SDLC & STLC

ICT304

Seance 2 & 3

Contents

SDLC

- Definition
- Les Etapes du SDLC
- La Modelisation des Processus du SDLC

SDLC

Definition

Le Cycle de vie du développement d'un logiciel (SDLC) est un ensemble d'étapes de la réalisation, de l'énoncé des besoins à la maintenance au retrait du logiciel sur le marché informatique.

De façon générale, on peut dire que le cycle de vie du logiciel est la période de temps s'étalant du début à la fin du processus du logiciel.

SDLC

le SDLC commence donc avec a proposition ou la décision de développer un logiciel et se termine avec sa mise hors service.

L'origine de ce découpage provient du constat que les erreurs ont un coût d'autant plus élevé qu'elles sont détectées tardivement dans le processus de réalisation.

Objectif du SDLC

Objectif

L'objectif principal du cycle de vie du développement d'un logiciel est de permettre la détection des erreurs au plus tôt et par conséquent, maîtriser la qualité du produit, les délais de sa réalisation et les coûts associés.

SDLC

- Definition
- Les Etapes du SDLC
- La Modelisation des Processus du SDLC

Les Etapes du SDLC

Le développement de logiciel impose d'effectuer un certain nombre d'étapes. Il existe de nombreux modèles de cycle de vie du développement d'un logiciel, les plus courants comportent les phases suivantes :

- La collection des exigences ;
- L'étude du préalable (faisabilité ou opportunité) ;
- Définition et analyse de besoins (Spécification) ;
- La conception du logiciel ;

Les Etapes du SDLC

- Le codage ;
- Les tests ;
- L'intégration ;
- L'installation ;
- La maintenance ;
- La disposition.

La collection des exigences

Cette étape (la collection des exigences) fait avancer le travail de l'équipe du développement du logiciel jusqu'à la visibilité du projet informatique.

A ce niveau, l'équipe de développement discute avec certains partenaires des divers problèmes du domaine et essaie de proposer autant d'informations possibles sur les différentes exigences.

La collection des exigences

A ce niveau, les exigences sont contemplées et isolées en fonction des besoins des utilisateurs; les exigences du système et les exigences fonctionnelles. Les exigences sont collectées en utilisant un nombre donné des pratiques telles que :

- Etude du système ou logiciel existant ou obsolète ;
- Conduite des interviews auprès des utilisateurs et développeurs;
- Référencer aux différentes bases de données ;
- Et la collection des réponses au moyen de questionnaires.

L'étude du préalable

Le développement est précédé d'une étude d'opportunité ou étude préalable. Cette phase a comme objectif de répondre aux questions suivantes :

Questions

- Pourquoi développer le logiciel ?
- Comment procéder pour faire ce développement ?
- Quels moyens faut-il mettre en œuvre ?

L'étude du préalable

Elle comprend à la fois des aspects techniques et de gestion. Parmi les tâches techniques, groupées sous le terme étude préalable, on peut citer :

- Dresser un état de l'existant et faire une analyse de ses forces et faiblesses;
- Identifier les idées ou besoins de l'utilisateur;
- Formuler des solutions potentielles;
- Faire des études de faisabilité;
- Planifier la transition entre l'ancien logiciel et le nouveau, s'il y a lieu;
- Affiner ou finaliser l'énoncé des besoins de l'utilisateur.

Définition et analyse de besoins

Lors de la phase d'analyse, également appelée phase de spécification (requirements phase, analysis phase, definition phase), on analyse les besoins de l'utilisateur ou du système englobant et on définit ce que le logiciel devra faire.

Resultat

Le résultat de la phase d'analyse est consigné dans un document appelé cahier des charges du logiciel ou spécification du logiciel

Définition et analyse de besoins

Une spécification comporte les éléments suivants :

- description de l'environnement du logiciel ;
- spécification fonctionnelle (functional specification), qui définit toutes les fonctions que le logiciel doit offrir ;
- comportement en cas d'erreurs, c'est-à-dire dans les cas où le logiciel ne peut pas accomplir une fonction ;
- performances requises (performance requirements), par exemple : temps de réponse, encombrement en mémoire, sécurité de fonctionnement ;

Définition et analyse de besoins

- les interfaces avec l'utilisateur (user interface), en particulier le dialogue sur terminal, la présentation des écrans, la disposition des états imprimés, etc.
- interfaces avec d'autres logiciels et le matériel ;
- contraintes de réalisation, telles que l'environnement de développement, le langage de programmation à utiliser, les procédures et normes à suivre, etc.

La conception du logiciel

La phase d'analyse est suivie de la phase de conception, généralement décomposée en deux phases successives :

Phase de Conception

- Conception Générale
- Conception Détaillée

Conception Générale

Conception générale ou conception architecturale (preliminary design ou architectural design) : Si nécessaire, il faut commencer par l'ébauche de plusieurs variantes de solutions et choisir celle qui offre le meilleur rapport entre coûts et avantages. Il faut ensuite figer la solution retenue, la décrire et la détailler.

Conception Détaillée

La conception détaillée affine la conception générale. Elle commence par décomposer les entités découvertes lors de la conception générale en entités plus élémentaires. Cette décomposition doit être poursuivie jusqu'au niveau où les entités sont faciles à implémenter et à tester, c'est-à-dire correspondent à des composants logiciels élémentaires.
Ce niveau dépend fortement du langage de programmation retenu pour l'implémentation. Il faut ensuite décrire chaque composant logiciel en détai

Le codage

Après la conception détaillée, on peut passer à la phase de codage, également appelée phase de construction, phase de réalisation ou phase d'implémentation (implémentation phase, construction phase, coding phase). Lors de cette phase, la conception détaillée est traduite dans un langage de programmation.

Les tests

La phase d'implémentation est suivie de la phase de test (test phase). Durant cette phase, les composants du logiciel sont évalués et intégrés, et le logiciel lui-même est évalué pour déterminer s'il satisfait la spécification élaborée lors de la phase d'analyse. Cette phase est en général subdivisée en plusieurs phases.

Phases de test

- Test Unitaire
- Test d'integration
- Test Système

L'intégration

Après avoir effectué avec succès la phase des tests de tous les composants, on peut procéder à leur assemblage, qui est effectué pendant la phase d'intégration (intégration phase). Pendant cette phase, on vérifie également la bonne facture des composants assemblés, ce qu'on appelle le **test d'intégration** (intégration test).

Installation

Après avoir intégré le logiciel, on peut l'installer dans son environnement d'exploitation, ou dans un environnement qui simule cet environnement d'exploitation, et le tester pour s'assurer qu'il se comporte comme requis dans la spécification élaborée lors de la phase d'analyse.

Maintenance

Après l'installation suit la phase d'exploitation et de maintenance (operation and maintenance phase). Le logiciel est maintenant employé dans son environnement opérationnel, son comportement est surveillé et, si nécessaire, il est modifié. Cette dernière activité s'appelle la maintenance du logiciel (software maintenance).

Maintenance

Il peut être nécessaire de modifier le logiciel pour corriger des défauts,pour améliorer ses performances ou autres caractéristiques pour adapter le logiciel à un nouvel environnement ou pour répondre à des nouveaux besoins ou à des besoins modifiés. on distingue:

Type de Maintenance

- La Maintenance Corrective,
- La Maintenance Adaptative,
- La Maintenance Perfective.

Type de Maintenance

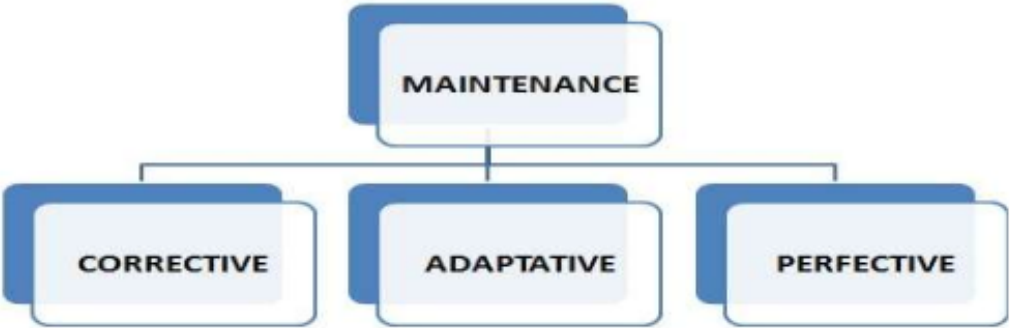

Figure: Type de maintenace

Maintenacne Corrective

c'est une maintenance qui Corrige les erreurs et les défauts d'utilité, d'utilisabilité, de fiabilité... cette maintenance Identifie également les défaillances, et les dysfonctionnements en localisant la partie du code responsable. Elle Corrige et estime l'impact d'une modification. Attention, La plupart des corrections introduisent de nouvelles erreurs et Les coûts de correction augmentent exponentiellement avec le délai de détection. Bref, la maintenance corrective donne lieu à de nouvelles livraisons (release).

Maintenance Adaptative

C'est une maintenance qui ajuste le logiciel pour qu'il continue à remplir son rôle compte tenu du l'évolution des Environnements d'exécution, des Fonctions à satisfaire et des Conditions d'utilisation. C'est par exemple le changement de SGBD, de machine, de taux de TVA, an 2000.

La Maintenance Perfective

C'est une maintenance qui sert a accroître et améliorer les possibilités du logiciel afin de donne lieu à de nouvelles versions. C'est par exemple ; les services offerts, l'interface utilisateur, les performances...

SDLC

Remarque

Ces étapes ne doivent pas être vues comme se succédant les unes aux autres de façon linéaire. Il y a en général (et toujours) des retours sur les phases précédentes, en particulier si les tests ne réussissent pas ou si les besoins évoluent.

La Modelisation des Processus du SDLC

La Modelisation des Processus du SDLC

STLC

Software Testing Life Cycle