L I C E N C E I N F O R M A T I Q U E

Projet de fin d'études

Sujets, Dates et Consignes

Année 2017-2018

Encadrant: Catherine Recanati
catherine.recanati@lipn.univ-paris13.fr)

3 mai 2018

Table des matières

Chapitre 1

Consignes générales

1.1 Introduction

Ce document regroupe l'ensemble des informations nécessaires à la réalisation du projet de Licence. Vous y trouverez les directives générales, un guide pour la réalisation du projet, les dates limites de remise des documents qui servent de base pour la notation de votre travail, et la liste des sujets proposés cette année. Si vous souhaitez travailler sur un autre sujet, vous devez nous le proposer par mail (avec une description un peu détaillée) avant la fin mai et nous vous rpondrons si votre projet est accepté ou non.

Vous devez impérativement respecter les consignes et les dates. En particulier, vous devez commencer par proposer par mail un groupe de 3 à 4 personnes, en indiquant si certains membres sont encore en recherche de stage. Vous pourrez aussi donner la liste de vos trois sujets préférés mais cela n'entraine pas que l'un de ces sujets vous soit attribué. La constitution du groupe et l'acceptation du sujet vous seront indiquées par mail avant la fin de la semaine suivante. Mais attention, vous ne faites que des propositions et un autre sujet ou un autre groupe peuvent vous être attribués.

1.2 Dates importantes

Vous commencerez donc par former un groupe de 3 à 4 personnes et indiquerez vos trois sujets préférés parmi ceux figurant dans ce document. Si vous souhaitez travailler sur un sujet différent de ceux proposés, envoyez rapidement une page de description du sujet au format pdf à Catherine Recanati qui validera ou non votre proposition de sujet.

Si vous n'arrivez pas à constituer un groupe, envoyez tout de même un mail à Catherine Recanati pour vous inscrire en Projet. Précisez si vous êtes en attente de stage, et quels sont vos sujets favoris.

Remarque importante : tous vos messages (et documents) doivent être envoyés systématiquement à (`catherine.recanati@lipn.univ-paris13.fr`))et en copie à tous les membres de votre groupe.

Vous devez donc tous envoyer **avant le 28 mai 2018** un mail contenant la liste des membres du groupe que vous proposez (**et leurs adresses mail**), en précisant ceux qui sont en attente de stage. Indiquez aussi la liste des 3 sujets qui intéressent votre groupe (dans l'ordre de vos préférences).

Vous serez informés dans la semaine suivante. par e-mail de

— votre numéro de groupe

— des membres du groupe et du sujet qui devra être traité par votre groupe.

La deuxième date butoir est celle de l'envoi de votre **premier rapport**. C'est **le 8 juin 2018**. Vous serez notés sur la qualité de ce premier document de spécification logicielle, et il est très important que vous suiviez scrupuleusement les consignes de travail et de rédaction figurant dans les deux prochains chapitres. Vous enverrez ce rapport de 15 pages maximum **au format pdf uniquement**.

Ce premier rapport inclura :

— une présentation générale et une analyse approfondie de votre projet (cf. 3.2)
— une liste détaillée des fonctionnalités de votre logiciel (cf. 3.2.2)
— une description de l'architecture logicielle fournie sous la forme d'une liste des différents modules ou packages prévus, et du schéma ou diagramme de dépendances entre ces modules. Sur ce schéma les modules seront représentés par des boîtes rectangulaires, et les appels (ou dépendances) entre modules seront représentés par des flèches (cf. 3.3.1)
— une liste des tâches du projet et un planning prévisionnel de répartition des tâches dans le temps (diagramme de Gantt, cf. 3.4.2)

Notez bien qu'aucune programmation n'est demandée à ce stade (il ne s'agit que de spécifications logicielles) **et il est très important que vous respectiez la date de remise du document.**

Vous aurez des retours sur votre rapport par mail. Il vous faudra alors le corriger en tenant compte des remarques qui vous auront été faites et votre note finale tiendra compte du soin que vous aurez apporté à ces corrections car le rapport corrigé fera partie de votre mémoire final.

Les soutenances auront lieu le **mardi 19 et/ou le mercredi 20 juin**. Vous ferez une présentation collective de 10 minutes (basée sur un fichier PowerPoint, OpenOffice ou pdf) et vous ferez l'après-midi ou le lendemain une démonstration de votre logiciel en salle machine, sur les machines de Galilée ou sur vos propres ordinateurs.
NB : Le fichier de la présentation orale (format .pdf, .ppt, .pptx ou .odp) sera envoyé par mail **deux jours avant la soutenance**, i.e. au plus tard le dimanche soir, pour que l'on puisse l'installer par avance sur un même ordinateur. Par ailleurs, vous devez préparer soigneusement le scénario de votre démonstration et le répéter dans l'environnement prévu pour la démo.

La semaine qui précéde celle des soutenances (semaine du 11 juin), vous devrez envoyer par mail :

— le mémoire final d'au maximum 25 pages hors annexes (format pdf uniquement) décrivant l'ensemble de votre projet (vous compléterez le premier rapport)
— le manuel utilisateur de votre logiciel (comportant aussi un manuel d'installation) au format pdf également

et avant le début de semaine suivante :

— votre présentation orale
— une archive `zip` ou tar.gz contenant le code commenté de vos programmes dans leur état actuel (tout autre format sera refusé - excepté une archive jar pour le code java)

NB : Le code pourra être modifié jusqu'au jour des soutenances mais à vos risques et périls (conservez toujours une version qui fonctionne).

1.3 Aide

Vous pouvez naturellement envoyer des messages électroniques pour avoir des éclaircissements sur votre sujet, une aide sur la conception ou sur un problème technique.

Chapitre 2

Note finale et livrables

L'évaluation du projet est faite au vu de :
- l'organisation du travail en groupe et le respect des délais
- le premier rapport
- la qualité du mémoire : corrections effectuées après les remarques des encadrants sur le premier rapport, manuel utilisateur et précision du manuel d'installation
- la soutenance orale finale
- la démonstration finale du logiciel
- la qualité technique de la réalisation (en particulier l'adéquation du logiciel au problème posé, les fonctionnalités proposées et la convivialité de l'interface utilisateur)
- la quantité de travail fournie (relativement au nombre de participants)

et enfin,
- la difficulté intrinsèque du sujet lui-même.

Les documents livrables seront envoyés par mail à Catherine Recanati, et en copie à tous les membres du groupe. Les textes seront au format pdf, et pour le code on enverra un fichier d'archive (format zip ou tar.gz uniquement).

La présentation orale finale sera livrée au format PowerPoint, OpenOffice ou pdf, au plus tard deux jours avant la soutenance.

Pour faciliter la recherche et classification de vos envois par mail, votre premier mail proposant un groupe et des sujets sera intitulé **[Projet L3] Choix sujet**. Ensuite, vous vous verrez attribuer un sujet et un numéro de groupe. Si vous êtes le groupe G12, vous utiliserez ensuite le préfixe **[G12]** pour tous vos envois. En particulier, les différents livrables demandés seront envoyés avec comme titres

[G12] Rapport, pour le premier rapport,

[G12] Mémoire, pour le mémoire final,

[G12] Soutenance, pour le fichier de présentation orale,

[G12] Code, etc.

2.1 Le premier rapport

Soignez en particulier la première page, et faites une table des matières paginée. La première page doit comporter :
- un titre (mentionnant le sujet traité)
- le nom des auteurs (et leurs courriers électroniques)
- la formation et l'année

— la date à laquelle cette version du rapport a été terminée et un numéro de version
— le nom de l'encadrant auquel le rapport est destiné

2.1.1 Plan du rapport

Le premier rapport doit comporter les chapitres suivants :

1. Présentation générale du sujet (on doit prévoir une introduction, et une présentation du domaine de l'application ; s'il s'agit d'un jeu, expliquez les rèlles du jeu par exemple).
2. Analyse approfondie, afin de décrire entièrement le projet à réaliser. On y précisera les problèmes théoriques ou pratiques sous-jacents à l'énoncé du sujet, le type d'utilisateur prévu, les principales fonctionnalités demandées ou nécessaires ainsi que la description des formats d'entrées/sorties s'il y a lieu ; les contraintes explicites ou non, l'environnement matériel et logiciel du projet doivent aussi être détaillés, et les choix effectués doivent être présentés avec leur justification (cf. 3.2).
3. La liste **complète** des fonctionnalités du logiciel (cf. 3.2.2).
4. La conception architecturale du logiciel (cf. 3.3.1). Le logiciel sera découpé en modules. Cette partie du rapport comprend au minimum **la liste complète de tous les modules, et le graphe des dépendances entre les différents modules**. Sur ce graphe les modules seront représentés par des boîtes rectangulaires, et les appels (ou dépendances) entre modules seront représentés par des flèches.
5. Planning. On fournira la liste des tâches et un planning prévisionnel représentant ces tâches en parallèle dans le temps (diagramme de Gantt). Remarque : à chaque module correspond déjà une tâche de conception et une tâche d'implémentation, mais il faut aussi prévoir les tâches de rédaction de documents, d'apprentissage de langage ou de librairie, mais aussi les tâches d'intégration de modules, et celles de debuggages ou de tests.

Vous recevrez des retours critiques sur ce premier rapport. Il vous faudra en tenir compte et le corriger avant de l'inclure dans le mémoire final du projet. Nous tiendrons compte positivement du fait que vous aurez corrigé chaque point que nous aurons mentionné comme problématique. A l'inverse, si vous ne tenez pas compte de nos remarques nous vous enlèverons des points.

2.2 Le mémoire du projet

Le mémoire final du projet reprend les chapitres du premier rapport en y ajoutant des compléments (conception détaillee, répartition des tâches entre les différents membres du groupe, planning prévisionnel initial et planning détaillé effectif), auquel s'ajoute un manuel utilisateur et une conclusion (bilan du projet).

Toutes les pages doivent être numérotées et le mémoire doit comporter une table des matières. La première page comportera :
— un titre (mentionnant le sujet traité)
— le numéro de votre groupe, par exemple G3
— le nom de(s) (l')auteur(s) et leur(s) courrier(s) électronique(s)
— la formation et l'année
— le nom de l'encadrants auquel le document est destiné
— la date à laquelle cette version du mémoire a été imprimée

2.2.1 Plan du mémoire

Le mémoire proprement dit doit comporter les chapitres suivants :

1. Une présentation générale du sujet
2. L'analyse approfondie
3. La liste complète des fonctionnalités
4. La conception architecturale (liste et description des modules et diagramme de dépendances)
5. La conception détaillée. Vous pourrez fournir par exemple
   — l'esquisse des algorithmes cruciaux
   — la liste des en-têtes de procédures ou fonctions classées par module,
   — la liste des données partagées et leurs structures
   — la liste des fichiers comportant les données ou les modules de programmes
   — si vous programmez en java, un diagramme de classe et éventuellement la javadoc du code des principaux modules en annexe
   — de manière générale, toute spécification logicielle (par exemple en format UML) que vous auriez utilisée. Mais rien ne vous est imposé car certains d'entre vous n'ont pas eu de formation au Génie Logiciel
6. Le planning prévisionnel initial détaillé et le planning effectif
7. Un manuel utilisateur qui inclura un manuel d'installation
8. Une conclusion (bilan technique et humain du projet)
9. Le cas échéant, en annexe, les fichiers de données avec lesquels vous avez testé votre logiciel
10. Une liste de références bibliographiques et de sites internet consultés.

Les références bibliographiques sont les ouvrages (livres, articles, ...) que vous avez utilisés pour l'analyse du problème et la réalisation de votre projet. Ne faites apparaître que les ouvrages que vous avez directement consultés. De même, ne mentionnez que les sites qui vous ont été utiles.

2.2.2 Planning prévisionnel et planning effectif

Le planning fourni initialement sera présenté sur un diagramme de Gantt et on listera les tâches en indiquant les membres du groupes en charge de chacune. On fournira aussi le planning qui aura finalement été réalisé (planning effectif) en indiquant au besoin les raisons ayant conduit à sa modification.

2.2.3 Le manuel utilisateur et le manuel d'installation

Le manuel utilisateur doit contenir un manuel d'installation du logiciel.

Le manuel d'installation doit fournir toutes les informations nécessaires pour installer le logiciel. En particulier il doit préciser l'environnement minimal requis pour pouvoir procéder à l'installation du logiciel et indiquer

— la liste commentée des fichiers nécessaires pour exécuter le logiciel (librairies, utilitaires, programmes sources, exécutables) ; ne pas oublier de préciser les numéros de version des systèmes ou compilateurs ; tout l'environnement nécessaire à l'exécution doit être décrit et on donnera éventuellement les adresses internet où le trouver
— les fichiers constituant le logiciel et la façon de les installer sur sa machine. (On donnera par exemple un fichier makefile et on dira précisément quelle commande make doit être tapée pour lancer son exécution).

Faites tester l'installation par une personne d'un autre groupe pour vérifier que vous n'avez rien oublié.

Le manuel utilisateur proprement dit décrit comment lancer le logiciel et comment l'utiliser. Il doit lister toutes les fonctionnalités accessibles, et les contraintes devant être respectées. Il doit indiquer comment se déroule le dialogue avec l'utilisateur (éventuellement, donner les écrans de l'interface utilisateur) et préciser ce qui se passe en cas d'erreur (liste des messages d'erreur par exemple).

Une bonne manière de procéder pour cela est d'écrire un premier chapître présentant la session typique d'un utilisateur. Ce premier chapitre présentera de manière naturelle le "parcours" d'un utilisateur lors d'une première session du logiciel. Ensuite, on peut décrire de manière plus systématique les différentes fontionnalités dans les chapitres suivants.

2.3 La soutenance orale

Vous devez faire le bilan du projet dans une présentation PowerPoint (ou OpenOffice, ou un fichier de transparents au format pdf).

Votre présentation ne doit pas durer plus de 10 minutes (il y aura 5 minutes supplémentaires pour des questions). Cette présentation doit être collective et vous vous passerez tour à tour la parole. Vous commencerez bien sûr par présenter le sujet. Vous pouvez ensuite reprendre certaines spécifications du premier rapport comme la liste des fonctionnalités proposées ou des schémas de spécifications fonctionnelles, ainsi que l'architecture globale du logiciel (par exemple le diagramme de dépendances entre les modules). Faites aussi un bilan en indiquant brièvement ce qui a été finalement développé (mais ne développez pas plus la présentation du logiciel et l'interface utilisateur, car ce sera fait dans la démonstration de l'après-midi).

Vous pouvez aussi parler de l'organisation du travail (répartition des tâches et planning). Enfin, vous pouvez indiquer les problèmes rencontrés et dire comment ils ont pu être résolus pour mettre en avant votre travail, et indiquer les possibilités d'extension ou d'amélioration du logiciel. Vous conclurez par un bilan (collectif ou individuel) sur ce que vous aura apporté le projet sur le plan technique et humain.

2.4 La démonstration du logiciel

La démonstration ne doit pas excéder 13 minutes, et il y aura 7 minutes de questions. C'est lors de la démonstration que la qualité technique de la réalisation sera évaluée. Il faut donc préparer et répéter la démonstration soigneusement.

Pour cela :

1. Définir des objectifs : ce que l'on veut montrer (si le code n'est pas complètement intégré vous pouvez néanmoins montrer le fonctionnement correct de certaines fonctions ou modules).
2. Préparer un scénario d'utilisation permettant de satisfaire ces objectifs. Cela peut nécessiter l'utilisation de fichiers préparés par avance.
3. Répéter et minuter votre démonstration en situation réelle, c'est-à-dire dans la configuration matérielle et logicielle qui sera celle de la véritable démonstration (en particulier, si vous avez réalisé le projet chez vous, il est très probable que vous devrez y apporter des modifications pour qu'il fonctionne correctement à l'Institut Galilée). Préparer aussi vos commentaires oraux.

Vous devez mettre en valeur ce que vous avez fait, et, seulement à la fin, expliquer les erreurs résiduelles et les fonctionnalités non encore opérationnelles. Les

problèmes de gestion de groupe que vous pouvez avoir rencontrés ne doivent pas apparaître lors de la démonstration (ils doivent avoir été évoqués bien avant, au moment où ils se sont produits, avec vos encadrants).

Chapitre 3

Guide de réalisation des projets

3.1 Préliminaires

3.1.1 Formation du groupe

Vous devez vous préoccuper de former un groupe dès le début du mois de mai. Une fois votre groupe formé vous pourrez discuter des sujets qui vous attirent le plus. Mais avant toute chose, il faut que vous échangiez vos adresses mails et vos numéros de téléphone personnels pour pouvoir vous joindre en cas d'urgence.

Vous pourrez éventuellement vous attribuer des rôles au sein du groupe (comme chef de Projet, chargé du planning, chargé de rédaction de documentation, chargé de l'interface graphique, etc.). Mais même si quelqu'un est chargé de la rédaction d'un document, il est très important que tous les membres du groupe relisent le document. Vos documents devront donc circuler entre vous par mail avant d'être envoyés à Catherine Recanati.

3.1.2 Spécification du logiciel

L'écriture d'un programme sur machine doit être impérativement précédée de l'analyse du problème et de la conception (papier) du logiciel. C'est la raison pour laquelle nous vous demandons de nous remettre un premier rapport avant de vous mettre à coder. Ce rapport nous permet de vous faire des remarques, et vous le corrigerez avant de vous mettre à la conception détaillée puis au codage. La version corrigée sera ensuite incorporée au mémoire que vous nous remettrez en fin de projet.

Votre projet devra être réalisé en fonction des besoins et caractéristiques des futurs utilisateurs du logiciel. Vous devez concevoir le logiciel afin qu'il soit facilement utilisable pour les personnes auxquelles il est destiné. Ainsi il peut s'agir d'un enfant ou d'un adulte, de quelqu'un qui ne connait rien à l'informatique ou au domaine exploré, ou au contraire d'un expert du domaine ou d'un expert en informatique. Il est donc important de savoir à qui le logiciel est destiné, car cela détermine la forme de l'interface utilisateur et les fonctionnalités proposées.

On doit déterminer quelle est la tâche effectuée par cet utilisateur potentiel. Le but du logiciel, c'est ce à quoi il sert, ce qu'il permet de faire, sa ou ses fonctions principales. Les ergonomes distinguent ici deux niveaux de description : celui de la tâche que permet d'accomplir le logiciel (par exemple, écrire une lettre pour un

traitement de texte, jouer aux échecs pour un jeu d'échecs, gérer une bibliothèque, etc.), et celui des fonctionnalités proposées par le logiciel pour permettre d'accomplir la tâche (par exemple, mettre en gras, justifier à droite, définir des marges, etc. pour un logiciel de traitement de texte, ou se connecter en réseau, jouer une partie, jouer un coup, pour un jeu d'échecs en réseau). Le niveau de la tâche est assez général et orienté "utilisateur", et celui des fonctionnalités est plus précis et détaillé. Notez cependant que les fonctionnalités sont indépendantes de l'interface utilisateur (qui peut tout autant être graphique que textuelle).

Pour bien fixer tout cela, nous vous demandons de nous rendre dans le premier rapport une présentation et une analyse approfondie de votre projet, ainsi que la liste détaillée des fonctionnalités du logiciel. La section qui suit vous donne des indications sur le contenu de cette première partie. La section suivante concerne la conception, et vous aidera à rédiger la seconde partie concernant l'architecture logicielle et la planification du projet.

3.2 Analyse approfondie et spécifications fonctionnelles

Cette étape de la description du projet doit permettre de déterminer ce que le logiciel à développer devra permettre de faire (le quoi ?). A cette étape, on ne se préoccupe pas de savoir comment le logiciel permettra de réaliser ces fonctionnalités (on ne se préoccupe donc pas de l'interface utilisateur), ni bien entendu de comment ces fonctionnalités seront programmées. Cela signifie, en particulier, que cette partie ne ne doit pas dépendre ou mentionner des aspects d'implantation comme les caractéristiques d'un langage de programmation ou d'une machine. On ne se préoccupe à cette étape que des services que le logiciel doit rendre à l'utilisateur. En principe, c'est seulement après cette première étape que le choix du langage de programmation et de l'environnement logiciel et matériel est effectué.

3.2.1 Analyse approfondie et analyse des besoins

L'analyse approfondie est consacrée à l'explicitation du contexte d'utilisation du logiciel et des besoins des utilisateurs. C'est la raison pour laquelle on parle d'analyse des besoins. Cette analyse répondra aux questions suivantes :

Quel est l'objectif du logiciel : à quoi doit-il servir, quel est le problème qu'il doit résoudre ? L'analyse commence par exposer ce sur quoi porte le logiciel, et ce qu'il permet de faire (la ou les tâches qu'il permet d'exécuter). En particulier, il faudra introduire les objets (ou entités abstraites) faisant partie du domaine sur lequel porte le logiciel (par exemple, des fichiers, des utilisateurs, et des arborescences de répertoires pour un gestionnaire de fichiers ; des pièces, des joueurs, des parties, des coups et des plateaux pour un jeu d'échecs ; des vins, des bouteilles et des étagères pour une cave à vin ; des expressions algébriques pour un logiciel de calcul formel, etc.). Il est important ici d'introduire et de fixer le vocabulaire lié au domaine.

Recommandation : pour décrire ce que fait votre logiciel, vous pourrez partir de l'énoncé du sujet et d'un premier texte (de vous) décrivant le logiciel que vous souhaitez faire. Soulignez-y les mots qui vous semblent importants (noms et verbes) et essayez d'isoler de cette manière les entités (objets ou concepts clés) qui font partie de ce petit monde (=le domaine de l'application). Un programme est toujours la simulation d'un petit univers. Vous devez dans cette première phase de conception, lister les objets faisant partie de cet univers en vous aidant des termes utilisés pour les désigner (la terminologie du domaine

de l'application). Cela vous permet de mieux cerner ce que fait votre logiciel et vous permettra plus tard de choisir les structures de données ou les objets qui feront partie de votre programme. Les entités de votre domaine correspondront normalement à des noms de votre texte, et les actions pouvant être exécutées correspondront à des verbes ou participes passés. Une fois cette analyse de texte effectuée, vous pourrez réécrire un texte de présentation du logiciel plus clair et plus précis, en fixant le vocabulaire du domaine de l'application.

A quel type d'utilisateur s'adresse le logiciel ? Quel est son âge et son niveau de compétence dans le domaine couvert par le logiciel ? Ces informations seront essentielles pour la conception de l'interface utilisateur et le choix des fonctionnalités qui seront offertes.

Quel est le contexte d'utilisation du logiciel ? A quelles occasions, l'utilisateur se servira-t-il du logiciel ? Quelles sont les différents types d'utilisation du logiciel ? A titre d'exemple, il arrive souvent qu'un logiciel nécessite une phase d'initialisation ou de configuration durant laquelle seules certaines fonctionnalités seront utilisées (et ces fonctionnalités ne seront que très rarement utilisées dans le contexte courant). Souvent, les différents contextes d'utilisation correspondent à des types d'utilisateurs différents, par exemple, des utilisateurs novices ou des utilisateurs expérimentés.

Déterminer précisément les entrées et les sorties du logiciel : l'utilisateur devra parfois fournir des données au logiciel (les entrées) pour qu'il puisse accomplir sa tâche. Vous devez déterminer sous quelle forme l'utilisateur disposera de ces données afin de choisir la procédure d'entrée des données la plus adaptée. De la même façon, vous devez déterminer quelle est la manière la plus appropriée pour présenter les résultats du programme (les sorties) à l'utilisateur, cela dépendra en particulier de l'utilisation que fera l'utilisateur des résultats produits par le logiciel.

Caractéristiques quantitatives : vous devez déterminer la taille et la quantité de données que l'utilisateur pourra traiter avec le logiciel. Indiquer aussi s'il existe des limitations.

3.2.2 Liste des fonctionnalités du logiciel

En prenant en compte les points précédents, vous devez déterminer précisément ce que le programme permet de faire, c'est-à-dire établir la liste exhaustive de ses fonctionnalités. C'est la partie la plus importante de cette étape, et c'est celle qui nécessitera le plus gros travail de rédaction. Chaque fonctionnalité doit être décrite sans pour autant entrer dans des détails d'implantation ou de configuration matérielle. Il ne faut pas non plus expliciter comment les fonctionnalités seront proposées à l'utilisateur par l'interface utilisateur. Pour bien décrire les fonctionnalités, il faut au contraire faire abstraction de l'interface utilisateur. Une bonne manière pour vous de faire abstraction de l'interface utilisateur est d'imaginer que le logiciel n'a pas d'interface graphique et ne permet d'interagir avec l'utilisateur que dans une fenêtre de type console dans laquelle il pourrait taper des commandes. On essayera aussi de présenter la liste des fonctionnalités du logiciel en les regroupant par sections thématiques. Ainsi par exemple, on regroupera ensemble les fonctionnalités de sauvegarde (enregistrer sous, enregistrer, etc.) et de chargement dans une même section, les fonctionnalités de connexion/déconnexions ou inscription d'un utilisateur, etc.

Pour chaque fonctionnalité, il faudrait idéalement aussi prévoir une procédure de test. Il s'agit ici de préparer la phase de test en décrivant des procédures et éventuellement des données qui permettront, une fois la programmation effectuée, de vérifier que la fonctionnalité est correctement implantée. (Ces tests pourront

aussi être utilisés lors de la démonstration finale pour la soutenance du projet). Ici, vous pouvez dresser un tableau de validation indiquant pour chaque fonctionnalité les valeurs problématiques et l'état du test de validation (à faire, en cours, ou validé). Ce tableau de validation pourra étoffer la conception détaillée mais il ne fera pas partie du premier document remis.

3.3 Conception

L'activité de conception consiste à enrichir la description du logiciel de détails d'implantation afin d'aboutir à une description proche d'un programme. Elle se déroule en deux étapes : l'étape de conception architecturale et l'étape de conception détaillée. Cette seconde étape peut éventuellement donner lieu à une révision de la première.

3.3.1 Conception architecturale

L'étape de conception architecturale a pour but de décomposer le logiciel en composants simples et indépendants les uns des autres - les modules - afin de simplifier l'activité de programmation qui suivra. Cette décomposition doit être faite de sorte que chaque module puisse être réalisé de manière totalement indépendante des autres, et par des programmeurs différents. La personne qui, à l'étape du codage, programmera un module donné doit pouvoir effectuer sa tâche sans avoir besoin de savoir comment les autres modules ont été, sont ou seront programmés.

C'est aussi à cette étape que l'on doit faire les choix contraignant l'implantation :

Configuration matérielle : sur quel type de machine fonctionnera le logiciel ? Votre logiciel nécessite-t-il la présence de périphériques particuliers (écran graphique, souris, imprimante, scanner, lecteur de CD-ROM, etc.) ?

Configuration logicielle : système d'exploitation, compilateur, bibliothèques de fonctions. (Pour chacun de ces logiciels, vous devrez préciser le numéro de version).

Réutilisation de logiciels existants : Dans le cas particulier où vous faites une nouvelle version d'un logiciel déjà opérationnel, il faut choisir entre reprendre et modifier le logiciel ou le refaire complètement.

Choix des langages de programmation correspondant à chaque module : il se peut que l'on programme l'interface graphique dans un langage différent de celui avec lequel on dénveloppe l'application proprement dite, ou qu'on ait à utiliser une architecture particulière imposant l'usage de langages particuliers (html, javascript, etc.).

Choix de l'environnement de développement : il faut déterminer l'ensemble des logiciels que vous utiliserez pour la programmation (par exemple gcc, java ou eclipse). Il comprend, au minimum, un compilateur, un éditeur de texte et un débogueur. Ces logiciels peuvent ou non être accessibles à partir d'une seule application.

Pour réaliser le découpage en modules, on part des fonctionnalités qui ont été définies précédemment. Si on a déjà regroupé les fonctionnalités en sections, chaque section donnera probablement lieu à un module. Pour chaque grande fonctionnalité, on définit précisément son interface avec le reste du logiciel, c'est-à-dire l'ensemble des informations dont elle aura besoin pour effectuer sa tâche, et celles qu'elle pourra modifier. Cela permettra de faire apparaître les dépendances entre les modules.

Un autre but important de cette décomposition est de mettre en évidence des modules possédant des caractéristiques suffisamment proches pour être fusionnés en un même module plus général (cette factorisation réduira d'autant le travail de programmation). Dans le même ordre d'idée, lorsque des bibliothéques logicielles sont disponibles, le découpage en modules devra, autant que possible, permettre la réutilisation de modules déjà existant dans ces bibliothèques.

Lors de ce travail, vous devez donner un nom le plus explicite possible à chaque module et aux données (ou modules) qu'il utilise ou produit. Notez qu'il y a aujourd'hui des modules relativement classiques, comme un module d'Interface Graphique Utilisateur si vous avez une interface utilisateur clavier/souris ; un module Gestion de Base de Données (qui permettra de faire des requêtes sur une BD) ; un module Connexion/Déconnexion pour un site web ou une application qui requiert une insciption ou un login ; un module Sauvegarde/Enregistrement pour gérer la relation aux système de fichiers.

En même temps que ce découpage, vous devez faire apparaître les dépendances entre modules : quel module utilise quel autre module ? Le résultat de cette analyse sera exprimé sous la forme d'un graphe de dépendances : chaque module est représenté par une boîte rectangulaire, et les dépendances sont représentées par des flèches orientées entre ces boîtes.

Pour décrire l'architecture du programme, on fournira donc dans le premier rapport un découpage en modules (en choisissant soigneusement leurs noms). On donnera une liste des modules avec une brève description, et on les représentera ensuite par des boites rectangulaires sur un graphe de dépendances.

Remarque : Il se peut que le logiciel soit constitué de plusieurs programmes, et qu'il existe une architecture particulière si ce dernier utilise plusieurs composants logiciels, comme par exemple, une architecture client/serveur. Mais on ne fait pas en réalité de distinction entre un programme et un module, et on représentera cette architecture globale sur le même diagramme en indiquant les relations de dépendances entre les différents programmes ou modules par des flèches.

On aura en particulier un découpage "classique" si le logiciel possède une interface graphique, ou s'il utilise une base de données. Dans le cas où l'on a une interface graphique, on prévoira un module destiné à l'interface graphique. En effet le monde de l'interface est un monde à part de celui du domaine du logiciel. Il est constitué d'éléments comme des boutons ou des menus, des boîtes de dialogue, etc. La construction et l'agencement de ces éléments interactifs sera donc effectuée dans le module de l'interface graphique, et ce dernier sera relié au reste de l'application par des appels de fonctions, allant parfois dans les deux sens.

De même, si l'on utilise une base de données, on regroupera dans un même module tout ce qui concerne la gestion de cette base de données (consultation, enregistrement, etc.). Ce module BD interagira alors avec le reste de l'application en permettant l'interrogation et la modification de la base de données. (Si on ne veut pas que ce module BD interagisse directement avec le reste de l'application, on pourra définir un module supplémentaire d'interface logicielle, chargé uniquement de la communication entre la BD et l'application proprement dite).

3.3.2 Conception de l'interface utilisateur

Bien que la description de l'interface utilisateur ne figure pas dans le premier rapport, vous devrez produire, **avant de vous lancer dans la conception détaillée et le codage**, les documents suivants :

Description d'une version préliminaire de l'interface : les différents écrans de l'interface utilisateur doivent être précisés, ainsi que la manière dont ils

s'enchaînent en fonction des choix de l'utilisateur. Cette description peut rester relativement informelle (sous forme papier-brouillon mais relativement exhaustive).

Rédaction d'une version préliminaire du manuel utilisateur : les fonctionnalités doivent être suffisamment bien décrites pour prévoir précisément comment le logiciel pourra être utilisé. Le manuel utilisateur ne vous est pas demandé initialement, mais vous devrez le fournir dans la documentation finale. Il sera rédigé du point de vue de l'utilisateur, et sera constitué en partie de la description des différents écrans. Vous pouvez bien sûr l'écrire avant de passer au codage, et il servira de document de spécification pour les programmeurs de l'interface.

Les différentes fonctionnalités du logiciel sont accessibles via l'interface utilisateur. Notez bien qu'une même fonctionnalité peut être accessible de différrentes manières : par exemple, par l'intermédiaire d'un menu et par l'intermédiaire d'un bouton, ou encore, via une combinaison de touches particulières (raccourci de commande) et par l'intermédiaire d'une barre d'outils. En outre, l'interface utilisateur n'est pas nécessairement une interface graphique avec clavier et souris. Néanmoins dans la plupart des cas, on fera une version préliminaire de l'interface en dessinant les différents écrans devant lesquels pourra se trouver l'utilisateur.

Pour la conception elle-même de l'interface, suivez soigneusement les recommandations qui figurent dans les sous-sections suivantes et en particulier celles de la section 3.3.4.

3.3.3 Distinction Interface/Application

Avant de réfléchir à l'interface, il convient de bien cerner cette notion. Pour l'organisation du code, on fait toujours, dit-on, une séparation nette entre l'interface et l'application. La justification, outre les avantages de la modularité, est qu'il existe différents systèmes de fenêtrages (Macintosh, X Window et Windows). L'application doit donc être écrite de manière indépendante pour que seul le code de l'interface ait besoin d'être réécrit s'il faut porter l'application dans un autre environnement.

La partie application, c'est le coeur du logiciel, ce à quoi il sert, ce qu'il permet de faire, sa ou ses fonctions principales. Les fonctionnalités proposées pour accomplir les tâches de l'utilisateur constituent finalement l'essentiel de l'application, et le travail de l'informaticien est d'abord de les définir, puis de les implémenter. Mais il n'est pas toujours facile de les distinguer de l'interface. L'interface guide l'utilisateur dans la mise en oeuvre des fonctionnalités pour l'aider à réaliser sa tâche. L'interface ne consiste donc pas uniquement en un dispositif permettant la mise en oeuvre des fonctionnalités. Des notions liées à la tâche ou à l'interaction entre l'homme et la machine peuvent y apparaître, en particulier si le logiciel est graphique (comme un logiciel de dessin).

La définition de l'interface passe par l'utilisation des dispositifs d'entrées/sorties. Ainsi, les interfaces utilisateurs ne sont pas nécessairement des interfaces graphiques utilisant un clavier et une souris. Les premiers ordinateurs avaient des interfaces non interactives, qui utilisaient des cartes perforées. Il y a trente ans, les logiciels n'utilisaient en général qu'une entrée (le clavier) et les sorties se faisaient sous forme d'impression de caractères (à l'écran ou sur imprimante). Ces interfaces ont été appelées alpha-numériques, car elles ne manipulaient que des nombres ou des caractères. Aujourd'hui la plupart des interfaces sont graphiques, et impliquent un dispositif de type écran/clavier/souris.

Le code de l'application est constitué des structures représentant les objets du monde sur lequel porte l'application (les fichiers et les répertoires pour un gestionnaire de fichiers, un damier, des pions, des joueurs dans le cas d'un jeu de dames,

etc.), et des structures utilisées pour organiser ou calculer sur ces données (structures arborescentes, attributs d'objets, listes, tableaux, tables de BD). C'est un univers sur lequel on définit des opérations et transformations permettant d'implémenter les fonctionnalités du logiciel.

La programmation objet vous permet de décrire facilement un univers d'objets (et vous donne divers avantages comme l'héritage, l'encapsulation, l'abstraction sur les procédures et les données, etc.), mais si vous programmez dans un autre langage, vous faites tout de même la même chose : vous représentez le monde de l'application en utilisant des structures de données et vous définissez les procédures nécessaires à la mise en oeuvre des fonctionnalités du logiciel.

Le code de l'interface graphique représente un autre monde : celui de notre interaction avec les écrans actuels. Ce monde qui gére l'interaction avec l'utilisateur est complexe mais basé sur des conventions. Il est constitué d'entités comme les fenêtres, les menus, les boutons, les ascenseurs, les boîtes de dialogues, etc., mais aussi d'entités plus abstraites comme des gestionnaires d'affichage, des modèles ou des vues d'objets complexes (par exemple les modèles et les vues de tables ou de listes en Java Swing). La structuration du code est imposée par la bibliothèque graphique utilisée. Si vous utilisez Java, vous ferez une modélisation objet et suivrez le modèle de programmation par événement basé sur les écouteurs. Avec les toolkits X11 ou les librairies graphiques, vous serez amené également à suivre le modèle de programmation par événements préconisé par les concepteurs de la librairie. Ces modèles prévoient tous la communication entre ce monde de l'interface et celui de l'application.

3.3.4 Conception de l'interface graphique

Même si vous ne pouvez pas développer une interface graphique, vous pouvez quand même en concevoir une - celle que vous auriez aimé programmer si vous en aviez eu le temps. Vous ne devez donc pas sauter cette étape, et vous pouvez fournir un manuel utilisateur décrivant l'interface graphique que vous aurez spécifiée. Votre travail de conception sera pris en compte dans la notation finale.

On dit toujours qu'une interface graphique doit être claire, simple et facile d'emploi. Nous avons ici cherché à isoler quelques règles à suivre pour réaliser cet objectif. La lecture de cette section vous permettra de focaliser sur les points importants à prendre en compte dans la spécification de vos écrans - points qui vous sembleront peut-être évidents, mais qui échappent pourtant souvent aux débutants.

Règle n^{o}1 : Définir les termes utilisés et l'iconographie

Pour la clarté de l'interface, il faut utiliser la terminologie du domaine de l'application. Il faut fixer les termes apparaissant dans l'interface avec soin. Cette tâche prend du temps mais est très importante. Ces termes apparaîtront dans les titres, les menus et les cadres. On réutilisera ensuite ces mêmes termes dans la documentation utilisateur et si possible dans le code. Cela permettra une meilleure compréhension pour l'utilisateur et le programmeur.

Dans les barres de menus, on unifiera la syntaxe des termes utilisés : soit tout nominal, soit tout verbal. Cette remarque s'applique aussi aux titres des boîtes de dialogues. N'hésitez pas non plus à préciser un terme par un complément ou un adjectif. Le titre d'un menu doit recouvrir sémantiquement les différents items du menu. C'est un titre de rubrique.

On veillera également à ne pas mélanger le français et l'anglais. Prévoir si vous avez le temps deux versions, en utilisant des tables pour les termes et les messages d'erreurs. Une fois fixée la terminologie, on évitera l'utilisation de synonymes dans

la documentation. Par exemple, si on utilise le terme Couper dans un menu, on gardera ce terme partout, et on évitera les synonymes comme Supprimer, Effacer, etc. Cela évite des ambigutés et supprime le risque d'incohérences.

De même que les éléments textuels doivent être précisément définis, les icônes, symboles et autres objets graphiques apparaissant dans l'interface doivent être choisis avec soin et en accord avec les futurs utilisateurs.

Etape suivante : concevoir les principaux écrans

Utilisez a priori une disposition standard : une fenêtre principale avec en haut une barre de menu et une barre d'icônes ; en bas, un ascenseur éventuel et/ou une petite fenêtre pour les messages ; à droite, un ascenseur.

Mais il y a d'autres dispositions possibles. En particulier, les fenêtres à panneaux multiples (réajustables par l'utilisateur), ou des fenêtres à onglets (faciles à implémenter en Java). On peut également choisir de démarrer le logiciel avec une barre d'icônes au lieu d'une fenêtre principale. Cette solution est préconisée quand le logiciel est complexe et possède de nombreux types de fenêtres.

S'appuyer sur les standards et s'inspirer des interfaces existantes : l'interface Macintosh a été inspirée par celle d'une machine graphique initialement conçue par Rank Xerox. Elle aura elle même beaucoup inspiré Windows. Aujourd'hui, il y a une certaine standardisation qui s'effectue entre les différents systèmes, et des conventions s'instaurent à partir du moment où elles sont reprises par plusieurs grands logiciels.

Pour que l'interface soit simple et claire, utilisez des conventions déjà connues de l'utilisateur. On procèdera ainsi pour le choix de certains termes figurant dans les menus, le choix des icônes, et pour la disposition générale des fenêtres et des objets graphiques.

Faciliter la navigation : l'utilisateur doit pouvoir se repérer dans les différents écrans du logiciel. Savoir où il en est dans sa tâche, ce qu'il est en train de faire et comment revenir à une étape antérieure. Si votre disposition est standard, le problème ne se pose pas vraiment. Mais si vous devez afficher de nombreuses fenêtres, vous devez organiser la cohérence graphique de vos fenêtres et la navigation entre les fenêtres. Vous avez plusieurs solutions classiques :

— Utiliser des fenêtres à onglets.
— Utiliser une vue des fenêtres permettant de naviguer entre elles. Cette vue peut se présenter comme une liste de boutons représentant les fenêtres, comme une fenêtre menu, ou comme un arbre si l'organisation du logiciel s'y prête.
— Utiliser les menus Modes ou Fenêtres de la barre de menu.

Règle n^{o}2 : Donner le contrôle et du feed-back à l'utilisateur

En fait, il faut distinguer plusieurs types d'utilisateurs, car on peut être novice ou expert de la tâche, et novice ou expert de l'application. Pour le novice de l'application, le parcours de la barre de menu devra donner un panorama des possibilités du logiciel. Pour l'expert de l'application, on offrira des raccourcis claviers et des barres d'icônes, ou d'autres moyens (menus courts/menus longs, possibilités de configuration de l'interface).

Guider l'utilisateur : l'utilisateur doit à chaque instant savoir ce qu'il est en train de faire (s'il vient de lancer une commande, si cette commande est en cours ou terminée, etc.) et pouvoir visualiser les résultats de ses commandes. On parle ici de *feed-back* (retour) utilisateur. Pour réaliser ce *feed-back*, on peut utiliser toutes

sortes de moyens : des clignotements, des grisés, des animations, l'entraînement des icônes à l'écran, le tracé de rectangles pointillés pour sélectionner des zones, la sélection d'objets par de l'inverse vidéo (ou faire apparaître des poignées), des curseurs ou pointeurs particuliers (montres, etc.), des barres de défilement pour montrer l'écoulement du temps, etc.

Il est important également que l'utilisateur puisse visualiser l'effet de ses commandes (mises à jour de vues, fenêtres de statuts ou de messages). L'utilisateur doit également savoir à chaque instant ce qu'il peut faire. On veillera à griser les commandes inaccessibles (textes ou icônes), indiquer par un beep sonore les actions interdites (textes non éditables), souligner à l'aide d'icônes l'état d'un document édité (sauvegardé, modifié), etc. Pour expliquer l'invalidation d'une commande, on affichera des messages d'erreurs. Ces messages apparaîtront dans des boîtes au centre de l'écran ou dans une petite fenêtre spécialisée (par exemple en bas de la fenêtre principale). Pour ces boîtes de messages, on utilisera les boîtes de dialogue toutes prêtes fournies par la bibliothèque utilisée.

Donner le contrôle à l'utilisateur : l'utilisateur doit toujours avoir la main. S'il lance une tâche, il doit aussi pouvoir l'interrompre. Un bouton Annuler d'une boîte de lancement inutilisable quand le curseur de souris est un sablier a de quoi l'énerver à juste titre. Il faut également veiller à ce que les actions explicites, comme la demande d'une sauvegarde, soient effectives (par exemple, pas de sauvegarde en mémoire locale pour optimiser le programme). Il est également souhaitable que l'utilisateur puisse revenir en arrière, et annuler l'effet de sa dernière commande.

Gérer les erreurs : il faut prévoir une gestion des erreurs concernant la saisie de données, avec éventuellement des corrections automatiques, ou des valeurs par défaut. Pour facilter la compréhension, on utilisera aussi les boites de messages standards prêtes à l'emploi.

Règle n°3 : Soigner la mise en page

il n'est pas toujours évident d'analyser pourquoi une interface est claire, mais la clarté de l'interface provient en grande partie de la disposition géométrique des éléments.

Un regroupement d'objets dans une zone spécifique doit correspondre à un regroupement conceptuel. Cette règle est fondamentale. Elle s'applique partout dans l'interface : dans les barres de menus, dans les barres d'icônes, dans l'organisation de l'interface principale et des boîtes de dialogue. On regroupe ensemble les commandes correspondant à un groupe de fonctionnalités (portant par exemple sur des objets de même type), et l'on regroupe ensemble les objets de l'interface permettant la mise en oeuvre d'une même fonctionnalité (par exemple les boutons sur lesquels ils déclenchent des actions). De manière générale, vous devez concevoir vos écrans par zones.

Créer des zones et les nommer avec des titres : mettez des titres bien pensés aux fenêtres. En particulier, donnez toujours un titre aux boîtes de dialogue. Groupez les éléments opérant sur des objets de même type (ou concernant un même groupe de fonctionnalités) dans une même région. Procédez ainsi pour les boîtes de dialogue, mais aussi dans les barres de menus, les menus, les barres d'icônes, et utilisez des séparateurs ou des cadres titrés pour faire ressortir ce découpage.

Organiser les boîtes de dialogues : encadrez et donnez des titres aux différentes zones d'une boîte de dialogue complexe. Pour cela utilisez des cadres (ou bords avec

titre en Java). N'hésitez pas à ce qu'il y ait une certaine redondance entre les titres des zones et les labels utilisés dans la boîte. Vous pouvez également scinder l'espace de la fenêtre horizontalement (avec des séparateurs ou des panneaux réajustables) pour séparer les zones groupant des fonctionnalités différentes. Ainsi, si l'on veut organiser la spécification de certains paramètres, on les présentera par exemple sous forme de cases à cocher ou de champs étiquetés dans une même zone de la boîte avec un titre rappelant la fonctionnalité mise en oeuvre (par exemple, le titre Paramètres). On n'oubliera pas également de donner un titre général à la boîte (comme Spécifier les paramètres). Les boutons qui apparaissent habituellement dans les boîtes sont les boutons Ok, Annuler, Valider, Appliquer, Enregistrer, et Aide. Notez qu'il y a des emplacements standards pour ces boutons.

Les boutons doivent être placés de sorte qu'on voit immédiatement ce sur quoi ils portent. L'encadrement des zones permet de délimiter la portée des actions sur ces boutons. Ainsi, le bouton Supprimer situé près d'une liste dans une zone encadrée contenant la liste portera naturellement sur les items sélectionnés de la liste.

Un dernier point sur les boîtes : il ne faut pas qu'il y ait trop d'enchaînements de boîtes, car cela ennuie l'utilisateur (le cot d'accessibilité est trop élevé). Si l'utilisateur doit configurer beaucoup d'éléments, pensez plutôt à des boîtes complexes, ou revoyez la façon dont vous organisez l'accès aux fonctionnalités du logiciel.

Règles de mise en page :

Aligner les objets. Par exemple, on alignera les bords gauches des labels étiquetant des champs textuels.

Laisser des marges suffisantes. Ne collez pas vos étiquettes ou vos champs de textes aux bords des zones qui les contiennent.

Eviter les espaces vides. Cette règle s'applique à la conception des boîtes de dialogue. Une zone de travail vide n'est, bien entendu, pas gênante.

Minimiser le nombre de lignes verticales. Par lignes verticales, on entend les lignes virtuelles qui apparaissent quand on aligne verticalement les bords gauche ou droit d'un groupe d'objets. Ces objets sont alors alignés sur une ligne (virtuelle) verticale que l'oeil perçoit. Cette ligne prend appui sur les bords alignés des objets. Ainsi, par exemple, si on aligne les bords gauches des labels et les bords gauches des champs de texte situés en vis-à-vis dans une boite de dialogue, on fera apparaître deux lignes verticales . Ne pas multiplier ces lignes consistera ici à aligner également d'autres objets de la même fenêtre sur ces lignes, comme par exemple, des boutons situés en dessous, ou le bord d'une zone encadrée située plus bas dans la boîte. Cela revient à aligner les différents objets parfois même au-delà des séparateurs de zones.

Placer toujours les mêmes choses aux mêmes endroits. C'est pour cette raison que l'on a choisi de griser les commandes inaccessibles dans un menu, plutôt que de les faire disparaître dynamiquement, car les items du menu ne se seraient plus trouvés à la même place dans le menu et cela aurait perturbé l'utilisateur. Dans cet esprit, uniformisez la mise en page des différentes boîtes ou fenêtres en plaçant les boutons, titres, etc. aux mêmes endroits s'ils sont communs à plusieurs boites similaires. Adoptez un certain point de vue sur la manière dont l'utilisateur doit par exemple configurer des paramètres et maintenez une certaine cohérence entre vos boîtes. En particulier, n'hésitez pas dans le code à utiliser les mêmes panneaux si vos boîtes ont des parties communes.

Règle n°4 : veiller à la cohérence

Veiller partout à la cohérence. Par exemple, si vous avez plusieurs façons de déclancher une commande, la procédure qui sera déclanchée doit être exactement la même. (ça vous évite aussi de dupliquer le code). En Java, on pourra utiliser les Actions pour déclencher les mêmes commandes dans la barre d'icônes et dans la barre de menu.

Pour que la documentation soit correcte, il faut également utiliser les termes apparaissant dans l'interface de manière cohérente avec le manuel utilisateur, et si possible, retrouver ces mêmes termes dans le code.

Il faut également avoir une utilisation cohérente des fontes et des couleurs. Les fontes doivent normalement être spécifiées par l'utilisateur. Mais vous pouvez permettre à l'utilisateur de choisir sa fonte, et réserver l'italique ou le gras de la même fonte pour un usage particulier.

Néanmoins, il doit y avoir une cohérence dans l'usage des fontes de votre logiciel. On donnera un sens à leur usage. Par exemple, on pourra utiliser une fonte particulière pour les items de menu et faire apparaître le titre du menu dans une autre fonte (par exemple, en gras).

Surtout, ne pas multiplier les fontes. Si vous voulez plusieurs fontes, utilisez la même fonte dans différentes tailles, avec éventuellement du gras ou de l'italique. Car, outre l'effet général de confusion, rien n'est plus laid qu'un mélange de fontes.

Les couleurs de la même façon devront être utilisées de manière cohérente. On donnera un sens à l'usage d'une couleur. Par exemple, on pourra mettre en relief les lignes d'erreur dans un éditeur de programmes en leur affectant une certaine couleur, une autre étant réservée aux mots clés.

Le choix des couleurs est supposé aussi être du ressort de l'utilisateur. Vous pouvez cependant fixer l'usage d'une couleur, et laisser l'utilisateur choisir cette couleur. Il faudra cependant veiller dans certains cas à ce que les couleurs choisies, en particulier pour des couleurs de fonte et d'arrière plan, aient un contraste suffisant. Le contraste idéal est celui du blanc et du noir. Si l'on veut mettre de la couleur, on peut utiliser une couleur claire à la place du blanc et écrire en noir, ou à l'inverse, prendre une couleur foncée pour le fond et écrire en blanc.

3.4 Conception détaillée et Planning prévisionnel

3.4.1 Conception détaillée

L'étape de conception détaillée fournit pour chaque composant (ou module) une description de la manière dont les fonctions du composant sont réalisées et une description précise de la manière dont les autres modules pourront interagir avec lui. Pour que vous puissiez travailler en parallèle sur plusieurs modules, il faut en effet définir soigneusement pour chaque module les structures de données et les fonctions dont les autres modules auront besoin.

C'est à cette étape que vous devez définir et nommer les structures de données qui seront utilisées dans le programme. Pour les principales structures de données vous devez donner une justification précise des choix effectués.

Lorsque le logiciel utilise des fichiers, c'est à cette étape que vous devez décrire très précisément le format de ces fichiers. Il s'agit de donner tous les détails nécessaires afin que chaque module utilisant ces fichiers puisse être programmé indépendamment des autres.

Pour les modules les plus importants, vous devez de plus donner un algorithme *en français* (et non pas dans un langage de programmation). A cette étape, il ne s'agit pas de commencer la programmation, mais seulement *de préparer* le travail de programmation, en précisant la manière dont les fonctionnalités seront réalisées.

3.4.2 Planning prévisionnel

Vous devez également établir un planning prévisionnel indiquant la liste des tâches à réaliser, la date et le début de chaque tâche, qui dans le groupe travaille sur cette tâche, en veillant à ce que l'enchaînement des tâches dans le temps soit correct : il faut réaliser d'abord les tâches nécessaires à la réalisation d'autres tâches. On cherche donc à établir les dépendances entre les tâches, et on les ordonnera et parallélisera. On listera en particulier les tâches de programmation des différents modules, mais il ne faudra pas oublier de lister aussi les tâches de documentation, d'apprentissage ou de recherche d'informations, ainsi que les tâches de coneption, de rédaction ou de préparation de la démonstration. L'intégration des différents modules (par exemple l'intégration de l'interface graphique) ou l'intégration d'un module de base de données, réserve aussi souvent des surprises et demande parfois des corrections. Il est donc important de réserver du temps pour la phase d'intégration des différents modules. On définira donc aussi des tâches d'intégration. Vous pouvez lister toutes les tâches sur un tableau.

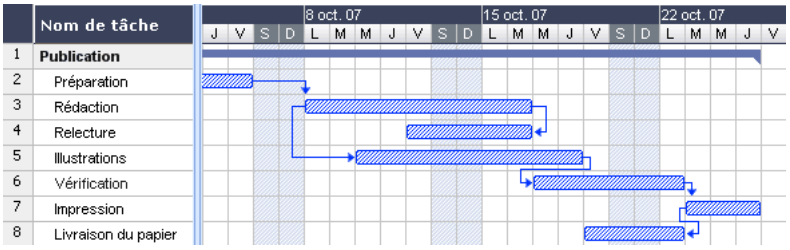
Les tâches pourront ainsi être représentées dans le temps en parallèle sur un schéma appelé diagramme de Gantt.

Dans un diagramme de Gantt on représente :

— en abscisse les unités de temps (exprimées en mois, en semaine ou en jours)
— en ordonnée les différentes tâches.

Initialement, le diagramme de Gantt ne visualise que le temps : les dates (début et fin) ainsi que la durée des tâches. Cependant il est fréquent de matérialiser par des flèches les liens de dépendance entre les tâches (la flèche relie la tâche précédente à la tâche suivante).

Vous devez fournir un tel diagramme dans le premier rapport, et dans le mémoire vous y ajouterez le diagramme plus précis correspondant au planning réel effectivement suivi (avec des tâches plus détaillées) .



Exemple de diagramme de Gantt

3.5 Codage

C'est seulement après l'étape de conception détaillée que commence la programmation. Si les étapes précédentes ont été passées correctement chaque module peut être programmé indépendamment des autres, et le travail de programmation pourra être réparti entre les différents participants du projet. Cette phase de programmation doit suivre certaines règles :

— se conformer strictement à la description du module qui a été faite lors de la conception détaillée (fonctionnalités à réaliser et interface avec le reste du logiciel) : tout écart à cette règle peut remettre en cause la réalisation de l'ensemble du projet et réduire à néant tout le travail déjà effectué sur les autres modules.
— adopter des standards de programmation pour le nommage des différents objets du langage (fichiers, type, variables, constantes, fonctions, procédures...), pour l'écriture des commentaires, pour la présentation du code (en particulier

pour l'indentation), pour le traitement des erreurs (détection systématique des erreurs pouvant être causées par les appels systèmes, format des messages d'erreurs, ...).

Vous devrez vous réunir régulièrement pour faire le point sur l'avancement des différentes tâches. Le planning prévisionnel évoluera donc en fonction de l'état d'avancement et vous serez amenés à le modifier lorsque des événements imprévus surgiront. Ainsi, si quelqu'un n'arrive pas à rśoudre un problème ou s'il programme trop lentement, vous pourrez lui adjoindre un membre du groupe pour l'épauler et modifier le planning ou la répartition initialement prévus.

3.6 Tests et mise au point

Le programme est exécuté sur les données de test et les résultats obtenus sont comparés avec ceux attendus. Lorsque les résultats obtenus diffèrent des résultats attendus, il faut corriger le programme et faire repasser tous les tests. La phase de tests est considérée comme terminée lorsque la version déboguée du logiciel passe tous les tests avec succès.

Si vous en avez le temps, vous reprendrez le tableau de validation suggéré à la fin de la section 3.2.2, en présentant cette fois, pour chaque module la liste des fonctionnalités à implanter, les valeurs d'entrées normales avec les résultats attendus (jeu de test) en précisant les cas particuliers qui peuvent se présenter, et l'état actuel de la validation (pas fait : les tests n'ont pas été passés ; en cours : les tests sont en cours, mais la mise au point n'est pas terminée, il reste des bogues ; OK : les tests ont été passés avec succès). La colonne "validation" sera remplie au fur et à mesure de la mise au point, après le codage.

Chapitre 4

Liste des sujets proposés

4.1 Gestion de réferences bibliographiques en bibtex et HTML

Le projet consiste à écrire un logiciel permettant de gérer des références bibliographiques contenues dans un fichier au format bibtex. Le logiciel offrira les fonctionnalités suivantes :

- chargement d'un fichier bibtex
- ajout de références
- modification de références
- suppression de références.
- sauvegarde des modifications
- Extraction/recherche des références à partir de noms d'auteurs, de mots du titre, de noms de références
- génération d'un fichier HTML valide permettant d'afficher la liste des références dans le format standard :

  author, title, publisher, series, booktitle, volume(number), pages, adress, year, note, annote.
- interrogation à distance, grâce à une architecture client/serveur. (Obligatoire si plus de deux étudiants).

Une base de données de références bibliographiques contient des références qui peuvent référencer des livres, des articles de revues ou des articles de conférences. A chacun de ces types de référence correspondent plusieurs champs dans le format bibtex. Ces différents champs sont donnés dans les exemples ci-dessous :

- Exemple de référence au format bibtex, pour un livre :

```
@Book{Mallatbook,
  author =   {S. Mallat},
  title =   {A Wavelet Tour of Signal Processing},
  publisher =   {Academic Press},
  year =   {1998},
  OPTeditor =   {},
  OPTvolume =   {},
  OPTnumber =   {},
  OPTseries =   {},
  address =   {Boston},
  OPTedition =   {},
  OPTnote =   {},
  OPTannote =   {}
}
```

— Exemple de référence au format bibtex, pour un article de revue :

```
@Article{Temlyakov,
  author =   {V. Temliakov},
  title =   {Nonlinear methods of approximation},
  journal =   {FOCM},
  year =   {2008},
  OPTkey =   {},
  volume =   {3},
  number =   {1},
  pages =   {33-107},
  month =   {Feb.},
  OPTnote =   {},
  OPTannote =   {}
}
```

— Exemple de référence au format bibtex, pour un article de conférence :

```
@InProceedings{pati93OMP,
  author =   {Y. Pati and R. Rezaiifar and P. Krishnaprasad},
  title =   {Orthogonal Matching Pursuit: Recursive Function
                Approximation with Applications to Wavelet Decomposition},
  booktitle = {27 th Annual Asilomar Conference on Signals, Systems,
    and Computers},
  pages =   {40-44},
  year =   {93},
  OPTeditor =   {},
  volume =   {1},
  OPTnumber =   {},
  OPTseries =   {},
  OPTaddress =   {},
  publisher = {IEEE},
  OPTnote =   {},
  OPTannote =   {}
}
```

Dans les références ci-dessus, les champs commençant par OPT, sont optionnels et ne sont pas renseignés. Les références portent ici les noms "Mallatbook", "Temlyakov" et "pati93OMP".

NB :

— Vous pourrez utiliser les outils Flex et Bison pour analyser un fichier Bibtex.

— Un fichier chargé pourra contenir des références incomplètes

4.2 Gestion de cave à vin

Il s'agit d'écrire un logiciel permettant de gérer le stock d'une cave à vin. On peut imaginer que cette cave appartienne à un particulier ou à un marchand de vin. On caractérisera un vin par sa région, son domaine, son château, sa couleur, son année, son cépage. Une bouteille de vin sera caractérisée par le vin qu'elle contient et la taille de la bouteille. La cave contient des vins. On voudra donner pour un vin, le nombre de bouteilles, une année de maturité, une localisation (un numéro de casier dans la cave), des commentaires (d'au plus 1000 caractères).

Le logiciel offrira les fonctionnalités suivantes :

— Chargement d'un fichier décrivant la cave

— modification de vin répertorié dans la cave

— ajout de vin

— suppression de vin

— sauvegarde d'une cave
— Recherche et affichage de vin par nom, par année de maturité, par couleur, par cépage, région, etc. Plusieurs critères pourront être combinés.
— interrogation à distance, grâce à une architecture client/serveur si le cas envisagé est celui d'un magasin

NB :

— Le nombre de vins ne sera pas limité (vous utiliserez une structure de données dont la taille n'est pas fixée par avance).
— Vous pourrez stocker les informations relatives à la cave dans une base de données. (Le langage Java possèdent des facilités pour la création et l'interrogation de base de données).
— Si vous prenez la décision de stocker les informations relatives à la cave dans des fichiers (au lieu d'une base de données) et que vous programmez en C, vous pourrez utiliser les outils Flex et Bison pour analyser ces fichiers. Si vous programmez en Java, vous pourrez alternativement définir vous-même un analyseur syntaxique de vos fichiers à l'aide de la classe StreamTokenizer.

4.3 Gestion d'arbres généalogiques

L'arbre généalogique d'une personne contient tous ses ancêtres connus (parents, grands-parents, arrière-grands-parents, etc.).

Le programme doit permettre à l'utilisateur de s'inscrire en tant que personne et de créer son arbre généalogique. Pour chaque personne, on peut connatre son nom et son prénom, sa date et son lieu de naissance, son sexe et éventuellement sa date et son lieu de décès.

Initialement, l'utilisateur a un arbre généalogique vide (ou si on veut, réduit à un noeud : contenant les informations sur sa propre personne, mais pas ses parents).

Le programme doit permettre à l'utilisateur de modfier son arbre généalogique (entrer un nouvel ascendant, le supprimer ou le modifier). Un arbre généalogique est constitué de noeuds contenant les mêmes champs que ceux d'une personne, complétés de deux ascendants : son père et sa mère. Certains champs peuvent ne pas être renseignés.

L'utilisateur devra aussi avoir la possibilité de sauvegarder son arbre généalogique de sorte qu'il le retrouve lors d'une nouvel session et il pourra aussi l'afficher sur l'écran pour le visualiser.

Il faudra maintenir les informations enregistrées par les utilisateurs dans une même base de données. Le logiciel devra aussi permettre d'effectuer des recherches par nom, prénom, date ou lieu de naissance, date ou lieu de décès, etc. sur des personnes connues de la base. Ces recherches pourront combiner plusieurs critères. L'utilisateur devra ainsi pouvoir charger les arbres généalogiques d'autres personnes que lui-même qu'il aura trouvé dans la base.

Extensions possibles : le couple des parents d'une personne peut avoir d'autres enfants. On pourrai vouloir stocker les autres enfants d'un même couple, ce qui amènerait à introduire les notions de frère et sœur.

Le logiciel pourrait avoir la notion de frère et sœur, ainsi que d'autres notions de parenté et permettre de rechercher dans la base un frère, une sœur, un cousin, ou tout autre relation de parenté comprise par le logiciel.

Vous pourrez spécifier des extensions du logiciel en indiquant les modifications qu'il faudrait apporter à votre logiciel initial pour qu'il comprenne de nouvelles notions. L'idéal ´etant que vos structures de données soient prévues pour faciliter ces extensions.

4.4 Gestion d'une bibliothèque

Dans ce projet, vous développerez un logiciel de gestion de bibliothèque. Il devra offrir les fonctionnalités suivantes :

— stocker des références de livres (auteur, titre, ...)
— ajout, suppression et modification de livres
— Recherche d'un ouvrage selon un ou plusieurs critères
— Gestion des prêts

Deux interfaces de visualisation et de manipulation sont demandées : d'une part, une interface en ligne de commande, d'autre part une interface graphique.

Le logiciel est destiné principalement au bibliothécaire d'un petit collège de 12 classes dont la bibliothèque contient environ un millier de livres. C'est le bibliothécaire qui gèrera les emprunts, les rendus d'ouvrages et les retards, ainsi que les inscriptions/desinscriptions des emprunteurs. Il gérera aussi le stock de livres et pourra ajouter, retirer un livre ou modifier l'état d'un livre.

Le logiciel permettra également à un utilisateur anonyme de rechercher un livre par auteur et/ou par titre, et de savoir s'il est disponible ou non en rayon.

Si vous pensez gérer la bibliothèque avec des fichiers, Il faudra gérer trois fichiers :

— le fichier des emprunteurs :
  référence (né d'identifiant unique)
  nom
  prénom
  classe
— le fichier des ouvrages
  référence (né d'identifiant unique)
  nom auteur
  prénom auteur
  titre
  état : emprunté, rendu, en rayon, en réparation
  référence d'emprunt
— le fichier des emprunts
  référence (né d'identifiant unique)
  référence livre
  référence d'emprunteur
  date d'emprunt
  date limite d'emprunt

Vous pourrez utiliser les outils Flex et Bison pour analyser les fichiers stockant les informations relatives à la bibliothèque, ou encore utiliser une BD.

4.5 Serveur de petites annonces

Le projet consiste à écrire un gestionnaire de petites annonces sous la forme d'une architecture client/serveur. Le programme serveur stockera un ensemble d'annonces et permettra aux clients de consulter les annonces, d'en créer de nouvelles, ou d'en supprimer. Une annonce sera constituée d'un titre et d'un texte (les deux sous forme de chaîne de caractères) et d'un auteur (l'utilisateur ayant créé l'annonce) qui sera propriétaire de l'annonce, i.e. le seul à pouvoir la supprimer.

Le serveur permettra la connexion de programmes clients, et ceux-ci pourront consulter la liste des titres des annonces, éditer le texte d'une annonce particulière, créer une nouvelle annonce, ou encore supprimer une annonce (mais on ne pourra supprimer une annonce que si on en est l'auteur).

Le programme client aura une interface graphique agréable affichant en permanence la liste des titres des annonces et leur nombre. Des boutons permettront

l'ajout, la suppression et l'édition d'une annonce sélectionnée par son titre. Un bouton de mise-à-jour permettra de recharger la liste des annonces disponibles au niveau du serveur.

On peut aussi envisager de proposer ce service sur une page web. Vous pourrez aussi envisager de développer d'autres fonctionnalités.

4.6 Assembleur MAMIAS

MAMIAS est un mini-langage assembleur (pour une machine virtuelle) utilisé pour l'enseignement en premier cycle. Le logiciel lira un fichier source contenant un texte de programme écrit en MAMIAS, et il générera un fichier binaire contenant le code machine de ce programme.

On pourra alors simuler un chargement de ce programme en mémoire centrale. Les lignes d'instructions seront placées dans les registres et on pourra exécuter le programme pas à pas.

Description de MAMIAS.

Les instructions et les données sont codées dans des mots mémoire (**huit positions binaires**). Chaque mot mémoire est repéré par un entier $n \geq 0$ appelé son **adresse**, le mot mémoire d'adresse n est désigné par `(n)`. On dispose en plus d'un mot mémoire particulier appelé **accumulateur** et désigné par *Acc*.

Une instruction est un mot mémoire décomposable en *CodeArgument*
- *Code* : est le numéro de code de l'opération (les 3 positions binaires de gauche),
- *Argument* : est
  - ou bien une adresse `n`,
  - ou bien un entier[5] `x` (les 5 positions binaires de droite).

On peut donc coder 8 opérations (puisque Code est sur 3 bits), adresser 32 mots mémoire (de 0 à 31, puisque sur 5 bits) et utiliser des entiers signés compris entre -16 et 15.

Une donnée est un entier[8] codé dans un mot mémoire adressé, ou figurant dans l'accumulateur.

Pour faciliter la lecture, chaque code opération sera désigné par une "expression mnémonique" (ce n'est pas ce que MAMIAS est censé lire : il ne comprend que les codes des 3 bits, mais cela rendra le logiciel plus lisible).

Les 8 opérations et l'effet des instructions correspondantes sont les suivants (où le symbole $\leftarrow$ désigne l'affectation) :
- `INIT` (Code = 000) : `INIT` est `Acc` $\leftarrow$ `x`
  Note : `x` est un entier[5] qui se trouve traduit en entier[8] dans l'accumulateur, plus précisément :
  - l'instruction 0000*abcd* (`INIT` 0*abcd*) donne la valeur 0000*abcd* à l'accumulateur,
  - l'instruction 0001*abcd* (`INIT` 1*abcd*) donne la valeur 1111*abcd* à l'accumulateur.
- `CHARGE` (Code = 001) : `CHARGE n` est `Acc` $\leftarrow$ `(n)`
- `RANGE` (Code = 010) : `RANGE n` est `(n)`$\leftarrow$ `Acc`
- `ET` (Code = 011) : `ET n` est `Acc`$\leftarrow$ `Acc` $\wedge$ `(n)`
  où $\wedge$ est l'opération de conjonction bit à bit (ou masquage) définie par $1 \wedge 1 = 1$, $1 \wedge 0 = 0 \wedge 1 = 0 \wedge 0 = 0$. Par exemple $10110001 \wedge 11010000 = 10010000$.
- `SAUTE` (Code = 100) : `SAUTE n` est "si `Acc` = 0 alors aller à l'adresse `(n)`"
- `ADD` (Code = 101) : `ADD n` est `(n)`$\leftarrow$ `(n) + Acc`

— `DEC` (Code = 110) : `DEC x` décale de `x` positions le contenu de `Acc`
Ce décalage se produit vers la gauche si `x` est positif, et vers la droite s'il est négatif : les digits qui "sortent" du mot sont perdus !
Par exemple, si `Acc` = 01110011 :
  — l'instruction 11000010 (`DEC2`) donne la valeur 11001100 à l'accumulateur,
  — l'instruction 11011101 (`DEC` −3) donne la valeur 00001110 à l'accumulateur.
— `STOP` (Code = 111) : `STOP` arréte l'exécution.
Les cinq positions binaires de droite n'ont aucune signification.

Sauf avis contraire (spécifié par l'opération `SAUTE`), les instructions s'exécutent dans l'ordre des adresses croissantes, à partir de 0.

Le logiciel lira un fichier source contenant un texte de programme du type

```
def     x       31
def     y       30

        INIT    0
        CHARGE  x
        SAUTE   suite
        RANGE   y
suite : ADD     x
        ....
        STOP
```

où les deux premières lignes définissent des 'variables', et dont les suivantes sont les instructions du programme écrites ici dans une notation symbolique. Un tel programme sera d'au plus 31 lignes (les 31 mots mémoire de la mémoire centrale).

Le logiciel générera un fichier binaire contenant le code binaire de ce programme (on a substitué à chaque opération symbolique son Code sur 3 digits, remplacé les variables tenant lieu de mot mémoire par le code binaire de l'entier correspondant, et substitué à l'adresse symbolique "suite" l'adresse 4 en binaire (soit 00100) puisque c'était la cinquième ligne du programme (numérotée à partir de 0) qui était labelisé "suite". Ainsi, le programme précédent aura pour code

```
000   00000
001   11111
100   00100
010   11110
101   11111
....
111   10110
```

Vous écrirez aussi un outil de visualisation de la mémoire centrale permettant d'exécuter un programme MAMIAS pas à pas. (La mémoire centrale est constituée de l'état des 32 registres et de celui de l'accumulateur).

4.7 Jeu d'échecs en réseau

Ce projet devra permettre à un joueur de jouer une ou plusieurs parties à travers le réseau avec d'autres partenaires. Un serveur gérera les plateaux correspondant aux différentes parties. Les clients se connecteront au serveur soit pour jouer, soit

pour assister, comme spectateur, au déroulement d'une ou plusieurs parties. Le logiciel développé doit posséder les fonctionnalités suivantes : connexion d'un futur joueur, affichage des parties en cours, affichage des joueurs en attente de partie, mise en relation de deux joueurs en quête de jeu. On peut aussi prévoir que les joueurs déclarent un niveau d'expertise afin de jouer contre un joueur de même force.

On doit pouvoir jouer une partie à distance, chaque joueur visualisant le même plateau que son adversaire sur son écran. Le déplacement des pièces de l'échiquier pourra être effectué en cliquant avec la souris sur les cases du plateau. Seuls les mouvements autorisés pourront être réalisés.

On doit également pouvoir sauvegarder une partie en cours, afin de la recharger plus tard si les deux joueurs participant sont connectés.

4.8 Le jeu de la vie

Le jeu de la vie, créé par le mathématicien John Horton Conway, est basé sur l'évolution de populations cellulaires. Les cellules sont disposées sur une grille. La naissance et la mort d'une cellule donnée sont conditionnées par la présence suffisante ou excessive de cellules vivantes dans son voisinage immédiat. On considère que l'espace est constitué d'une grille, dans laquelle chaque case a 8 voisins au maximum (car il y en a moins pour les cases situées sur les bords ou dans les coins).

Une case ne peut contenir qu'une seule cellule et les règles de vie et mort sont les suivantes :

— si le voisinage d'une case vide contient trois cellules, alors une cellule nait ;
— si une cellule est entourée de plus de 4 cellules, elle meurt étouffée ;
— si une cellule est entourée d'une cellule ou moins, elle meurt d'isolement.

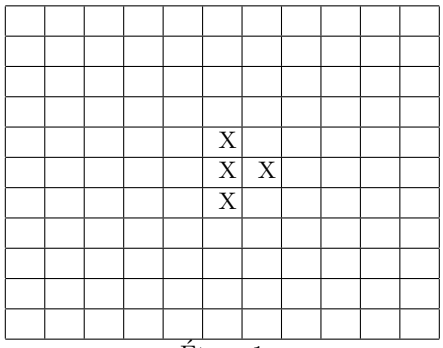
Le logiciel doit permettre de simuler l'évolution d'une population de cellules initialement placées sur une grille. Cette simulation devrait pouvoir être réalisée en différent mode : un mode pas à pas enchaînant les évolutions successives, ou un mode plus rapide dont la vitesse pourra être fixée. On doit également pouvoir à tout moment interrompre cette simulation et sauvegarder l'état d'une grille, ou en charger une nouvelle. On préparera pour la démonstration des exemples intéressants, préalablement sauvegardés dans des fichiers. Le logiciel permettra le chargement et la lecture de tels fichiers.

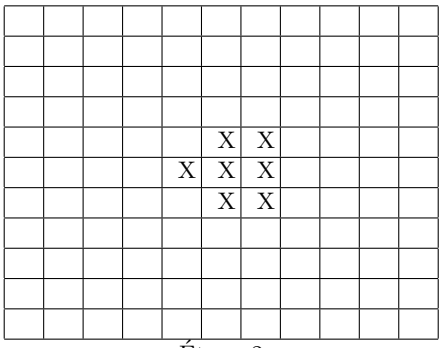
Vous pourrez définir un format naturel de fichier de sauvegarde sous forme de lignes constitué de 0 et de 1(séparés par des caractères blancs) pour définir la présence ou l'absence de cellules sur la grille, et en permettre l'édition dans un éditeur de texte normal. Mais vous permettrez aussi une édition et une interaction graphique (avec la souris) plus sophistiquée (vous pourrez vous inspirer de l'interface graphique de la commande bitmap) permettant de créer une grille et d'y placer des cellules.

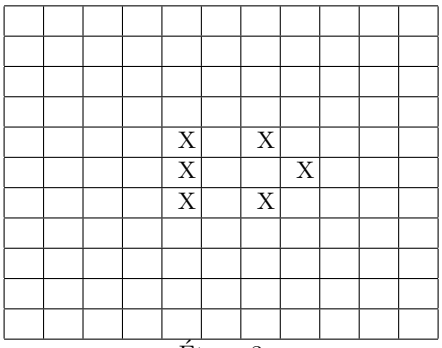
Vous inclurez les fonctionnalités suivantes :

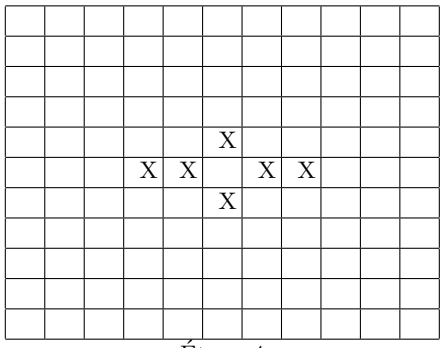
— considérer un espace torique (le haut et le bas se rejoignent, ainsi que la droite et la gauche)
— fixer les dimensions d'une grille
— éditer une configuration de grille
— choisir la vitesse de déroulement d'une simulation
— lancer une simulation en mode pas à pas ou automatique
— interrompre une simulation
— permettre à l'utilisateur de modifier l'état du jeu pendant son déroulement
— sauvegarder/recharger l'état d'une grille à partir d'un fichier

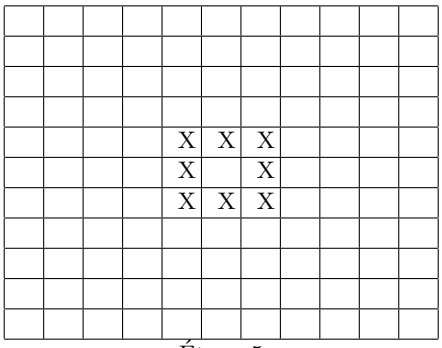
Exemple d'évolution d'une grille de dimension 11x11

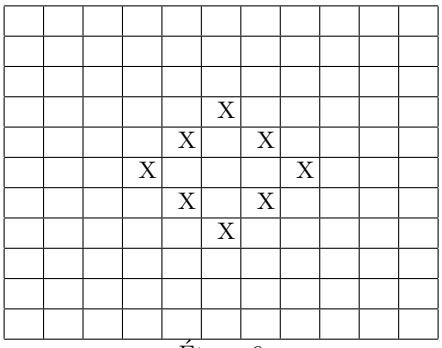

Étape 1

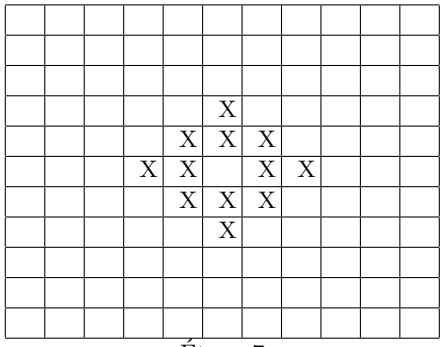

Étape 2

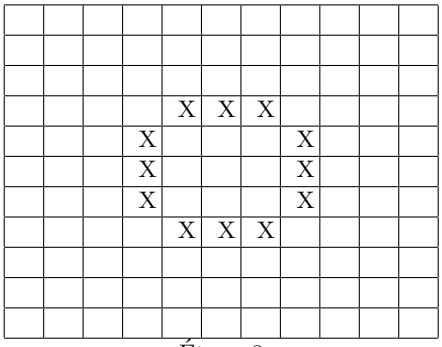

Étape 3

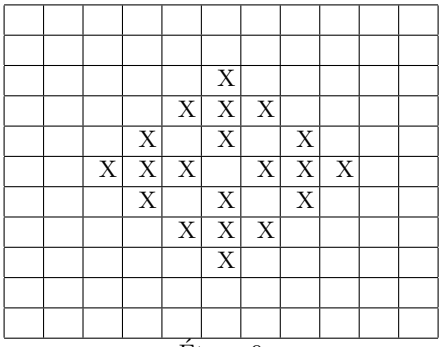

Étape 4

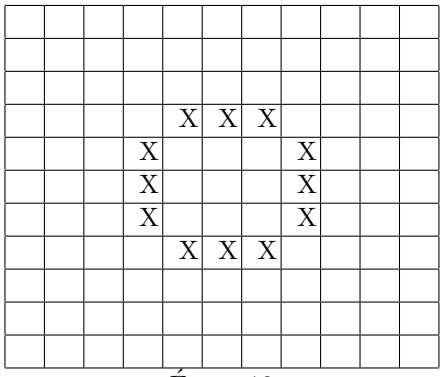

Étape 5

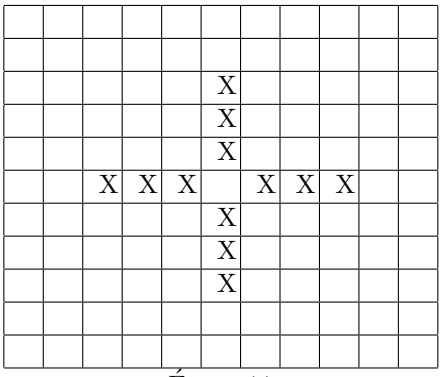

Étape 6


Étape 7


Étape 8


Étape 9


Étape 10


Étape 11

4.9 Simulation de circuits électroniques

Les circuits électroniques sont constitués de composants reliés entre eux par des entrées et des sorties. La simulation consiste à envoyer un signal à l'entrée du circuit (des 0 ou des 1) et à observer ce qui se passe à la sortie (encore des 0 ou des 1). Le logiciel devra permettre de créer, de sauvegarder et de charger des circuits électroniques.

On voudrait simuler le fonctionnement d'un circuit électronique logique, c'est-à-dire donner les valeurs de sortie (0 ou 1) en fonction des valeurs données en entrée (0 ou 1).

Un circuit électronique est un ensemble de composants connectés entre eux. Ces composants peuvent être des portes AND (2 entrées, 1 sortie), des portes OR (2 entrées, 1 sortie), des portes XOR (2 entrées, 1 sortie), des inverseurs NOT (1 entrée, 1 sortie), des entrées IN (0 entrée, 1 sortie) ou des sorties OUT (1 entrée, 0 sorties). Les tables de vérités de ces composants sont les suivantes :

AND	0	1
0	0	0
1	0	1

OR	0	1
0	0	1
1	1	1

XOR	0	1
0	0	1
1	1	0

NOT	0	1
	1	0

Des circuits pouront être chargés à partir d'un fichier. Une ligne dans un fichier décrit soit un composant (`composant NomComposant TypeComposant`), soit une liaison entre deux composants (`liaison NomComposant NéSortie NomComposant NéEntrée`), avec `TypeComposant`$\in$ {`AND,OR,NOT,IN,OUT`} et `NomComposant` un mot (suite de caractères sans espace).

Exemple de fichier de circuit :

```
composant   in1    IN
composant   in2    IN
composant   in3    IN
composant   out1   OUT
composant   and1   AND
composant   or1    OR
liaison     in1    1     and1   1
liaison     in2    1     and1   2
liaison     and1   1     or1    1
liaison     in3    1     or1    2
liaison     or1    1     out1
```

Un tel fichier pourra être analysé par le logiciel, et la syntaxe vérifiée. Vous inclurez les fonctionnalités suivantes :

- produire la table de vérité d'un circuit (tableau dont les colonnes contiennent les valeurs en entrée et les valeurs en sortie) et l'afficher. La sauvegarder dans un fichier.
- éditer un circuit avec le logiciel sous forme textuelle (fichier) ou sous forme graphique
- modifier un circuit : ajouter d'autres composants : noeud1-2 (noeuds qui envoient sur leurs deux sorties la même chose que ce qu'ils ont en entrée), multiplexeurs, etc.
- et animer les circuits (difficile)

4.10 Thématiseur général

Un thématiseur général est un logiciel qui permet de catégoriser un document en déterminant les thèmes dont il parle. il prend en entrée un document et retourne une

liste de thèmes correspondants au contenu du document (par exemple : science, tourisme, humour, horoscope, etc.). Un document est un fichier texte. On considèrera qu'il s'agit d'une suite de mots (chaîne de caractères sans espace) séparés par des espaces ou des caractères de ponctuation.

Un thématiseur général fait appel à des thématiseurs spécialisés (pour un thème).

Un thématiseur spécialisé est initialisé pour un thème particulier, par exemple "Science". Il a comme ressource une liste de mots (mots thématiques) correspondant au thème. Par exemple pour le thème "Science" les mots 'informatique', 'physique', 'chimie', 'fonction', 'mathématique', 'démonstration', 'preuve' etc. feront partie de sa liste de mots-clés. Sa fonction principale, étant donné un document, est de retourner le pourcentage de mots du document qui font partie de sa liste de mots thématiques. Si ce pourcentage est supérieur à un certain seuil, le document est considéré comme rattaché à ce thème.

Un thématiseur général doit consulter tous les thématiseurs spécialisés pour découvrir les thèmes correspondant au document, mais l'utilisateur pourra aussi sélectionner au préalable une liste de thèmes à parcourir (i.e. une liste de thématiseurs spécialisés à consulter) pour limiter la recherche dans un sous-ensemble de thèmes initiaux. L'utilisateur doit aussi pouvoir interroger le thématiseur sur l'appartenance d'un document à un thème donné, et modifier les seuils définis pour chaque thème particulier. Il devrait aussi pouvoir définir de nouveaux thèmes à partir d'une liste de mots-clés et d'un seuil.

Le logiciel aura une interface graphique permettant de charger le fichier d'un texte, de fixer des seuils, et de lancer la recherche de thèmes. On pourra aussi entrer des listes de mots-clés thématiques permettant de créer ou d'enrichir de nouveaux thématiseurs spécialisés. On peut imaginer diverses fonctionnalités ayant trait aux thématiseurs spécialisés comme au thématiseur général. Le logiciel pourra utiliser une base de données pour enregistrer les mots-clés associés aux thématiseurs spécialisés. On pourra prévoir une interface graphique, ou encore un site web. A vous de spécifier clairement le logiciel et son architecture.

Quelques idées pour vous permettre d'implémenter un jeu d'essais :

Thème astronomie : astronomie astronome soleil lune vénus saturne pluton neptune mercure galaxie planète planètes étoile étoiles satellites mars jupiter cosmique terre lunette télescope ciel lumière éclipse exploration astrophysique cosmogonie cosmos cosmographie longue-vue lorgnette astre comète nova filante météorite météore astérode aérolithe spoutnik lactée astral.

Thème science : science scientifique technologique médecine astronomie astrophysique chimie informatique mathématique biologie physique géologie paléontologie histologie géométrie empirisme logique empirique découverte scientifique cognition savoir expérimental algèbre algébrique formule preuve connaissances.

Thème horoscope : horoscope horoscopes poisson verseau bélier taureau gémeau cancer lion vierge balance scorpion sagittaire capricorne amour travail santé sensible coeur chance argent carrière célibataire soleil lune vénus mars saturne jupiter rencontre étoile ciel prédiction astrologie astrologue astrologues devin ésotérisme divination destin destinées signe signes astral ascendant décan.

Thème humour : rire histoires belges blondes blonde belge toto comique blague blagues comédie charade plaisanterie drôle galéjade canular farce farces hilarité pouffer esclaffer sourire rigoler amuser amuse amusant hilarant jeu bouffonnerie bouffon clown ironie malice noir anglais.

Document1 : Les astrologues sont de plus en plus consultés par les personnes fragiles lire leur horoscope les rassure et les amuse le ciel et les planètes ont toujours

fasciné l homme et l astronomie n a pas su faire disparaître l astrologie

Document2 : Le rire est le propre de homme mais il y a des sens de l humour bien particulier j aime l humour anglais mais je déteste l humour des étudiants de médecine leurs farces sont souvent macabres et ne me font pas rire

Document3 : Je ne regarde jamais mon horoscope sur Internet je préfère y lire des blagues mes blagues préférées sont celles sur les blondes

Document4 : Une blonde entre dans une bibliothèque et demande un livre à la bibliothécaire de quel auteur interroge la bibliothécaire heu répond la blonde de vingt centimètres

4.11 Réservation de chambres d'hôtel

Le logiciel est destiné au directeur d'un hôtel d'une vingtaine de chambres. Il doit pouvoir enregistrer des réservations, afficher le planning des disponibilités et procéder à toute opération. Le logiciel sera également accessible via un site web aux futurs clients qui pourront l'utiliser pour consulter les disponibilités et faire des réservations en ligne. Il y a tout un travail de conception et de spécification à faire pour réaliser ce projet, et l'on s'inspirera des sites de réservations d'hôtels existants.

4.12 Motus

Il s'agit de simuler le jeu télévisuel, pour que les joueurs puissent s'entrainer. Le logiciel pourra être livré comme application, ou accessible sur une page web. A titre d'exemple on pourra s'inspirer du jeu en ligne sur //motus.france2.fr/.

Le but du jeu Motus est la découverte d'un mot caché de 7 lettres, en au plus 6 tentatives. Les différentes tentatives (proposition de mots) seront placées sur une grille de 6 rangées (6x7 cases). Une variante serait de chercher un mot de 8 lettres en au plus 7 tentatives. Initialement, la première lettre du mot recherché est donnée, et placée sur la première case en haut de la grille. Si cette lettre figure plusieurs fois dans le mot, les différentes occurrences de la lettre seront placées correctement sur cette ligne, les autres cases restant vides.

Le joueur doit alors proposer un mot de 7 lettres dans un délai maximum de 8 secondes. Son mot est quelconque et peut ne pas commencer par la première lettre fournie. Mais ce mot doit contenir exactement 7 lettres et être correctement orthographié. Si le mot n'existe pas, il est refusé et le logiciel ne renseigne pas le joueur sur les lettres proposées sur la ligne. Si le mot est correct, les lettres bien placées sont coloriées en rouge, et les lettres mal placées entourées d'un cercle jaune. Toute lettre du mot proposé qui est aussi dans le mot recherché est signalée, soit comme bien placée (si elle se situe à la bonne place) et elle apparaît alors en rouge, soit comme mal placée, et elle est alors entourée d'un cercle jaune. Pour une lettre, on ne peut avoir au maximum que le nombre d'occurrences de cette lettre dans le mot qui se retrouvent coloriées (soit en jaune, soit en rouge si certaines sont bien placées).

Les coups s'enchainent alors, le joueur devant proposer un nouveau mot sur la ligne suivante dans le délai maximum supposé (fixé par défaut ici à 8 secondes). Si le joueur trouve le mot en moins de 6 tentatives, il a gagné, sinon, il a perdu, et la bonne réponse s'affiche.

On permettra au logiciel de fixer initialement la taille des mots (ici 7 lettres, mais ce nombre pourra varier de 6 à 10), ainsi que le nombre de coups (tentatives) maximal autorisé (ici 6) et la durée du délai donné au joueur pour entrer une nouvelle proposition (ici 8 secondes).

Pour vérifier l'orthographe des mots proposés, on devra faire des recherches sur les composants TALN gratuits disponibles, afin de réutiliser l'un d'entre eux .

4.13 Organiseur (Agenda)

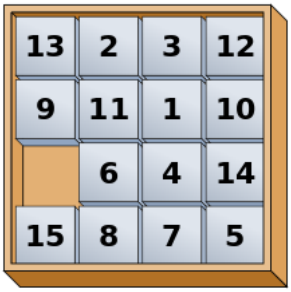
Un organiseur est un logiciel qui permet à un utilisateur de gérer (stocker, consulter) son emploi du temps tout au long de l'année (c'est un Agenda). Le logiciel doit au moins permettre de noter des rendez-vous sur un an, de 6h00 à 23h00, à raison d'un rendez-vous par heure. L'organiseur devra permettre de modifier, stocker, et consulter facilement son emploi du temps.

L'interface sera graphique et devra offrir des facilités de consultation (affichage par mois, avec des flèches permettant d'avancer mois par mois, ou semaine par semaine).

4.14 Jeu de taquin

Le taquin est un jeu solitaire en forme de damier créé vers 1870 aux Etats-Unis. Il est composé de 15 petits carreaux numérotés de 1 à 15 qui glissent dans un cadre prévu pour 16. Il y a donc une case vide. Le jeu consiste à remettre dans l'ordre les 15 carreaux à partir d'une configuration initiale quelconque, en déplaçant à chaque coup un carreau voisin de la case vide (horizontalement ou verticalement), et cela en un minimum de coups. L'interaction avec le damier pour déplacer les carreaux se fera avec la souris.

On consultera la page wikipedia https ://fr.wikipedia.org/wiki/Taquin pour des informations sur les configurations solubles et insolubles. Sachez en tous cas que si vous générez une disposition aléatoire de nombres dans les cases, vous risquez de proposer un jeu sans solution. Mais votre jeu ne doit bien entendu soumettre que des problèmes ayant une solution.



Exemple de Jeu de Taquin

L'utilisateur pourra à tout moment interrompre la partie et sauver la configuration du damier afin de la recharger plus tard.

Amélioration : Le logiciel pourra proposer aussi une version dans laquelle on peut choisir une image de fond à reconstituer. L'image sera découpée en 16 morceaux et 15 d'entre eux seront disposés sur les carreaux. Le jeu ressemblera alors à un puzzle.

On pourra aussi proposer une variante dans laquelle le nombre de carreaux pourra être choisi avant de commencer à jouer.

4.15 Borne de location de films

On s'intéresse à la programmation d'une application permettant d'emprunter des DVD de films (disponible sur des bornes de distribution semblables aux distributeurs

de billets). On pourra interroger la borne concernant la disponibilité d'un film donné. Un film aura pour caractéristiques un titre, un (ou plusieurs) réalisateur, des acteurs, une langue originale et une année de sortie. A un film pourra correspondre plusieurs exemplaires de DVD aux caractéristiques de langues éventuellement différentes. Les DVD pourront être disponibles ou en location avec une date de retour prévue.

L'utilisateur aura accès aux informations concernant les langues audio disponibles, et les langues des sous-titres présentes sur un DVD. Il pourra effectuer des recherches multi critères, sur les films et les DVD disponibles associés. Si un DVD est disponible, il pourra en obtenir la location pour un temps déterminé, et sortir le DVD après communication de son numéro de carte bleue (qui permettra aussi de l'identifier comme emprunteur).

Le logiciel comportera les fonctionnalités suivantes : Recherche de film (avec un ou plusieurs critères, on pourra également rechercher un film par mot clés apparaissant dans le titre), et Recherche de DVD associés (langues) ; Affichage de l'état des DVD associés disponibles ; Emprunt de DVD ; Retour d'un DVD ; Facturation (émission d'une facture indiquant le prix facturé en cas de retour dans les temps, et indication de la somme prélevée en cas de retard. En cas de retard, un certain montant - correspondant au prix de vente du DVD, sera en effet prélevé automatiquement sur la carte bleue.

On utilisera une base de données pour inventorier les films et les DVD associés, ainsi que gérer les emprunts des clients. Les clients seront répertoriés dans la base de données par leur numéro de carte bleue, saisi automatiquement lorsqu'ils se connecteront à la borne.

On ne développera pas l'interface de la borne proprement dite, mais on simulera une connexion d'utilisateur par l'entrée d'un numéro l'identifiant (son numéuro de carte bleue). L'interface sera ici une interface graphique permettant d'effectuer des recherches dans la base de données, d'obtenir une locationn de DVD ou de rendre un DVD. On pourra aussi consulter l'état de ses différents emprunts en cours.

Pour la gestion des films, un utilisateur particulier (l'administrateur) pourra crḿer de nouveaux films et de nouveaux DVD. Il sera aussi habilité à modifier lun film, ou à en supprimer de la base.

4.16 Master Mind

Le but du jeu est de retrouver une combinaison ordonnée de 4 couleurs. Plus précisément, il s'agit de découvrir une rangée (cachée) de 4 billes de couleurs (le nombre de billes d'une rangée et le nombre de couleurs possibles pourra varier en fonction du niveau de difficulté souhaité).

Déroulement du jeu : Une rangée secrète (ou combinaison) est choisie par l'ordinateur. Le joueur qui cherche à la découvrir propose alors sur une ligne une rangée de 4 billes de couleurs.

Le programme lui indique alors (sous forme de points noirs et de points blancs) combien de billes sont bien placées (i.e. sont de la bonne couleur et à la bonne place dans la rangée), et combien de billes sont mal placées (i.e. sont d'une couleur appartenant à la combinaison, mais pas à la bonne place dans la rangée). Pour cela il affiche des points, noir ou blanc en guise de réponse, à côté de la rangée proposée.

- Un **point noir** indique la présence d'une bille bien placée (mais ne précise pas laquelle, i.e. ne précise ni sa couleur, ni son emplacement dans la rangée)
- Un **point blanc** indique qu'il existe une bille d'une bonne couleur qui est mal placée (mais ne précise ni la couleur, ni son emplacement sur la ligne)

A l'aide de ces informations, le joueur peut proposer une nouvelle rangée de billes de couleurs sur une deuxième ligne. L'ordinateur lui répond à nouveau par

un certain nombre de points noirs ou blancs, ce qui lui fournit encore d'autres informations, combinables avec les précédentes. Le joueur peut alors entrer une nouvelle rangée sur une nouvelle ligne, et ainsi de suite. Les lignes restent affichées (avec leurs points blancs et noirs en vis-à-vis) sur un même plateau, et permettent au joueur de faire des hypothèses sur l'appariement des points (noirs ou blancs) et des billes sur chaque ligne. Cela lui permet de raisonner avant de faire une nouvelle proposition.

Au bout de 10 lignes proposées sans découvrir la bonne combinaison, le joueur a perdu, et la combinaison secrète s'affiche. A l'inverse, s'il obtient sur une ligne une réponse de 4 points noirs avant la dixième ligne, c'est qu'il a découvert la bonne combinaison et qu'il a gagné.

On pourra aussi prévoir d'affecter un score au joueur. Dans ce cas, il y aura une base de données de joueurs sauvegardant leurs scores et leur pseudonymes. On aura une architecture Client/Serveur permettant la connexion, l'inscription au jeu, l'enregistrement et la consultation des scores. On peut aussi prévoir l'affichage des meilleurs scores.

Le jeu de Mastermind du site `http://www.litterales.com/jeux` devrait vous aider à comprendre le jeu et à préciser les fonctionnalités du logiciel que vous souhaiterez développer. Parmi celles-ci vous ajouterez la possibilité de modifier le nombre de billes de la combinaison (5 billes aulieu de 4). Le joueur devrait également pouvoir choisir le nombre de couleurs avec lesquelles il doit jouer. (Vous fixerez par exemple un nombre de couleurs maximum, et l'utilisateur pourra choisir de jouer avec un nombre plus petit de couleurs. Une fois le nombre de couleurs fixé, les couleurs seront imposées par le programme et affichées pour que le joueur sache avec quelles couleurs jouer).

Remarque : on peut aussi prévoir un mode de jeu à deux joueurs. Dans ce cas, l'un des joueurs choisira la combinaison à découvrir par l'autre. L'application aura une architecture Client/Serveur ce qui permettra à plusieurs joueurs de se connecter, et l'interface pourra être une interface graphique classique, ou une page web.

4.17 Le jeu Othello

Le jeu Othello, appelé aussi Reversi, se joue sur un damier de 8×8 cases. Les pions ont deux faces de couleurs différentes : l'une blanche, l'autre noire. Chaque joueur choisit une couleur au début du jeu, et le but est de colorier le plus de cases possible de l'échiquier avec sa couleur. Au début du jeu, chaque joueur dispose deux de ses pions dans sa couleur en diagonale au centre du damier. Le joueur qui a les blancs commence alors à jouer en plaçant un pion sur une case vide en respectant les conditions suivantes :

- le pion est placé de façon à former un alignement vertical, horizontal ou diagonal avec un autre pion de même couleur ;
- entre les deux extrémités de l'alignement formé, il ne doit y avoir que des pions adverses (il n'est pas possible de laisser des espaces vides). Les pions de couleur adverse ainsi pris au piège changent de couleur, et c'est au tour de l'autre joueur de jouer.
- Si un joueur ne peut pas jouer, il passe son tour (autant de fois que nécessaire).

Le jeu s'arrête lorsque le damier est plein ou lorsque plus personne ne peut jouer. On compte alors le nombre de cases occupées par chaque couleur. Le gagnant est le joueur dont les pions occupent le plus grand nombre de cases.

Le logiciel développé aura une architecture client/serveur et permettra à deux joueurs connectés sur des machines différentes de jouer sur un même plateau. Le serveur gérera les connexion/inscription des joueurs, le lancement d'une partie, sa

sauvegarde (en cours de partie, pour pouvoir la finir plus tard), le chargement de partie interrompue. L'interface utilisateur proposée sera une interface graphique.

4.18 Le compte est bon

Il s'agit dans un premier temps de permettre à un utilisateur de s'entrainer au jeu télévisé "Des chiffres et des lettres" concernant la partie "chiffres" du jeu. Ensuite, on permettra à deux joueurs de s'affronter, en réseau, sur une série de dix tirages constituant une partie.

Les règles du jeu sont de former un nombre approchant au mieux un nombre compris entre 0 et 999 tiré aléatoirement, en utilisant (au maximum une fois) 6 nombres, tirés également au hasard parmi les nombres suivants : 1, 2, 3, ..., 9, 10, 25, 50, 75 et 100. Les six nombres peuvent être combinés à l'aide des opérations arithmétiques de base : addition, soustraction, multiplication et division (entière).

Ainsi par exemple, pour atteindre le nombre 266 avec les six nombres : 3, 75, 10, 4, 25 et 2, on peut proposer le calcul en trois opérations :

25 + 2 = 27
27 x 10 = 270
270 - 4 = 266

Ce n'est pas la seule solution, car on peut aussi faire d'autres calculs, comme par exemple :

3 x 75 = 225
225 + 25 = 250
250 + 10 = 260
4 + 2 = 6
260 + 6 = 266

Une fois le tirage du nombre à atteindre et des 6 nombres à utilisés affiché, le ou les joueurs ont 45 secondes pour trouver une combinaison exacte, ou un calcul approché. (En mode d'entrainement mono joueur, le joueur pourra modifier ce temps et bénéficier de plus de secondes).

Si un joueur trouve le nombre exact dans le temps imparti, il marque 10 points. Il marque également 10 points si le nombre qu'il obtient est la meilleure approximation possible du nombre cherché. En mode deux joueurs, s'il trouve un nombre approchant meilleur que celui trouvé par son adversaire, il marque seul 7 points. Si les deux joueurs trouvent la même approximation, ils marqueront tous les deux 7 points.

L'ordinateur proposera dans tous les cas un "meilleur" calcul, c'est-à-dire un calcul décomposable en le minimum d'opérations possibles.

L'interface devra être plaisante et aussi conviviale que possible. On pourra s'inspirer de l'application disponible sur tablette intitulée "DesChiffresDesLettres" qui est très réussie. On pourra aussi regarder l'application en ligne du site france3.fr pour apprendre les règles du jeu, mais l'interface est moins agréable.

4.19 Bataille navale

Initialement, chaque joueur place des bateaux de différents types sur une grille. Puis, chacun son tour, chaque joueur essaie de tirer sur les bateaux ennemis en donnant les coordonnées d'une case adverse. Si le tir tombe à proximité d'une case d'un bateau, le programme le signale par un message "bateau manqué" (sauf si la grille n'est pas suffisamment grande, auquel cas il n'y a aucun message).

Si la case atteinte est occupée par un bateau, le programme doit le signaler par un message “bateau de type T touché” (et la case atteinte sera alors marquée spécialement). Si le bateau est entiérement touché (toutes ses cases ont été découvertes), il est alors considéré comme coulé, sa représentation sera modifiée et un message affiché. Le joueur qui réussit à couler tous les bateaux ennemis le premier gagne la partie.

On définira plusieurs types de bateaux (occupant plus ou moins de cases) : torpilleurs, sous-marins, frégates, porte-avions, etc. Le plus simple étant d'avoir des bateaux occupant de une à quatre cases (horizontales ou verticales). On pourra aussi spécifier la taille de la grille, et cela déterminera le nombre et le type de bateaux qu'un joueur devra disposer sur sa grille. Ces informations seront clairement indiquées sur l'interface. La taille de grille par défaut sera de10x10 cases (les cases seront numérotées de 1 à 10 et de A à J, en ligne/colonne). Le joueur devra valider sa configuration de bateaux avant de commencer à jouer. (Pour vous inspirez concernant l'interface utilisateur, vous pouvez consulter le jeu en ligne http ://fr.battleship-game.org ou encore http ://boomboomboat.com).

Le logiciel proposera deux modes de jeu : en mode individuel, le joueur jouera contre l'ordinateur, qui tiendra lieu de second joueur ; en mode 2 joueurs, le joueur se connectera sur un serveur dans l'attente d'un adversaire contre qui jouer.

Le logiciel affichera initialement une interface comportant deux grilles : celle du joueur (sur laquelle il devra disposer ses bateaux) et celle de l'adversaire (initialement vide). Chaque joueur pourra voir les coups portés par son adversaire sur sa propre grille, et les coups qu'il porte lui-même sur la grille de l'adversaire. Lorsqu'un bateau est touché, les cases touchées apparaitront dans un marquage différent.

En mode individuel (mono-utilisateur), le joueur pourra interrompre sa partie en cours contre l'ordinateur, et la sauvegarder pour la reprendre plus tard.

Le logiciel peut être une application lancée sur une machine potentiellement distante de la machine sur laquelle tourne le serveur (architecture client/serveur), ou présenter initialement son interface sur une page web.

4.20 Appli Smartphone : Qui-est-ce ?

L'application permet de jouer au célèbre jeu de société 'Qui est ce ?' sur un téléphone portable (doté d'un système Android), en se connectant au réseau.

Une partie de 'Qui-est-ce ?' se joue à deux joueurs. Chaque joueur dispose d'un plateau sur lequel sont représentés des personnages (normalement 24). Au début de la partie, chaque joueur choisit secrètement l'un de ces personnages. Le but du jeu est alors de deviner le personnage choisi par l'adversaire, en posant des questions sur son apparence physique et sur sa personnalité.

Chaque joueur pose à tour de rôle une question à son adversaire, question pour laquelle il est possible de répondre par oui ou par non. Le premier joueur à deviner l'identité du personnage choisi par son adversaire gagne la partie.

Les principaux objets composant l'application sont :
— des joueurs, acteurs principaux du jeu
— des parties opposant deux joueurs
Chaque partie est constituée :
— d'un plateau contenant la liste des personnages
— d'une interface de dialogue permettant de poser les questions à l'autre joueur
Les tâches principales pouvant être effectuées par un utilisateur seront :
— s'inscrire (se connecter) sur le logiciel et ainsi devenir joueur
— jouer une partie, soit en choisissant un adversaire, soit en acceptant une partie proposée par un autre joueur connecté.

Lors d'une partie, le joueur pourra :
— voir le plateau de jeu correspondant à sa partie et interagir avec (éliminer, choisir ou proposer un personnage)
— poser des questions à son adversaire et lui répondre, quitter l'application.

4.21 Appli Smartphone : Qu'est-ce qu'on mange ce soir ?

Cette application est une application Smartphone, et tourne sur un téléphone muni d'un système Android. Elle permet de trouver des recettes ludiques utilisant des ingrédients particuliers (les restes du frigo ou des placards). On effectue des recherches de recettes (ou de video) en partant de la liste des ingrédients avec lesquels on souhaite cuisiner. On pourra dans cette recherche préciser éventuellement la nature du plat que l'on veut cuisiner : entrée, plat ou dessert.

4.22 Appli Smartphone : Gestion de comptes

Il s'agit d'écrire une appli permettant de gérer les comptes d'une colocation ou de vacances entre amis. Il y a plusieurs utilisateurs, et chacun peut y ajouter une facture de frais collectif. Le logiciel devra utiliser une architecture client-serveur (pour que chacun puisse ajouter lui-même ses factures). On pourra s'inscrire sur l'appli initialement pour adhérer à un groupe, puis chacun y entrera ses frais, et on pourra aussi demander le calcul du solde des comptes, qui indiquera combien on doit d'argent et à qui si l'on a un solde négatif, et dans le cas inverse, qui doit nous rembourser quelle somme.

Exemple : On a des vacances partagées entre Albert, Bertrand, Christophe et Daniel. Daniel a payé 150 d'essence. Bertand a payé 350 de logement. Christophe a payé 300 de courses diverses.

La balance totale est donc de -50 pour Daniel, 150 pour Bertrand, 100 pour Christophe and -200 pour Albert (la somme est bien entendu toujours 0). Donc Albert doit verser 200 et Daniel 50 pour rembourser Bertrand et Christophe. L'appli pourra proposer à Albert de donner 150 à Bertrand et 50 à Christophe, et à Daniel de donner 50 à Christophe.

4.23 Appli Smartphone : L'anglais tout de suite !

Le but de cette appli est de permettre à un touriste franais se rendant dans un pays anglophone de se débrouiller rapidement en anglais. Une telle appli peut rivaliser avec les sections des guides de tourisme qu'on achète dans les aéroports - lesquels contiennent traditionnellement un petit lexique, des phrases utiles, une section sur comment prononcer les mots, former les nombres, etc. Ici, l'appli Smartphone poursuit un but analogue, c'est-à-dire faire apprendre rapidement à un locuteur du franais de quoi se débrouiller dans la langue du pays visité. Mais pour réaliser cela, au lieu d'un lexique, on trouvera la liste des mots "que l'on connait déjà sans le savoir", c'est-à-dire la liste des mots qui sont les mêmes, en franais et en anglais. Ainsi il n'y aura rien à apprendre, si ce n'est lire plusieurs fois cette liste des mots communs aux deux langues. Et rien n'empêche bien entendu de proposer d'autres fonctionnalités (comme celles énumérées plus haut).

L'architecture choisie devra permettre de faire des extensions (multilingues). La langue choisie ici est l'anglais, mais si vous préférez l'italien, ou une autre langue

voisine du franais, il n'y a pas de problème. Il s'agit dans ce développement de mettre en place les grandes lignes de l'architecture finale d'une l'appli multilingue. Cette architecture doit permettre l'intégration facile (voire immédiate) d'autres langues que l'anglais, et chaque groupe de fichiers dans une langue donnée doit permettre de "créer" une nouvelle appli.

Pour vérifier que tout cela fonctionne bien, on pourra développer l'appli pour l'anglais, et vérifier l'intégration d'un noyau pour l'italien. Ce point qui concerne l'intégration des extensions de l'appli, doit être compris comme une contrainte concernant son architecture.

Fonctionnalités attendues :

A. La principale fonctionnalité est l'affichage de listes de mots : La liste des mots communs aux deux langues (rendez-vous, justice, train, hot-dog, wc, dance, chance, etc. pour l'anglais) ; la liste des mots anglais faciles à apprendre pour un franais (consciencious, intellectual, dancing, etc.) ; celle des mots que tous les franais connaissent (go, self-made-man, happy, new, year, etc) ; ensuite la liste des faux-amis avec leurs traductions ; la liste des prépositions, postpositions, adverbes de temps, et de manière, si nécessaire.

B. Autres fonctionnalités à ajouter éventuellement :

— incorporer un logiciel de prononciation de mots, comme le font aujourd'hui les petites machines dédiées de traduction automatique.
— Incorporer des jeux comme le pendu (avec la liste des mots connus par exemple) ou d'autres jeux de mots (mots-croisés par exemple).
— fournir (comme les guides touristiques) la traduction de quelques phrases prêtes à l'emploi : "O se trouve la gare ?", "Combien a cote", "O est l'hpital", etc.
— afficher la liste des premiers nombres et donner les règles de formation des nombres.
— aides diverses : conjugaisons de verbes, syntaxe minimale (syntaxe du "petit nègre" ?), liste des verbes courants les plus utilisés (être, avoir, faire, etc.) ou listes des prépositions, postpositions, principaux adverbes, etc.

C. Contexte fonctionnel : on souhaite une architecture ouverte permettant de réutiliser le travail des groupes des années antérieures, et permettant aussi l'intégration facile d'une nouvelle langue.

D. Environnement logiciel et matériel : l'application sera développée en java, sous Android, avec Android Studio. Prérequis techniques : java et Android studio. (Si on utilise une base de données, il faut connatre le langage SQL, mais l'intégration n'est pas très compliquée en java).