

Méthodes de développement

Introduction

1 - Objectifs	2
2 - Risques d'un projet logiciel	2
3 - Préparation et conduite de projet	3
4 - Caractères particuliers du logiciel et conséquences.....	3
4.1 Caractère immatériel	3
4.2 Complexité	4
4.3 Erreurs résiduelles	4
4.4 Evolutivité	4
4.5 Fiabilité.....	5
5 - Diversité des logiciels	5
6 - Evolutions technologiques et productivité.....	6
7 - Conclusions.....	6
Bibliographie	8

1 - Objectifs

Un logiciel d'une certaine taille est un produit complexe. Comme tout développement de produit complexe, le développement de logiciel est un développement à risques, risques qui sont accrus par les caractères particuliers du logiciel (le logiciel est immatériel).

Pour limiter les risques, il y a besoin de suivre des méthodes éprouvées. Cependant, il y a une grande diversité dans les développements de logiciels, et il n'y a pas de méthodes universelles ni universellement admises.

Il ne s'agira donc pas d'enseigner une méthode de développement, mais des techniques de base permettant de construire des méthodes adaptées aux problèmes posés.

Les objectifs du cours sont donc :

- apprendre les techniques de base nécessaires pour maîtriser la réalisation de logiciels complexes,
- montrer l'utilisation de ces techniques sur un cas concret, par la réalisation d'un projet suffisamment complexe.

Les principales techniques de base sont :

- la préparation du projet,
- la conduite de projet et le travail en équipe,
- le développement itératif,
- la spécification du logiciel,
- la conception du logiciel,
- la fabrication du logiciel,
- la gestion de la configuration,
- la préparation à la maintenance,
- la documentation,
- l'assurance de la qualité.

2 - Risques d'un projet logiciel

L'expérience des projets logiciels importants menés ces dernières années, montre qu'une proportion non négligeable n'a pas abouti, et que pour pratiquement tous les autres on relève :

- un dérapage des délais,
- une augmentation des coûts,
- une mauvaise adéquation aux besoins des clients,
- et un grand nombre d'erreurs résiduelles.

Les conséquences peuvent être :

- l'insatisfaction du client qui pourra se traduire sur le plan des paiements, de la suppression de nouvelles commandes...
- des pertes financières pour le développeur qui peut avoir à supporter les surcoûts,
- le ternissement de l'image de marque du développeur,
- l'arrêt du projet, avec non paiement des travaux...

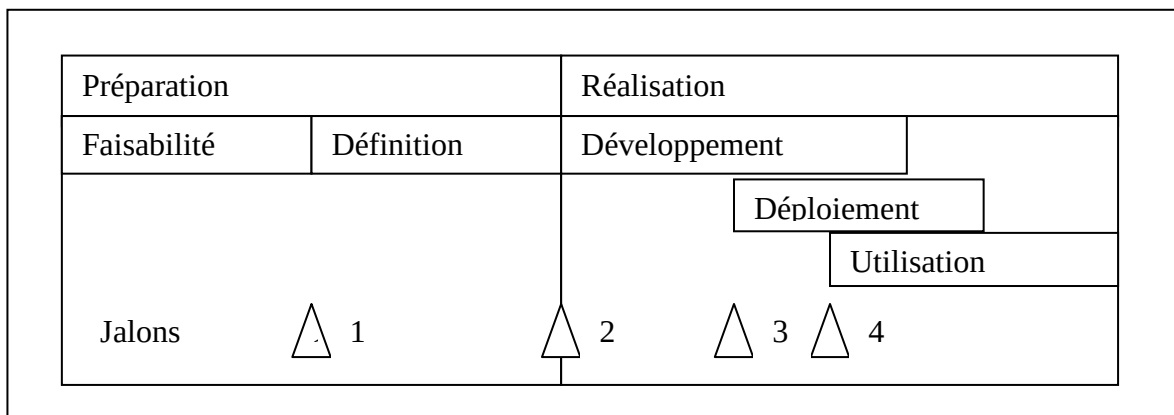
L'objectif d'une méthode de développement est de limiter ces types de risques.

3 - Préparation et conduite de projet

Pour limiter les risques, on appliquera au niveau de la préparation et de la conduite de projet les mêmes méthodes et techniques que les grands projets industriels.

Le projet devra être phasé avec des jalons pour passer d'une phase à l'autre.

Le schéma ci-dessous montre un exemple de phasage tiré des grands projets militaires. On distinguera les phases de préparation (faisabilité, définition), de développement, de déploiement ou production, et d'utilisation. On notera qu'au niveau réalisation un chevauchement des phases est possible. Les 4 jalons pourraient être le choix de concept, le lancement de la réalisation, la définition de l'architecture approuvée, et la qualification de la première version utilisable.



Le projet doit être soigneusement préparé avant de lancer le développement :

- comparaison de solutions (phase de faisabilité),
- rédaction de cahier des charges,
- planification, budget, organisation.

La conduite du projet nécessite :

- la mise en place d'une organisation projet avec un chef de projet,
- le suivi de l'avancement avec la tenue à jour des plannings,
- la mise en place de la gestion de configuration,
- la mise en œuvre des dispositions d'assurance qualité.

4 - Caractères particuliers du logiciel et conséquences

4.1 Caractère immatériel

Un logiciel n'est pas un produit matériel, c'est de l'information. Il n'est pas visible, pas palpable, pas mesurable...

C'est une cause de difficulté par rapport à la réalisation d'un matériel :

- l'état d'avancement ne peut être constaté visuellement,
- le contrôle ne peut être fait avec des appareils de mesure,
- il ne peut pas être étiqueté,

- toutes les aberrations sont possibles, il n'y a pas les limitations des lois de la physique...

Le développement du logiciel demande donc la mise en œuvre de procédures particulières pour pouvoir suivre son avancement (suivi de projet) et maîtriser l'état du produit (gestion de configuration).

4.2 Complexité

Un logiciel est une suite d'éléments binaires (bits). Chaque élément est simple, mais il y en a plusieurs millions ou dizaines de millions.

C'est un autre ordre de grandeur qu'au niveau des matériels, ou même un matériel complexe ne comporte que quelques centaines de pièces élémentaires.

Un logiciel est donc toujours un produit très complexe, et le développement présente des risques qu'il faut réduire en utilisant une méthode éprouvée.

4.3 Erreurs résiduelles

Un logiciel est un produit très complexe. Lors du codage du logiciel, le programmeur introduit un grand nombre d'erreurs dans le code.

Il faut donc effectuer des tests pour détecter les erreurs et les corriger. Cependant les tests ne détectent qu'une partie des erreurs, et il reste toujours des erreurs. Cela n'empêche pas le logiciel de fonctionner dans les conditions courantes, les erreurs résiduelles pouvant avoir des effets limités ou ne concerner que des conditions particulières.

Le nombre d'erreurs résiduelles dépend des méthodes employées et de l'effort de test. Il n'est évidemment pas connu, mais on l'estime entre 0,1 et 20 erreurs par 1000 lignes de code selon les efforts faits pour éviter et corriger les erreurs.

Pour limiter les erreurs résiduelles, il est recommandé :

- de suivre une méthode de développement telle qu'elle vous sera présentée,
- de mettre en place des contrôles qualité,
- d'utiliser des techniques de réalisation permettant de limiter les erreurs (programmation objet, langages objet...),
- de planifier la réalisation en laissant un temps important pour faire des tests étendus et corriger les erreurs.

4.4 Evolutivité

Le logiciel est un produit très évolutif :

- il reste toujours des erreurs qu'il faut corriger lorsqu'elles sont détectées et qu'elles sont gênantes pour le fonctionnement (il n'y a pas lieu de corriger toutes les erreurs),
- les besoins des utilisateurs évoluent,
- l'environnement informatique évolue rapidement (ordinateurs, systèmes d'exploitation, outils...)

Lorsqu'on réalise un logiciel, il faut presque toujours prévoir qu'il évoluera, et préparer cette capacité à évoluer. C'est la préparation de la maintenance qui doit être faite pendant le développement du logiciel auquel elle doit être intégrée.

Une bonne méthodologie permet une préparation implicite de la maintenance pendant le développement.

Nota

La notion de "maintenance" du logiciel n'a pas la même signification que pour les matériels. On ne répare pas un logiciel, mais on réalise un nouveau produit (nouvelle version de logiciel) qui vient remplacer le produit logiciel en service (on peut faire une analogie avec les matériels jetables).

4.5 Fiabilité

La fiabilité (ou taux de défaillance) d'un logiciel n'est pas évaluable actuellement.

Les théories de la fiabilité sont faites aujourd'hui pour des matériels fabriqués en grand nombre. Dans ce cas les défaillances des composants des matériels représentent l'essentiel des pannes, et on calcule le taux de défaillance d'un matériel en additionnant le taux de défaillance de ses composants et en négligeant les autres causes de défaillance.

Dans le cas d'un logiciel, les composants sont les bits, et il n'y a pas de défaillance de composants... On constate pourtant des défaillances du logiciel qui sont dues aux erreurs résiduelles et sont donc des défauts de conception.

On peut bien sur faire un relevé des défaillances constatées lors de l'utilisation d'un logiciel, mais on est loin de pouvoir évaluer un taux de défaillance. Des solutions ont été proposées par des chercheurs, mais elles ne semblent pas satisfaisantes.

L'utilisation des logiciels pose donc des problèmes dans les systèmes sûrs où on doit calculer la fiabilité du système pour garantir une probabilité de défaillance très faible.

Dans le cas des systèmes de haute sécurité, tels les systèmes nucléaires, on fait l'hypothèse que le logiciel a une fiabilité nulle pour l'évaluation de la sécurité. Les matériels mis en œuvre par le logiciel sont conçus pour conserver la sécurité quelques soient les bêtises faites par le logiciel.

Dans le domaine de l'aviation civile, on ne peut aujourd'hui se passer de logiciel et il faut pourtant assurer la sécurité des passagers des avions (c'est à dire leur garantir que la probabilité du crash est inférieure à 10^{-9}). Pour les matériels, on peut faire des calculs de fiabilité, mais pour les logiciels, à défaut de pouvoir évaluer la fiabilité, on demande de suivre une méthode de développement permettant de limiter les erreurs et de faire des tests extensifs.

5 - Diversité des logiciels

Les logiciels apparaissent aujourd'hui dans tous les domaines : calcul scientifique, gestion, bureautique, jeux, automatismes industriels...

Il y a une très grande diversité, certains logiciels doivent être très surs, d'autres sont très volumineux, d'autres manient des grandes masses de données...

Classiquement, on distingue 3 types de logiciels :

Type	Caractéristiques
Scientifique	Algorithmes complexes Volumineux
Industriel	Contraintes de temps (temps réel) Sûreté de fonctionnement Interfaces multiples et spécifiques
Gestion	Masses de données

Face à une telle diversité, il n'y a pas de méthode de développement universelle, la méthode doit être adaptée au type de logiciel et à sa criticité.

6 - Evolutions technologiques et productivité

La production du logiciel est un travail intellectuel. Il requiert du personnel de haut niveau. Le coût ne peut être réduit comme pour le matériel par des techniques d'industrialisation et d'automatisation de la production.

Les logiciels sont de plus en plus complexes, et pour que le prix reste abordable, il faut sans cesse améliorer la productivité des réalisateurs.

Depuis les débuts de l'informatique, il y a eu des améliorations technologiques considérables qui ont accru la productivité :

- dans les langages : langages symboliques, langages structurés, langages typés, langages objets,
- dans les techniques de conception : conception structurée, conception orientée objet,
- dans les outils : introduction de l'interactivité puis du graphisme, débogueurs, éditeurs de textes, gestionnaires de configuration...
- méthodes de développement : développement prototype, développement itératif...

La productivité s'est ainsi bien améliorée. Cependant cette évolution nécessite de se remettre en cause perpétuellement pour rester compétitif.

La productivité ne sera jamais infinie, et pour réduire les coûts des logiciels, l'enjeu actuel est de pouvoir réutiliser des morceaux de logiciels existants sans avoir à les adapter et les revalider. Les techniques objets devraient permettre d'atteindre cet objectif. Il faudra voir en particulier si le concept d'EJB (Entreprise Java Bean ou objets métiers) se développe.

7 - Conclusions

Le logiciel est un produit complexe et qui présente des caractères particuliers qui rendent le développement à risques.

Il est donc nécessaire :

- de préparer soigneusement le développement,
- d'adopter une méthode de développement ayant fait ses preuves,
- de bien organiser le projet et d'en assurer le suivi,
- de mettre en place des dispositions d'assurance qualité.

Bibliographie

Méthodologies

Norme ISO 12207 Processus du cycle de vie des logiciels
Description des activités du cycle de vie des logiciels

B.W Boehm The spiral model
Le modèle en spirale

Philippe Kruchten Introduction au RUP
Présentation du RUP (Rational Uniform Process) méthode de développement itérative basée sur UML (Uniform Method Language - notation standardisée pour la conception objet) commercialisée par Rational

Documentation du RUP (Rational Uniform Process) sur le site internet de IBM Rational
www.rational.com

G Booch Conception orientée objet et applications Addison Wesley Paris 1992
Un des pères de la conception objet

J Rumbaugh Modélisation et conception orientée objet
L'autre père de la conception objet

DO178B Considérations sur le logiciel en vue de la certification des systèmes sûrs RTCA - EUROCAE 1992
Méthode de développement de logiciels sûrs pour l'aéronautique

Charroux, Osmani et Thierry-Mieg UML2 - Pearson Education 2005

Documentation sur les logiciels

B.W Boehm Software engineering economics - Prentice Hall 1981
Etude des coûts des logiciels

Gestion de projet

Jean Cavaillès Méthodes de management de programme - Teknéa 1991
Gestion des grands projets

RG AERO 000 40 Spécification de management des programmes aéronautiques et spatiaux - Bureau de normalisation aéronautique
Guide de rédaction d'une spécification de management

ISO 10006:1997, Management de la qualité — Lignes directrices pour la qualité en management de projet

ISO 10007:1995, Management de la qualité — Lignes directrices pour la gestion de configuration

Guides documentaires

RG AERO 000 08 Guide pour l'élaboration d'une spécification technique de besoin
Guide plutôt orienté matériel

Norme ANSI/IEEE STD 830 Guide to Software Requirements Specifications 1984
Guide pour la spécification technique de besoin logiciel

Norme IEEE STD 828 IEEE standard for software configuration plan 1990
Guide pour le plan de gestion de configuration

Qualité logiciel

Norme ISO 10005:1995, Management de la qualité — Lignes directrices pour les plans qualité

Norme IEEE STD 730.1 IEEE standard for quality assurance plan 1989
Guide pour le plan qualité logiciel

Norme ISO/CEI 9126 Information technology - Software product evaluation - Quality characteristics and guidelines for their use 1991
Caractéristiques de qualité

Norme ISO 9000 version 2000 : Systèmes de management de la qualité — Principes essentiels et vocabulaire

Norme ISO 9001 version 2000: Systèmes de management de la qualité - Exigences

Norme ISO 9004 version 2000: Systèmes de management de la qualité - Lignes directrices pour l'amélioration des performances

ISO 9000-3:1997, Normes pour le management de la qualité et l'assurance de la qualité — Partie 3: Lignes directrices pour l'application de l'ISO 9001:1994 au développement, à la mise à disposition, à l'installation et à la maintenance du logiciel
Application de l'ISO 9001 aux logiciels