

# **CHAPITRE 01 :**

## **INTRODUCTION A LA REINGENERIE**

### **Introduction :**

le domaine d'informatique parmi les domaines plus intéressants dans la vie intellectuelle ,l'informatique possède maintenant plus d'espace d'environnement permet à l'humain l'exploitation dans chaque domaine .

le développement des logiciels depuis des années est atteint un haut degré , le système de programmation était toujours pour accéder à l'application dotée , mais aujourd'hui nous pouvons inverser l'action , l'évolution de programmation permet découvrir les processus et les instructions implicites qui ont été cachés sous une interface graphique .

la réingénierie a fait ce raisonnement très possible , La réingénierie de logiciels est maintenant perçue comme un processus permanent, une évolution continue en raison de changements perpétuels des processus d'affaires et des technologies de l'information.

### **1 Définition**

La réingénierie en français, est une l'étude et l'analyse d'un système pour en déduire son fonctionnement interne, et était appliqué dans plusieurs domaines ."LeReverse Engineering, ou rétro-ingénierie / ingénierie inverse en français, représente l'étude et l'analyse d'un système pour en déduire son fonctionnement interne, et se retrouve dans de nombreux domaines de l'ingénierie : génie civil, mécanique, ingénierie navale, aéronautique, etc...

Dans le domaine informatique, faire du reverse revient souvent à utiliser des outils d'analyse comme le désassembleur ou le décompilateur dans le cadre d'analyse en boîte blanche. Nous reviendrons par la suite sur les différents contextes d'analyse. On retrouve également des outils permettant d'analyser les entrées/sorties d'un programme en sniffant le réseau comme Wireshark, ainsi que de nombreux autres outils, sur lesquels nous reviendrons par la suite". [9]

### 2 Les origines de la rétro ingénierie

La rétroingénierie c'est le domaine le plus complexe et elle est défini par Chikofsky en 1990 comme une processus d'analyse d'un n'importe quel système après elle devient le plus utilisable par les cracker et les hacker "Dans le contexte de l'ingénierie logicielle, le terme rétro-ingénierie a été défini en 1990 par Chikofsky et Cross comme le processus d'analyse d'un système pour identifier les composants du système et leurs interrelations, et créer des représentations du logiciel dans une autre forme ou dans un niveau d'abstraction plus élevé. La rétro-ingénierie a été traditionnellement vue comme un processus en deux étapes : l'extraction des informations et l'abstraction. L'étape d'extraction des informations analyse les artefacts du logiciel pour recueillir des données brutes, tandis que l'étape d'abstraction crée des vues et documents orientés utilisateur à partir des données brutes . Chikofsky et Cross ont présenté une structure de base pour les outils implémentant la rétro-ingénierie . Le logiciel étudié est analysé et les résultats de cette analyse sont stockés dans une base d'informations. Par la suite, ces informations sont utilisées pour produire différentes vues du logiciel, comme les diagrammes, les rapports, et les métriques".

### 3 Mise en œuvre

- Il doit être fait en conjonction avec les tests dynamiques.
- Sa réalisation nécessite l'utilisation d'environnements de développement appropriés (p.ex. Eclipse, NetBeans).
- Respecter le « code propre ». [10]

### 4 Objectifs:

Mais pourquoi faire du La rétro ingénierie?

Dans le domaine de l'informatique, le reverse a de nombreuses utilités :

- Il peut être utilisé pour comprendre en détail le fonctionnement d'un logiciel,
- Améliorer la sécurité et qualité d'un logiciel (recherche de failles, étude d'un virus pour l'éradiquer, ...),
- Permettre de contourner les protections logiciels (numéro de licence, etc...),
- Reproduire le comportement d'un logiciel.[9]

### 5 Compétences

Pour pouvoir rentrer dans ce cercle de "reverser", il convient de disposer d'un minimum de bagages, dont voici une liste non exhaustive :

- Savoir développer,
- Connaître les spécifications du processeur cible (CISC (Complex Instruction Set Computing): x86, RISC (Reduced Instruction Set Computing): ARM, etc...).

Un bon commencement consiste à connaître les techniques d'optimisation réalisées par un processeur, liées à la théorie de la compilation : déroulement de boucle (si le code généré par une boucle peut tenir dans une page de code (4Ko sur x86/x64), la boucle est supprimée et le code est copié autant de fois que le nombre d'itérations de la boucle), l'alignement, qui consiste à travailler avec la taille native des mots du processeur (ne pas travailler sur 1 octet sur un bus 32 bits, etc...), etc...,

- Avoir des bonnes bases concernant la compilation (Analyse lexicale, syntaxique, sémantique, parseur, génération de code, etc...),
- Savoir faire de la reconnaissance de formes, de pattern,
- Et bien sûr, pour pouvoir être un bon rétro-ingénieur, il faut savoir être ingénieux!! [9].

### 6 Raisons

- Pour ne pas accumuler de dette technique.
- Pour améliorer la logique de conception du logiciel.
- Pour rendre le système plus simple à comprendre et donc à maintenir, étendre et vérifier.
- Pour aider à trouver les bugs. [10]

### 7 Les droits du La rétro ingénierie

#### 7.1 Le droit d'analyse :

Un nouveau droit d'analyser le logiciel est prévu par la loi de 94. Il est défini comme celui d'observer, d'étudier ou de tester le fonctionnement de ce logiciel lorsque l'utilisateur effectue

toute opération de chargement, d'affichage, d'exécution, de transmission ou de stockage du logiciel.

C'est donc un droit au reverse engineering, défini comme l'analyse d'un système destinée à rechercher ses principes de conception. L'analyse n'est autorisée, tout comme la décompilation, que si elle ne cause pas de préjudice injustifié aux intérêts légitimes de l'auteur, et qu'elle ne porte pas atteinte à l'exploitation normale du programme, conformément au droit international.

Enfin, la loi prévoit que toute stipulation contractuelle contraire à ce droit est nulle et non avenue. Mais le contrat pourra fixer le cadre de l'utilisation normale du logiciel.[11]

### 7.2 Le droit de décompilation :

Le droit de décompiler le logiciel, c'est à dire de tenter de retrouver le programme source à partir du programme objet, est ajouté à la condition qu'il soit effectué à des fins d'interopérabilité. La décompilation doit donc servir à interfacer le logiciel avec des logiciels de coordination, c'est à dire permettre l'articulation des logiciels entre eux. Elle doit se limiter aux parties nécessaires à l'interfaçage. En effet, cette décompilation suscite des inquiétudes notables quant à la protection du programme.

Si la loi interdit toute stipulation contraire à ce droit l'auteur pourra prévoir, contractuellement, de l'organiser, en prévoyant une procédure d'information à son égard, par l'utilisateur, de son désir de décompiler, suivi d'une communication des interfaces utiles et des points d'ancrage.

Puisqu'elle n'est autorisée que dans le cas où les informations nécessaires n'ont pas déjà été rendues facilement et rapidement accessibles, elle ne pourra être considérée comme indispensable au sens de la loi dans le cas d'une standardisation du logiciel, d'une publication ou d'une réponse à une demande individuelle.[11]

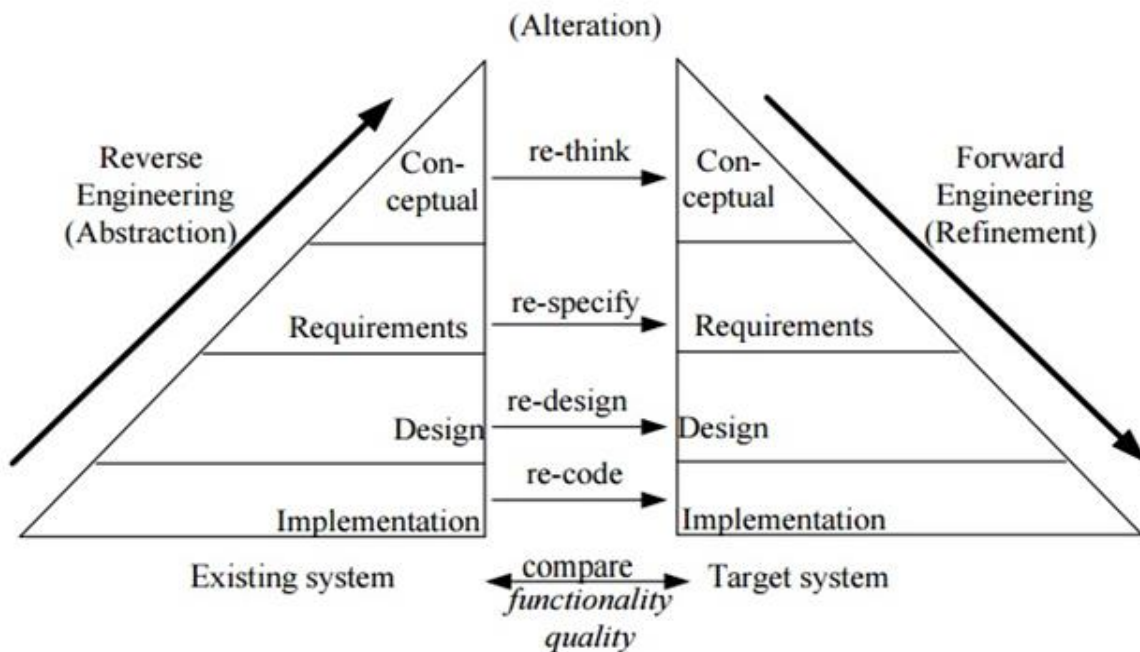
## 8 Les Outils :

Le meilleur désassembleur est sans équivoque Windasm, qui tourne sur Windows. Il vous faudra aussi un éditeur hexadécimal, alors là je ne peux pas citer de nom vu le nombre qu'il existe à vous de faire votre choix. Ensuite il vous faudra un debugger, le meilleur est Soft Ice. Mais il existe aussi OLLYDBG. On peut citer plein d'autre programme très utile dans le reversing comme Procdump (détecter lorsqu'un logiciel est compresser), SmartCheck....[11].

## 9 Les applications en informatique

A partir des circuits informatiques, il est possible, à partir d'algorithmes spécifiques (déassembleur, décompilateur), de retrouver les bases du programme informatique qu'il génère. La "**rétro ingénierie logicielle**" peut également s'appliquer à la récupération des données entre un ordinateur et son périphérique, qui permet de communiquer à son tour avec le périphérique. Ce type de rétro-ingénierie est en particulier utilisé pour la production de logiciels libre pour la fabrication de pilotes (pour imprimantes, webcam, cartes graphiques...).[12]

La rétro ingénierie logiciel pose le problème de la légalité. Le film *Paycheck*, inspiré d'une nouvelle de Philip K. Dick, met d'ailleurs en scène au début du film un personnage qui est payé pour comprendre la technologie utilisée par un concurrent à partir de l'objet physique. En évitant de voler les codes et les plans de ses adversaires, le "copieur" peut faire croire à une découverte simultanée et améliorer encore la technologie qu'il a copiée.[12]



**Figure1.1:** principes de réingénierie [13]

La figure ci-dessus correspond à trois principes de réingénierie : l'abstraction, l'altération et le raffinement. L'abstraction est une augmentation progressive du niveau

abstrait du système. Elle produit une représentation qui en souligne les caractéristiques du système par la suppression de certaines informations sur les autres. Le mouvement à la hausse que l'on appelle rétro-ingénierie est lié à des sous-processus, des outils et des techniques. L'altération est l'élaboration d'un ou plusieurs changements à une représentation du système sans changer le degré d'abstraction, y compris l'addition, la suppression et la modification de l'information, mais pas la fonctionnalité. Le raffinement est la diminution progressive du niveau d'abstraction de la représentation du système et est provoqué par la substitution successive d'information du système existant avec des informations plus détaillées. L'ingénierie qui ressemble au développement de logiciel de nouveau code, mais avec quelques améliorations de processus. Toutefois, pour la respécification des exigences, les techniques de rétro-ingénierie doivent être appliquées pour la mise en œuvre et la conception pour obtenir les caractéristiques fonctionnelles [13].

### **10 Des méthode générique de rétro-ingénierie:**

#### **10.1 Une méthode générique de rétro-ingénierie de bases de données:**

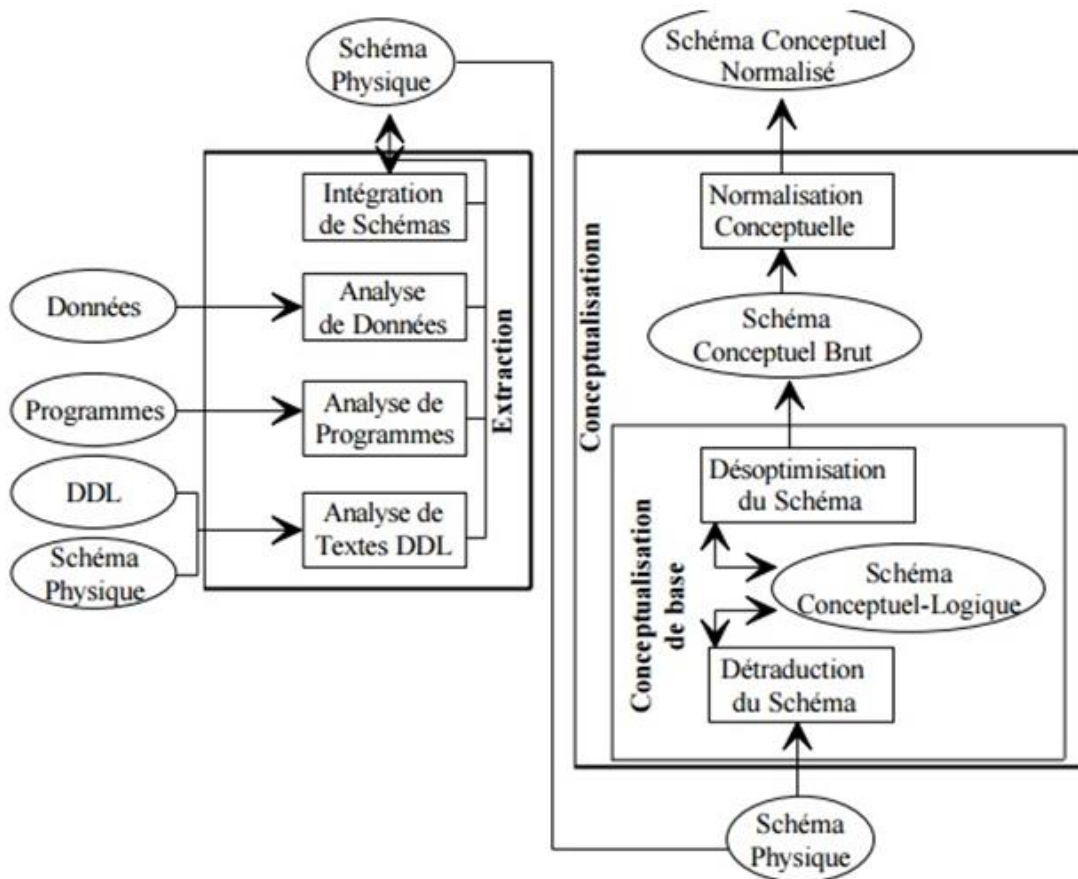
Cette méthode comporte deux processus principaux (figure 2), à savoir l'extraction des structures de données et la conceptualisation de ces structures[14]. Ces deux parties correspondent à des schémas différents qui requièrent des raisonnements, des concepts et des outils différents. De plus, cette division est à peu près l'inverse des processus de conception physique et logique habituellement admis en ingénierie des bases de données.

#### **10.2 L'extraction des structures de données :**

Cette phase a pour objet la reconstruction du schéma logique complet selon le modèle du SGD2, y compris toutes les contraintes et les structures de données non explicitement déclarées. Certains SGD (les SGBD par exemple) offrent, sous une forme ou une autre, une description du schéma global des données. Bien que ce schéma soit déjà assez complet, il devra être enrichi au moyen d'une analyse des autres composants de l'application (vues, code procédural, données, écrans de saisie, ...).

Le problème est beaucoup plus complexe quand il s'agit de recouvrer le schéma conceptuel de fichiers classiques. Chaque programme source devra être analysé pour retrouver une partie de la

structure des données. Cette analyse doit aller bien au-delà de la simple recherche de la structure des fichiers déclarés dans les programmes. [14]



**Figure 1.2:** méthode générique de retro-ingénierie de base de données.

En particulier, deux types de problèmes peuvent être rencontrés, quelque soit le SGD : les structures sont définies dans les programmes ou il y a perte de spécifications. Il est possible que lors de la conception de l'application l'on n'ait pas utilisé toutes les possibilités du SGD et implémenté dans le programme certaines contraintes qui auraient pu être déclarées dans le SGD.

Les structures définies dans les programmes sont des structures ou des contraintes qui ne sont pas déclarées dans le SGD mais qui sont représentées ou vérifiées de façon procédurale dans le programme. Par exemple : des contraintes référentielles. La perte de spécifications vient de la non-implémentation dans le SGD et dans le programme, de certaines contraintes du schéma

conceptuel. Elles peuvent, par exemple, être vérifiées par construction : les données sont importées et sont correctes; l'exécution du programme ne permet pas de violer ces contraintes.

Recouvrer les structures définies dans les programmes ou perdues est une tâche complexe, pour laquelle il n'existe pas encore de méthodes déterministes. Seule une analyse méticuleuse de toutes les sources d'informations disponibles permet de retrouver les spécifications. Le plus souvent ces informations doivent être consolidées par la connaissance du domaine de l'application. Les processus principaux de l'extraction des structures de données sont les suivants :

- analyse des déclarations des structures de données dans les scripts de définition du schéma et dans les sources du programme;
- analyse du code source du programme pour recouvrer les structures définies dans les programmes;
- analyse des données sur lesquelles travaille le programme pour en trouver les structures et les propriétés ainsi que pour confirmer ou infirmer certaines hypothèses;
- intégration des différents schémas obtenus lors des étapes précédentes. Dans cet article, nous tenterons d'illustrer une technique particulièrement puissante d'analyse de programmes qui permet de détecter des structures algorithmiques qui suggèrent des structures de données et des contraintes

### **10.3 La conceptualisation des structures de données :**

Le deuxième processus consiste en l'interprétation du schéma obtenu lors de l'extraction des structures de données pour en dériver un schéma conceptuel.

Il détecte et transforme (ou élimine) les redondances et les structures non conceptuelles introduites lors de la conception de la base de données.

La conceptualisation des structures de données se fait en trois étapes : la préparation du schéma, la conceptualisation de base et la normalisation. La préparation du schéma consiste en la modification de certains noms de manière à les rendre plus significatifs. Lors de la conceptualisation de base, toutes les structures de données qui ne sont pas conceptuelles sont éliminées ou transformées. Elle est elle-même décomposée en :



### 10.3.1 détraduction du schéma :

le schéma de départ de ce processus est conforme au modèle du SGD utilisé, toutes ses structures de données spécifiques au SGD doivent être détectées et transformées en structures de données conceptuelles équivalentes

### 10.3.2 désoptimisation du schéma :

le schéma a été restructuré et enrichi pour des raisons d'optimisation. Il faut détecter et éliminer ces structures. Finalement nous disposons d'un schéma conceptuel qui peut encore être transformé, de manière à le rendre conforme à un standard méthodologique par exemple. Il s'agit d'un processus classique de normalisation conceptuelle. [14]

## 10.4 Fragmentation de programmes :

Nous décrivons brièvement une technique d'analyse et de transformation de programme utilisée dans la phase d'extraction des structures de données. Cette technique, appelée fragmentation de programme (program slicing), permet d'extraire d'un programme le fragment (idéalement) nécessaire et suffisant pour comprendre le comportement de ce programme à un point déterminé. Ce point est appelé critère de fragmentation. Un fragment de programme est constitué des parties du programme qui affectent les valeurs calculées par rapport au critère de fragmentation. Idéalement, un fragment est constitué de toutes les instructions qui affectent les valeurs calculées par rapport au critère de fragmentation et uniquement celles-là. Le concept original de fragments de programmes a été introduit par Mark Weiser [Weiser 84]. Mark Weiser prétend qu'un fragment correspond à l'abstraction mentale d'un programmeur quand il débogue un programme. Différentes notions de fragmentation de programme et méthode de calcul ont été définies depuis, répondant chacune à des exigences particulières [14].

## 10.5 L'outil DB-MAIN :

L'environnement d'ingénierie de bases de données DB-MAINest dédié à l'ingénierie des bases de données, ce qui englobe la rétro-ingénierie. En particulier son but est d'assister le développeur dans la conception, la rétro-ingénierie, la migration, la maintenance et l'évolution de bases de données.[14]

### **conclusion:**

Dans ce chapitre nous avons parlé sur larétroingénierie qui amène de nombreux bienfaits dans l'analyse globale de la qualité logicielle. Les métriques nous indiquent les faiblesses du code. Cela nous fait mettre en place des démarches d'amélioration du code. Entraînant une meilleure adaptabilité aux futurs rajouts, et augmentant les performances. Cependant les métriques ne sont que des indicateurs pour les responsables de qualité du code. Ce sont à eux de faire des démarches pour regrouper ces facteurs et en tirer le coté qualitatif. Les tableaux de bords aident dans la démarche de prise de décision, en regroupant automatiquement les indicateurs vers les facteurs de qualité logicielle. Ces vérifications régulières permettent une remise en cause du code récurrente et aisée, en construisant suivant les logiciels les suivis automatisés.