Synthèse Vocale
et
Logiciel Pédagogique

Rapport de stage

Documentation version 1.0 créée le 2 août 2006

Dernière mise à jour le 14 septembre 2006

Auteur :

Vincent BOUCHET vbouchet.@gmail.com
Master Sciences et Technologies, mention Systèmes Intelligents
spécialité : Informatique et Traitement de l'information

Tuteurs :

Entreprise : Sébastien DELCROIX, Responsable Développements & Exploitation
Université : Jacques LOTTIN, Professeur des Universités

Résumé

Mon stage de fin d'études s'est déroulé au sein du CRI74. Il s'inscrit dans le cadre d'un projet de développement d'outils destinés à faciliter la création de logiciels éducatifs. Le domaine choisi comme exemple est celui de l'apprentissage de la lecture. J'ai commencé à participer à ce projet, avant le stage, pendant mon projet de fin d'études.

La première partie de ce stage est l'analyse, et le développement d'un prototype de framework pour des exercices de lecture. De plus, le domaine étant celui de la lecture, le synthétiseur vocal conçut en projet à été amélioré, avec l'ajout notamment d'un catégoriseur de Brill, et une gestion de la prosodie temporelle. Enfin, nous disposons maintenant des bases, pour poursuivre le travail sur la synthèse vocale en ajoutant un moteur vocale.

Mots clés :

Logiciels éducatifs, Framework, Synthèse Vocale, Catégoriseur de Brill, Squeak

Remerciements

Je remercie M. Sébastien Delcroix, le responsable développement du CRI pour m’avoir admis à ce stage.

Je remercie également M. Jacques Lottin qui a accepté d’être le professeur tuteur responsable de mon stage, et M. Stéphane Ducasse qui a suivi techniquement mon stage.

Je souhaite également remercier l'ensemble du CRI 74 pour m’avoir reçu pendant ces six mois dans une ambiance de travail agréable.

Enfin, je remercie la société Ontologos pour son aide dans la réalisation d'un catégoriseur de Brill.

Table des matières

Ajouter un tableau de participants

Veuillez vous reporter au lexique pour les mots suivi du caractère *.

Index des illustrations

1 – Introduction

Dans le cadre de ma dernière année de Master Informatique et Traitements de l'information, j'ai été amené à effectuer, du 1er mars au 31 août un stage en entreprise. Ce stage s'est déroulé à Archamps au sein du CRI de Haute Savoie (centre de ressources informatiques). Cet organisme a pour mission le développement des nouvelles technologies en Haute Savoie. La personne qui m'a accueilli est M. Sébastien DELCROIX, le responsable développement du CRI. Il a assuré mon encadrement durant ce stage.

Mon stage s'inscrit dans le projet, du CRI, de réalisation d'outils d'aide à la création de logiciels pédagogiques. L'objectif de mon stage était de faire un logiciel de démonstration pour le domaine de l'apprentissage de la lecture, d'où la nécessité d'une synthèse vocale. Une première approche avait été réalisée au cours de mon projet de fin d'études. En participant à ce stage, j'étais désireux de continuer à travailler sur un nouveau domaine découvert en projet ainsi que d'être en relation avec des non spécialistes en informatique.

Tout d'abord, nous allons nous intéresser au contexte de travail, c'est-à-dire au CRI, aux objectif et au déroulement du projet. Ensuite nous verrons le travail réalisé sur le logiciel développé, puis, dans la dernière partie, nous étudierons la synthèse vocale.

2 – Contexte de travail

2.1 – Présentation du CRI 74

2.1.1 – Introduction

Le Centre de Ressources Informatiques de Haute-Savoie (CRI74) est situé à Archamps, près de Genève, sur la frontière Suisse. Il dépend de l'Agence Économique Départementale, association loi 1901, chargée de la promotion économique du département et subventionnée par le Conseil Général. Le CRI74 a pour mission de sensibiliser les acteurs du secteur public à l'informatique en réseau et favoriser une diffusion massive et équitable des usages liés aux TIC.

Pour mener à bien ce rôle, le CRI74 est devenu le fournisseur d'accès et de services Internet des structures publiques départementales. Pour aider ses utilisateurs à devenir autonomes dans leur utilisation informatique quotidienne, il a mis en place de nombreux services par l'installation, au sein de leur structure, de serveurs de communication. De plus, des formations sont proposées par le CRI74. Cela permet aux utilisateurs de se familiariser avec les différents outils informatiques. Ainsi ils administrent leur propre service de messagerie et d'intranet par la distribution Linux que le CRI74 développe : PingOO.

Aujourd'hui le CRI74 compte près de 60 000 utilisateurs, avec 300 serveurs PingOO déployés sur le département. Site web : http://www.cri74.org

2.1.2 – Historique

En 1994, bénéficiant de la proximité du CERN, inventeur du Web, le Département présente un Internet, encore balbutiant en France, lors de la Fête de la Science. Parmi les publics, des acteurs de l'Éducation saisissent immédiatement le potentiel pédagogique de cet outil et interpellent le Conseil Général pour qu'il les aide à en bénéficier. Le CRI74 est né, jouant avant l'heure le rôle de FAI, dont la notion commerciale n'existe pas encore en France.

Il imagine alors les services à mettre en place pour aider ce public, qui sera rapidement élargi à l'ensemble des structures départementales.

Pour cela, il joue à la fois le rôle de prestataire d'accès au réseau Internet et celui d'un laboratoire de recherche et d'intégration. Les activités de recherche sont tournées vers la mise au point de solutions informatiques permettant d'utiliser les possibilités des réseaux à des fins professionnelles, de manière fiable et au coût le plus bas possible.

2.1.3 – Les Missions du CRI74

❖*Fournisseur d'accès*

Le Centre de Ressources Informatiques de la Haute-Savoie est avant tout un prestataire de services se rattachant à l'informatique. En effet, le CRI74 a pour mission de servir de fournisseur d'accès Internet à divers intervenants du service public, principalement l'Éducation Nationale mais aussi les mairies, les offices du tourisme ou encore le conseil général de Haute-Savoie

❖*Formation*

Les publics du CRI74 sont de manière générale démunis face aux TIC, et possèdent peu de moyens internes pour une gestion autonome. La formation est donc un service essentiel dans la mission de diffusion des usages informatiques du CRI74. Elle permet aux utilisateurs de

s'approprier les outils, et permet au CRI74 de maîtriser le service d'assistance. Ces formations visent à doter les organismes de compétences, qui leur permettent à leur tour de dispenser les mêmes formations en interne.

Les thèmes :

- Internet Pratique
- Spip
- OpenOffice.org
- PingOO

Des formations sont ensuite dispensées « à la carte », en fonction de la demande des utilisateurs. Comme pour l'ensemble des services proposés par le CRI74, seules les demandes émises par le plus grand nombre aboutiront.

Enfin, la formation est le seul service payant du CRI74.

En effet, beaucoup de personnes dans les secteurs ciblés, comme les enseignants de l'éducation nationale, ne sont pas assez formées pour appréhender correctement les nouvelles technologies. Or, il est vital que les enseignants puissent avoir un niveau correct dans le maniement de ces technologies pour développer ce savoir chez leurs élèves, par la création de site web notamment. Ces sites sont tout d'abord créés dans les réseaux locaux des établissements puis mis en ligne au CRI74.

❖*Recherche et développement*

L'activité de recherche et de développement est essentielle : le CRI74 agit sur les usages, il est donc normal que ceux-ci se développent et que les utilisateurs deviennent de plus en plus précis dans leurs demandes et exigeants dans leurs attentes. Il faut alors déployer rapidement et économiquement de nouvelles solutions répondant à ces demandes.

Le CRI74 a initié le projet Edres 74 (Éducation Réseau 74) qui a débuté en avril 1995 et qui concerne la connexion à Internet de tous les établissements scolaires du département de la Haute-Savoie. Le but avoué est, ici, d'amener les nouvelles technologies de l'information à la portée de tous, élèves comme enseignants, pour préparer le département à son entrée dans la société de l'information.

❖*L'administration et l'assistance déléguées*

Étant donné le nombre croissant d'établissements connectés à Internet par le CRI74, il devient plus difficile d'assurer convenablement cette mission ainsi que le support technique et administratif nécessaire au bon fonctionnement du réseau, comme la création des comptes utilisateurs... Le CRI ne dispose plus d'assez d'effectifs par rapport au nombre croissant d'établissements à desservir.

Il a donc été décidé de déléguer toute la partie administrative concernant la hotline, ainsi que la gestion des utilisateurs et des serveurs de communication.

Le CRI74 travaille en partenariat avec les structures fédératrices de ses utilisateurs et leur délègue certaines actions. L'Association des Maires de Haute-Savoie pour les collectivités, l'Inspection Académique pour l'éducation, l'Agence Touristique Départementale pour le secteur du tourisme, sont autant de relais et de partenaires. Seul un fonctionnement cascadé permet au dispositif de fonctionner.

2.1.4 – Équipe

Voici un organigramme illustrant l'organisation au CRI74, suivi d'un court descriptif des rôles techniques.

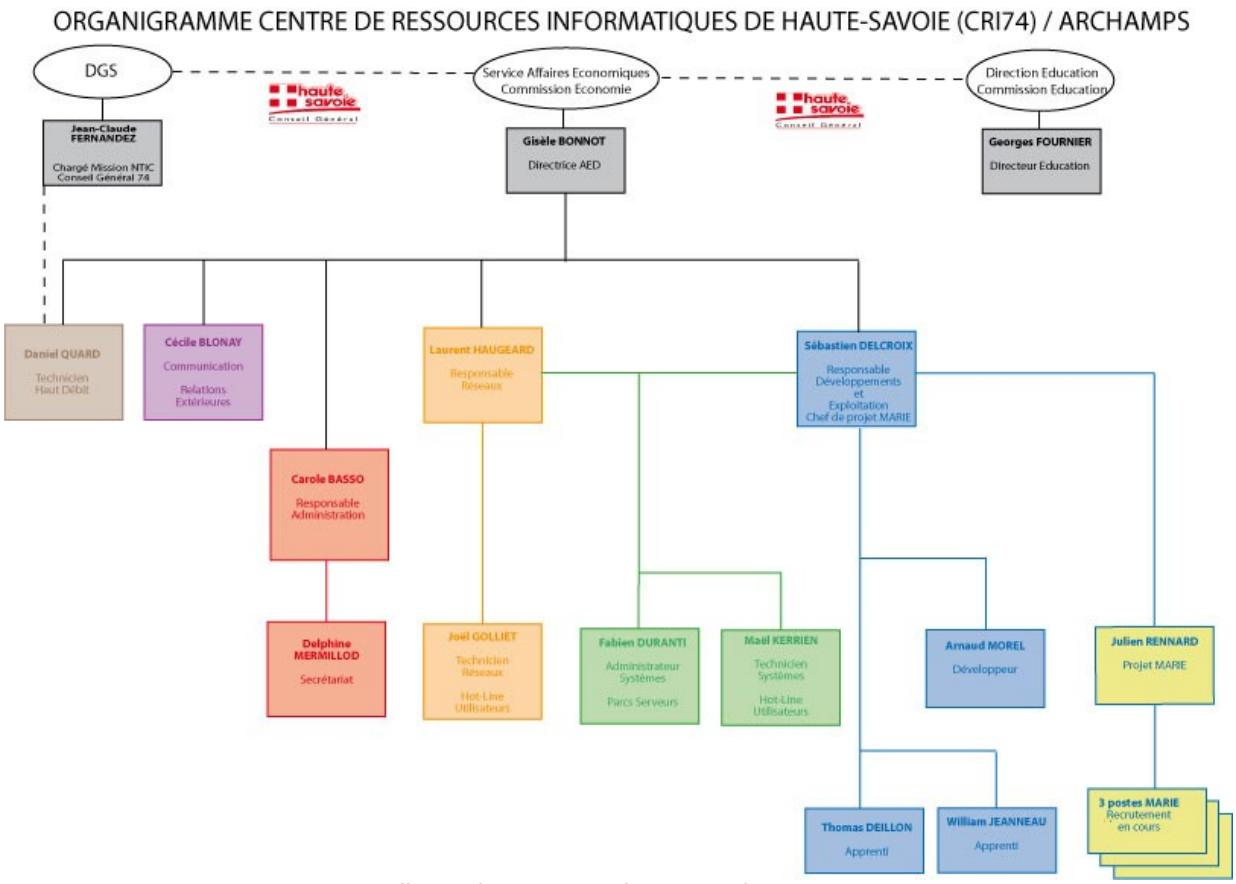


Illustration 1: Organigramme du CRI74

❖*Administrateur Réseau*

Disponibilité (empêcher toute interruption de service) et sécurité (empêcher toute tentative d'intrusion ou perte de données) des réseaux informatiques empruntés par les utilisateurs pour aller sur Internet depuis leur structure ;

Interconnexion du réseau du CRI avec les autres réseaux (Amplivia, ...) ;

Veille technologique.

❖*Technicien réseau et technicien système*

Disponibilité et sécurité du réseau (technicien réseau) et des machines (technicien système) ;

Test de nouvelles solutions et intégration de nouveaux services ; Assistance aux utilisateurs (hotline).

❖*Administrateur système*

Disponibilité (empêcher toute panne) et sécurité (mise en place d'antivirus, protection des serveurs) des machines proposant les services et les logiciels aux utilisateurs (messagerie, site web, ...) ;

Mise à jour des services, mise en place et intégration de nouveaux logiciels sans interruption de service ;

Veille technologique.

Technicien haut-débit

Conduite du projet de réseaux à haut-débit et suivi du projet téléphonie mobile.

❖*Développeur*

- *Développement de PingOO ;*

Intégration de solutions au système informatique du CRI74, pour les utilisateurs ou pour besoins internes.

- *Responsable développements*

Étude de faisabilité et rédaction des cahiers des charges concernant de nouveaux outils demandés par les utilisateurs ; Cohérence entre l'outil à développer, les contraintes liées au réseau et les contraintes liées au système.

- *Responsable cohérence exploitation*

L'exploitation est la réunion des réseaux et des systèmes : étroitement liés, les deux domaines doivent être architecturés et construits en cohérence :

Étude de la mise en place de nouveaux services répondant aux besoins des utilisateurs et aux contraintes techniques ou économiques ;

intégration des nouveaux services à l'existant sans causer de dysfonctionnements auprès des utilisateurs ;

Coordination entre réseaux et systèmes et définition des stratégies de déploiement ou de mises à jour ;

2.2 – Formation sécurité : challenge Sécuritech

Au CRI74, la sécurité a toujours été au centre des préoccupations. La nature de son activité le rend particulièrement exposé aux dangers extérieurs. Aussi, nous avons été encouragé à participer à un challenge sécurité.

❖*Présentation*

Le challenge Securitech est un concours de sécurité informatique en ligne. Il s'est déroulé du samedi 29 avril au jeudi 18 mai. Dans le cadre de mon stage, j'ai participé challenge 2006 avec d'autres stagiaires du CRI74, c'était une façon de se sensibiliser à la problématique de la sécurité informatique.

Durant ces trois semaines, une quinzaine d'épreuves on été proposées, couvrant différents aspects de la sécurité tels que les failles web et applicatives, l'injection SQL*, l'analyse réseau, le reverse engineering*, la cryptanalyse*, la stéganographies* ou les forensic*.

Chaque épreuve résolue apporte un certain nombre de points qui permettent le classement des participants. Une épreuve dispose d'un nombre inconnu de validateurs. Pour augmenter la difficulté du challenge aucune information n'est donnée sur les moyens de les valider. Dans la majorité des cas on dispose seulement d'une trace réseau comme point de départ. Il faut ensuite chercher tous les moyens possibles de valider l'épreuve.

Le déroulement des épreuves est expliqué dans l'annexe : Déroulement du challenge Sécuritech

❖*Bilan*

Quelques niveaux étaient bien trop complexes et très peu de participants les ont résolus. Ils nécessitaient des connaissances très complètes (Exemple : casser un cryptage en RSA 1024* pour trouver la clé d'un texte crypté en AES256*).

Personnellement, le challenge m'a permis une réelle prise de conscience de l'existence des

problèmes de sécurité en informatique. Ceux-ci sont encore trop peu pris en compte pendant le développement de logiciel, et pourtant un formulaire avec des accès à une base de données peuvent mettre en danger la stabilité du programme, la confidentialité des données, ou même du réseaux. Nous en avons déjà eu une première approche pendant notre formation avec une conférence, la mise en pratique de démarches que pourrait avoir des individus malveillants permet de mieux s'y préparer et de pouvoir y faire face.

2.3 – Objectifs du stage

❖*Contexte*

Actuellement, même si l'informatique est présente dans les écoles, elle n'est pas toujours utilisée à bon escient. L'utilisation de Linux a permis de résoudre les problèmes de sécurité et de stabilité.

Cependant, probablement en raison d'un manque de volonté de la part des développeurs il manque beaucoup d'applications pédagogiques sous Linux.

Le CRI a donc pour objectif le développement d'un certain nombre de logiciels en open source, en espérant créer ainsi une émulation. Afin de faciliter et encourager les développements, il est envisagé de créer un framework* pour la réalisation d'applications pédagogiques.

Afin de convaincre de l'utilité du projet, il a été décidé de réaliser une démo d'un logiciel pédagogique avec Squeak durant mon stage. Le domaine de l'apprentissage de la lecture a été choisi comme cadre. J'avais montré la faisabilité du logiciel, et développé un prototype de synthèse vocale au court d'un projet de fin d'études. En effet, une application d'apprentissage de la lecture nécessite une synthèse vocale.

❖*Objectifs*

L'objectif du stage était de poursuivre et de terminer le travail réalisé en projet de fin d'étude : développer un logiciel éducatif pour l'apprentissage de la lecture. Ce logiciel devait être un logiciel de démonstration, visant à fabriquer des outils de création de logiciel pour les enseignants.

Aussi, il s'agissait donc, d'une part, de voir si il était possible de trouver un méta modèle* pour les exercices de lecture et, d'autres part, de développer une interface graphique adaptée aux besoins des instituteurs. Le domaine de l'apprentissage de la lecture servant d'exemple, il fallait également perfectionner la synthèse vocale.

Cependant, durant le stage, le CRI a décidé de réorienter le projet de développements de logiciels pédagogique. En effet, le CRI est en contact avec M. Hilaire Fernandes, un professeur de mathématique, qui est en train d'effectuer une thèse sur les logiciels pédagogiques et les EIAH*. L'éducation, la pédagogie sont les domaines de M. Fernandes, et, actuellement la communauté du logiciel libre ne dispose pas d'une synthèse vocale. Il était donc logique de travailler en partenariat plutôt qu'en concurrence.

Aussi, il a donc été décidé de se concentrer sur la plus grande valeur ajouté que pouvait apporter ce stage au monde du logiciel libre et de Squeak : la création d'une synthèse vocale pour le français. La réalisation d'un logiciel pédagogique est passé au second plan.

2.4 – Déroulement du stage

2.4.1 – Plannings prévisionnels :

Le planning initiale du stage était le suivant :

	Mars	Avril	Mai	Juin	Juillet	Aout
amélioration de la synthèse vocale,						
sélection de trois types d'exercices						
programmation d'un prototype						
Méta modélisation						
Ajout de fonctionnalités sur l'application						
Documentation, Intégration, rapport.						

Cependant, comme nous l'avons vu il a été décidé de recentrer le stage sur la synthèse vocale. De plus, nous avons été encouragé à participé au challenge Securitech. Le planning a donc été revu fin mars :

	Avril	Mai	Juin	Juillet	Aout
Participation au challenge Sécuritech					
Logiciel pédagogique (méta modélisation, protoype)					
Poursuite des améliorations de la synthèse vocale					
Étude et remplacement de la Dll Mbrola					
rédaction du rapport					

La poursuites des améliorations de la synthèse vocale a été un peu plus longue que prévue. Le temps planifié nécessaire pour remplacer la Dll Mbrola était insuffisant pour la réalisation complète .

Le planning réel a donc été le suivant :

	Mars	Avril	Mai	Juin	Juillet	Aout
amélioration de la synthèse vocale,						
sélection de trois types d'exercices						
Challenge Sécuritech						
Méta modélisation, prototype						
Étude pour replacement de la Dll						

	Mars	Avril	Mai	Juin	Juillet	Aout
Mbrola						
rédaction du rapport						

3 – Logiciel Pédagogique

3.1 – Projet

3.1.1 – Contexte

Le fonctionnement d'une salle de classe, dans la cas de l'utilisation de logiciel, a été étudié pendant le projet de fin d'études avec une visite dans une école. Une vingtaine d'élèves travaillent de manières individuelles sur un ordinateur. Ils sont encadrés par deux personnes, qui vérifient leur sérieux et les aident lorsqu'ils ont des difficultés.

L'institutrice nous a également expliqué les points essentiels que doit posséder un logiciel pédagogique pour l'apprentissage de la lecture :

- Il est très important d'avoir un suivi visuel sur l'écran du résultat de l'exercice que l'élève est en train d'effectuer. Cela permet à l'institutrice de contrôler les élèves en difficulté.
- L'interface doit être simple.
- Enfin, il est important de conserver une trace écrite, notamment pour montrer les progrès des enfants à leurs parents.
- La synthèse vocale peut être utilisée tout au long de l'application, pour faciliter l'utilisation du logiciel et améliorer le contenu des exercices.

3.1.2 – Présentation

L'analyse réalisée au court du projet de fin d'études à montrée qu'il existe un public assez large susceptible d'utiliser l'application. Celle-ci se devait donc de répondre aux attentes de chacun. Quatre types d'utilisateurs ont donc été définis. De plus, pour faciliter la lisibilité des données d'un exercice, celles-ci sont mises dans des fichiers de configuration.

On peut donc schématiser l'application est les utilisateurs de la manières suivante :

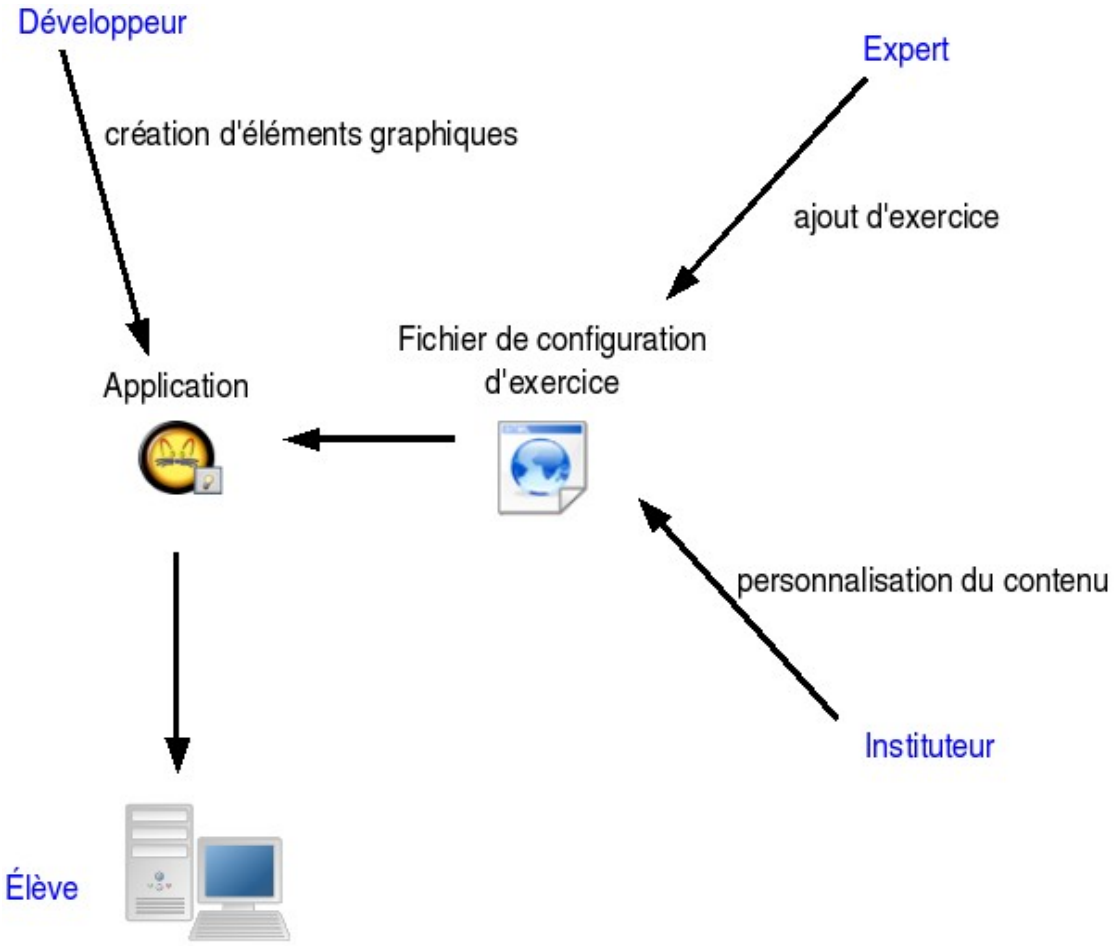


Figure 1: fonctionnement de l'application

Les utilisateurs sont :

- L'élève : il s'agit de l'utilisateur standard, et final du logiciel. Il peut effectuer les différents exercices définis par l'instituteur.
- L'instituteur : c'est un utilisateur avancé. En effet, il peut modifier le contenu des exercices, afin des les adapter au contenu de son enseignement. Pour cela une interface graphique lui permettra d'affecter les valeurs désirées au fichier de paramétrage des exercices.
- L'expert : la principale différence avec l'instituteur, c'est son implication. En effet, l'expert a la possibilité de choisir la suite d'exercice qu'effectuera l'élève, parmi les exercices qui ont été programmés dans l'application. Il peut créer les fichiers de paramétrage des exercices (et il peut aussi y mettre des valeurs ou les laisser en « blanc »
- Le développeur : c'est un informaticien. Le logiciel doit être structuré de manière ouverte, afin de permettre et de faciliter la programmation est l'ajout de nouveaux types d'exercices.

Pour faciliter le paramétrage et la lecture des données du logiciel, elles sont stockées dans un fichier XML*, et non dans Squeak.

Comme nous l'avons déjà vu dans le planning, la création d'interface graphique pour

l'instituteur et l'expert a été abandonnée au profit de la synthèse vocale.

3.1.3 – Méthode de travail

Un des objectifs du stage était de voir s'il était possible de trouver un méta-modèle pour des exercices de lecture. Ne connaissant ni la, ou les, librairie(s) graphique(s) sous Squeak, ni la structure d'un exercice il a donc été choisi de commencer par le développement de « brique » logicielle. L'idée étant d'étudier le domaine à travers ces premiers développements et de trouver comment méta-modéliser les exercices de lecture.

Ainsi, le développement a commencé, par une montée en compétence sur le développement d'interface graphique avec Squeak. Ensuite il a continué par le choix de l'apparence générale de l'application ainsi que des trois exercices représentatifs mettant en valeur les possibilités graphiques de Squeak.

3.2 – Application

Les choix techniques pour développer une interface graphique avec Squeak sont simples. En effet, il n'existe actuellement qu'un seul framework disponible : Morphic. (l'autre framework, tweak, est en cours de développement).

3.2.1 – Morphic

Morphic a été développé à l'origine pour le langage Self avant d'être porté sous Squeak. Le nom « Morphic » vient du mot « Morph » qui signifie « forme » en Grec,

Une morph, est le niveau central d'abstraction de Morphic. C'est un objet qui possède une représentation visuelle qui peut être reprise et déplacé. Elle peut être composée de plusieurs sous morphs.

Une morph peut également :

- effectuer des actions en réponse à des entrées de l'utilisateur.
- effectuer une action quand une morph est déposée sur elle, ou quand elle est déposée sur une autre morph.
- effectuer une action à interval régulier
- contrôler le placement et la taille de ses sous morphs.

Morphic offre de grande possibilités pour le développement d'une interface graphique. (cf Annexe : Exemple de Morph).

3.2.2 – Apparence générale

Les élèves travaillent individuellement sur le logiciel, avec une institutrice qui les surveille, et les aide quand ils sont en difficulté. Aussi, il est indispensable d'avoir un suivi visuel sur l'écran du résultat de l'exercice que l'élève est en train d'effectuer. L'affichage du résultat des exercices précédents indique à l'enseignante si l'élève a eu des problèmes (ou si il a travaillé sérieusement ou non).

De plus, les élèves travaillant de manière individuel, l'énoncé de l'exercice doit être lui aussi

affiché. La forme choisie est donc la suivante :

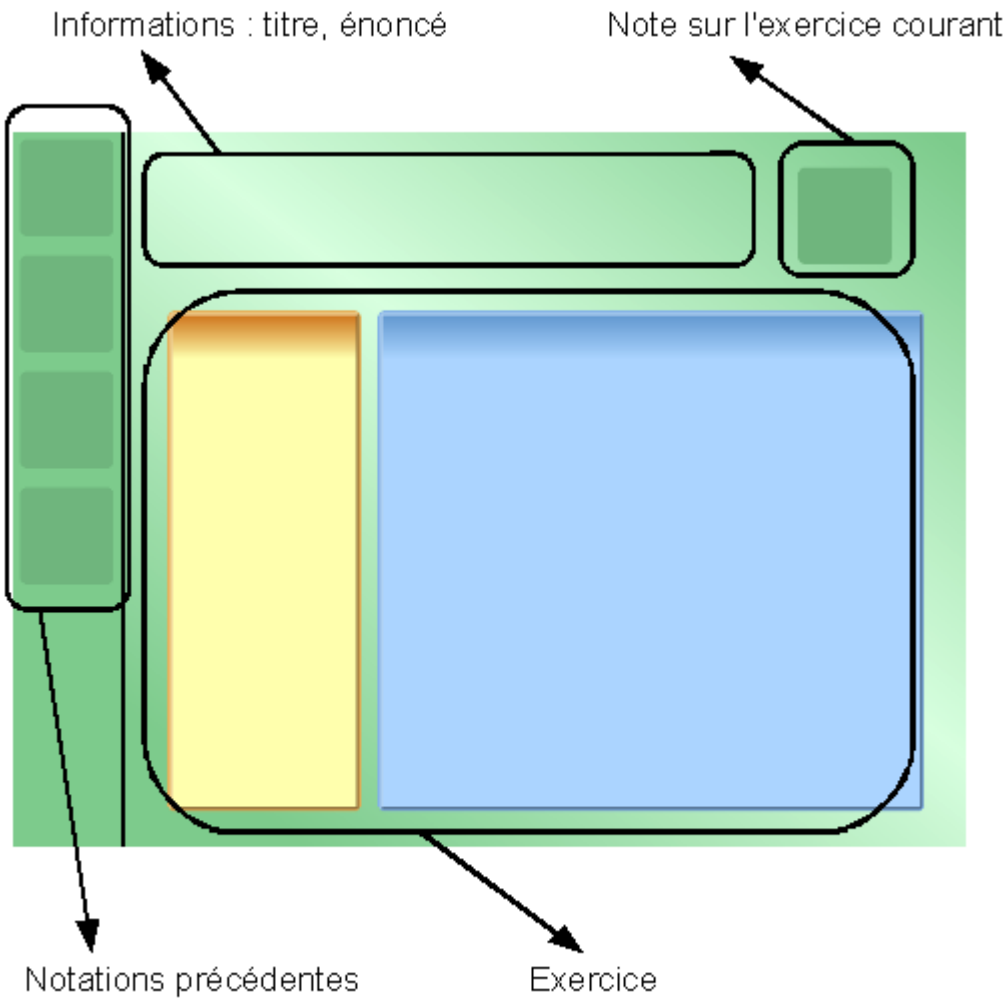


Figure 2: Apparence générale de l'application

Il existe quatre zones différentes :

- Exercice : c'est la zone principale, dans laquelle l'élève effectue l'exercice.
- Informations : cette zone contient les informations nécessaires pour effectuer l'exercice, comme l'énoncé.
- Note : affiche la note actuelle de l'exercice qui est en train d'être réalisé.
- Notations précédentes : c'est la place réservée aux notes des exercices précédents.

Le système de notation est une partie importante, il se devait d'être suffisamment précis, et lisible pour que quelqu'un qui passe derrière le poste de travail puisse le comprendre immédiatement.

3.2.3 – Notations

La solution adoptée est similaire à celle employée par le logiciel Lectra mini étudié en projet. Il s'agit d'une image qui change au fur à mesure que l'élève fait des erreurs. Par exemple, dans Lectra mini, un arbre dont les feuilles vertes tombaient au fur à mesure indiquait à l'enseignant le nombre d'erreurs.

Ce système est visible, et suffisamment précis, en effet, une image se voit plus facilement que du texte, et on est dans le contexte de l'apprentissage : la note existe seulement pour montrer des difficultés, et non pour évaluer l'élève. Ce système est aussi beaucoup plus ludique est adapté aux enfants.

Comme l'application était destinée à être utilisé sous Linux, « l'évolution » d'un manchot (la mascotte de Linux)a été préférée. Les exercices sont très courts, cinq images sembles suffisantes :

Figure 3: Évolution de la représentation de la notation

3.2.4 – Exercices développés

Le but initial du logiciel était de faire une démonstration des possibilités offertes par Squeak pour le développement de logiciel pédagogique. On peut donc définir deux contraintes dans le choix de ces exercices :

- D'une part, les exercices choisis doivent mettre en valeur graphiquement Squeak.
- D'autres part, ils devaient être représentatifs des exercices, (informatique ou non) qui sont actuellement utilisés par les instituteurs

Après une études de plusieurs logiciels pédagogiques utilisés pendant le projet de fin d'études, suivie d'une analyse de plusieurs méthodes écrites, trois exercices fréquemment utilisés ont été sélectionnés.

❖*Le texte à lire.*

L'enfant lit à haute voie un texte étudié pendant une leçon. Cet exercice est un exercice d'entraînement personnel à la lecture. Il est très utile pour les enfants, car au court d'une leçon, ils n'ont pas tous le temps de s'entraîner à lire le texte a haute voix.

En cliquant sur un mot, il le faire lire par la synthèse vocale. Cela lui permet d'une part, de s'aider pour lire des mots difficiles, et d'autres part de s'auto-évaluer.

Cet exercice met surtout en avant l'apport de la synthèse vocale, et la facilité d'utilisation de celle-ci dans l'interface graphique.

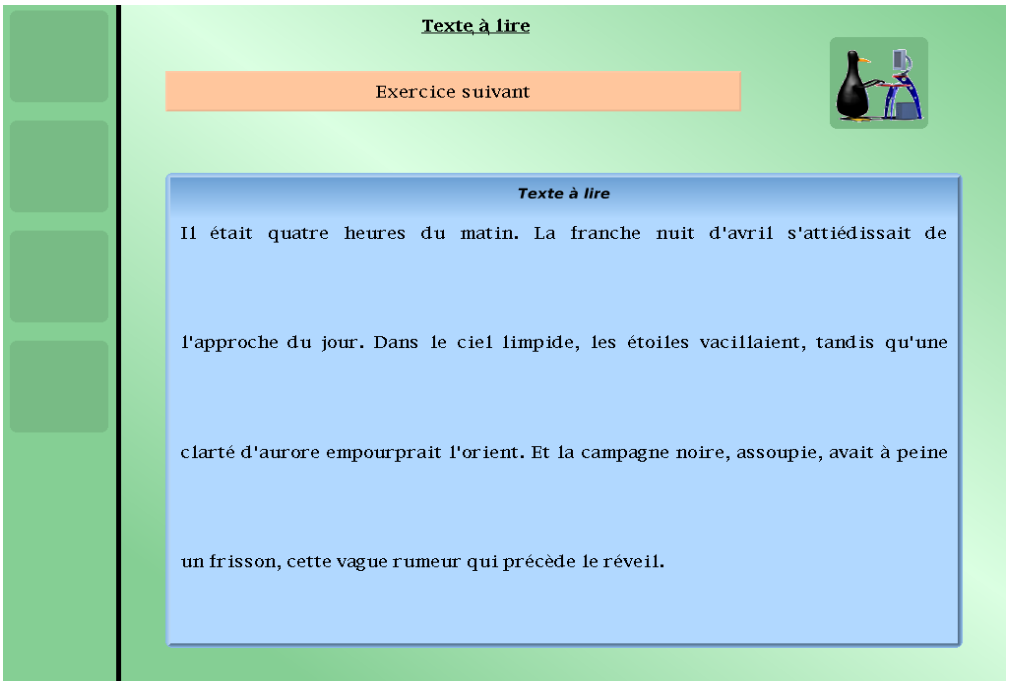


Figure 4: Exemple d'exercice : "texte à lire"

L'élève qui décide quand il passe à l'exercice suivant.

❖*Les phrases à relier*

L'objectif est de relier un début de phrase avec sa suite. Les phrases sont extraites de textes que l'élève a étudié pendant une leçon avec l'institutrice.

Pédagogiquement cet exercice met en avant la compréhension du sens d'une phrase pour l'élève. Pour faciliter cette compréhension, chaque morceau de phrase est lu lors d'un clic de souris : c'est donc l'élève qui détermine le niveau de difficulté.

D'un point de vue graphique, cette exercice montre qu'il est possible de relier des composants

entre eux.

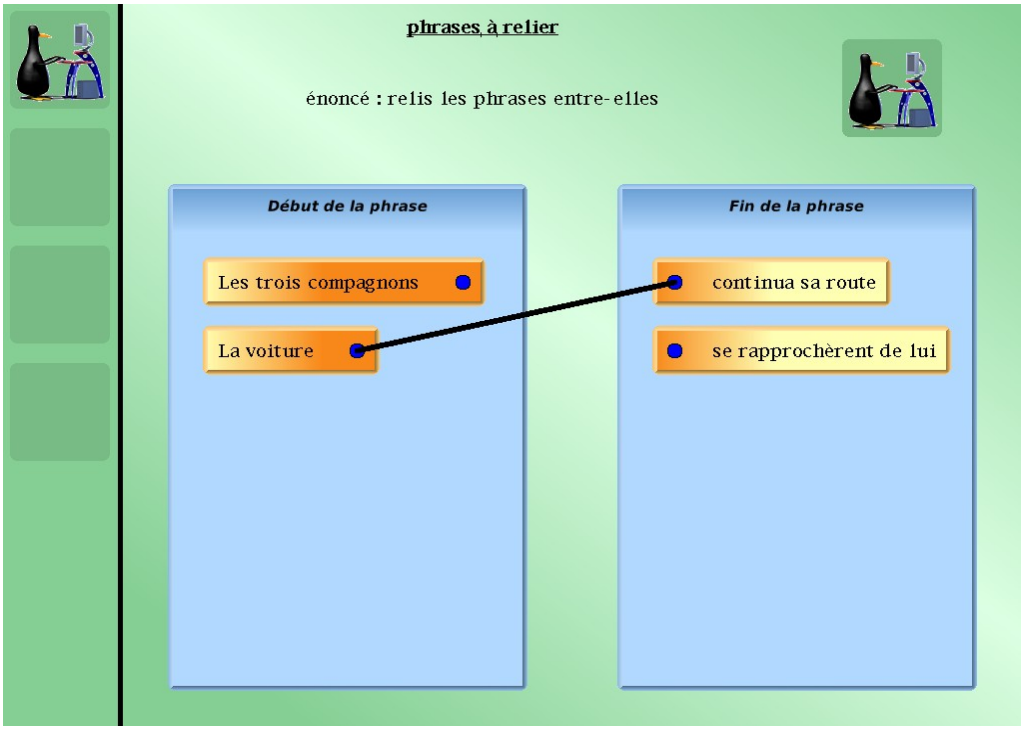


Figure 5: Exemple d'exercice : "phrases à relier"

L'élève passe à l'exercice suivant quand l'exercice est terminé correctement.

❖*Le texte à trou*

Il s'agit de compléter les « trous » d'un texte en prenant les mots disponibles dans la liste à gauche, avec un « glisser-déposer ». Comme pour le premier exercice, le texte utilisé a déjà été étudié auparavant.

Cet exercice est un exercice de compréhension. La synthèse vocale apporte une assistance à l'enfant en lisant un mot du texte lors d'un clic. Lorsqu'il a complété tous les « trous » les étiquettes mal placées sont supprimées et remises dans la liste de gauche.

Enfin, cet exercice met en avant, d'une part la possibilités d'effectuer des glisser déplacer avec Squeak et d'autres par la capacité de l'interface graphique à se réorganiser

automatiquement.

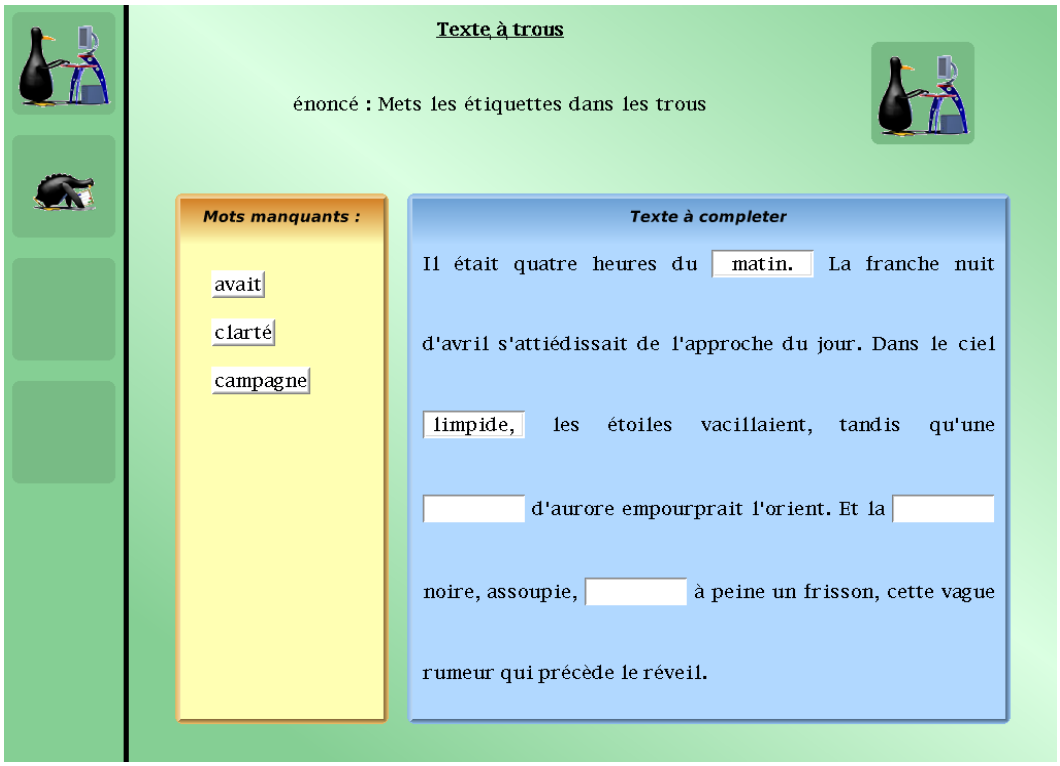


Figure 6: Exemple d'exercice : "texte à trous"

Ici aussi, l'élève passe à l'exercice suivant quand l'exercice est terminé correctement.

❖*Bilan*

Ces trois exercices représentent assez bien les possibilités graphiques offertes par Squeak. Il est possible de faire du « glisser-déposer», de relier des éléments entre eux, etc. On peut aussi envisager des exercices basés sur la synthèse vocale, comme une dictée par exemple. La limite dans le développement se situe surtout au niveau de l'imagination du développeur, et non dans les contraintes graphiques du langage.

La synthèse vocale est utilisée à la fois comme un élément de l'exercice (avec un clic sur un mot difficile à comprendre), mais aussi comme une aide à la compréhension de celui-ci, avec par exemple la lecture de l'énoncé (toujours en cliquant dessus). De manière générale, un clic sur n'importe quel texte présent sur l'écran déclenchera la lecture de ce texte par la synthèse vocale.

3.2.5 – Récapitulatif

L'institutrice fait un suivi sur papier pour les exercices réalisés. Ce suivi sert à montrer les progrès des enfants auprès de leurs parents.

Ainsi, chaque enfant remplit, à la fin, une feuille avec ses résultats. Pour faciliter ce travail, un écran récapitule les exercices et les nombres d'essais nécessaire pour les réussir. Comme pour les exercices, il est possible de faire lire un mot, ou un chiffre, par la synthèse vocale en cliquant dessus.

En récapitulant à la fin de la suite d'exercice les résultats, et en laissant l'élève les écrire lui

même au lieu de les imprimer, cela lui permet de prendre davantage conscience des difficultés qu'il a eu.

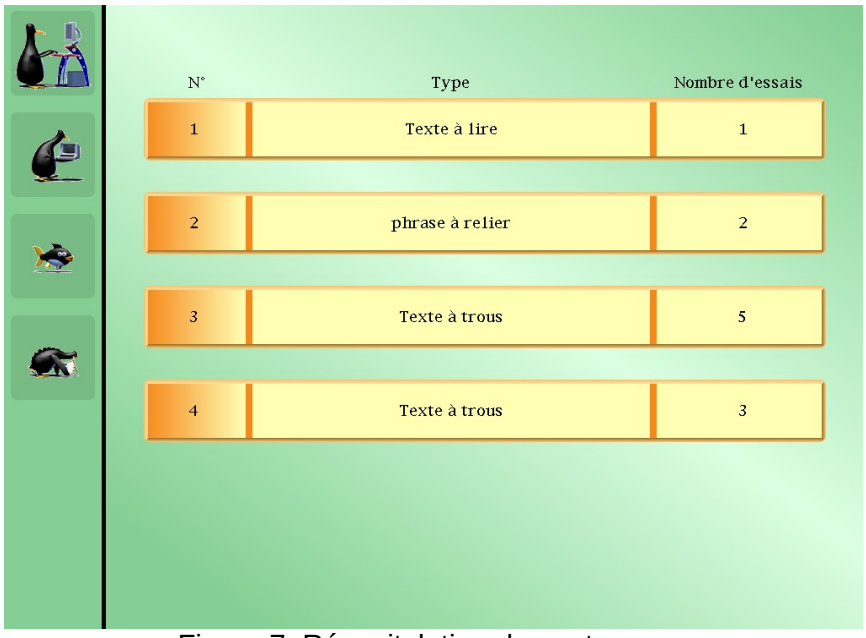


Figure 7: Récapitulation des notes

L'interface à l'apparence suivante :

3.2.6 – Fichier de paramétrages

Le contenu de chaque exercice est stocké dans un fichier XML. XML Étant un standard informatique, le contenu des fichiers peut donc être facilement accessible à partir d'autres logiciels.

La structuration du fichier XML n'a pas été vraiment étudiée, et elle est actuellement extrêmement simple, mais suffisante pour l'utilisation actuelle. Elle prouve que Squeak peut accéder à des fichiers XML et qu'il est donc possible de se baser sur ce système de stockage facilement lisible comme l'illustre l'exemple suivant.

Exemple :

Paramétrage d'un exercice de lecture de texte :

```
<exercice type="lecture de texte">
        <data>Il était quatre heures du matin. La franche
nuit d'avril s'attiédissait de l'approche du jour. Dans le
ciel limpide, les étoiles vacillaient, tandis qu'une clarté
d'aurore empourprait l'orient. Et la campagne noire,
assoupie, avait à peine un frisson, cette vague rumeur qui
précède le réveil." </data>
</exercice>
```

Deux éléments sont contenus dans le fichier :

- *le type, qui indique l'exercice*
- *les données (entre les balises data) qui spécifient les données de l'exercice*

3.3 – Analyse

3.3.1 – Présentation

Comprendre un logiciel requiert de le représenter sous forme de modèles. Un méta modèle est un langage permettant d'exprimer des modèles. Un méta modèle sert ainsi à exprimer des concepts communs à l'ensemble des modèles d'un même domaine.

3.3.2 – Méta-modélisation pour des exercices

Les exercices standard déjà utilisés pour l'apprentissage de la lecture ont été recensés lors du projet de fin d'études:

- exercices d'écoute (recherche de la position d'une syllabe dans un mot, ...)
- exercices de reconnaissance basés sur la recherche d'indices (associer un son à une image, ...)
- exercices de combinatoire (jeu de pendu, ...)

Tous les exercices utilisés par l'apprentissage de la lecture sont différents. Ils sont également extrêmement simples en terme de contenu (quelques composants de base sont nécessaires pour créer un exercice), et de fonctionnalités. Les trois exercices sélectionnés illustrent bien la simplicité d'un exercice pour l'apprentissage de la lecture. De plus le seul point commun entre ces exercices vient du fait que les données proviennent d'un texte étudié au court d'une leçon.

Il a donc été impossible, et inutile, de trouver un méta modèle pour les regrouper : il est inutile de modéliser le monde des exercices.

L'accent a donc été porté sur la manière de structurer l'application, dans l'objectif de faciliter l'ajout d'exercice.

3.4 – Framework pour l'apprentissage de la lecture

3.4.1 – Framework, définition

Un framework (cadre d'application ou cardiciel en français, mais la dénomination anglaise est celle couramment employée) est constitué d'un ensemble de classes abstraites collaborant entre elles pour faciliter la création d'un, ou d'une partie, d'un système logiciel. Il fournit suffisamment de « briques logicielles » pour pouvoir produire une application aboutie. Ces composants sont spécialisés pour un type d'application, et sont organisés pour être utilisés en interaction les uns avec les autres.

Il implémente le coeur du logiciel c'est à dire à la fois statiquement en stipulant les composants centraux et aussi dynamiquement en implémentant la dynamique du système les interactions entre composants : Le framework contrôle tout. Il suffit juste ensuite de spécifier des composants selon le projet.

3.4.2 – Structuration de l'application

N'ayant pas trouvé de méta-modèle pour des exercices, je me suis contenté de modéliser la structure de l'application. L'objectif de cette structure est double :

- faciliter les développements futurs, en offrant des facilités pour s'insérer dedans.
- contraindre les choix graphiques (couleurs des fenêtres, taille des polices, ...) afin de

conserver la cohérence de l'application.

l'écran se décompose donc de la manière suivante :

```
Ecran = Rappel de Notation + Exercice
```

Le rappel de notation (une suite de notation) est géré automatiquement, le développeur se contente de développer un nouvel exercice.

Un exercice contient :

```
Exercice = Informations + Zone(s) + Structure de Donnée
```

Les informations sont des constantes à affecter : le titre, l'énoncé et le nombre d'essai que l'on peut effectuer pour cet exercice. Ces constantes décrivent l'exercice.

La structure de donnée indique de quelle manière on récupère les informations dans le fichier XML.

Enfin, là, où les zones correspondent à l'exercice proprement dit.

```
Zone = Informations + Composants
```

Chaque zone est constituée de un, ou plusieurs composant graphique. Un composants est un élément de base d'une interface graphique (liste déroulante, bouton, etc.)

Plusieurs de ces composants (liste, texte lut par la synthèse sur un clic, ...) ont déjà été développés pour la programmation des trois exercices de démonstration. Les informations à spécifier lors de la création de zone sont son titre et son type. Le type de la zone (zone servant à entreposer des données, comme la liste de mot d'un texte à trou, ou zone où l'élève effectue l'exercice) sert a déterminer sa couleur.

Cette application correspond à la définition d'un framework que nous avons vu plus haut. L'ajout d'exercices se fait facilement, en s'insérant d'une part dans la hiérarchie de classe

abstraite fournie, et en structurants des composants graphiques fournis dans une ou plusieurs zones.

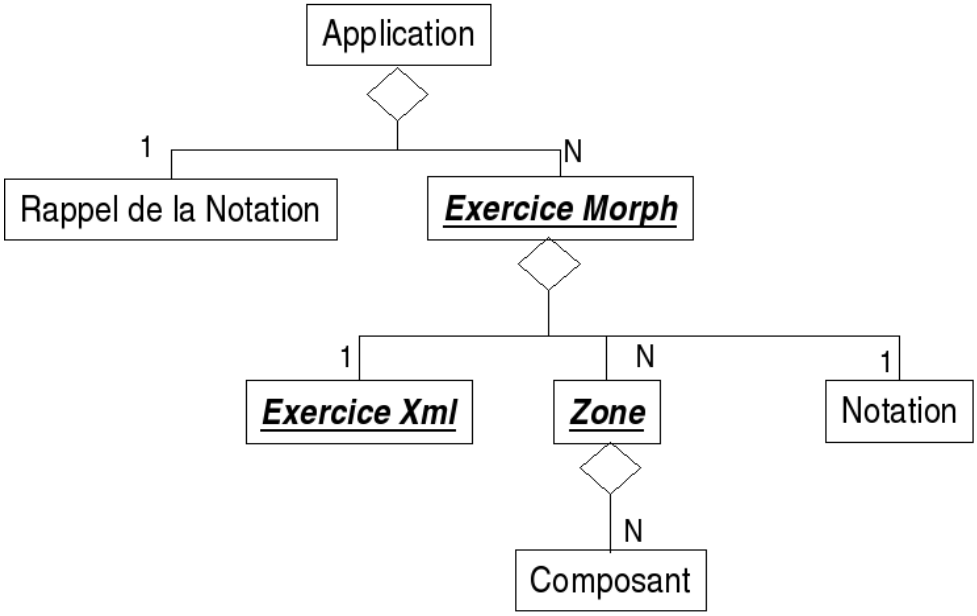


Figure 8: Schéma UML simplifié de l'application

Il y a trois classes abstraites à redéfinir :

- Exercice Xml : elle permet de spécifier l'utilisation des données du fichier XML : celles-ci sont lues sous la forme d'une collection de chaîne de caractères. L'utilisateur doit donc parser les éléments de cette collection pour obtenir la forme voulue.
- Exercice Morph : son rôle principal est de gérer les paramètres de l'exercice (énoncé, titre). Elle contient aussi les différentes zone de l'exercice. Enfin, elle permet d'affecter les données structurées par la première classes aux différentes zones.
- Enfin, il faut spécifier les différentes zones nécessaire : indiquer quel sera son type et quels composants elle va contenir.

3.5 – Bilan

Seulement la partie utilisateur standard du logiciel est développée. Cette partie fonctionne correctement et prouve que Squeak peut être utilisé pour le développement de logiciel pédagogique.

Comme nous l'avons vu, nous pouvons parler de framework pour cette application. En effet, le travail du développeur, pour l'ajout d'exercice, se résume à s'insérer dans une hiérarchie de classe et à structurer (éventuellement à créer) des composants graphiques.

L'objectif de développer un outil pour la création d'exercice est donc atteint. Cet outil n'est cependant pas vraiment utilisable dans l'état actuel. Il reste à développer les parties manquantes pour qu'il soit vraiment adapté aux utilisateurs finals possibles.

4 – Synthèse vocale

4.1 – Présentation

L'objectif d'un synthétiseur vocale est de lire n'importe quel texte, que celui-ci soit introduit directement dans un ordinateur par un opérateur ou scanné et soumis à un système de reconnaissance de caractères. La lecture doit être intelligible et naturelle. La synthèse vocale est encore un domaine de recherche, notamment au niveau de la génération automatique de l'intonation.

4.1.1 – Fonctionnement d'un synthétiseur vocal

Un synthétiseur vocal (Text To Speech, TTS en anglais) est toujours composé de deux composants :

❖*Traitement du langage naturel : (Natural Language Processing, NLP en anglais)* :

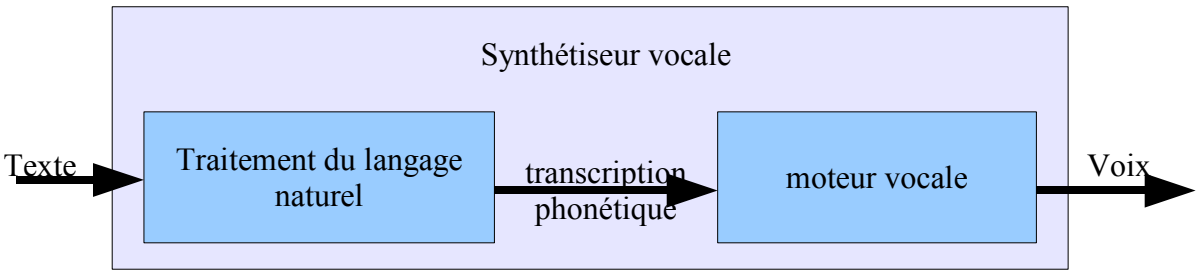
Il permet la conversion d'un texte en phonème. C'est également lui qui génère la prosodie (la manière de prononcer : intonations, ...). Ce module détermine la façon de prononcer le texte passé en entrée.

Une phonème est une unité linguistique abstraite. C'est la plus petite unité significative d'un langage. Les phonèmes n'ont pas d'existence indépendante : elles constituent un ensemble structuré d'unités dont chaque élément est intentionnellement différent des autres. Les phonèmes apparaissent dans le langage parlé sous différentes formes articulatoires : les sons.

❖*Le moteur vocal (Digital Speech Processing, DSP en anglais):*

Il synthétise la voix, à partir des information du module de traitement du langage naturel.

On peut donc schématiser un synthétiseur vocale de la manière suivante :



4.1.2 – Analyse de l'existant

J'avais développé une application de synthèse vocale en français sous Squeak pendant mon projet de fin d'étude. Elle consistaient en un module du traitement du langage naturel, utilisant la librairie MBROLA (un moteur vocal utilisant la synthèse de diphone) avec Squeak.

Une diphone est une combinaison de deux sons : la fin du premier son est le début du second son.

Cette librairie permet de transformer une suite de phonèmes avec sa prosodie en voix. On passe en paramètre à la Dll un texte sous la forme suivante :

```
K 150   0 100 50 110
```

K : phonème (son 'k')
150 : durée de la phonème en milliseconde (150 ms)

Cependant, cette synthèse vocale est destinée pour l'apprentissage de la lecture, il est donc important qu'elle ne fasse pas de fautes de prononciation (exemple : mauvaise liaison). Il a donc fallu perfectionner l'application réalisée en projet 120.

4.2 – Traitement du langage naturel

Le module du traitement du langage naturel contient trois parties : l'analyseur de texte, la transformation du texte en phonèmes et la génération de prosodie.

4.2.1 – Analyseur de texte

L'analyseur de texte crée la structure de données qui est utilisée pour les autres opérations.

4.2.1.1 – Fonctionnement

Ce module est essentiellement un parseur* qui découpe le texte en expressions. Celles-ci sont déterminées par la ponctuation : '.' '!' '?' ' :' et ','. Il effectue également divers pré-traitements :

- transcription des nombres : les caractères numériques sont remplacés par leur équivalent en texte (exemple : 1 - > un)
- transcription de caractères : les caractères spéciaux sont également remplacés (exemple : € -> euro).
- résolution des abréviations et des acronymes (exemple : Mr -> Monsieur).
- résolution des utilisations spécifiques des caractères. (exemple : le caractère 'h' peut aussi être utilisé pour l'heure 12h15 - > douze heures quinze).

Ainsi, on obtient une liste d'expressions et leur types (début de phrase, milieu de phrase, fin de phrase affirmative, phrase interrogative, phrase entre parenthèses et phrase exclamative).

4.2.1.2 – Limites

Dans la langue Française, certains mots se prononcent de manière différentes selon leur catégorie grammaticale. Par exemple, couvent se prononce « couve » (verbe conjugué) ou « couvan » (nom). L'identification de la catégorie grammaticale à laquelle appartiennent les mots du texte est donc essentielle pour une bonne prononciation.

Une solution simpliste est de regarder la présence, ou l'absence de déterminant devant le mot dont on cherche la prononciation pour déterminer sa catégorie grammaticale.

Exemple :

```
le couvent -> nom
```

Cependant, cette solution est très limitée : on peut facilement attribuer des fausses catégories.

Exemple : le verbe conjugué « couvent » est après un déterminant.

```
les poules ont pondu des oeufs dans le couvent, elles les couvent
```

Il est donc nécessaire d'utiliser un logiciel permettant de déterminer la catégorie grammaticale d'un mot de manière fiable.

De nombreux logiciels de catégorisation grammaticale automatique existent. Ils produisent un peu moins de 5 % d'erreurs pour l'anglais. Les logiciels de catégorisation grammaticale automatique sont beaucoup plus efficaces pour traiter l'anglais parce qu'il y a eu plus de recherche appliquée à l'anglais. Aussi, le choix d'un catégoriseur pour le Français est donc restreint.

Dans le cadre de notre projet, nous avons utilisé un catégoriseur à base de règles, celui de Brill. Le catégoriseur de Brill a été choisi car il était le seul à respecter les conditions pré requises pour être facilement ajouté :

- des bonnes performances (6 % d'erreurs).
- un code source disponible sous une licence libre (MIT).
- une documentation complète.

4.2.1.3 – Catégoriseur de Brill

❖*Présentation*

Le catégoriseur de Brill atteint une performance d'environ 94 % pour le français (97 % pour l'anglais). Cette performance dépend des données utilisées pour créer les bases de connaissances nécessaires. Pour créer son approche, Brill s'est inspiré des écrits de Zellig Harris, un linguiste qui a formulé plusieurs règles pour l'analyse de langues inconnues. Toutefois, Brill n'a retenu que peu des règles formulées par Harris, puisque l'analyse de la langue que peut faire un linguiste diffère nécessairement de celle que peut faire un ordinateur. Mais le catégoriseur de Brill fait tout de même une analyse linguistique d'une langue qu'il ne connaît pas et ne se base que sur les faits de surface (mots, et catégorie des mots proche du mot à catégoriser), tout comme le conseillait Zellig Harris.

❖*Réalisation*

Il y a deux étapes pour créer un catégoriseur de Brill. Il faut tout d'abord créer les bases de connaissances via un premier programme. Un dictionnaire est créé ainsi que deux types de règles : contextuelles et lexicales. Les règles contextuelles vont servir à catégoriser les mots connus ambigüs selon le contexte. Les règles lexicales vont servir à catégoriser les mots inconnus d'après leur morphologie et un contexte très restreint.

Ensuite, on utilise ces bases de connaissances pour catégoriser un nouveau texte avec un second programme.

➢ *Entraînement*

Pour créer les bases de connaissances, il faut posséder un texte correctement catégorisé. Celui-ci m'a été fourni par la société Ontologos.

J'ai utilisé le programme de Brill en C et en Perl pour créer les bases de connaissances. Il permet de créer dans un premier temps les données nécessaires, puis de générer dans un second temps le dictionnaire et les règles.

- Création du dictionnaire :

Le dictionnaire est créé à partir des mots qui se trouvent dans le texte correctement catégorisé et des différentes catégories qui leur sont attachées. Il place la catégorie qui se retrouve le plus souvent pour un même mot en tête de liste dans le dictionnaire. Les autres

catégories possibles sont placées à la suite de celle-ci.

Exemple d'un mot du dictionnaire :

```
complet ADJ SBC
```

« complet » peut être un adjectif ou un nom commun.

- Création les règles contextuelle :

Le logiciel utilise une copie du corpus duquel les catégories ont été enlevées.

Les mots de ce corpus recevront dans un premier temps leur catégorie la plus fréquente. Ensuite, le logiciel compare sa catégorisation avec celle du corpus pré-catégorisé. À la première erreur trouvée, il crée plusieurs règles. Puis, il applique chacune de ces règles à tout le texte pour voir si les règles créées sont généralisables. La règle qui réduit le plus le nombre d'erreurs est enregistrée. Puis l'entraînement continue pour corriger la deuxième erreur.

Exemple de règle contextuelle :

```
A B SURROUNDTAG X Y
```

Le mot catégorisé A va être catégorisé B si le mot précédent est catégorisé X et le mot suivant est catégorisé Y.

- Création des règles lexicales

Pour créer les règles lexicales, le logiciel utilise des données concernant deux corpus : le corpus de référence et un corpus non catégorisé aussi gros que possible.

A partir de ces fichiers on crée :

- la liste des mots contenus dans le texte catégorisé avec la fréquence pour chacune des catégories qui leur sont associées . Cette liste servira à déterminer la bonne catégorie pour un mot.
- la liste des bigrammes* contenus dans le corpus non étiqueté en ordre inverse de fréquence qui servira à créer des règles lexicales utilisant le contexte.
- la liste de tous les mots figurant dans le corpus non étiqueté en ordre inverse de fréquence. Cette liste servira à vérifier des opérations morphologiques comme l'ajout et la suppression de préfixes et de suffixes.

À partir de ces données, des règles seront formulées, permettant d'assigner la bonne catégorie aux mots du corpus. Tous les mots du corpus de référence reçoivent comme catégorie nom singulier s'ils commencent par une minuscule et nom propre singulier s'ils commencent par une majuscule. Puis le résultat obtenu est comparé avec la liste mot-catégorie-fréquence et, pour corriger les erreurs, des règles sont créées à partir d'une liste de modèles de règles. Les règles créées seront appliquées à tous les mots du texte. Les résultats obtenus sont comparés avec la catégorisation du corpus de référence. La règle qui a permi d'améliorer le plus les résultats est enregistrée avec son score, les autres sont écartées et l'apprentissage se poursuit, pour corriger les autres erreurs. Une fois l'apprentissage terminé, les règles seront placées en ordre inverse de score. La règle la plus productive en premier est la règle la moins productive en dernier.

Exemple de règle lexical :

```
A t fhassuf B
```

Un mot catégorisé A, qui a comme derniers caractères « t » va être catégorisé B.

➢ *Ajout des bases de connaissances dans Squeak*

On obtient trois fichiers textes :

- le dictionnaire : 22 000 mots.
- les règles contextuelles : 218 règles.
- les règles lexicales : 255 règles.

Une classe créé sous Squeak permet de charger les différents fichiers textes créés par le module d'entraînement de Brill. En stockant directement toutes les informations dans l'image, on évite ainsi de charger des données de taille non négligeable chaque fois que l'on en a besoin, et de déplacer des fichiers textes en plus de l'image.

Le principe est simple : il suffit de créer des méthodes permettant de générer un dictionnaire et des collections de règles. Ces méthodes contiennent la totalité des informations des fichiers textes et servent à déplacer et reconstruire la structure de donnée utilisée par le catégoriseur. Une classe a été crée pour chaque type de règles lexicales et contextuelles.

Cependant, Squeak ne peut pas contenir de grandes méthodes. Le maximum autorisé est de 250 lignes, mais il vaut mieux éviter de dépasser les 100 lignes de codes. La taille des données étant très importante (un dictionnaire de 22 000 mots) il était impossible de rentrer les données « à la main ». Aussi, trois parseurs ont été créés pour cette tache.

Le fonctionnement de ces parseurs est similaire : ils parcourent les fichiers textes et construisent une première méthode, qui fait appel à des méthodes d'une centaine de lignes constituées d'un morceau du dictionnaire ou de la collection à construire. Ainsi, on peut créer dans Squeak nos bases de connaissances.

➢ *fonctionnement du catégoriseur*

Nous allons étudier le fonctionnement du catégoriseur au travers d'un exemple.

Exemple :

- *texte à catégoriser :*

```
Les petites poules couvent
```

- *catégories :*

```
DTN : déterminant
SBC : nom commun
ADJ : adjectif
VCJ : verbe conjugué
```

- *dictionnaire :*

```
Les DTN
couvent SBC VCJ
```

- *règles lexicales :*

```
SBC fgoodleft Les ADJ
```

Le mot est catégorisé nom (SBC) est catégorisé comme adjectif (ADJ), si le mot à gauche est « Les ».

- *règles contextuelles :*

```
SBC ADJ PREVTAG SBC
```

Le mot est catégorisé nom (SBC) est catégorisé comme adjectif (ADJ), si la catégories du mot à gauche est nom (SBC).

```
ADJ VCJ PREVTAG SBC
```

Le mot est catégorisé adjectif (ADJ) est catégorisé comme verbe (VCJ), si la catégorie du mot à gauche est nom (SBC).

- Détermination d'une catégorie par défaut

Chaque mot du texte reçoit une catégorie par défaut. Il s'agit de la catégorie d'en-tête de liste dans le dictionnaire pour les mots connus, sinon de noms communs pour les mots commençant par une minuscule et de noms propres pour les mots commençant par une majuscule.

On obtient :

```
Les/DTN petites/SBC poules/SBC couvent/SBC
```

- Application des règles

Les mots inconnus sont traités par les règles lexicales (mots non présents dans le dictionnaire). Les règles contextuelles seront utilisées pour les mots connus ambigüs (mots présents dans le dictionnaire et possédant plusieurs catégories possibles). Les règles lexicales sont appliquées en premier. Le fonctionnement est similaires pour les deux types de règles.

Les mots sont traités un à un et pour chacun des mots, l'application de chacune des règles sera tentée. La règle ayant le meilleur score sera la première à être appliquée puisqu'elle est la plus générale. La dernière règle appliquée, celle ayant le plus faible score, corrigera peu d'erreurs. Si la règle peut être appliquée au mot, la catégorie du mot est modifiée. Puis, que le mot ait changé de catégorie ou non, on passe à la deuxième règle la plus productive et ainsi de suite, jusqu'à la fin des règles. Ainsi, un même mot peut changer plusieurs fois de catégorie pendant la phase d'assignation de catégorie.

Règles lexicales :

Pour que la règle s'applique, il faut que le mot de gauche du mot courant soit «Les» et que le mot soit catégorisé « nom ». Le seul mot sur laquelle elle s'appliquera est « petites ».

On obtient :

```
Les/DTN petites/ADJ poules/SBC couvent/SBC
```

Règles contextuelles :

Il n'y a que le mot « couvent » qui possèdent deux catégories dans le dictionnaire : les règles contextuelles ne vont s'appliquer que sur ce mot.

La première règles indique qu'un mot catégorisé comme nom est catégorisé adjectif si le mot précédant est catégorisé « nom » : la catégorie de poules est nom donc la règle s'applique :

```
Les/DTN petites/ADJ poules/SBC couvent/ADJ
```

On continue avec la seconde règle : un mot catégorisé « adjectif » est catégorisé « verbe » si la catégorie du mot précédant est « nom »:

```
Les/DTN petites/ADJ poules/SBC couvent/VCJ
```

➢ *Implémentation en objet du catégoriseur*

Le catégoriseur de Brill à été « traduit » du C vers du smalltalk, et couplé au module de

synthèse vocale. On utilise en entrée la structure de donnée crée par l'analyseur de texte : un texte est découpé en mots et en phrases. Ensuite la catégorisation commence, et cette structure de donnée est enrichie par les informations du catégoriseur (chaque mot se voit attribué une catégorie).

Le fonctionnement est celui décrit précédemment : chaque phrase de la structure de donnée est parcourue mot à mot trois fois, comme décrit précédemment :

- Les catégories par défaut sont attribuées.
- Les règles lexicales sont appliquées les unes après les autres, sur chaque mot, pour attribuer des catégories aux mots absents du dictionnaire.
- Enfin, les règles contextuelles permettent de déterminer la catégorie à choisir dans le dictionnaire, quand plusieurs catégories sont présentes.

➢ *correction d'erreurs*

Divers tests ont montré qu'il y avait quelques erreurs. Celles-ci ont principalement pour cause un corpus d'entraînement de taille limitée, et contenant quelques catégories erronées. Ainsi, plusieurs règles ont été corrigées manuellement. D'autres règles ont été ajoutées. De plus, les erreurs sur les bases de connaissances ont été corrigées au fur et à mesure de l'avancement.

Exemple :

catégories erronée pour un mot :

```
bleu VCJ ADV
```

le mot « bleu » est considéré comme un verbe conjugué (VCJ) ou un adjectif (ADV).

Après correction manuelle :

```
bleu ADV
```

« bleu » ne peut être qu'un adjectif (ADV).

4.2.1.4 – Bilan

La synthèse vocale dispose désormais d'un catégoriseur. Il serait nécessaire de faire des tests poussés pour calculer les performances de celui-ci, et de corriger les erreurs restantes. Cette correction ne peut se faire que manuellement.

Cependant, dans l'état actuel il est suffisant pour la synthèse vocale, car on ne l'utilise que pour déterminer la prononciation des mots ambigüs de la langue Française. Des règles de transformation en phonèmes ont été adaptées pour prendre en compte la catégorie grammaticale des mots.

4.2.2 – Transformation en phonème

4.2.2.1 – Fonctionnement

Ensuite, on utilise la structure précédente pour générer la liste de phonèmes. L'approche choisie durant le projet est basée sur des règles (établies par des experts en linguistique). Elle consiste à établir des règles permettant de passer d'un ensemble de lettres à une ou plusieurs phonèmes. Chaque règle possède des conditions indiquant les caractères devant entourer l'ensemble de lettres à transformer.

Les différentes règles sont stockées sous la forme suivante :

caractère(s) précèdent(s), caractère(s) suivant(s) , texte, code de(s) la(les) phonème(s).

Exemple de règle

```
('' 'V' 'on' 'o~')
```

'' → aucun caractère avant
V → Voyelle après
on → règle concernant la suite de caractères « on ».
o~ → codage de on avec une phonème.

Le code de la phonème provient de l'alphabet SAMPA pour le français. Il s'agit d'une version « machine » de l'alphabet phonétique, chaque caractère de cet alphabet pouvant être facilement tapé au clavier. (cf annexe 2 : Alphabet SAMPA pour le Français).

Ces règles sont ensuite triées en fonction de la taille totale de leurs conditions et du texte qu'elles transcrivent (dans l'exemple précèdent, la règle portant sur « ien » aura une priorité de 4 : 3 caractères pour la règles + 1 caractère de pré-condition).

Enfin, on cherche la règle qui s'applique au début de la phrase (en les essayant les unes après les autres dans l'ordre) et lorsqu'on la trouve, on génère le code correspondant. Puis, on se déplace dans la phrase en fonction du nombre de caractères que la règle trouvée a pu transcrire.

Exemple : Règles utilisées pour transcrire le texte « bonjour » en phonèmes « b o~ Z u R »:

Texte restant : « bonjour »

```
'' ''   'b'  ->  'b'
```

Texte restant : « onjour »

```
'' 'V'  'on' ->  'o~'
```

Texte restant : « jour »

```
'' ''   'j'  ->  'Z'
```

Texte restant : « ou «

```
'' ''   'ou' ->  'u'
```

Texte restant : « r »

```
'' ''   'r'  ->  'R'
```

Texte restant : « »

Il existe des exceptions dans la langue française qui ne peuvent pas être exprimées avec ces règles (exemple : dans « tabac » on ne prononce pas le 'c'). Il a donc fallu ajouter des règles portant sur ces mots.

Exemple :

```
('' '' 'tabac' 't a b a')
```

4.2.2.2 – Limites

Il a fallut dans un premier temps modifier, ou ajouter des règles, afin de prendre en compte les informations ajoutées par le catégoriseur de Brill.

Exemple :

Le verbe « reporter » se prononce avec les règles standard, mais on prononce « reportaire » le nom « reporter » :

```
('' 'P' 'reporter' 'R 2 p O R t E R' 'SBC')
```

« ent » se prononce « an » dans si le mot est un adjectif, un nom ou un adverbe :

```
('' 'P' 'ent' 'a~' 'ADV|SBC|ADJ')
```

De plus, à l'issue du projet 120, la synthèse vocale a gérée de manière simpliste les liaisons : Lorsqu'un mot se terminait par un s ou un x, le son 'z' était ajouté à la fin de ce mot. Cette gestion était insuffisante pour avoir une synthèse vocale précise.

4.2.2.3 – Gestion des liaisons

Ce module se base sur les règles gérant les liaisons en Français. On parcourt tous les mots de chaque phrase et on vérifie si une liaison est possible :

❖*Tests si une liaison est possible :*

Le module est entièrement paramétrable : les liaisons de la langues française ne sont pas régies par des règles strictes, il existe de nombreuses exceptions. Les bases de connaissances (est ce que le mot est une exception sans liaison, ...) peuvent être modifiées.

➢ *Tests sur le mot courant :*

Un mot ne peut donner lieu à une liaison que si :

- c'est un nom commun finissant par s ou x,
- il finit par une consonne autorisant les liaisons
- ce n'est pas une exception

➢ *Tests sur le mot suivant :*

Le mot suivant le mot courant va autoriser les liaisons si :

- le premier caractère de ce mot est une voyelle
- le premier caractères est un h (qui n'est pas un h aspiré*)
- ce n'est pas une exception

➢ *Test sur les deux mots :*

Il n'y a pas de liaison entre un nom commun et un verbe.

❖*Ajout de la liaison :*

Tout d'abord, on regarde s'il ne faut pas dénasaliser une voyelle nasale* : si oui, et si le mot n'est pas une exception, on la remplace (ex : 'un' -> 'une').

Enfin, on ajoute le son de la liaison au début du mot suivant. En effet, si il y a une pause entre

les deux mots, la liaison est prononcée au début du deuxième mot

Exemple de prononciation : « deux amis »

```
On prononce :       « deu »      (pause)       « zami »
Et non :            « deuz »     (pause)       « ami ».
```

4.2.3 – Génération de la prosodie

4.2.3.1 – Définition de la prosodie

La prosodie est en quelque sorte la musique du langage. Elle concerne le rythme de la parole et sa mélodie. Sa description suppose l'observation des variations de paramètres physiques tels que la fréquence fondamentale, et la durée des événements linguistiques. La fréquence fondamentale définit l'intonation, qui rend compte des variations de hauteur de la voix. La durée des événements linguistiques, celle des syllabes par exemple, est également un élément essentiel de la description prosodique.

Ainsi, avec une suite de phonèmes et la prosodie l'accompagnant, on peut décrire une phrase, et la façon de la prononcer.

4.2.3.2 – Fonctionnement

❖*Intonation*

Il n'existe pas encore de solution universelle pour la génération de prosodie, celle-ci dépendant étroitement du langage. En 1966, Delattre a mis en évidence les dix intonations de base du français. Ces travaux restent d'actualité et donne une première approche pour calculer le pitch en fonction du type de la phrase. Il a analysé l'intonation d'une phrase avec une grille de quatre échelons couvrant chacun une octave.

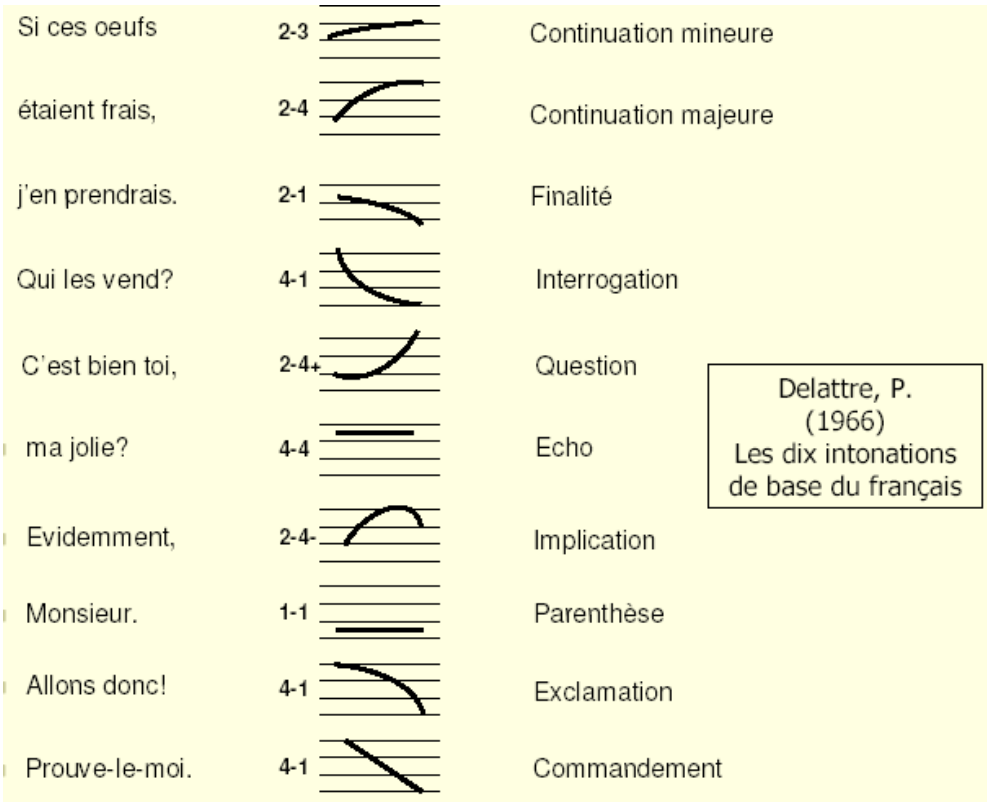


Figure 9: Les dix intonations de base du français

On utilise les intonations fondamentales de Delattre vu précédemment pour déterminer les modifications sur le pitch. Pour cela, des tests ont permis de trouver des courbes mathématiques (basées sur les fonctions puissances, exponentielle et logarithme) similaires à celles de Delattre permettant d'obtenir la valeur du pitch pour chaque position des phonèmes d'une phrase. La fréquence trouvée pour une phonème est appliquée sur toute sa durée.

Exemple d'utilisation de la fonction exponentielle pour obtenir la valeur du pitch pour une fin de phrase affirmative :

```
(fréquence +   (1 - (    ( (e^position) / (e^(taille)) ) *
palier) )
```

Fréquence : valeur de départ pour la fréquence
Position : position de la phonème
Taille : nombre totale de phonème
Palier : valeur entre chaque niveau

❖*Durée des événements linguistiques*

La durée de chaque phonème reçoit dans un premier temps une valeur prédéterminée qui a été stockée sous Squeak. Tomas Dubida (2003), a montré que dans la langue française, les deux dernières syllabes sont allongées. Aussi la durée affectée précédemment est allongée de 10 ms pour les deux dernières phonèmes : cela ne correspond pas tout le temps aux deux dernières syllabes mais les tests ont montré que cette modification rompt la monotonie et l'apparence « hachée » du débit de la parole.

Cette modification est simpliste. Afin d'améliorer la qualité du discours, une meilleure prosodie temporelle a été choisie.

4.2.3.3 – Prosodie temporelles

❖*Présentation*

Diverses contraintes linguistiques (phonologique, syntaxique, sémantique), psycholinguistique (contraintes liées à la production de la parole), émotives et pragmatiques s'exercent dans la dimension temporelle de la parole.

La plupart des algorithmes existant actuellement se basent sur la structure intonative, et sur la localisation des accents. Cependant le français n'est pas une langue accentuelle comme l'anglais ou l'allemand, ce qui explique que les structures temporelle calculées avec ces modèles différent des structures observées par un locuteur. La parole généré paraît donc moins fluide et moins naturelle.

La structure temporelle choisie se base sur les travaux de Brigitte Zellner. La production orale d'un énoncé étant structurée dans le temps, deux concepts sont retenus. Tout d'abord, pour désigner les éléments de cette organisation temporelle, on parlera de groupe temporel, et non pas de groupe prosodique, puisque seule la dimension temporelle est ici prise en compte. Une structuration à trois niveaux est retenue. Ainsi, une phrase peut être structurée en plusieurs groupes majeurs (GM) qui eux mêmes sont ou non structurés en plusieurs groupes mineurs (Gm).

On obtient donc la structure hiérarchique suivante :

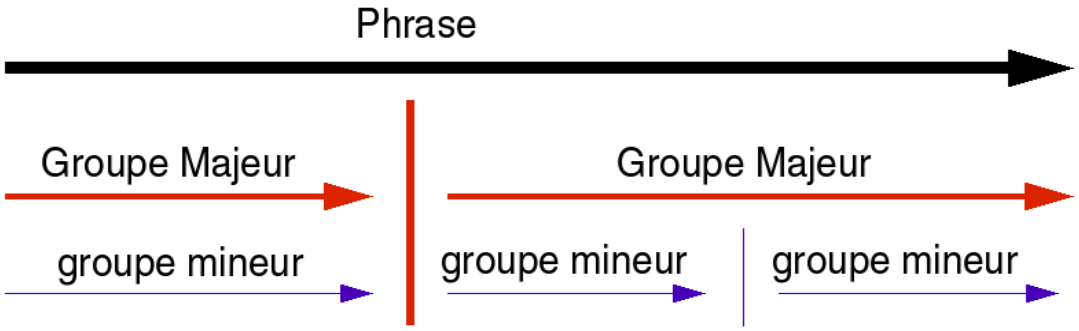


Figure 10: Structure hiérarchique temporelle

Le deuxième concept porte sur la relation qui s'établit entre les durées. La durée d'une syllabe est relative car elle est aussi fonction des durées précédentes et suivantes. Ainsi, une impression de ralentissement du débit peut être créée par un allongement de la syllabe, tout aussi bien que par une compression des syllabes précédentes et/ou suivantes.

Brigitte Zellner a analysé une centaine de phrases déclaratives, et a mis en évidence qu'une syllabe voit sa durée affectée en fonction de sa place dans la structure temporelle. Elle a retenu quatre types de frontières pour expliquer quatre niveaux de durées syllabiques :

- Début de Groupe mineur -> syllabe brève
- Milieu de Groupe mineur -> syllabe brève ou médium

- Fin de Groupe mineur -> syllabe longue
- Fin de Groupe majeur -> syllabe très longue

Enfin, une pause sépare les différents groupes majeurs.

❖*Calcul*

Le calcul de la structure temporelle est fait en fonction de la catégorie morpho-syntaxique de chaque mot. Un mot est déclaré comme étant mot lexical (l) si c'est un nom, verbe, adjectif ou adverbe). Sinon, il sera déclaré comme grammatical (g).

Exemple :

```
La fille s'est déguisée en une jolie petite fée espiègle
 g    l      g       l       g  g    l     l     l     l
```

Les frontières temporelles se déterminent ensuite en lisant la phrase de gauche à droite : On découpe le texte en groupe majeur à chaque fin de phrase et devant les virgules.

Exemple : (| : représente une frontières de groupe majeur)

```
La fille s'est déguisée en une jolie petite fée espiègle ||
 g    l      g       l       g  g    l     l     l     l
```

Ensuite chaque groupe majeur est découpé en groupe mineur. Un nouveau groupe mineur est crée quand :

- un mot grammatical apparaît après un mot lexical.
- trois mots lexicaux se suivent et le dernier mot est le verbe. Le nouveau groupe commencera avec le dernier mot.
- au moins quatre mots lexicaux se suivent. Le nouveau groupe commencera après le premier mot si c'est le verbe, sinon après le second.

Exemple : découpage de la phrase en groupes mineurs.

```
(La fille) (s'est déguisée) (en une jolie petite) (fée espiègle)|
 g    l       g       l       g   g    l     l       l       l
```

De plus, une marque de fin de groupe majeur (GM) est créée à l'intérieur de la phrase lorsque celle-ci comprend au moins treize syllabes. En fonction du nombre de mots, une frontière de GM est posée à mi-phrase, sur la frontière de Gm la plus proche.

Exemple : le 5ème mot est le centre de la phrase

```
(La fille) (s'est déguisée)| (en une jolie petite) (fée espiègle)|
 g    l       g        l       g  g    l     l       l      l
```

❖*Attribution d'une durée aux phonèmes*

Tout d'abord, on attribut à chaque phonème une durée par défaut. Ensuite, on modifie la durée de cette phonème en fonction de la classe du mot. Brigitte Zellner utilise cinq classes de durée syllabique :

Classe de mot	1	3	3	4	5
type	compression majeur	compression mineur	proche des durée par défaut	allongement mineur	allongement majeur
Modifications de la durée	première syllabe : -20 % seconde syllabe: -10 %	première syllabe : -10 %	pas de modification	dernière syllabe : +10 %	dernière syllabe : +20 % avant dernière syllabe : +10 %

La classes de chaque mot est attribué en fonction de leurs positions dans la structure : La ligne encadrée du schéma suivant illustre la classe du mot en fonction de sa position. Si on prend le cas du premier groupe mineur du premier groupe majeur, on a : 1 (3)* 4. Ceci signifie que le premier mot du groupe mineur sera de classe 1. Le dernier mot sera de classe 4. Enfin, tous les mots présents entre le premier et le dernier seront de classe 3.

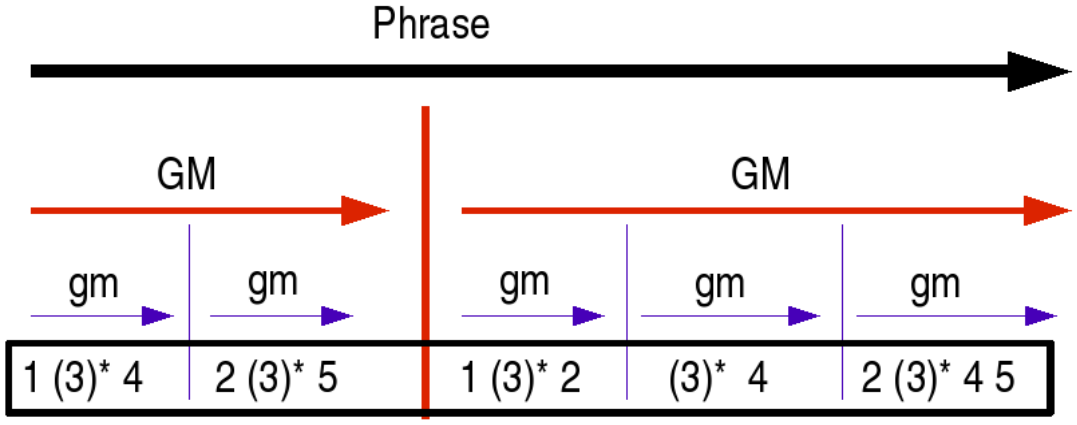


Figure 11: Classe d'un mot dans la structure hiérarchique

Exemples :

- le deuxième mot du second gm du premier GM est un mot de la classe 2.
- le premier mot, du premier gm du dernier GM est de classe 1.

❖*Attribution des pauses*

Une pause est ajoutée entre chaque groupe majeur. Sa durée dépend de la ponctuation (Exemple : 100 ms après une virgule).

❖*Résultats :*

Les tests que présente Brigitte Zellner montrent une bonne correspondance entre la prosodie temporelle générée et celle qui est prononcé par un locuteur humain. Ces tests n'ont pas pus être vérifié dans le cadre de notre application car ils sont très long effectuer : enregistrement d'une centaine de phrases et comparaison manuelle de la correspondance, via un logiciel d'édition de son.

Concrètement, la différence entre la durée de prononciation de chaque phonème est la suivante pour le mot bonjour :

Exemple : prononciation de « bonjour » sans le module de gestion temporelle :

```
b 76 o~ 106 Z 81 u 88 R 55
```

les caractères représentent le code de la phonème, et le nombre la durée de la prononciation de chaque phonème.

Exemple : prononciation de « bonjour » avec le module de gestion temporelle :

```
b 68 o~ 112 Z 93 u 105 R 68
```

La durée des phonèmes en début du mot a été raccourcie, et allongée en fin de mot.

Exemple : prononciation de « bonjour » avec le module de gestion temporelle, dans la phrase : « bonjour, comment va tu ? » :

```
b 68 o~ 102 Z 83 u 91 R 57
```

La durée des phonèmes en début du mot a été raccourcie, et allongée en fin de mot, mais les modifications sont moins importantes que dans l'exemple précédent.

Ces exemples illustrent que les modifications apportées par le module de gestion temporelles sont bien différentes en fonction de la position du mot dans la phrase. Ainsi, la synthèse vocale prononcera de plusieurs manières le même mot (dans des contextes différents). Le discours prononcé sera donc moins régulier et monotone, donc plus naturel.

❖*Bilan*

Dans modèle temporel est proposé par Brigitte Zellner, aucune information accentuelle n'est requise. En effet, l'accent peut être intégré plus tard dans le modèle prosodique car il n'a que peu d'importance pour la structure temporelle en français, même si sa présence peut localement introduire des variations de durée.

Ce modèle permet de générer des structures temporelles proches de celles que pourraient produire un locuteur, et améliore la qualité de la prononciation du synthétiseur. Pour l'améliorer, il faudrai encore prendre en compte la longueur de la phrase, la catégorie type morpho-syntaxique et le nombre de mots de la phrase. Cependant je n'ai pas trouvé d'article indiquant comment modifier la durée d'une phonème en fonction des ses paramètres.

4.2.4 – Bilan

La structure du module de traitement du langage naturel est terminé. Elle consiste une hiérarchie de 49 classes avec un total de 519 méthodes et accesseurs, le tout contrôlé par 117 tests unitaires*. La dernière étape pour le perfectionner est, d'une part le contrôle de toutes les données (règles de prononciation, liaison, ...), d'autre part une amélioration de la génération de la prosodie.

Cependant, le travail sur la génération de la prosodie doit être fait en relation avec le moteur vocal

4.3 – Moteur Vocal

4.3.1 – Présentation

Les opérations impliquées ici sont celles qui correspondent aux muscles articulatoires et aux vibrations des cordes vocales humaines. Ces opérations peuvent être effectuées de deux manières :

- Explicitement, avec une série de règles qui décrivent formellement l'influence d'une phonème sur une autre.
- Implicitement, en stockant un exemple de toutes les transitions phonétiques et coarticulation* dans une base de donnée, et en les utilisant comme unité acoustique (de la même manière que les phonèmes sont utilisés pour un module de traitement du langage naturel).

4.3.2 – Stratégies

4.3.2.1 – utilisation de règles

Les synthétiseurs basés sur des règles sont ceux qui sont le plus en faveur des phonéticiens car ils constituent une approche cognitive et générative des mécanismes de phonation. L'un de ces synthétiseur, Klatt, est très répandu car il aide dans l'étude des caractéristiques d'une voix naturelle.

La première étape pour la constitution d'un synthétiseur basé sur des règles est l'enregistrement d'un grand nombre de mots par un orateur professionnel. Ces mots sont choisis de façon à constituer un corpus représentatif des transitions et coarticulations de la langue choisie. Des expert en linguistique, aidés par des analyseurs vocaux, vont ensuite déterminer les règles. Elle sont souvent exprimées sous forme de segments représentant chacun une durée pour une valeurs de fréquences. Ces multiples segments qui décrivent une phonème sont appelé allophones.

Un signal vocal paramétrique peut être produit à partir de ces règles. Ce signale vocal paramétrique est transformé en signal digital.

Cette stratégie requiert des compétences très complètes en linguistique. De plus, les systèmes actuels ne sont pas d'une grande qualité. Cette méthode est donc inappropriée pour la réalisation du moteur vocal de notre synthèse vocale

4.3.2.2 – concaténation de segments

Le principe de ces synthétiseurs est de stocker toutes les coarticulations et transitions d'une langue. En effet, les parties les plus dures à générer sont les transitions entre les différents sons. Dans sa partie médiane la fréquence du son est stable, et ne change peu. L'idée est

donc de stocker toutes les segments : les diphones (fin d'un son et début d'un autre) nécessaires pour un langage.

Il existe plusieurs moyens de stocker les segments :

❖*sous une forme paramétrique :*

Ce principe se base sur le fait que la voix est produite par un filtre à l'extrémité d'un type. Aussi, au lieu de stocker un son, il est plus simple et léger de ne garder que les paramètres du filtre.

Le LPC, *Linear Predictive Coding,* se base sur cette méthode. Il utilise une base de diphones de moins de 1Mo.

Les son nécessaires sont re-créés en applicant le filtre stocké sur un son blanc (son contenant toutes les fréquences sonores possibles).

❖*sous une forme d'encodage Wave :*

L'encodage Wave est un standard pour stocker des formats audio non compressés. Les sons sont donc stockés pour être concaténés les uns à la suite des autres, la phase de génération d'un signal sonore par son est donc supprimée. La taille d'une base de diphones destinée à cet usage est d'environ 10 Mo.

❖*sous les deux formes*

Il existe également des formes de codages mixtes. Ils sont destinés à offrir plus d'informations pour faciliter la concaténation.

❖*concaténation*

La partie intelligente de ce type de synthétiseur se situe dans la manière dont les segments stockés sont concaténés les uns à la suite des autres.

Les segments sont sélectionnés, puis modifiés en fonction des informations du module de traitement du langage naturel (allongement ou diminution de la durée, modification de la fréquence). Ensuite les fréquences sont lissées. Ces manipulations sont plus ou moins lourdes, selon les algorithmes utilisés. Le choix de l'algorithme dépend aussi de la manière dont sont stockés les sons.

Il existe une autre solution qui consiste à stocker une très grande quantité de diphones. Ainsi le synthétiseur ne modifie plus les diphones lors des concaténation, mais se contente de choisir celles qui correspondent le mieux. Cette méthode donne de très bon résultat, proche d'une voix naturelle. Cependant la taille d'une telle base de diphones avoisine les 200 Mo.

4.3.3 – Bilan

La méthode la plus simple à implémenter est probablement la concaténation de diphones. C'est la méthode qui est utilisée par la plupart des synthétiseurs vocaux actuel de bonne qualité. Il est nécessaire de concevoir en premier une base de diphones pour le français.

Il n'existe aucune base de diphones sous une licence entièrement libre actuellement pour la langue française. La licence de Mbrola interdit l'utilisation de ses bases avec une autre application. Le temps nécessaire estimé pour la création d'une base de diphones est d'un mois (temps moyen observé pour la constitution des bases utilisées par la librairie Mbrola). Ce temps comprend l'enregistrement d'un corpus de mots représentatifs de la langue et le découpage manuelle des diphones.

Le temps de création de cette base peut être réduit en utilisant le corpus de mot enregistré « La Base Audio Libre De Mots Français ». C'est une base de données d'enregistrements sonores, distribué librement, d'une liste d'environ 7000 mots ou expressions en langue

française. Il ne reste plus que le travail de sélection et de découpage des diphones. Cependant, pour une bonne qualité il est nécessaire de stocker les diphones à une fréquence identique. Un premier travail de manipulation est nécessaire.

Squeak dispose de plusieurs fonctionnalités pour manipuler la fréquence du son. Cependant, on ne peut modifier le son que proportionnellement, par rapport à sa valeur actuelle. Il est actuellement impossible de spécifier la fréquence à laquelle on souhaite prononcer le son.

La création d'un moteur vocal est une étape complexe. De plus il est très long de créer une base de diphones. Ce module n'a donc pas été développé pendant le stage.

4.4 – Perspectives

Le travail effectué sur la synthèse vocale est surtout visuel. Il consiste à transformer du texte en une suite de paramètres compréhensibles pour la Dll Mbrola.

Exemple de résultats du module de traitement du langage naturel :

Texte tapé par l'utilisateur :

```
Je suis un synthétiseur vocale.
```

Sortie du traitement du langage naturel :

```
Z 64 25 124 75 124#2 108 13 124 38 124 63 124 88 124#s 125 10
124 30 124 50 124 70 124 90 124#y 76 17 124 50 124 83 124#i
80 17 124 50 124 83 124#_ 10#z 78 17 124 50 124 83 124#9~ 104
13 124 38 124 63 124 88 125#s 125 10 125 30 125 50 125 70 125
90 125#e~ 97 17 125 50 125 83 125#t 90 17 125 50 125 83 125#e
87 17 125 50 124 83 124#t 90 17 124 50 124 83 124#i 80 17 124
50 124 83 124#z 87 17 124 50 124 83 124#9 101 13 124 38 124
63 124 88 124#R 55 25 123 75 123#v 80 17 123 50 123 83 123#O
107 13 123 38 122 63 122 88 122#k 95 17 122 50 122 83 121#a
102 13 121 38 121 63 121 88 120#l 63 25 120 75 120#_ 300#
```

Ce programme a été conçu pour être facilement repris. Plus que la première partie d'une synthèse vocale, c'est aussi une base destinée à évoluer. Il est entièrement paramétrable, la sortie peut donc être facilement modifiée (rythme du langage différent, voix plus ou moins aiguë). Un expert en prosodie peut facilement effectuer des modifications pour en améliorer la qualité.

Enfin, il peut donc être repris et complété pour la création d'une synthèse vocale complète et entièrement sous Squeak (suppression de la Dll Mbrola). Quelqu'un désirant créer un moteur vocal disposera de la première partie d'une synthèse vocale pour le tester.

La seconde partie du travail, l'analyse des différents types de synthèse vocale, donne les bases nécessaires pour continuer le travail.

5 – Conclusion

La réalisation d'une synthèse vocale de qualité nécessite des connaissances avancées en informatique, en linguistique et en traitement du signal. Étant un expert dans un seul de ces trois domaines, je me suis donc basé sur les travaux de linguistes pour concevoir un module de traitement du langage naturel. Ce module est le premier élément d'une synthèse vocale. Je n'ai pas eu le temps de réaliser le second module de la synthèse vocale, je n'ai effectué qu'une rapide présentation reprenant les bases de ce que j'ai étudié. Une des principales difficultés pour la réalisation ce module est la création d'une base de diphones française « libre ».

La module de traitement du langage naturel peut être amélioré avec une utilisation intensive de la synthèse vocale. En effet, il reste encore probablement un grand nombre d'exceptions de la langue française qui ne sont pas stockées dans les bases de données. Ce module est disponible sous une licence libre.

Ce stage a été une très bonne première expérience professionnelle. J'ai pu travailler dans une totale autonomie sur un projet et choisir moi même les grandes directions à prendre. De plus, la compréhension d'un domaine auparavant inconnu, la synthèse vocale, est une bonne expérience du monde de la recherche.

6 – Lexique

AES256 : Algorithme de chiffrement asymétrique avancé.

Bigrammes : Mot en deux parties, chacune de ces parties n'ayant aucun sens individuellement. Exemple : Aujourd'hui.

Chiffre de César : Méthodes de chiffrement la plus simple connue. C'est une technique de codage par substitution, c'est-a-dire que chaque lettre du texte claire est remplacée par une autre lettre a distance fixe dans l'alphabet. Par exemple, si l'on utilise un décalage de 3, `A` serait remplacé par `D`, `B` deviendrait `E`, et ainsi de suite. Cette méthode a été nommée d'après Jules César, qui utilisait cette technique pour communiquer avec ses généraux.

Coarticulation : Fait que, selon les phonèmes qui l'entourent dans une phrase, une phonème n'est pas prononcé de la même manière.

Cryptanalyse, **décryptage** : Coté « opposé » du déchiffrement : l'objectif est de retrouver le texte d'origine sans clé.

Déchiffrement : Utilisation d'une clé pour obtenir le texte clair à partir d'un texte .

EIAH : *Environnement Informatique d'Apprentissage Humain*. Domaine de recherche sur la manière de concevoir un programme dans le but de faciliter l'apprentissage.

XML : *Extensible[Markup Language*, ou *langage de balisage extensible*. Langage informatique générique. Son objectif initial est de faciliter l'échange automatisé de contenus entre systèmes d'informations hétérogènes.

Forensic : Analyse d'une machine après crash ou compromission.

Framework : Ensemble structurés de bibliothèques pour le développement d'un programme (voir partie 3.4).

H aspiré : Souvent représenté par un *h* écrit en début de mot, il empêche la liaison. (exemple : dans *les haches*, le *s* n'est pas prononcé) et l'élision (*la hache*, au lieu de *l'hache* incorrect). Le h aspiré s'oppose au h muet.

Parseur : programme permettant de « découper » un texte en un ensemble structuré.

Reverse engineering : La **rétro-ingénierie** est l'activité qui consiste à étudier un objet pour en déterminer le fonctionnement.

RSA 1024 : Algorithme de chiffrement asymétrique.

SQL : *Structured query language* , ou *langage structuré de requêtes*, pseudo-langage informatique standard et normalisé, destiné à interroger ou manipuler une base de données.

Stéganographies : Consiste à dissimuler des données à l'intérieur d'autres données.

Tests unitaires : procédé permettant de s'assurer du fonctionnement correct d'une partie déterminée d'un logiciel.

XPath : syntaxe (non XML) pour désigner une portion d'un document XML. Xpath est utilisé comme langage de requêtes dans les bases de données.

7 – Bibliographie

7.1 – Rapport de stage

Gaël Garcin (2006) *Nessus et la sécurité.*

Julien Arnoux (2006) *Mise en place d'un outil de veille technologique et sécurité.*

Jérôme Déprez (2006) *Pingoo FS, Partages de Fichiers et contrôleur de domaine.*

Vincent Bouchet(2006) *Synthétiseur vocal, rapport de projet 120.*

7.2 – Articles de recherche

Gestion de la prosodie temporelle :

Tomas Dubida, (2003), *De l'acoustique au conventionnel: une vue configurative et multiparamétrique de l'accent en français et en tchèque*

Brigitte Zellner (1996, Revue Française de Linguistique Appliquée, n°1, Paris), *Structures temporelles et structures prosodiques en français lu.*

Brigitte Zellner (1998, Acte des XXIIèmes Journées d'Etude sur la Parole (JEP 98)). *Caractérisation du débit de parole en français.*

Brigitte Zellner (2002), *Revisiting the Status of Speech Rhythm*

7.3 – Site web

Carégoriseur de Brill :

Thibeault, Mélanie (2004), http://www.theses.ulaval.ca/2004/22225/22225.html : *La catégorisation grammaticale automatique: adaptation du catégoriseur de Brill au français et modification de l'approche.*

Eric Brill, http://www.cs.jhu.edu/~brill/ : *Eric Brill's Home Page.*

Morphic :

http://minnow.cc.gatech.edu/squeak/

Alphabet SAMPA :

http://www.phon.ucl.ac.uk/home/sampa/ : SAMPA computer readable phonetic alphabet (pour le français : http://www.phon.ucl.ac.uk/home/sampa/french.htm).

Moteur Vocale :

http://shtooka.moostik.net/dico-fr/ : *Base Audio Libre De Mots Français*

http://www.otolith.com/otolith/olt/lpc.html Linear Predictive Coding (LPC)

Divers :

http://fr.wikipedia.org/ : *L'encyclopédie libre.*

7.4 – Documentation (livre):

Thierry Dutoit (1997), *An Introduction to Text-to-Speech Synthesis, ISBN 0-7923-4498-7 (285 pages).*

8 – Annexes

8.1 – Déroulement du challenge Sécuritech

Nous avons participé au challenge en équipe, afin de se répartir les épreuves en fonction des connaissances de chacun et d'augmenter nos chances de succès.

Les challenges que nous avons réussi étaient basés sur des injections SQL et Xpath*, décryptage d'un code et reconstruction d'image contenant des données stéganographiques* :

- Injection

Le principe de l'injection est simple : Une technique utilisée pour valider un login est d'entrer un morceau de requête dans le formulaire. Ainsi, on « complète » la requête de recherche du formulaire pour obtenir une requête toujours vraie.

Exemple :

Le formulaire utilise la requête suivante :

```
SELECT * FROM maTable WHERE login = 'texte_entrée'
```

On entre un morceau de requête :

```
' OR 'A' = 'A
```

La requête qui sera utilisée sera donc la suivante, elle est toujours vrai !

```
SELECT * FROM maTable WHERE login = '' OR 'A' = 'A'
```

Une autre méthode pour l'injection consiste à trouver un champ dans un formulaire qui retourne un message. Comme pour la première technique, on rajoute des morceaux de requêtes pour manipuler le message et afficher des informations de la table.

- Décryptage

Dans ce challenge, on disposait d'un texte crypté, avec comme seule information qu'il s'agissait d'un code classique modifié. L'objectif était de retrouver le 1000 ème mot du texte qui avait été volontairement mal orthographié.

Extrait du texte a décrypter :

```
RVSICEVV XXP
OJ U. EV KXMLS XQD WIERXB VIYMKIR WEWVIBI TCWZ H'LPL FDTM DD
WYSHVM HZUT WO
```

Comme on peut le voir, ce type de cryptage conserve la ponctuation, ce qui est typique du chiffre de César*.

Tout d'abord, j'ai fais des recherches dans des grosses base de données de texte pour trouver si un texte avait la même ponctuation. Ainsi, j'ai découvert qu'il s'agissait du chapitre XVI des Trois Mousquetaires d'Alexandre Dumas.

Ensuite, j'ai utilisé des outils de décryptage* qui existent sur Internet, et j'ai constaté que l'un d'entre eux (permettant de décrypter le chiffrage de Vigénère, une variante du chiffre de César) renvoyait une clé lisible :

```
POSTULERACHAPITREXVIOUMLEGARDEDESSCEAUX
```

On constate que la clé est constitué du mot POSTULERA suivit du texte à décrypter.

En programmant en Java une classe qui utilisait « postulera » suivit des caractères trouvés au

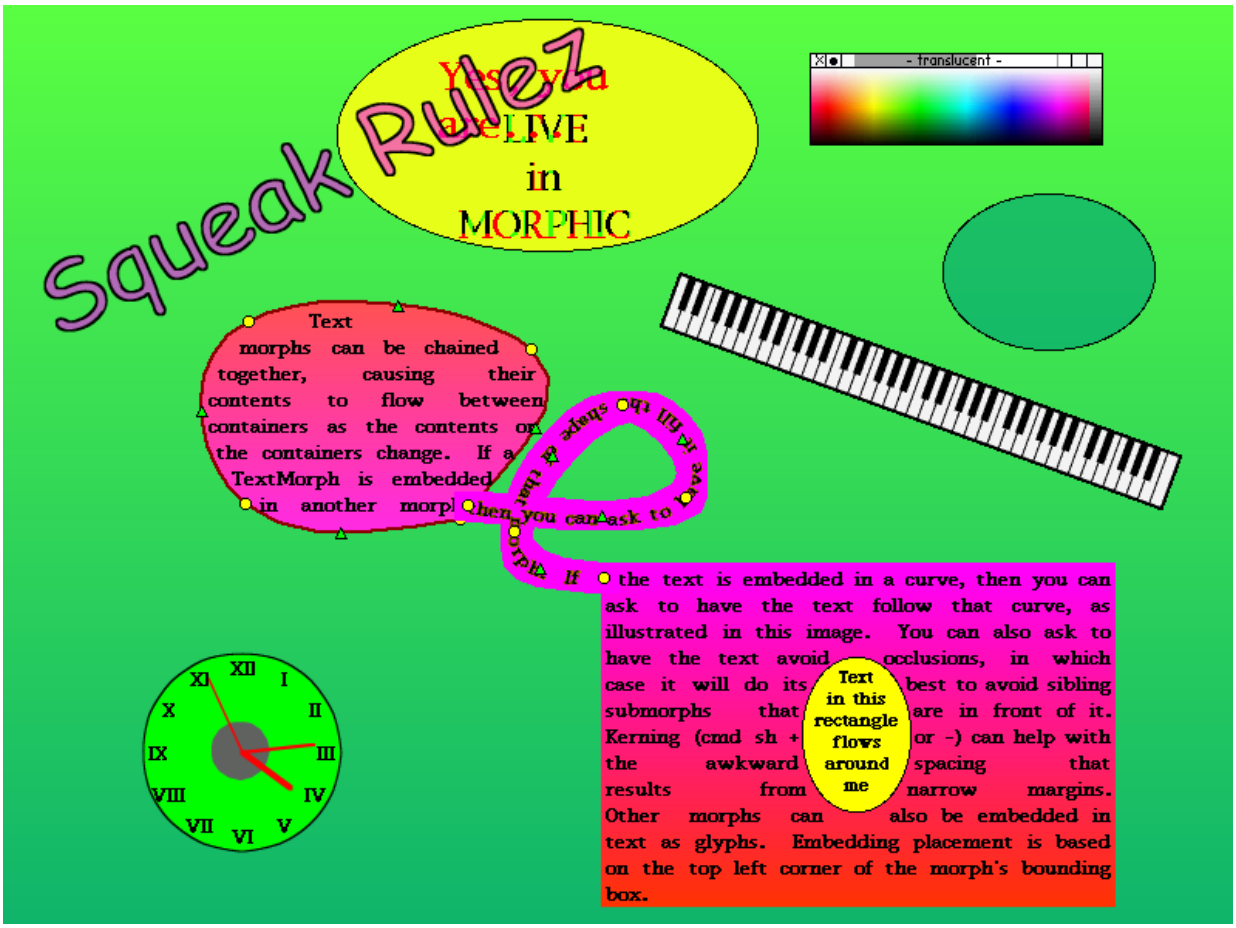
fur et à mesure comme clé de déchiffrement* j'ai pu trouver le mot permettant de valider le challenge.

- Reconstruction d'image

Nous disposions d'une cinquantaine de morceaux d'image. Nous avons tout d'abord conçu un script en Python pour recoller ces morceaux et reformer ainsi l'image originale . Ensuite nous avons utilisé un outil de stéganographie pour extraire des informations (du texte) dissimulées dans l'image.

Le niveau de difficulté est très élevé. Avec notre équipe composée uniquement de stagiaires nous n'avons trouvé que 6 validateurs sur 21. Malgré cela nous sommes fiers d'avoir terminé 60ème sur 2260 inscrits (seulement 384 inscrits ont eu plus de 0 points).

8.2 – Exemple de Morphs :



8.3 – Alphabet SAMPA pour le Français.

Symbol	Mot	Transcription
p	pont	po~
b	bon	bo~
t	temps	ta~
d	dans	da~
k	quand	ka~
g	gant	ga~
f	femme	fam
v	vent	va~
s	sans	sa~
z	zone	zon
S	champ	Sa~
Z	gens	Za~
j	ion	jo~
m	mont	mo~
n	nom	no~
J	oignon	oJo~
N	camping	ka~piN
l	long	lo~
R	rond	Ro~
w	coin	kwe~
H	juin	ZHe~
j	pierre	pjER
i	si	si
e	ses	se
E	seize	sEz
a	patte	pat
A	pâte	pAt
O	comme	kOm
o	gros	gRo
u	doux	du
y	du	dy
2	deux	d2
9	neuf	n9f
@	justement	Zyst@ma~
e~	vin	ve~
a~	vent	va~
o~	bon	bo~
9~	brun	bR9~