aucun dépassement ne doit ni ne peut être accepté. Cependant, un chef de projet dont les projets dépassent systématiquement les délais et les budgets qu'il avait prévus devra revoir ses méthodes de prévision ou ses méthodes de gestion de projet. Dans tous les cas, il devra s'efforcer d'expliquer les écarts.

Cette explication des écarts et la documentation au sujet du projet et en rapport avec la gestion du temps et du budget seront des renseignements précieux pour l'organisation. En effet, si un effort est fait pour expliquer le degré de réussite de chaque projet de développement de système, l'organisation accumulera de l'expérience dans le domaine et sera mieux à même de gérer efficacement les projets à venir. Les auteurs qui s'intéressent au concept de risque identifient en effet le manque d'expérience d'une équipe ou d'une organisation avec les projets d'informatisation comme étant un facteur de risque important.

Inversement, l'accumulation d'expérience contribuera à faire diminuer celui des projets futurs.

En plus d'évaluer la gestion du projet, on devra se préoccuper d'évaluer le système lui-même. Cette évaluation ne pourra être faite immédiatement après la mise en place; il sera alors trop tôt pour déterminer si le système atteint ses objectifs ou non. Il faudra attendre que le système ait été utilisé pendant un certain temps avant de pouvoir procéder à cette évaluation. L'évaluation de caractéristiques telles que le temps de réponse, la facilité d'utilisation, la qualité des outputs produits pourra être faite assez rapidement, c'est-à-dire trois ou quatre mois après que le système aura été mis en place. Pour d'autres aspects, il faudra attendre plus longtemps, selon le système. Pour les aspects reliés aux coûts/bénéfices, par exemple, il faudra souvent attendre que le système ait été utilisé pendant une année complète avant de procéder à l'évaluation. De la même façon, il ne sera parfois possible de mesurer le degré d'atteinte des objectifs qu'après que le système aura été utilisé pendant une période assez longue. Peu importe cependant la période qui aurait dû s'écouler avant qu'on puisse être en mesure de procéder à une évaluation, cette dernière est primordiale. Pourtant, nombreuses sont les organisations où cette activité est négligée.

Sources :

- Le processus unifié de développement de logiciels par Ivar Jacobson, Grady Booch et James Rumbaugh
- *http://lmi17.cnam.fr/~anceau/Documents/test1_99.pdf*, *http://uml.free.fr/*
- Le développement de systèmes d'information de Suzanne Rivard et Jean Talbot