

2014-2015

Projet Enigma

Compte-Rendu

DUT Informatique (S4)

TROUVE Robin
KISSI Naïm
DE LIMA Marianna

Encadré par:

M. ROY

Sommaire

Introduction	2
1. Enigma : Machine de cryptage	3
1.1. Création d'Enigma (remise en contexte)	3
1.1.1. Remise en contexte	
1.1.2. Utilisation d'Enigma	
1.2. Fonctionnement d'Enigma	
1.2.1. Les rotors	
1.2.2. Le plugboard	
1.2.3. Le réflecteur	
1.3. Cryptanalyse par Turing	
2. Enigma : Simulation informatique	
2.1. Organisation du projet (architecture mvc, méthodes de travail, outils)	
2.1.1. Choix et contraintes	
2.1.2. Outils de travail	
2.1.3. Gestion de l'équipe (réunion, scrum ?, diagramme de gantt, micro -taches)	
2.2. Représentation d'Enigma (ex : rotor = objet ... comment on le conceptualise et comment il fonctionne dans le code, comment on a essayé de reproduire la machine physique)	
2.2.1. Etude d'Enigma	
2.2.2. Conception UML	
2.3. Difficultés rencontrées et solutions retenues	
3. Manuel utilisateur	
Conclusion	
Annexes	

Introduction

Dans le cadre de notre DUT Informatique, en semestre 4, nous sommes amenés à réaliser un projet alliant mathématiques et informatique afin d'appliquer nos connaissances et la méthodologie que nous avons pu acquérir au cours de notre formation.

Le sujet de ce projet est la machine Enigma, machine de cryptage la plus connue et utilisée lors de la Seconde Guerre Mondiale. Il s'agit donc ici de comprendre son fonctionnement, ses mécanismes et de pouvoir reproduire son comportement de manière informatisée.

Ce compte-rendu détaille notre démarche pour aborder le sujet, le comprendre et enfin réaliser techniquement le projet en s'organisant au sein du trinôme, en utilisant des outils et les méthodes appris en cours.

Nous allons donc voir dans une première partie ce qu'est la machine Enigma, son rôle, son fonctionnement et nous nous intéresserons aussi à sa cryptanalyse par Alan Turing pour comprendre ses failles. Nous verrons ensuite comment nous nous sommes organisés pour mener le projet à bien en mettant toutes nos connaissances en œuvre et nous verrons aussi les difficultés nous avons rencontrés et comment nous les avons surmontées.

1. Enigma : Machine de cryptage

1.1. Création d'Enigma

Ecrire en 12 Times New Roman

Pour 1.1.1 remise en contexte :

Créateur ? Date ? Quelle méthode de cryptage existait avant? (voir méthode de vigenère) Quel était le but d'Enigma ?

Pour 1.1.2 Utilisation

Dire qu'il y avait des versions commerciales mais que nous on s'intéresse à la versio militaire.

Qui s'en servait ? Comment il s'en servait (il s'envoyait le code pour paramétrer la machine, les messages étaient codés en morse, les 2 machines devaient être identiques)? Ne pas parler du fonctionnement interne mais externe si tu veux

1.2. Fonctionnement d'Enigma

1.2.1. Les rotors

Nous allons à présent voir en détail comment Enigma fonctionne, quelles sont ses particularités qui l'ont rendu si difficile à « casser ».

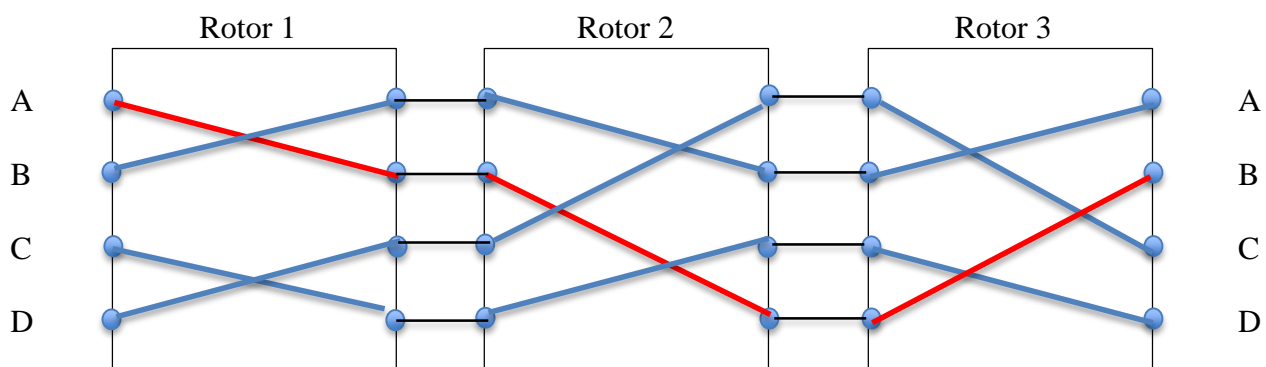
La puissance d'Enigma réside dans le fait qu'elle est imprévisible en terme de chiffrement des mots rentrés et cela est directement permis par sa structure interne. C'est la combinaison de ses composants qui la rend si performante mais quels sont-ils ? En tant que machine électromécanique, Enigma est composée de rotors fonctionnant avec des contacts électriques permettant la substitution poly-alphabétique, c'est d'ailleurs une de ses particularités.



Les rotors servent de connexion électriques, ils sont au nombre de 3 dans la machine et pouvaient être choisis parmi 5 rotors en tout. Leur particularité vient du fait qu'ils sont cylindriques et fixés sur un axe autour duquel ils peuvent tourner. A chaque lettre tapée, le rotor effectue une rotation qui change complètement la donne pour la lettre suivante car du coup, la permutation sera différente pour chaque lettre même si celle-ci est retapée.

C'est un des points forts d'Enigma car de ce fait, l'analyse par fréquence qui consiste à repérer les lettres qui reviennent fréquemment dans un message crypté est impossible. Mais comme signalé, il y a 3 rotors en tout. Chaque rotor représente les lettres/chiffres que l'on veut crypter. Par exemple, si on prend l'alphabet de 26 lettres, chaque rotor aura 26 positions. Le 1^{er} rotor tourne d'un cran à chaque fois qu'une lettre est tapée, de sa 26^e position à sa position initiale, il déclenche le 2^e rotor qui lui aussi tourne d'un cran. Lorsque le 2^e rotor effectue sa rotation de sa 26^e lettre à sa position initiale, il déclenche à son tour le 3^e rotor. Les rotors reviennent à leur position initiale lorsqu'ils ont tous parcouru leur 26 positions. Chaque rotor pouvait être positionné de 1 à 26 avant de commencer à taper le message.

Nous avons schématisé les 3 rotors avec 4 lettres selon un circuit électrique de cette manière :



On peut voir ici que si on tape lettre « A », elle sera permutée en « B » à l'issue du rotor 1 puis en « D » et enfin en « B ». Le rotor 1 tournera d'un cran et ses sorties seront décalées ce qui permettra d'avoir une lettre cryptée différente même si on retape à nouveau « A ».

Si on calcule le nombre de possibilités de cryptage qu'offrent les rotors, nous avons :

$$5 * 4 * 3 = 60$$

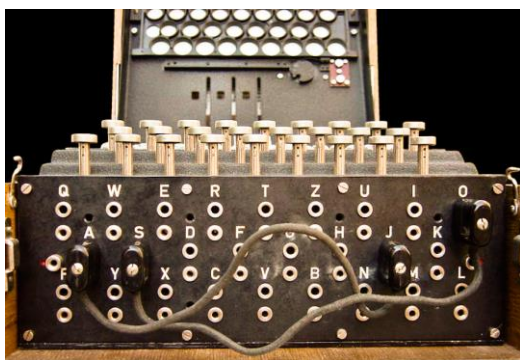
$$26^3 = 17\,576$$

$$\text{Soit : } 60 * 17\,576 = 1\,054\,560$$

Jusqu'ici nous avons donc 60 possibilités de choix de rotors (3 parmi 5) et 17 576 (pour un alphabet de 26 lettres) positions possibles des rotors car chaque rotor à 26 positions donc $26 \times 26 \times 26$ soit un total de 17 576 possibilités de cryptage. Si l'alphabet comporte plus de caractères, cela augmentera le nombre de positions possibles des rotors et donc le nombre de possibilités total.

1.2.2. Le plugboard

Mais Enigma ne s'arrête pas à ces trois rotors tournant, en effet, un des composants central de cette machine est le tableau de connexion (plugboard), situé devant la machine et permettant d'effectuer 10 paires de permutation. Le plugboard était spécialement conçu pour les machines utilisées à des fins militaires, les versions commerciales en étaient dépourvues.



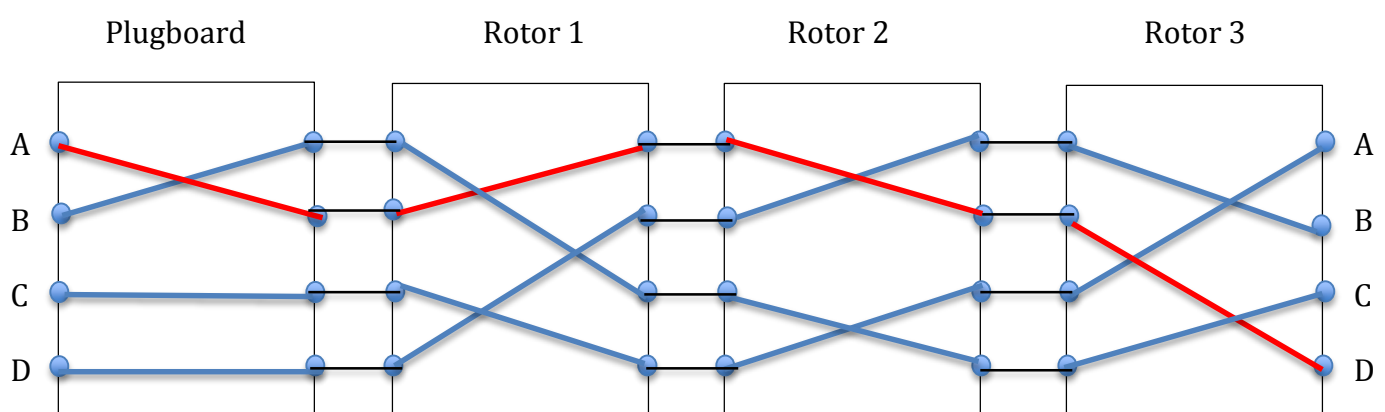
Les permutations sont commutatives c'est-à-dire que si la lettre « A » est reliée à la lettre « B » alors la lettre « A » sera permutée avec la lettre « B » mais la lettre « B » sera permutée avec la lettre « A ».

Il y a donc 20 lettres permutées et 6 inchangées dans un alphabet de 26 lettres.

Le but du plugboard était de brouiller les pistes car une lettre tapée était d'abord permutée suivant le tableau de connexion puis codée.

Concrètement, cela signifie que si « A » était permutée avec « B », c'est la lettre « B » qui serait effectivement cryptée en passant dans les rotors.

Nous avons donc enrichi notre schéma (cf rotors ci-dessus) avec le plugboard ce qui nous donne sur un alphabet de 4 lettres :



Dans ce cas-ci, les lettres « A » et « B » sont permutées alors que les lettres « C » et « D » restent inchangées. Si on veut crypter la lettre « A », c'est donc en fait la lettre « B » qui va être cryptée et on obtiendra donc en permutations successives: A->B, B->A, A->B, B->D.

Si nous calculons les possibilités qu'offrent les branchements frontaux pour un alphabet de 26 lettres nous avons :

$$\frac{26!}{6!10!2^{10}} = 150\,738\,274\,937\,250$$

En effet, nous avons 26 combinaisons de lettres (26x25x24...x1) ce qui justifie le « 26 ! ». Cependant, nous ne voulons faire que 10 paires de lettres donc 6 lettres restent non permutées. Etant donné que l'ordre des combinaisons n'importe pas, nous pouvons diviser par « 6 ! » et multiplier par le nombre de combinaison possible pour les 10 paires soit « 10 ! ». Enfin les paires sont constituées de deux lettres interchangeables, on peut alors diviser par « 2 » et comme il y a 10 paires de lettres cela nous donne « 2¹⁰ »

Le plugboard est donc un élément central car c'est lui qui augmente considérablement le nombre de possibilités.

Si on rajoute les possibilités qu'offraient les rotors nous avons :

$$150\,738\,274\,937\,250 * 1\,054\,560 = 158\,962\,555\,217\,826\,360\,000$$

Voilà donc le nombre de possibilités de cryptage que permettait Enigma grâce à la combinaison de ses composants.

1.2.3. Le réflecteur

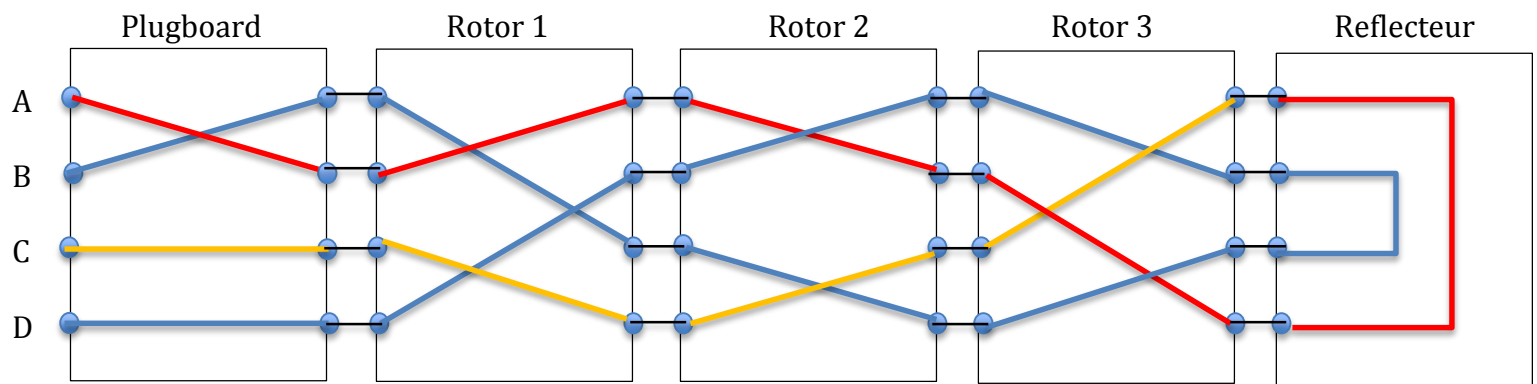
Cependant, à ce stade, la machine est capable de crypter mais pas de décrypter. Il faudrait donc une machine pour crypter et créer une seconde machine pour décrypter. Pour pallier à ce problème, Enigma s'est dotée d'un réflecteur, une sorte de rotor fixe qui a pour but de faire une ultime permutation et de renvoyer le courant dans l'autre sens. De ce fait, il inverse par paires les lettres et permet de lier les lettres de façon à pouvoir rendre le cryptage réversible. Concrètement, si on tape la lettre « A » et qu'elle est cryptée en « D » alors lorsqu'on décryptera la lettre « D » elle renverra « A » (la machine doit avoir les mêmes paramétrages pour le cryptage et le décryptage).






Il existait plusieurs versions de réflecteur. La version B était la plus couramment utilisée mais il a aussi existé les versions C et D. Certaines de ces versions permettant de positionner le réflecteur de différentes manières (différentes positions un peu comme les rotors).

Le principe même du réflecteur empêchait toute lettre d'être codée par elle-même ce qui sera largement exploité lors de la cryptanalyse de Turing que nous verrons dans la prochaine partie.

Nous pouvons enrichir une dernière fois notre schéma pour montrer une version complète du fonctionnement d'Enigma :



Légende : Cryptage : chemin aller 
chemin retour 
Decryptage : chemin aller
chemin retour 

Reprenons notre exemple pour crypter la lettre « A ».

ALLER

REFLECTEUR

RETOUR

Plugboard : « A » devient « B »

Réflecteur : « D » devient « A »

Rotor 3 : « A » devient « C »

Rotor 1 : « B » devient « A »

Rotor 2 : « C » devient « D »

Rotor 2 : « A » devient « B »

Rotor 1 : « D » devient « C »

Rotor 3 : « B » devient « D »

Plugboard : « C » reste « C »

« A » est donc cryptée en « C » et si on veut décrypter « C », on remonte le chemin retour jusqu'au début du chemin aller et on obtient bien « A ». Lorsque la lettre est passée par tous les composants, une pile électrique, alimentant Enigma, permet d'allumer la lampe correspond à la lettre cryptée/décryptée que l'on doit alors noter.

Le fonctionnement d'Enigma permet donc de faire une machine à crypter et décrypter très puissante, avec très peu de failles. Cependant, la seule faiblesse apparente qu'elle a (une lettre ne peut être cryptée par elle-même) va être le point de départ de la cryptanalyse d'Alan Turing.

1.3. Cryptanalyse par Alan Turing

Transition :

Après avoir terminé l'étude d'Enigma, son fonctionnement, sa manière de crypter et de décrypter, nous pouvons passer à la réalisation du projet c'est-à-dire simuler la machine Enigma en informatique.

2.Enigma : Simulation informatique

2.1. Organisation du projet

Avant de nous lancer dans le développement du projet, nous avons dû dans un premier temps nous organiser sur différents points.

2.1.1. Choix et contraintes

Nous avons dû effectuer certains choix avant de commencer à développer et même avant de pouvoir conceptualiser le projet.

Nous nous sommes tout d'abord posé la question du choix du langage de programmation. Celui-ci étant complètement libre avec pour seule obligation, avoir une interface graphique, nous pouvions partir sur du développement logiciel (C, C++, Java) mais aussi sur du développement web (html, css, javascript). Parmi les langages que nous connaissons, celui que nous avons le plus utilisé et où nous nous sentons le plus à l'aise reste le langage Java.



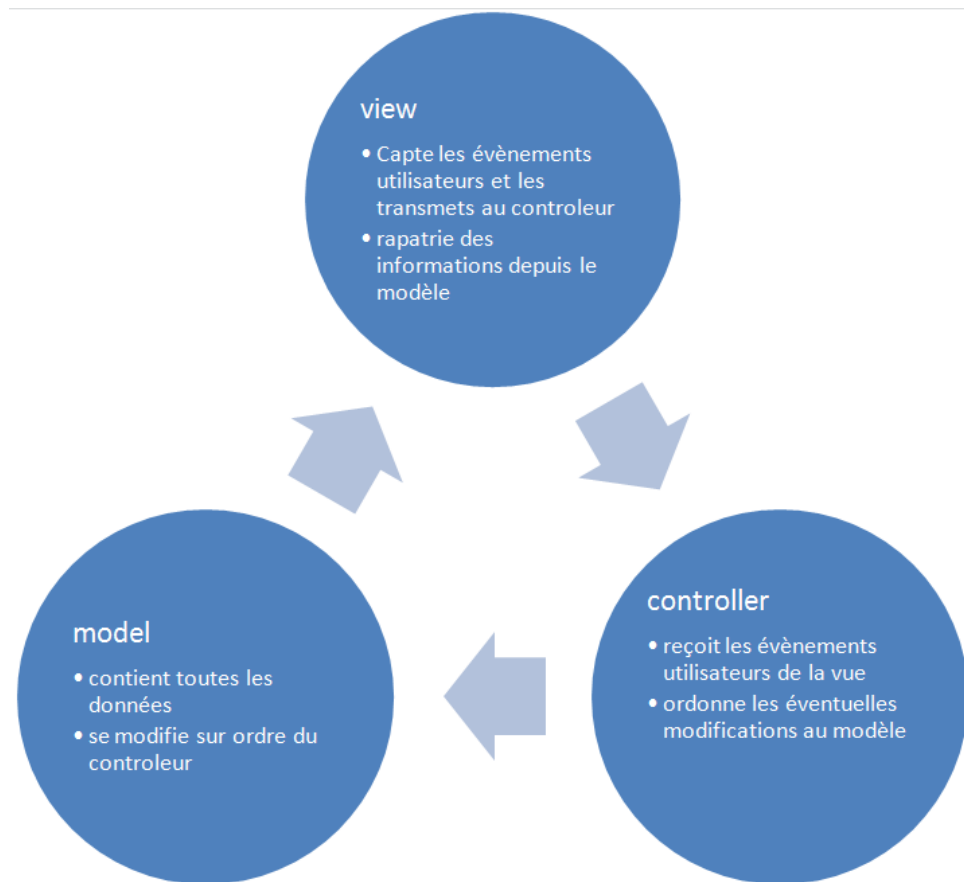
C'est donc le langage de programmation Java, langage orienté objet, que nous avons retenu étant donné que nous avons déjà eu un cours d'interface Homme Machine facilitant le développement de l'interface graphique et que nous sommes à l'aise avec la notion d'objet qui nous a paru cohérent avec les composants de la machine Enigma (nous détaillerons ce point dans une prochaine partie).

Nous avons eu, au semestre dernier, un cours sur les principaux design pattern que l'on peut utiliser afin d'avoir une bonne conception d'un projet permettant de décrire une solution standard à un problème de conception logiciel qui peut être par la suite réutilisé sans avoir à modifier le code existant (dans le cas de rajout de classes par exemples). Les design pattern permettent de respecter un certain nombre de bonnes pratiques permettant une conception optimisée et de ce fait un code clair et organisé.

Dans le cadre de notre projet, nous avons décidé d'utiliser le design pattern MVC (Modèle Vue Contrôleur) permettant de faire communiquer les différentes couches de notre projet de façon événementielle c'est-à-dire qu'une action de l'utilisateur via l'interface graphique (la vue) déclenche automatiquement un événement dans le contrôleur qui lui-même se chargera de notifier la couche modèle (c'est le cœur du projet, elle contient toutes les données principales et se charge des principaux calcul).

Afin de faciliter cette programmation événementielle, nous avons implémenté les classes observer et observable définie par défaut en Java et pour créer l'interface graphique, nous avons utilisé la bibliothèque Swing de Java. Nous avons ainsi défini l'architecture globale de notre projet et commencé la phase de conception.

Voici un petit schéma explicatif du design pattern MVC:



Source : <http://raphael-waeselynck.developpez.com/tutoriels/java/java-me/mvc/>

2.1.2. Outils de travail

Ces choix nous ont permis de définir les outils de travail que nous allons utiliser pour partager notre travail et communiquer efficacement.



Comme appris en cours de méthodologie, nous avons utilisé Github, une plateforme de développement collaboratif basée sur git, un outil de versionning. Cet outil permet de mettre en ligne notre projet et les évolutions apportées de façon à ce que tous les membres de l'équipe aient la dernière version du projet disponible et puisse rendre accessible instantanément toutes les modifications effectuées. Cela facilite grandement la création de projet et nous permet d'être très réactifs quant aux erreurs qui pourraient y avoir sur le projet pour les corriger rapidement et ne pas continuer de développer sur une mauvaise base.

Concernant le développement, nous avons décidé d'utiliser Eclipse du au choix du langage de programmation qui est un IDE (Integrated Development Environment) permettant de développer des logiciel en langage Java et de faciliter les tests fonctionnels. C'est un des IDE les plus utilisé pour le développement Java (avec netbeans) et c'est aussi celui que nous connaissons le mieux car nous l'avons déjà utilisé lors d'un projet ultérieur et lors des séances de programmation à l'IUT.



Nous avons aussi été amenés à utiliser le logiciel Mumble, un logiciel VoIP, afin de communiquer oralement en équipe lorsque nous ne pouvions pas nous voir à l'IUT. Cela nous a permis de communiquer efficacement, de réfléchir aux éventuelles difficultés et de proposer des solutions facilement sans avoir à attendre de se retrouver à l'IUT pour en parler.

Ces outils ont été une aide pour avoir une meilleure gestion de l'équipe mais celle-ci s'est aussi faite autour d'autres points que nous allons détailler.

2.1.3. Gestion de l'équipe

Dans un projet, il est important d'avoir une équipe qui sait communiquer, qui s'investi et il est aussi important de répartir les tâches à chaque membre afin d'optimiser le développement du projet.

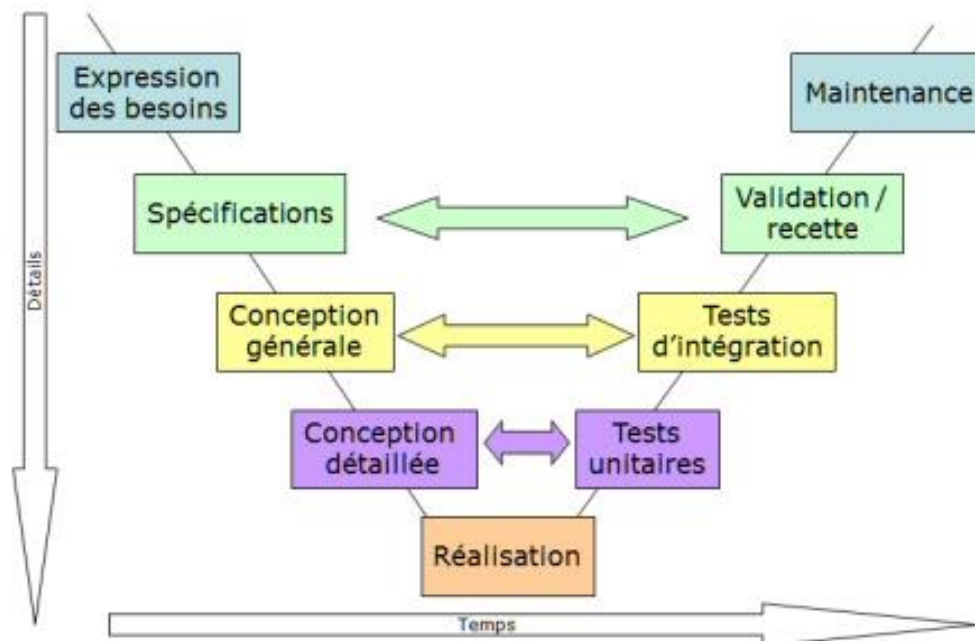
Dans un premier temps, il était important de se mettre au point sur les connaissances que nous avons acquises sur la machine Enigma lors de nos recherches afin d'avoir une vue générale et cohérente du projet par tous les membres de l'équipe. Pour ce faire, nous avons organisé des petites réunions où nous avons expliqué à chacun ce que nous avons compris ou ce que nous n'avions pas compris pour que l'on puisse lever les zones d'ombres restantes qui nuisent au développement du projet. Ces réunions ont été l'occasion de schématiser les problèmes et d'apporter des solutions de conception. Les avis sur les différentes possibilités de conception nous ont permis d'enrichir le projet et de réfléchir sur des solutions plus optimisée que ce qui

était proposé initialement. Cela nous a permis d'être d'un commun accord sur la façon dont nous allions procéder pour développer le projet. Ces petites réunions ont été la base d'un travail d'équipe solide et efficace.

Par la suite il a été nécessaire de diviser le projet en plusieurs tâches notamment dû au choix de conception que nous avons fait (architecture MVC) et qui nous a permis de développer indépendant et parallèlement chaque partie. De ce fait chaque membre de l'équipe s'est occupé d'une des couches de l'architecture 3-tiers c'est-à-dire : la partie métier, la partie interface et la partie contrôleur. Pour combler les dépendances entre ces couches nous avons mis en place un « Mock » qui permet de simuler une classe même si celle-ci n'est pas encore opérationnelle. Plus concrètement, il s'agit d'une classe simplifiée qui garde le squelette de la classe réelle mais qui renverra juste les valeurs attendues sans effectuer de calcul (contrairement à la classe réelle).

Afin de respecter les délais, et d'avoir un point de repère temporel sur le travail effectué, nous avons conçu un diagramme de Gantt prévisionnel nous permettant de mieux nous situer dans le temps et d'améliorer notre efficacité (voir annexe diagramme.pdf).

Nos méthodes de travail se sont apparentés aux méthodes agiles pour la gestion de projet et plus précisément Scrum du fait de nos réunions, de nos découpage de tâches mais aussi parce que lorsque nous avons terminés une micro-tâche (ex : une fonction complexe d'une classe), nous la testions et nous étions susceptible de modifier pour l'adapter. Plus précisément, nous n'avons pas suivi un cycle en V :



Source : <http://www.methodesagiles.info/Agilite.php>

Nous avons réalisé chaque fonctions puis effectués des tests fonctionnels et corriger/adapter au besoin. Cette méthode à l'avantage d'être moins rigide que le cycle en V et de permettre une évaluation quotidienne des fonctionnalités.

Grâce à cela, la gestion de l'équipe a été grandement facilitée et le travail d'équipe a été plus efficace ainsi que la mise en place du projet.

2.2. Représentation d'Enigma

Une part importante dans la simulation d'Enigma a été de modéliser la machine pour mettre en valeur les composants et les fonctionnalités primordiales de la machine.

2.2.1. Etude d'Enigma

Nous avons dû effectuer certains choix avant de commencer à développer et même avant de pouvoir conceptualiser le projet.

Annexes :

https://www.youtube.com/watch?v=G2_Q9FoD-oQ par « numberphile »

https://www.youtube.com/watch?v=7dpFeXV_hqs par « e-penser »

https://interstices.info/encart.jsp?id=c_9752&encart=0&size=790,700

<http://www.apprendre-en-ligne.net/crypto/Enigma/>