École Centrale Paris
Option SEM
Filière MP

Samuel Hocevar
SNCF - Direction de la Recherche et de la Technologie

Rapport de stage de 3e année
Avril - Novembre 2002
$X_i + Y^2$

Résumé

TVi est un démonstrateur multimédia à la place pour sièges de TGV utilisant entre autres la technologie de diffusion vidéo Mediabase et les formats de vidéo MPEG et QuickTime. Ces technologies sont anciennes, réduisant l'impact du démonstrateur comme vitrine du savoir-faire de la SNCF dans le domaine des Nouvelles Technologies de l'Information et de la Communication. Elles sont aussi presque toutes propriétaires, rendant très coûteuse une éventuelle industrialisation de TVi.

Pour régler le problème du coût, j'ai procédé entre autres à la réécriture du code de TVi dans des langages standardisés et remplacé les logiciels coûteux par des équivalents gratuits ou libres. Lorsque ces équivalents n'existaient pas, je les ai développés, en particulier un *plugin* pour navigateurs Web utilisant les logiciels de diffusion vidéo VideoLAN. L'utilisation de VideoLAN a aussi permis de pallier l'ancienneté de TVi, grâce à des technologies de pointe telles que MPEG4, ou par la diffusion de vidéo via réseau sans fil sur des ordinateurs de poche.

Pour éviter une nouvelle obsolescence de TVi, il reste encore à supprimer le peu de logiciels propriétaires restant. À plus long terme, il faudra développer des extensions pour les formats de vidéo numérique de demain.

Les travaux effectués ont permis de diviser le coût de TVi par 3, et en particulier son coût en logiciels a été ramené à zéro. TVi est aujourd'hui à la pointe de la technologie en matière de diffusion de vidéo sur réseau. J'ai pu sur le plan technique parfaire mes connaissances dans le domaine de la vidéo numérique. Sur le plan organisationnel, j'ai pu expérimenter avec rigueur et succès une méthode moderne de développement logiciel dite « légère ».

Table des matières

Chapitre 1

Remerciements

Je tiens à remercier les personnes suivantes pour leur aide et leur soutien précieux et sans qui mon stage n'aurait certainement pas été aussi passionnant qu'il ne l'a été :

- Jean-Noël Temem, responsable de l'unité de recherche Nouvelles Technologies de l'Information et de la Communication, pour m'avoir accueilli au sein de son équipe ;
- Philippe David, pour m'avoir proposé ce stage et pour m'avoir suivi et conseillé durant toute la durée de ce stage, et peut-être pour plus longtemps encore ;
- Michel Tarralle, développeur initial de TVi, pour le travail énorme qu'il a pu accomplir, ainsi que toute son équipe, pour leur code mais aussi pour leur travail graphique et leurs bonnes idées ;
- toute l'équipe VideoLAN de l'École Centrale, pour le travail extraordinaire qu'ils accomplissent chaque année sur ce projet, ainsi que Jean-Philippe Rey et Dimitri Dagot ses encadrants au niveau de l'École ;
- tous les autres développeurs de VideoLAN, en particulier Christophe Massiot, Gildas Bazin et Sigmund Augdal, pour leurs conseils avisés mais aussi leurs critiques acerbes et toujours justifiées, et surtout pour le travail bénévole énorme qu'ils accomplissent ;
- Sébastien, Carl, Daniel, Chems, Grégory, Jérôme et Marc-Antoine pour leur compagnie agréable durant ce stage, ainsi que David, Sylvain et François, et surtout Christian pour son éternelle bonne humeur ;
- Mathieu Poumeyrol, pour `npunix.c`, pour son code d'exemple Unix et Win32, puis pour `npwin.cpp`, pour les headers Mozilla pour Win32, et

pour toutes ses informations qui m'ont fait gagner un temps précieux ;

- Colin Delacroix pour son code d'exemple OS X et Win32 ;
- Mumit Khan, pour son document *Howto create Netscape Client Plug-in with Cygwin/Mingw GCC* ;
- Michal Trojnara pour `stunnel`, Lars Brinkhoff pour `httptunnel`, Lool, MaXX, Prof, Xav et Walken, pour les avoir installés ou pour avoir redémarré mes ordinateurs lorsque j'en avais besoin ;
- tous les contributeurs de `vlc@videolan.org` et tous les utilisateurs de `#videolan` pour leurs tests, leurs *bug reports*, et leurs suggestions.

Chapitre 2

Introduction

Ce chapitre d'introduction présente brièvement les objets techniques impliqués dans le sujet du stage, ainsi que les raisons qui m'ont poussé à choisir ce stage.

2.1 TVi

Développé en 1997-1998 au sein de l'unité NTIC de la Direction de la Recherche de la SNCF, TVi (*Télévision, Vidéo et Informations à la place*) est un démonstrateur multimédia destiné à présenter un éventail des possibilités qu'offrent les nouvelles technologies en matière d'agrément et de divertissement mais aussi d'information à la place dans les TGV.

TVi est composé d'un serveur et de deux clients, tous trois des PC. Il ne s'agit pas d'un produit final mais est présenté comme un outil de communication vers les décideurs, première étape pour l'élaboration future d'un éventuel produit.

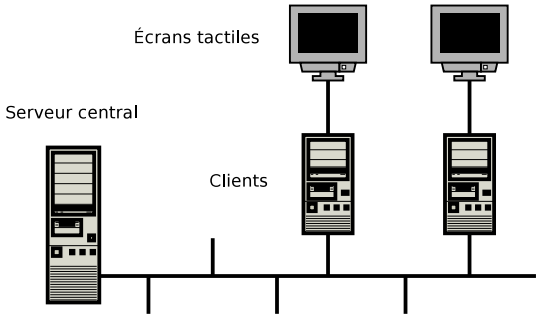

FIG. 2.1 – L'architecture de TVi

L'architecture de TVi s'articule autour d'une technologie principale qui est le web. Cette approche a permis de s'affranchir du coûteux développement d'une application graphique (puisqu'il suffit alors d'utiliser un navigateur web), au prix d'une rigueur supplémentaire au niveau des possibilités de l'interface (cette rigueur peut cependant être contournée par le biais de *plugins* pour navigateurs web).

Parmi les autres technologies utilisées par TVi, on peut citer le format d'animation *Macromedia Flash*, les formats de vidéo MPEG, Cinepak et Sorenson.

2.2 VideoLAN

VideoLAN est un projet de diffusion de vidéo numérique originaire de l'École Centrale. Originellement simple projet étudiant, c'est devenu depuis l'année 2000 un projet d'envergure mondiale grâce à sa diffusion publique et à l'accueil de développeurs extérieurs à l'École.

Une centaine de développeurs y contribue, et sa base d'utilisateurs est estimée à près de 100.000 utilisateurs. Des grands comptes tels que Phillips, AT&T, Motorola ou British Telecom utilisent VideoLAN et y contribuent.

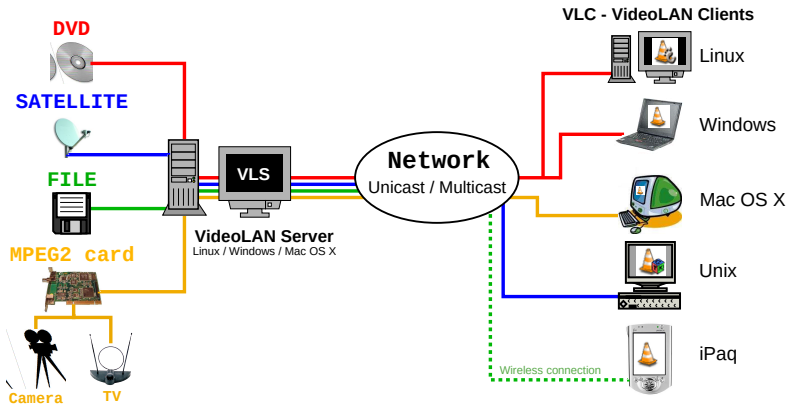

FIG. 2.2 – L'architecture de VideoLAN

La solution VideoLAN comprend les composants majeurs suivants :

– une partie serveur, VLS, destinée à envoyer des flux vidéo sur un réseau depuis une source de données (disque dur, DVD, carte satellite, ...) ;

– une partie client, VLC, destinée à récupérer les flux envoyés par le serveur et à les restituer sur un ordinateur.

Ce sont ses qualités techniques et novatrices qui ont fait retenir VideoLAN comme principale solution de modernisation de TVi.

2.3 Les besoins du stage

L'idée originale du stage était d'une part de développer des *plugins* pour navigateurs web basés sur VideoLAN, pour deux systèmes d'exploitation différents (Windows et Linux) afin de permettre à TVi de bénéficier de toutes les possibilités de diffusion vidéo supportées par VideoLAN, et d'autre part de tester les applications de cette diffusion vidéo à des ordinateurs de poche sur réseau sans fil.

2.4 Le choix du stage

2.4.1 La motivation

L'offre de stage de la SNCF mentionnant explicitement VideoLAN comme l'un des composants nécessaires du produit final, et ayant travaillé plusieurs années sur ce projet – dans un premier temps dans le cadre de mon projet de deuxième année à l'École Centrale, puis en tant que contributeur extérieur bénévole, mais aussi conseiller et formateur auprès des élèves ayant repris le projet les années suivantes – celui-ci me tenait particulièrement à cœur.

Le sujet du stage m'intéressait donc *a priori* beaucoup, mais l'attrait de la nouveauté était plus important dans d'autres stages à ma dispostion, sans compter que certaines de ces autres offres concernaient aussi VideoLAN, la SNCF n'étant absolument pas seule à s'intéresser à ce projet pour ses qualités à la fois techniques et économiques.

2.4.2 Le choix final

Mon choix s'est finalement porté sur ce stage pour plusieurs raisons supplémentaires :

– ma connaissance du projet VideoLAN et ma participation au design histoirique de celui-ci me donnait des atouts certains pour la gestion de ce projet ; personne selon moi n'aurait pu démarrer les développements requis sans une solide et coûteuse mise à niveau préliminaire.

– les développements proposés entraient dans la lignée des intérets du projet, et j'avais déjà réalisé des études préliminaires à titre personnel qui s'étaient avérées prometteuses ;

– la perspective de pouvoir communiquer lors de mon travail avec des gens dont je connaissais précisément les qualités humaines et les compétences techniques m'enchantait ;

Première partie

Objectifs

Chapitre 3

Problématique et enjeux

Dans ce chapitre je développerai les problèmes posés par TVi dans son état actuel, en particulier les problèmes d'obsolescence et de coût. Je donnerai aussi un aperçu d'autres axes de recherche qui seront explorés durant le stage ainsi que les enjeux qu'ils représentent pour la Direction de la Recherche et de la Technologie.

3.1 TVi

3.1.1 Problème du vieillissement

Le système TVi se fait vieux, non seulement en raison des logiciels qui le composent, mais aussi de par les technologies qu'ils implémentent elles-mêmes. Les progrès effectués durant les trois dernières années permettent aujourd'hui des applications jugées impossibles lors de la réalisation de TVi (comme par exemple le décodage logiciel de MPEG2, ou la diffusion de MPEG4 via un réseau), mais rendent aussi obsolètes certaines technologies utilisées, comme par exemple la norme HTML3.2 ou le langage JavaScript spécifique au navigateur Netscape 4.7.

3.1.2 Problème du coût

Le déploiement total de TVi nécessiterait l'implémentation de 170.000 unités ; on comprend aisément la nécessité de réduire les coûts de déploiement au maximum, même au prix de développements légèrement plus coûteux.

L'utilisation d'un maximum de logiciels libres, dont le coût en licences est quasi-nul, a été considérée comme une approche saine et sera un axe de développement majeur. Mais d'autres moyens de réduire le coût pourront être envisagés, comme par exemple l'utilisation de deux écrans sur une seule unité centrale.

3.1.3 Problème de la maintenance

Certains des logiciels qui composent TVi sont, comme nous l'avons déjà vu, extrêmement dépassés. Mais se pose aussi le problème des logiciels n'existant tout simplement plus ; par exemple, le serveur web de Netscape est un produit complètement abandonné. Le logiciel de serveur vidéo Mediabase a lui aussi été abandonné par SGI. Par ailleurs les écrans tactiles utilisés pour le prototype ne sont plus fabriqués.

Au niveau du produit lui-même se posent d'autres problèmes de maintenance ; à l'obsolescence des langages s'adjoint le problème de la modification de la mise en page. Par exemple, le graphiste ayant réalisé les icônes et animations de TVi n'est probablement plus disponible pour en réaliser d'autres. Autre exemple, celui de la résolution de l'écran : toutes les pages web de TVi sont prévues pour être visualisées sur un écran de taille 800*600, ce qui correspond effectivement à la taille des écrans tactiles existant, mais deviendrait problématique si l'on était amenés à changer ces écrans. Par ailleurs, les pages web de TVi étant toutes statiques, beaucoup de duplication de code serait nécessaire pour en créer une nouvelle.

3.1.4 Améliorations possibles

En plus des problèmes et limitations déjà évoqués et qu'il tiendra de corriger, un lot d'améliorations possibles a été envisagé ; des axes de recherche tels que la diffusion de MPEG4, le *streaming* de radio ou d'un signal caméra (*webcam*) en temps réel ont retenu notre attention.

3.2 Autres applications du *streaming* vidéo

3.2.1 Le *wireless*

Les technologies de communication sans fil (*wireless*) ont connu un essor explosif au début des années 2000 avec l'avènement de la téléphonie mobile. Aujourd'hui les tentatives pour accorder les solutions de *streaming* vidéo aux technologies *wireless* n'en sont encore qu'à leurs balbutiements, et les développements réalisés sur VideoLAN sont toujours à la pointe de la technologie. Des sociétés comme Phillips, ou British Telecom utilisent d'ailleurs VideoLAN dans leurs développements.

Des tests et éventuellement des développements seront donc réalisés pour avoir un aperçu des possibilités qu'offre VideoLAN en matière de diffusion de vidéo sur *wireless*, l'unité de recherche étant dotée d'un réseau *wireless* et de plusieurs types d'ordinateurs de poche de type PDA.

L'intéret pour l'unité Nouvelles Technologie de l'Information et de la

Technologie de s'orienter vers de tels axes de recherche est de pouvoir offrir une vitrine de son savoir-faire et des possibilités concrètes qu'offrent ces technologies.

Chapitre 4

Cahier des charges

Ce chapitre décrit de manière précise les livrables attendus en fin de projet, ainsi que les moyens de validation à mettre en œuvre pour juger de leur conformité avec les attentes.

4.1 Définition des objectifs

4.1.1 Modernisation de TVi

La modernisation de TVi passe par plusieurs sous-objectifs :

- une mise à jour de tous les logiciels et, par extension, technologies composant TVi ;
- une adaptation de VideoLAN à TVi ;
- une rénovation des pages web de TVi suite aux réalisations sus-citées ;
- une adaptation fonctionnelle de TVi à la modernisation des concepts et paradigmes que le projet incarne, autant que la technique le permet.

4.1.2 Réduction du coût de déploiement

La réduction du coût de déploiement passe au minimum par les objectifs suivants :

- l'utilisation d'un maximum de logiciels libres ou gratuits après l'étude de leurs qualités intrinsèques et extrinsèques (*ie.* leurs fonctionnalités objectives mais aussi la comparaison de celles-ci avec les produits propriétaires concurrents) ;
- l'étude de la diminution de l'encombrement et de la réduction des ressources matérielles nécessaires à un déploiement ;

– l'automatisation du déploiement.

4.1.3 Réduction du coût total de d'exploitation

Se posera aussi, dans le cas d'un éventuel déploiement, le problème du coût d'exploitation. On peut citer les objectifs suivants :

– la réduction des coûts d'administration en utilisant des procédures de maintenance automatiques ;

– l'utilisation de technologies ne nécessitant pas le versement de royalties pour leur utilisation, et estimées stables dans le temps afin de ne pas risquer un redéveloppement complet (on s'attachera en particulier aux formats de flux vidéo) ;

– la facilitation de la mise à jour des logiciels développés.

4.2 Les moyens de validation

4.2.1 Tests de régression

Une plate-forme de tests de régression devra être mise en place, à la fois pour la partie VideoLAN du projet et pour la partie TVi. Les tests devront pouvoir valider les critères suivants :

– les pages web de la nouvelle version de TVi devront être fonctionnellement au moins équivalentes à l'ancienne version ;

– les pages web de la nouvelle version de TVi devront être visuellement équivalentes à l'ancienne version ;

– tous les formats de vidéo supportés par la première version de TVi devront être supportés par la partie VideoLAN du projet ;

– la partie VideoLAN du projet devra être adaptable à la partie TVi de manière transparente, avec en cas de besoin la possibilité de revenir à l'ancienne version du lecteur vidéo.

4.2.2 Tests de performance

Afin de ne pas dilapider les économies faites en termes de matériel, il conviendra de réaliser des tests de performance pour s'assurer que les développements réalisés ne nécessitent pas de matériel informatique trop coûteux. La batterie de tests pourra par conséquent porter sur l'utilisation de la mémoire et du processeur, mais aussi sur des appréciations plus subjectives de vitesse de réaction.

4.2.3 Calcul des réductions de coût

Un calcul de la réduction du coût de TVi devra être effectué, et devra faire intervenir les résultats (ou en cas d'impossibilité à chiffrer, les estimations) suivants :

- coût des logiciels
- coût du matériel
- coût de déploiement
- coût d'administration

4.2.4 Documentation

Pour valider les possibilités d'exploitation des développements réalisés, une documentation devra être réalisée, permettant à la fois une exploitation des outils réalisés (la documentation utilisateur) et une éventuelle poursuite des développements par un tiers (la documentation développeur).

Deuxième partie

Organisation

Chapitre 5

Étude préliminaire

Ce chapitre présente les résultats d'une étude préliminaire effectuée sur TVi, ainsi qu'un état de l'art dans le domaine de la diffusion de vidéo numérique. Ceux-ci ont permis de faire des choix techniques et organisationnels préliminaires que je présente aussi.

5.1 Étude superficielle de TVi

Durée : 2 semaines.

Les premières semaines de mon stage ont été employées à l'étude de l'existant, en particulier TVi : lecture du rapport, étude rapide de l'architecture, mais sans rentrer dans une étude trop poussée qui, si elle s'était avérée nécessaire, aurait été du ressort d'une phase complète du projet. Le but de cette phase était d'estimer la charge des développements nécessaires, mais aussi de faire des choix techniques initiaux.

Les considérations suivantes sont le fruit de mon étude :

- il y a énormément de redondance dans le code de TVi, et l'utilisation d'un langage auteur comme PHP permettra de la supprimer ;
- les pages web de TVi sont loin d'être conformes aux différents standards qu'elles sont censées implémenter, un gros travail de transcription sera donc nécessaire ;
- TVi utilise des plugins propriétaires pour certains formats, il faudra donc s'assurer que les formats en question puissent être lus par la partie VideoLAN du projet. Dans le cas contraire, il faudra que les plugins en question soient réutilisables dans la version modernisée de TVi.

5.2 État de l'art en *streaming* vidéo

Durée : 1 semaine.

J'ai réalisé pendant une semaine une petite étude afin de déterminer les possibilités réelles du *streaming* vidéo à l'heure actuelle. Les formats cités précédemment (MPEG, Cinepak et Sorenson) sont toujours employés, mais de nouvelles technlogies (MPEG2, MPEG4, DivX) sont apparues et il faudra les prendre en compte.

5.3 Choix initiaux

Un choix nécessaire majeur a été celui de la nouvelle plate-forme de TVi ; en effet ce choix conditionnait l'achat de nouvelles machines et les développements effectués sur VideoLAN ; à la lumière de l'étude de TVi et de ma connaissance de VideoLAN, le choix s'est porté sur le navigateur web *Mozilla*. Il restait à choisir un système d'exploitation, mais nous nous sommes laissé du temps pour faire ce choix, car les développements préliminaires pouvaient être faits quel que soit le système d'exploitation.

J'ai de plus décidé de reporter les travaux de recherche en diffusion sur réseau sans fil à la fin du stage. En effet, des recherches sur VideoLAN sont en cours dans le domaine, et il sera possible de profiter des résultats de ces recherches.

Chapitre 6

Méthodologie

Dans ce chapitre, je présente brièvement une méthode de développement dite « légère » (*eXtreme Programming*) et son application au projet. Pour plus de détails le lecteur pourra se reporter à l'annexe décrivant en détail cette méthode.

6.1 Les méthodes légères

Un des buts des méthodes légères est de diminuer le nombre de documents intermédiaires produits durant la conduite du projet, et n'apportant pas de vraie valeur ajoutée au projet. J'ai décidé d'implémenter, pour la partie VideoLAN du projet, les grands traits de la méthode XP (ou *eXtreme Programming* – voir annexe correspondante), à savoir :

- dépendance d'un petit noyau central : la base des développements se fait sur un noyau destiné à recevoir des incréments successifs ;
- méthode incrémentale : plutôt que de suivre un plan de développement linéaire, la méthode procède par courts incréments d'une à deux semaines ;
- travail par binôme : les développements se font à deux, avec revue croisée du code de chacun, afin de détecter au plus vite les erreurs commises plutôt que de s'en apercevoir plus tard lors de tests de validation ;
- livraisons fréquentes : après chaque incrément, le logiciel est opérationnel et implémente une nouvelle fonctionnalité, que le client peut tester pour permettre une remontée de problèmes plus rapide que lors des tests d'intégration finaux comme pour les méthodes traditionnelles.

La méthode XP est une méthode adaptative qui me permettra de réagir au plus vite au dynamisme du projet VideoLAN. En effet un développement long devra se baser sur un état du code à une date précise, et nécessitera une synchronisation des développements à la fin ; il est possible que des divergences importantes soient intervenues, ce qui rajouterait une surcharge de travail lors de la synchronisation. Il est même possible que la synchronisation devienne impossible sans remettre tous les développements en cause.

6.2 Application au projet

Je n'utiliserai la méthode XP que pour la partie VideoLAN du projet ; en effet, celle-ci se prête bien au développement d'un noyau central et à de petits incréments. De plus, des livraisons fréquentes sont possibles et j'en attends un bon retour car les utilisateurs de VideoLAN sont nombreux et font habituellement beaucoup de remontées de problèmes. Si ce retour venait à ne pas être suffisant, je serai amené à effectuer des tests internes plus poussés.

Afin de pouvoir travailler en binôme, je choisirai à chaque incrément un développeur de l'équipe de développement de VideoLAN avec qui je travaillerai étroitement.

Il est possible de résumer mon application de la méthode XP dans le schéma suivant :

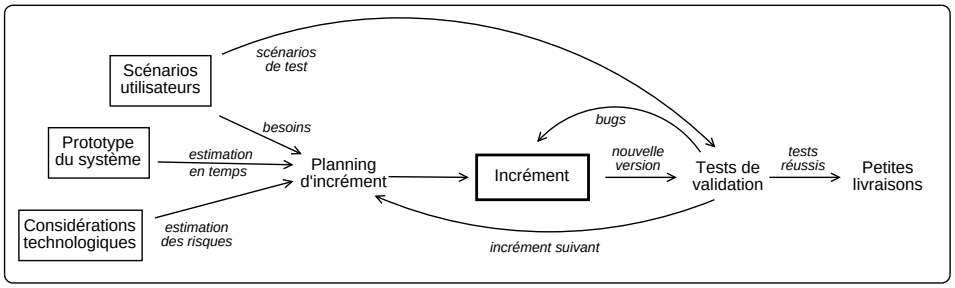

FIG. 6.1 – La méthode XP

Chaque incrément est basé sur l'implémentation d'un ou plusieurs scénarios utilisateur. Les scénarios utilisateur ne sont pas des spécifications techniques, mais de simples descriptions d'une utilisation du logiciel, comme par exemple « l'utilisateur appuie sur le bouton stop et la vidéo s'arrête ».

Les scénarios seront décidés en fonction des possibilités que j'aurai étudiées dans TVi, et serviront de base à chaque incrément. Ils serviront aussi à définir des tests de validation avant chaque livraison. Après chaque incrément, de

nouveaux scénarios seront proposés, jusqu'à ce que toutes les fonctionnalités requises soient implémentées.

6.3 La méthode classique

Les autres développements seront menés selon une méthode plus classique suivant le modèle en V :

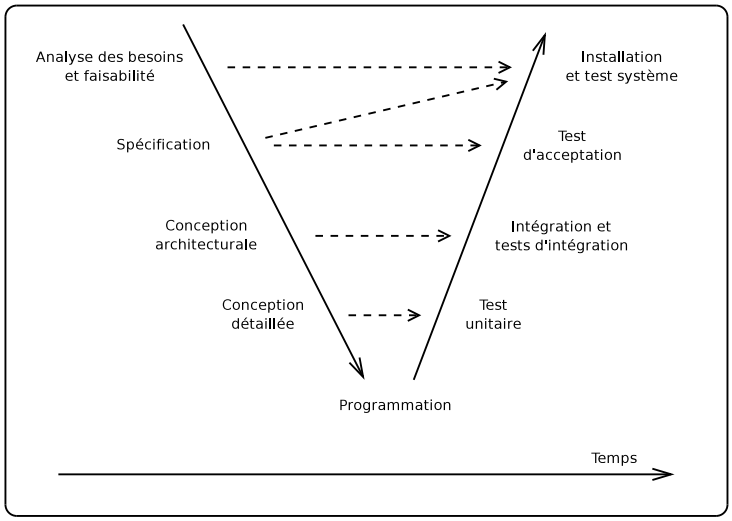

FIG. 6.2 – Le modèle en V

Étant seul développeur disponible pour les parties ne concernant pas VideoLAN, il m'est impossible de choisir un binôme, ce qui élimine d'office la méthode XP. J'ai aussi éliminé les autres méthodes légères car celles-ci sont adaptées aux développements dont les objectifs changent avec le temps, ce qui n'est pas le cas ici.

Il m'appartiendra de définir en début de chaque phase les tests de validation nécessaires à l'application de cette méthode.

Chapitre 7

Organisation

Ce chapitre présente de manière détaillée l'organisation du projet, en particulier les moyens de communication internes et externes mis en œuvre et le système de remontée d'information. Sont ensuite présentés une ébauche de planning, une étude de risques, ainsi que le planning détaillé résultant de cette étude.

7.1 Ressources matérielles

Tout le matériel nécessaire aux développements est déjà disponible dans le laboratoire dans lequel je serai amené à travailler. De plus les logiciels de développement que je vais utiliser sont tous gratuits et disponibles sur Internet, ce qui évitera toute dépense supplémentaire et éliminera les risques de commande en retard.

Pour la mise en place de la nouvelle plate-forme de TVi, un serveur et des clients ont déjà été commandés et seront livrés dans le mois.

7.2 Organisation interne

7.2.1 Interactions avec la hiérarchie

Mon responsable de stage, Philippe David, sera aussi mon interlocuteur privilégié avec ma hiérarchie. Comme il nous est possible de nous voir physiquement très souvent, nous avons décidé de la stratégie de remontée d'information suivante :

- quotidiennement, une brève entrevue pour faire un rapide point de l'avancement ;
- hebdomadairement, une réunion plus longue permet de faire le point de manière moins poctuelle sur l'avancement, et est l'occasion de rendre

compte des réunions hebdomadaires du projet VideoLAN.

- à des dates qui seront définies au cours du projet, des présentations ou des démonstrations auront lieu pour avoir une vision plus globale de l'avancement.

Par ailleurs, les entrevues et présentations seront consignées dans un journal de stage qui servira à la rédaction du rapport final.

7.3 Interactions externes

7.3.1 VideoLAN

Le projet VideoLAN se réunit hebdomadairement pour une réunion ayant trait aux directions prises par le projet, à la fois sur le plan technique et sur le plan communication. Je participe habituellement à ces réunions à titre personnel, faisant partie des développeurs majeurs de VideoLAN, et j'entretiens d'excellents rapports avec toute l'équipe. Le plan des réunions, sauf cas exceptionnel, est le suivant :

- récapitulatif des points d'action de la semaine passée ;
- bilan des activités techniques et non-techniques de la semaine ;
- dégagement des objectifs atteints, des points d'action manqués ;
- réflexion commune sur les causes possibles de retard ou d'échec, étude des mesures correctives et solutions de repli possibles ;
- dégagement de nouveaux points d'action à entreprendre pour la semaine suivante et désignation des responsables pour chacun de ceux-ci.

On peut rapprocher le cadre des réunions VideoLAN à la réunion de jeu de livraison (avant et après chaque itération) dans un cycle d'XP ; c'est en effet la méthode de développement que je compte employer dans le cadre de mes développements conjoints avec l'équipe VideoLAN, et je continuerai donc à prendre part à ces réunions.

L'équipe VideoLAN utilise par ailleurs des moyens de communication plus directe liés à Internet, qui sont les suivants :

- des *mailing-lists* destinées soit aux utilisateurs soit aux développeurs des différents produits ;
- un canal IRC de discussion sur lequel peuvent avoir lieu des discussions en temps réel lorsque la nécessité et l'occasion se présentent, et sur lequel les développeurs majeurs sont présents en permanence ;

- un système de gestion de version de logiciels, qui n'est pas à l'origine un moyen de communication, mais qui permet d'adjoindre un message informatif ou interrogatif à chaque modification du code qui sera automatiquement envoyé dans une *mailing-lists* de développeurs sans avoir à utiliser d'autres outils, ce qui est souvent un gain de temps certain.

7.3.2 IDM

IDM est une société ayant manifesté un intéret pour le développement d'un plugin VLC pour Mozilla. Il s'agit d'un développement que je serai amené à faire, le Directeur de la Recherche d'IDM sera donc disponible pour me fournir des informations qui pourraient m'être nécessaires dans mon travail.

Il est à noter qu'aucune base contractuelle n'a été établie pour cet accord ; en effet tout contrat aurait été contraignant pour moi, alors que le présent accord tacite me laisse toute liberté, en particulier au niveau du calendrier. En contrepartie je n'ai pas de garantie totale de retour d'information, c'est un point qu'il ne faudra pas négliger.

7.3.3 Le grand public

Le grand public est une partie primordiale du projet, car il constitue le meilleur moyen de tester des petites livraisons de la méthode XP. Il s'agit d'un avantage énorme d'un logiciel libre comme VLC, les utilisateurs sont des testeurs sans le savoir.

7.4 Ébauche de planning

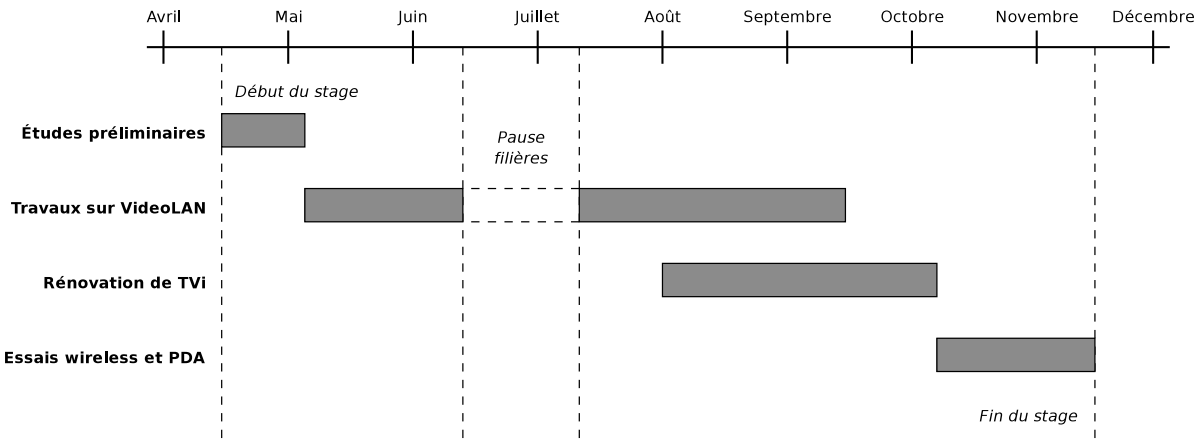

FIG. 7.1 – Ébauche de planning

Cette prémière ébauche de planning fait apparaitre les deux grands chantiers du projet que sont d'une part le travail de fond sur VideoLAN, et d'autre part son adaptation à TVi ainsi que la modernisation de ce dernier, sur la base d'estimations de charge de travail qui découle des études préliminaires.

7.5 Étude de risques

7.5.1 Les risques

On peut énumérer les risques majeurs classiques dans le cadre du développement d'un système informatique :

- défaillance de personnel : c'est un risque à gravité très importante car je suis en gros le seul personnel disponible, mais à probabilité très faible tant du point de vue technique (car j'ai une longue expérience personnelle des technologies que je serai amené à utiliser) que du point de vue humain (je connais les personnes avec lesquelles je serai amené à travailler) ;

- calendrier irréaliste : les risques sont à gravité moyenne, mais la probabilité est importante (je pense notamment à des vices cachés de TVi) ;

- budget irréaliste : les risques sont à gravité failble car il s'agit essentiellement d'un projet de recherche, et la probabilité est très faible aussi puisqu'il a justement été décidé une réduction du coût de l'existant ;

- interfaces utilisateurs inappropriées : le risque est à gravité importante (s'agissant d'un projet dans lequel l'egonomie tient une grosse part) mais à probabilité quasi-nulle puisque les interfaces ont déjà été réalisées ;

- tâches externes défaillantes : le risque est à forte probabilité puisque les développeurs extérieurs de VideoLAN ne sont pas contractuellement réttachés à mon projet, mais la gravité n'est que moyenne car je pense être capable de réaliser les développements moi-même ;

- problèmes de performance : le risque est à gravité moyenne (car il impliquerait un achat de matériel supplémentaire et donc un dépassement de budget) et à probabilité faible compte tenu des tests de régression prévus ;

- technologie inadéquate : le risque est à gravité importante car il rendrait inutiles tous les travaux effectués, mais la probabilité est faible du fait de mon expertise technique et des analyses préliminaires réalisées.

Par ailleurs les méthodes légères ne sont pas une méthode miracle et ont elles aussi leur lot de risques :

- remise en cause du noyau : le risque est à gravité très importante puisque tous les développements de la partie VideoLAN vont s'articuler autour du noyau, et que la partie TVi en est dépendante aussi, cependant la probabilité est très faible car les moyens (humains et techniques) engagés dans le design sont considérables ;

- remise en cause des incréments précédents : c'est bien entendu le risque majeur d'une méthode itérative, la probabilité est importante mais la gravité est mineure car le découpage en incréments sert justement à minimiser la perte de temps induite ;

- impossibilité d'intégration d'un incrément : c'est un risque à probabilité peu importante si le noyau est correctement pensé, mais à gravité importante car elle remet éventuellement le noyau en cause.

On peut observer un résumé des risques présentant une appréciation relative de leurs gravités et probabilités respectives :

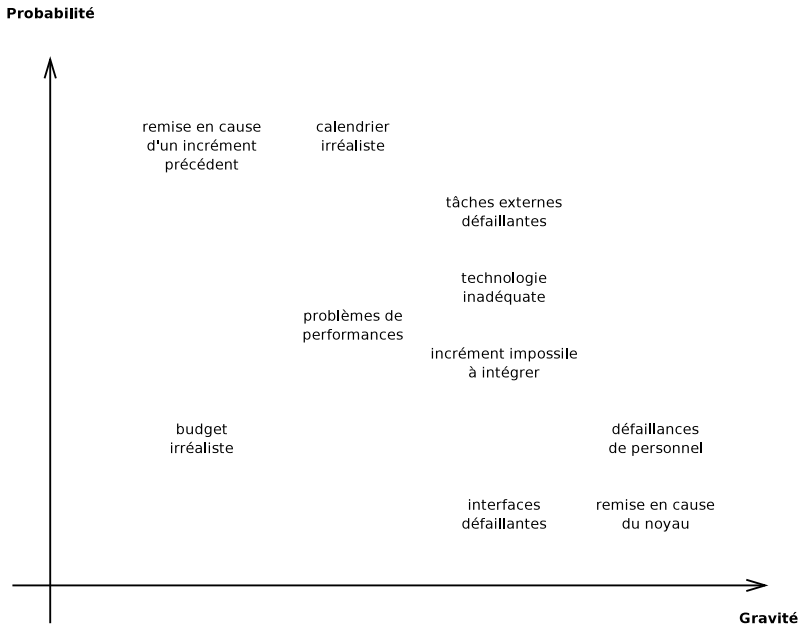

FIG. 7.2 – Récapitulatif des risques

7.5.2 Solutions de repli prévues

Pour chacun des risques évoqués, j'ai dégagé des moyens de minimiser le risque et des solutions de repli envisagées (celles-ci pourront très bien évoluer avec le temps) :

- défaillance de personnel : il faudra opérer une réorganisation des groupes de travail, et dans le pire des cas il me faudra continuer tout seul : c'est un risque à vraiment minimiser, il faudra donc convenablement choisir mes binômes pour la méthode XP.

- vices cachés de TVi : la seule solution sera ici de persévérer et d'accepter un retard dans le développement, au prix peut-être de la suppression de quelques développements annexes, c'est pourquoi ces derniers ont été placés en fin de planning pour faire tampon ;

- budget irréaliste : les risques sont *de facto* minimisés par les objectifs du projet, et l'utilisation de logiciels gratuits pour le développement lui-même permettra encore de minimiser ceux-ci ;

- interfaces utilisateurs inappropriées : les petites livraisons fréquentes permettront au logiciel d'être testé et de détecter au plus vite ces problèmes et de les corriger ;

- tâches externes défaillantes : ce risque est lié à celui de la défaillance de personnel, une solution de repli éventuelle est soit de changer de binômes pour tenter d'être plus efficace, soit de cesser de faire intervenir des extérieurs sur le projet, ce qui retardera par contre le développement ;

- problèmes de performance : les tests de performance effectués à chaque itération permettront de détecter ces problèmes et d'y remédier au mieux, si le dimensionnement initial a été correct il s'agit habituellement d'une simple optimisation d'algorithme à effectuer, sinon un dépassement de budget est à craindre pour se doter de machines plus puissantes ;

- technologie inadéquate : l'état de l'art effectué en début de projet est là pour exclure complètement ce risque ;

- remise en cause du noyau : il s'agit d'un point extrêmement critique qu'il faut absolument minimiser, c'est pourquoi une durée très importante a été accordée à l'étude de celui-ci, de même qu'à son développement ;

- remise en cause des incréments précédents : c'est un point controversé, car je suis parfois d'avis que la réécriture d'un élément permet de le parfaire (ce qui est d'ailleurs la seule solution de repli sensée), mais il faudra veiller à minimiser ce risque en attachant le plus grand soin à l'étude précédant chaque itération ;

- impossibilité d'intégration d'un incrément : la réduction de ce risque passe par une estimation préliminaire du contenu des incréments, et surtout par le soin apporté à la création du noyau (celui-ci devrait être suffisamment souple pour permettre des modifications mineures acceptant l'intégration) ; si toutefois cela venait à se produire, il faudrait réétudier l'incrément car remettre le noyau en cause est inaceptable.

7.6 Planning détaillé

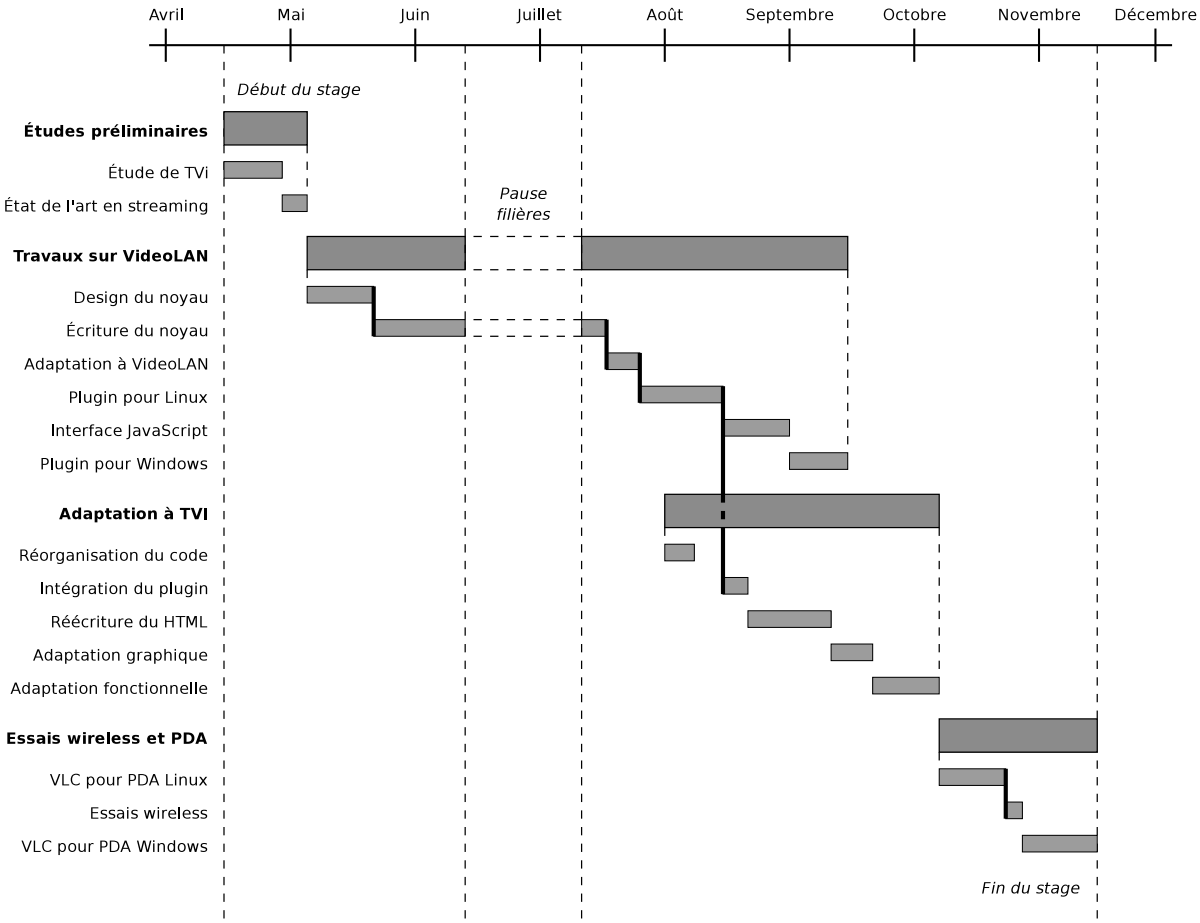

FIG. 7.3 – Planning détaillé

Les traits verticaux gras représentent des tâches interdépendantes dont la seconde ne peut se faire que lorsque la première a été accomplie. On remarque le soin tout particulier qu'il faudra apporter aux travaux sur VideoLAN.

Troisième partie

Déroulement

Chapitre 8

Déroulement

Ce chapitre retrace le calendrier du projet dans ses grands points, en le mettant en rapport avec le planning et en mettant en évidence les variations de ce calendrier avec le planning. Dans un deuxième temps, je détaille les raisons de ces variations, les solutions apportées, ainsi que les résultats intermédiaires obtenus durant le déroulement du projet.

8.1 Calendrier

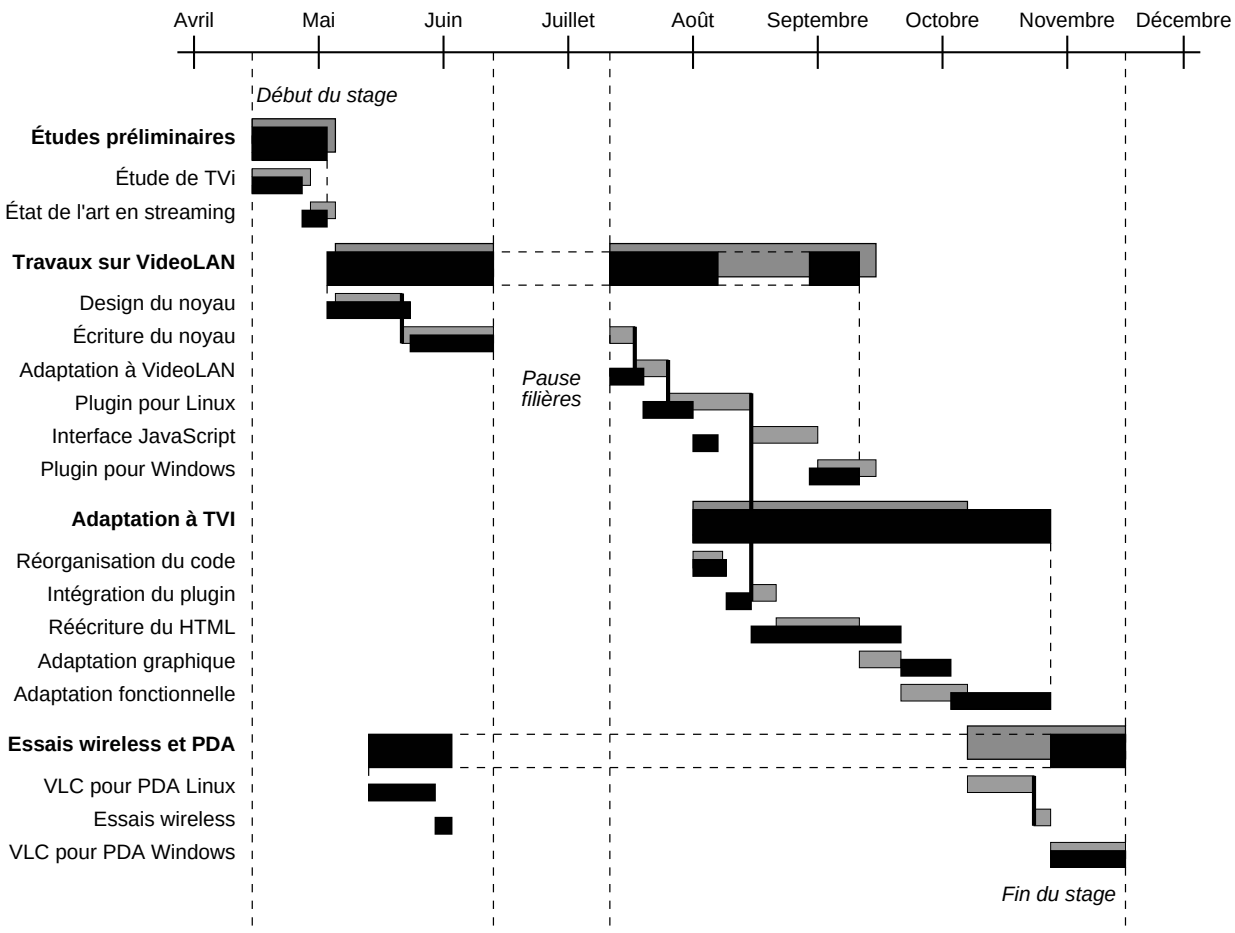

FIG. 8.1 – Calendrier des réalisations

Les intervalles noirs présentent le temps effectivement passé sur chaque tâche prévue. On remarque trois variations importantes avec le planning initial :

- la phase *Travaux sur VideoLAN* a commencé et fini légèrement en avance, mais on y observe surtout plus de trois semaines d'inactivité, qui ont donc pu être consacrées à d'autres tâches ;
- la phase *Essais wireless et PDA* a commencé près de 5 mois à l'avance, et s'est terminée dans les temps ;
- la phase *Adaptation à TVi* a commencé dans les temps, mais s'est terminée avec près de 3 semaines de retard.

8.2 Déroulement

8.2.1 Études préliminaires

Cette phase du projet a duré de mi-avril à début mai, comme initialement prévu. Elle s'est en fait terminée légèrement plus tôt car l'étude de TVi a été plus courte que prévu d'environ trois jours.

Il est ressorti de cette phase l'ébauche de planning présentée dans la partie précédente, ainsi que le planning détaillé. J'ai de plus réalisé un travail de recherche d'outils de documentation, et d'outils de gestion de problèmes (*bug tracking system*). Les résultats de ces travaux sont présentés en annexe.

8.2.2 Travaux sur VideoLAN

Ces travaux ont démarré avec un peu d'avance, et le temps gagné a permis de commencer le design du noyau plus tôt. Celui-ci a d'ailleurs duré plus longtemps que prévu en raisons de discussions techniques très longues avec l'équipe VideoLAN et une certaine latence dans les communications. Mais au total la phase d'écriture a fini avec une semaine d'avance, ce qui montre que la durée du design a été très utile.

La suite des développements s'est alors effectuée très vite : deux semaines d'avance pour le *plugin* Linux, et trois semaines d'avance pour l'interfaçage de celui-ci dans le langage JavaScript. Il a donc été possible de commencer plus tôt les traavaux sur TVi, et de se concentrer sur ceux-ci plutôt que de mener de front les deux sous-objectifs. Une des raisons d'une telle avance a été la masse de contributions par des extérieurs au projet, qui a nécessité une adaptation rapide. La méthode *xp* a permis cette adaptation puisque chaque contribution a pu être intégrée au plus tard lors du nouvel incrément.

La dernière partie de cette phase, l'écriture du *plugin* Windows, a été commencée à la date prévue et naturellement terminée dans les temps.

Il est ressorti de cette phase tous les livrables requis, et largement dans les temps : les *plugins* demandés et correctement interfacés, ainsi que les différents systèmes de tests de régression et la documentation correspondante, effectués en même temps que le développement.

8.2.3 Adaptation à TVi

Cette phase a été la plus délicate du projet. Elle a débuté dans les temps, avec même un gain de temps pour l'intégration du *plugin* vu les résultats excellents de la phase précédente. Cependant la réécriture du code HTML a duré beaucoup plus longtemps que prévu et a fait perdre toute l'avance du

départ. Le retard de cette phase a ensuite empiré, et la dernière phase d'adaptation fonctionnelle s'est terminée avec près de trois semaines de retard.

Le retard peut s'expliquer très schématiquement par des « vices cachés » de TVi. En effet, pour prendre un exemple précis et simple, le rapport mentionnait que les pages web composant TVi avaient été réalisées grâce à un logiciel auteur (Netscape Composer). Une lecture de ces pages confirmait cela *a priori*. Malheureusement il se trouve que ces pages ont été ensuite modifiées manuellement, rendant plus difficile leur modification.

De tels problèmes ne peuvent pas être détectés aisément, même par un spécialiste ; de plus une étude plus approfondie aurait nécessité une ou deux semaines supplémentaires, pour en arriver à une conclusion qui n'aurait malgré tout pas permis d'éviter le retard final. On est là en présence d'un exemple où une étude poussée ne permet pas de gagner de temps, et dans ce cas il vaut mieux prévoir une durée tampon suffisamment large.

Malgré leur retard, les objectifs de cette partie ont été correctement remplis, et deux présentations d'avancement du projet à ma hiérarchie ont permis de valider l'avancement durant la période de développement. Le résultat principal a été une version totalement rénovée de TVi, n'utilisant plus aucun logiciel propriétaire, et reposant sur VideoLAN pour toute la partie vidéo.

8.2.4 Essais wireless et PDA

J'avais prévu cette dernière partie pour la fin du stage car je prévoyais que des développements externes allaient être faits dessus, et je voulais permettre un intervalle tampon à la fin du projet dans l'éventualité où la phase *Adaptation à TVi* prendrait du retard.

Comme je m'y attendais, les développements pour PC de poche sous Linux et réseaux sans fil ont été effectués très tôt par des contributeurs du projet VideoLAN ; j'y ai personnellement participé pour orienter les développements dans les directions qui me seyaient, mais la charge de travail a été extrêmement légère et n'a pas causé de désagréments dans les objectifs que je poursuivais à ce moment. Par contre, aucun intéret n'ayant été manifesté par le projet VideoLAN pour la version PC de poche sous Windows, j'ai dû m'en charger, et j'ai terminé ce travail à la date prévue dans le calendrier, le retard de la phase précédente ayant été compensé par l'économie de temps qu'a représenté le développement anticipé de la partie Linux.

Au final, une version de VLC pour PC de poche a été réalisée, à la fois sous Linux et sous Windows, et des essais fructueux de diffusion vidéo par réseau sans fil ont été effectuées pour la version Linux. La version Windows n'a à ce jour pas encore été testée.

8.2.5 Divers

Des recherches annexes ont été faites durant la durée projet, visant à améliorer l'efficacité des tâches principales mais sans s'inscrire précisément dans l'une de ces tâches. Les recherches d'outils de documentation et de gestion de problèmes en sont un exemple.

Parmi les autres exemples de travaux de recherche, on peut citer de brèves recherches pour faire fonctionner les écrans tactiles de l'ancienne version de TVi (infructueuses et ayant mené à la commande de nouveaux écrans), ou des tentatives de réduction de coût et d'encombrement en faisant fonctionner deux écrans sur la même machine (fructueuses celles-ci).

Quatrième partie

Bilan et conclusions

Chapitre 9

Bilan

Ce bref chapitre présente un inventaire exhaustif des réalisations effectuées durant le stage, ainsi que les moyens mis en œuvre pour les valider.

9.1 Réalisations

À l'issue du stage, les livrables disponibles (ou en cours de finition au moment de la rédaction du rapport) sont les suivants :

- une version du logiciel VLC de VideoLAN sous forme de *plugin* pour navigateurs web, fonctionnant sous Linux et Windows ;
- une version totalement rénovée de TVi, fonctionnant sur les deux plateformes Linux et Windows, et prête à supporter de nouveaux types de médias grâce aux *plugins* sus-cités ;
- une version des clients TVi dépourvue du maximum de logiciels propriétaires et utilisant des formats standard ouverts, et dont l'installation automatisée est facilitée par l'utilisation de packages ;
- des clients TVi de coût réduit car pouvant fonctionner avec un seul ordinateur pour deux écrans ;
- un serveur TVi dépourvu de tout logiciel propriétaire, et à l'administration facilitée par l'automatisation des mises à jour de logiciels.
- une version de VLC fonctionnant sur PDA, à la fois sous Linux et sous Windows, et permettant la réception de vidéo transitant sur un réseau sans fil.

9.2 Validation des réalisations

Les différents moyens de validation suivants ont été employés :

- pour la partie VideoLAN, les tests de régression et de performance mis au point durant le développement ont été effectués et sont tous passés avec succès ;
- pour la partie TVi, l'intégralité des fichiers vidéo présents sur l'ancienne version ont été testés sur la nouvelle avec succès ;
- toujours pour la partie TVi, un parcours de toutes les pages et différentes simulations d'utilisation réelle ont permis de valider la conformité aux attentes en termes de non-régression des fonctionnalités ;
- une documentation pour développeurs de VLC, une documentation pour utilisateurs du *plugin*, et une documentation présentant la nouvelle configuration de TVi destinée aux administrateurs (en annexe du présent document) ont été réalisées ;
- le coût en logiciels de TVi a été ramené à zéro ; avec une estimation de départ pour deux machines de 600 EUR en logiciels, 1200 EUR d'écrans et 1800 EUR d'unités centrales et de connectique diverse, l'économie est de l'ordre de 600 EUR pour deux postes clients ; de la même manière, la réduction de coût du serveur est de l'ordre de 3000 EUR ;
- la possibilité pour les clients TVi de pouvoir utiliser une seule unité centrale pour deux écrans permet encore de réduire le coût de deux clients de 600 EUR, soit une réduction totale du coût du démonstrateur d'environ 60%.

Chapitre 10

Conclusions

Ce chapitre tire les enseignements du stage en matière de technique, d'organisation et de relations humaines. Des perspectives sont ensuite ouvertes pour d'éventuelles suites données au projet.

10.1 Enseignements

10.1.1 L'importance de la technique

Les compétences techniques ont été décisives à plusieurs stades du projet et ne sont pas à négliger, et il est indispensable d'avoir à la fois une vue d'ensemble du projet et des compétences variées pour pouvoir prendre les bonnes décisions.

Par exemple une erreur de jugement dans le design du noyau aurait pu avoir des conséquences catastrophiques, et la seule connaissance de VideoLAN ne permettait pas d'anticiper tous les risques, il fallait connaître précisément les besoins de TVi. De même les retards de la partie TVi sont dus à une mauvaise estimation qui aurait pu être affinée par un expert ayant une vision globale du projet.

10.1.2 L'intéret des méthodes légères

La leçon de la méthode XP est sa rapidité d'adaptation aux conditions changeantes. Une méthode trop planifiée dans les détails n'aurait pas permis d'avancer aussi vite, alors que cette méthode incrémentale a permis d'adapter très rapidement au projet les contributions extérieures inattendues.

Cependant il ne faut pas perdre de vue que les méthodes légères ne sont pas une réponse à tous les problèmes ; en particulier elle ne se serait pas du tout prêtée à la partie TVi du projet, très linéaire et peu sujette à des

changements importants, c'est pourquoi une méthode classique en V était plus adaptée à la conduite de cette partie.

10.1.3 Leçons personnelles

J'ai pu découvrir dans le cadre d'un même projet les différences entre le travail exclusivement personnel et le travail en équipe ; une leçon capitale que j'en ai retirée est l'efficacité du travail en binôme avec revue croisée du code, qui augmente considérablement la qualité du code et donc réduit les risques d'erreurs ultérieures.

L'apprentissage technique a été lui aussi non-négligeable. J'ai pu parfaire mes connaissances dans le domaine de la vidéo numérique, en implémentant des standards à la pointe de la technologie. Les logiciels que j'ai développés sont sur certaines plate-formes sans équivalent commercial, et lorsque ces équivalents existent, leurs performances sont toujours inférieures.

10.2 Perspectives

Il reste encore certains logiciels propriétaires dans la solution TVi ; pour éviter les problèmes liés à ces logiciels (coût, risque d'obsolescence, impossibilité de changer de plate-forme hôte, disparition du fournisseur, ...) il peut être intéressant de trouver des logiciels de remplacement, ou si nécessaire d'en développer.

De plus, les tests de diffusion vidéo sur PC de poche et réseau sans fil laissent entrevoir des possibilités directes d'adaptation à TVi. On peut imaginer par exemple l'économie de connectique que représenterait une solution TVi sur réseau sans fil. Un autre axe de recherche pourrait être l'adaptation de l'interface complète de TVi à des écrans de la taille de ceux des PC de poche, permettant de s'afranchir des écrans tactiles encombrants.

Cinquième partie

Annexes

Chapitre 11

Bibliographie

Bibliographie

[1] Kent Beck. *Extreme Programming Explained : Embrace Change*. 1999.

[2] MPEG-2 commitee. *Generic Coding of Moving Pictures and Associated Audio (ISO/IEC 13818-1)*. Norme MPEG-2.

[3] Microsoft Corporation. *Microsoft Developer Network Library*. http ://msdn.microsoft.com/library/.

[4] Martin Fowler. *Refactoring : Improving the Design of Existing Code*. 1999.

[5] Martin Fowler. *The New Methods*, 2000. http ://www.martinfowler.com/articles/newMethodology.html.

[6] Don Wells. *Extreme Programming : A gentle introduction*, 1999. http ://www.extremeprogramming.org/.

Chapitre 12

Présentation de la SNCF

12.1 Présentation de la SNCF

12.1.1 Historique

En 1914, la longueur du réseau national ferroviaire atteint 39400 km. Il est exploité par des compagnies privées. La prolifération des lignes dont beaucoup ne sont pas rentables met de nombreuses compagnies dans une situation financière difficile, nécessitant une intervention de l'État dans la gestion des chemins de fer.

Les réseaux étant pour la plupart déficitaires, les chemins de fer sont alors nationalisés en 1937 et donnent naissance à la Société Nationale des Chemins de fer Français.

La convention du 31 août 1937 permet entre autres de fusionner les divers réseaux de chemins de fer en un réseau unique, de placer celui-ci sous la responsabilité de l'État et de gérer le réseau de façon industrielle et non administrative afin d'équilibrer le budget économique. La réunion des capitaux privés et publics fait de la SNCF une société d'économie mixte.

Mais l'entreprise doit affronter dès 1937, et plus encore à partir de 1946, la concurrence croissante des transports routiers, de l'aviation et de l'oléoduc. Afin de la mettre sur un pied d'égalité avec les autres transporteurs, l'État apporte de profondes modifications au régime financier de l'entreprise. La SNCF devient pleinement responsable de son équilibre budgétaire.

Libre de sa gestion, la SNCF se lance dans une politique commerciale active. La qualité du service se renforce par la succession de performances techniques qui placent l'entreprise à l'avant-garde du progrès ferroviaire dans le monde.

En 1955, deux locomotives électriques battent le record du monde de

vitesse sur rail en roulant à la vitesse de 331 km/h.

En 1974, la construction de la ligne nouvelle Paris-Lyon à grande vitesse est approuvée par le gouvernement. 1981, le TGV Sud-Est établit le nouveau record du monde à 380 km/h. Puis c'est au tour du TGV Atlantique qui atteint la vitesse de 515.3 km/h en mai 1990.

Depuis mai 1995, le TGV Eurostar reliant Paris, Bruxelles et Londres, est accessible aux voyageurs. Ce service est offert par un Groupement Européen d'Intérêt Économique regroupant la SNCF, la SNCB (Société Nationale des Chemins de fer Belges) et la BR (chemins de fer britanniques). Dès juin 1995, d'autres partenariats se sont joints à ce groupement : l'Allemagne et la Hollande, pour mettre en service le Thalys (Paris-Bruxelles-Köln-Amsterdam).

À l'échelle des moyennes distances, le rail concurrence désormais sérieusement l'avion et l'automobile.

12.1.2 La SNCF

Introduction

La SNCF est chargée du transport ferroviaire de voyageurs et de fret sur le réseau national ainsi que de l'exploitation et de la gestion de l'infrastructure (voies, signaux). Elle répond aux besoins des clients en proposant des produits et services de qualité à des prix compétitifs. Elle assure sa mission dans les meilleures conditions de confort et de ponctualité et réalise des performances techniques remarquables.

En 1996, le gouvernement a procédé à une réforme du système ferroviaire, d'où la création d'un établissement public « Réseaux Ferrés de France » (RFF), chargé d'assurer, pour le compte de l'État, la responsabilité de propriétaire du réseau des infrastructures ferroviaires. La SNCF est devenue une société exploitante utilisant un réseau appartenant à RFF.

Afin d'assurer son développement et la reconquête de ses clients, la SNCF s'est engagée dans un projet industriel, qui repose sur trois thèmes : le client, l'entreprise, les hommes. Il prévoit un certain nombre d'actions à mener et propose à chaque cheminot de participer à leur mise en œuvre. Elles garantissent le développement de l'activité ferroviaire et un redressement financier durable de la SNCF. « Pour être, en 2002, l'entreprise de service public de référence en France et en Europe ».

Statut économique

Le 1er janvier 1993, conformément à la loi d'Orientation des Transports Intérieurs (LOTI), la SNCF est devenue un Établissement Public Industriel et

Commercial (EPIC). Son cahier des charges lui confie la mission d'exploiter, d'aménager et de développer, selon les principes du service public, le réseau ferré national en même temps qu'il définit certaines obligations :
- entreprise publique devant tenir compte de l'intérêt général ;
- entreprise en situation de concurrence pour toutes ses activités ;
- service public et compétitivité (réalisation de nouveaux projets, adaptation de son organisation) ;
- équilibre financier (l'équilibre des comptes est une exigence fondamentale) ;
- concours à l'exportation (présence sur les marchés d'ingénierie et d'études à l'étranger grâce à sa filiale SOFRERAIL).

Stratégies

Elles s'ordonnent autour de trois objectifs essentiels :
- développer les liaisons rapides de voyageurs ;
- améliorer les transports de la vie quotidienne ;
- consolider la situation du fret (transport de marchandises).

Leur mise en œuvre passe par l'écoute de la clientèle car le développement du trafic suppose une adéquation permanente aux besoins de transport. La SNCF doit également diffuser une image forte par l'offre d'une grande variété de produits et posséder une gestion dynamique de son domaine d'activité.

12.1.3 La direction de la recherche

Cette direction est apparue avec le projet C03 qui a donné naissance au TGV, il y a tout juste 30 ans. Elle a aujourd'hui pour mission de piloter la Recherche et le Développement de l'entreprise et de conduire les programmes de recherche dans le cadre d'un fonctionnement en réseau.

Elle est organisée en trois domaines de responsabilités définis à partir des trois principaux axes d'activité de la Direction :
- l'animation des pôles de compétences du réseau de la Recherche ;
- la production de la recherche ;
- l'élaboration du Programme de recherche, prospective et développement.

12.1.4 L'unité NTIC

Les activités de l'unité de recherche Nouvelles Technologies de l'Information et de la Communication (NTIC) sont par nature pluridisciplinaires. Les techniques utilisées et mises en œuvre relèvent à la fois de l'informatique, des télécommunications, du traitement du signal, de l'intelligence artificielle, de la communication interactive et des facteurs humains.

Outre les aspects veille technologique et fonctionnelle, qui permettent de surveiller les nouvelles technologies et d'imaginer de nouveaux concepts, les travaux de recherche et de développement s'appuient sur la réalisation d'études et de projets qui prennent en compte les besoins des utilisateurs et des clients. Ces travaux sont une première étape pour assurer la maturation qu'impose l'innovation, et permettre le passage vers l'opérationnel.

Les développements en cours concernent principalement les sujets suivants :

- multimédia ;
- traitement automatique de la parole ;
- systèmes de dialogue ;
- réalité virtuelle ;
- Internet ;
- télécommunications ;
- commerce électronique.

Chapitre 13

L'*eXtreme Programming*

Ce chapitre a pour but de présenter l'*eXtreme Programming* (ou XP), en commençant par les raisons qui ont poussé à son apparition ainsi qu'à d'autres méthodes de développement dites « légères », puis en décrivant les grands principes de la méthode, et enfin en montrant la façon dont j'ai pu adapter cette méthode au projet.

13.1 L'approche classique : le « modèle en V »

Cette méthode a été inventée pour faire face à une crise du logiciel dans les années 70, et pour pallier les défauts suivants :

- des coûts et temps de développement quasi impossibles à prévoir ;
- une qualité du produit livré déficiente ;
- une très faible réutilisation des produits, et une maintenance difficile, onéreuse et souvent source de nouvelles erreurs.

Le modèle en V est une méthode articulée en différentes phases, enchaînées linéairement, mais avec des retours en arrière possibles.

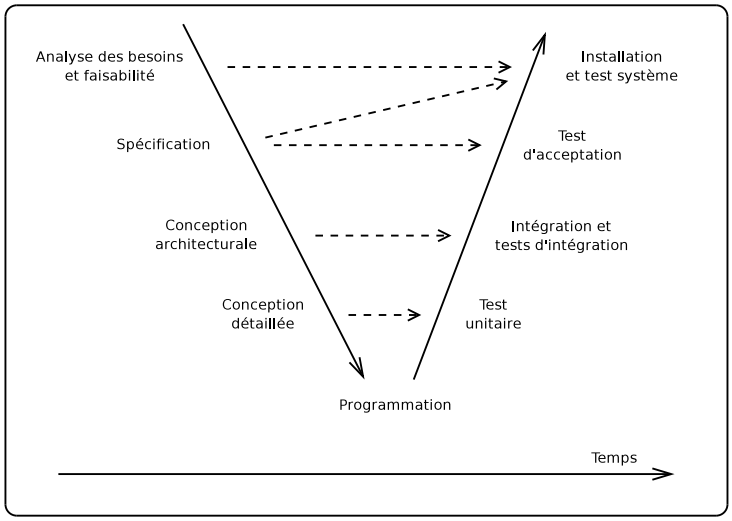

FIG. 13.1 – Le modèle en V

Si au cours d'une des phases du développement, on découvre des erreurs ou des manques dans une phase antérieure, cette dernière est invalidée et le processus de développement reprend à partir de cette phase révisée. Il convient donc d'accorder la plus grande importance aux premières phases.

Cette approche classique du développement logiciel souffre malheureusement de défauts chroniques :

- les décisions les plus importantes sont prises au début du projet, lorsqu'on en sait le moins sur le sujet : il faut souvent revenir sur ces décisions par la suite et ces retours sont d'autant plus coûteux qu'ils sont inattendus ;
- le client perd la visibilité sur le projet pendant la phase de réalisation (effet tunnel) :
  - des erreurs ou des manques graves peuvent être découverts au moment de la livraison ;
  - l'effort de développement n'est pas toujours adapté à l'importance de chaque fonctionnalité ;
  - le fonctionnel peut avoir évolué pendant le développement, rendant le logiciel obsolète dès sa sortie ;
- le processus en lui-même comporte des activités (rédaction de documents, etc.) qui ne participent pas directement à la valeur ajoutée du produit ;

– tout le développement converge vers un produit bien défini, souvent difficile à faire évoluer par la suite.

Cette approche classique fonctionne quand la problématique est bien connue et que l'environnement est prévisible, mais il faut une autre méthode pour les projets innovants. Malheureusement, étant à la fois seul développeur et seul bénéficiaire de certaines parties du projet, mettre en place une équipe de développement ne m'est pas possible dans ces cas précis. La connaissance des limites de l'approche traditionnelle est cependant une donnée précieuse pour l'évaluation des risques, l'estimation des temps de développement, et donc l'élaboration du planning.

13.2 L'*eXtreme Programming*

13.2.1 Présentation de l'*eXtreme Programming*

L'*eXtreme Programming* fait partie de ces nouvelles méthodes de développement qualifiées de méthodologies « agiles », ou légères. Elles sont couramment considérées comme un remède à la bureaucratie et un encouragement au bricolage, mais soulèvent un intéret certain de la part de l'ensemble du paysage informatique car pour beaucoup justement, l'attrait de ces nouvelles méthodes est d'être un habile compromis entre aucun processus formel et une surcharge de processus.

Cependant il ne faut pas voir ces nouvelles méthodes comme une simple réduction de paperasse ; la principale différence avec les méthodes plus anciennes est qu'elles sont adaptatives plutôt que prédictives. Selon Martin Fowler (ThoughtWorks), les méthodes lourdes tendent à planifier de grosses parts du processus logiciel en détail pendant une durée importante, ce qui fonctionne correctement tant qu'il n'y a pas de changements. Donc leur nature est de résister au changement. Les méthodes agiles, par contre, s'accommodent favorablement du changement. Elles s'efforcent d'être des processus qui s'adaptent et s'épanouissent au changement, au point de se changer elles-mêmes. Les méthodes agiles sont orienté vers l'humain, plutôt que vers le processus. Elles s'efforcent explicitement de travailler avec la nature de l'homme plutôt que contre elle, et de souligner que le développement informatique devrait être une activité agréable.

Origine

Au début des années 90, Kent Beck réfléchissait à de meilleurs moyens de développement logiciels. Il avait travaillé quelque temps avec Ward Cunningham, et ensemble ils avaient expérimenté une approche du développement logiciel qui rendait tout très simple et plus efficace. Kent s'est demandé ce qui rendait la réalisation d'un logiciel facile, et ce qui la rendait difficile ; en

Mars 1996 Kent commença un projet chez DaimlerChrysler en utilisant ces nouveaux concepts du développement logiciel ; le résultat fut la méthodologie *eXtreme Programming* (XP).

Kent réalisa qu'il y avait quatre dimensions dans lesquelles n'importe quel projet de développement logiciel pouvait être amélioré. Il faut améliorer la communication. Il faut chercher la simplicité. Il faut du retour d'information (*feedback*) sur la qualité et l'avancement du projet. Et il faut être sans cesse audacieux.

Aperçu de la méthode

Communication, simplicité, *feedback* et audace sont les quatre valeurs de l'XP. Cette méthode fonctionne en conditionnant l'équipe autour de méthodes simples, avec suffisamment de retour d'information pour leur permettre de voir où ils en sont, et d'ajuster leurs méthodes à leur situation unique.

En XP, chaque contributeur du projet est une partie intégrante de « l'équipe entière » ; cette équipe se forme autour d'une entité qu'on appelle *Client*, qui peut être une ou plusieurs personnes, représente le commanditaire et fait partie de l'équipe et travaille avec eux en permanence.

Les équipes d'XP utilisent une forme simple de planning et de suivi pour décider de ce qui doit être fait à un instant donné, et pour prévoir la date de finition du projet. Focalisée sur la valeur du produit, l'équipe produit le logiciel en passant par une série de *releases* intermédiaires opérationnelles qui satisfont à tous les tests que le *Client* aura définis.

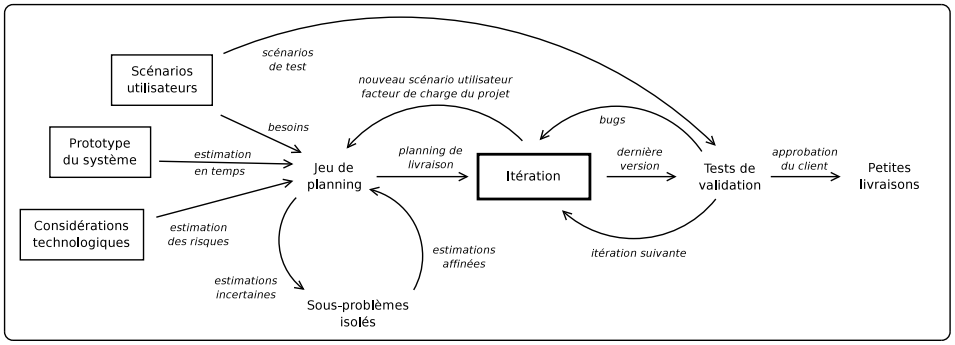

FIG. 13.2 – La méthode XP

L'XP utilise un mode de développement itératif, et accorde un temps minimal aux estimations préliminaires. L'idée est ici de sans cesse réajuster ces estimations au cours du développement, à chaque itération.

Les développeurs pratiquant l'XP fonctionnent à la fois en binômes et en tant qu'équipe complète. Ils font du design simple et testent leur code de manière extrêmement intensive, améliorant constamment le design en le remettant continuellement au niveau des besoins courants. Cela implique que le système doit toujours rester consistant et en état de production, et que tous les programmeurs adoptent les mêmes conventions de code.

Qui utilise cette méthode ?

Les principaux acteurs du mouvement sont naturellement Kent Beck, mais aussi Robert C. Martin, expert UML/C++ à la tête de la société de conseil Object Mentor.

Malgré la jeunesse de cette discipline, l'XP a déjà été expérimentée avec succès chez de grands noms tels que Bayerische Landesbank, Credit Swiss Life, First Union National Bank, DaimlerChrysler, Ford Motor Company, CSEE Transport.

13.2.2 Mise en œuvre

Les prérequis

L'*eXtreme Programming* est adapté aux petites équipes, celles-ci doivent donc être de petite taille (jusqu'à 10 personnes). De plus, les développeurs doivent avoir une compétence suffisante pour construire des applications réellement évolutives, et l'entente entre les membres de l'équipe doit permettre une réelle communication.

Il faut de plus rester conscient qu'XP introduit un changement fondamental dans l'approche du développement : la culture de l'entreprise doit permettre ce changement, et chacun doit se former à ces pratiques.

Les limites

Les conditions du projet ne permettent malheureusement pas de mettre en œuvre toutes les méthodes d'XP développées à ce jour, et ce rien qu'en raison du nombre limité de participants au projet ; il a donc fallu trouver des méthodes annexes pour pallier cela.

Le meilleur moyen d'adopter les méthodes d'XP est de commencer un nouveau projet, alors que les développements nécessaires au sujet de stage sont articulés autour de projets existants. Heureusement, les développeurs et utilisateurs de VideoLAN sont eux-mêmes rompus à des méthodes similaires et leur application sera complètement transparente.

Application à la partie VideoLAN du projet

En ce qui concerne les prérequis cités précédemment, je n'ai aucune inquiétude quant aux compétences techniques des développeurs qui seront amenés à travailler sur VideoLAN, connaissant leurs réalisations passées et ayant eu des conversations techniques personnelles avec eux. J'ai opté pour le cycle standard XP :

1. Le client écrit ses besoins sous forme de scénarios.
2. Les développeurs évaluent le coût de chaque scénario.
3. Le client choisit les scénarios à intégrer à la prochaine livraison.
4. Chaque développeur prend la responsabilité d'une tâche.
5. Le développeur choisit un partenaire.
6. Le binôme écrit les tests unitaires correspondant au scénario à implémenter.
7. Le binôme prépare l'implémentation en réorganisant le code existant, puis il procède à l'implémentation proprement dite.
8. Le binôme cesse l'implémentation lorsque les tests unitaires passent.
9. Le binôme intègre ses développements à la version d'intégration, en s'assurant que tous les tests de non-régression passent.

Étant seul développeur disponible, je peux adapter chaque point du cycle à mon projet précis, en prenant compte des limites du projet et de ses particularités.

Pour une application complète au projet, il conviendra de développer en plus des tests de régression VideoLAN une batterie de tests de validation et de régression spécifiques à TVi.

13.2.3 Avantages attendus

Les avantages usuellement attendus de cette méthode, et qui ont pu être vérifiés sur d'autres projets, sont les suivants :

- chaque développement est moins complexe ;
- les intégrations sont progressives ;
- la livraison et la mise en oeuvre peuvent être effectuées et testées après chaque incrément ;
- l'effort est constant dans le temps (à l'exception d'un pic dans les spécifications détaillées).

13.3 L'*eXtreme Programming* au service de l'approche classique

Les valeurs de Kent Beck (communication, simplicité, *feedback* et audace) sont certes à l'origine de l'*eXtreme Programming* qui s'oppose au premier abord à l'approche classique, mais des idées importantes peuvent en être retirées pour être adaptées à cette dernière. En particulier, l'idée de simplicité est ici absolument cruciale.

13.3.1 La partie TVi

La partie TVi du projet se prête d'autant moins à l'XP qu'elle est issue d'un projet déjà réalisé. Des principes simples de l'XP tels que l'adoption des mêmes conventions de code ou la validation par tests de régression ne peuvent donc pas s'y adapter facilement. De plus, l'effectif alloué à cette partie n'est que d'une seule personne (moi-même) et je ne peux compter sur les intérets communs d'autres développeurs pour espérer fonctionner en binôme.

Cependant, une idée majeure, la simplicité, va pouvoir être mise en avant. En première estimation, une factorisation du code de TVi permettra à première vue d'en diminuer la longueur à peu près de moitié. Un tel gain est énorme, n'est pas explicitement justifié par le cahier des charges, mais facilitera grandement la maintenance ultérieure.

Sixième partie

Annexes techniques

Chapitre 14

Configuration des machines

Le document n'a pas été inclus à la compilation.

Chapitre 15

Arborescence de TVi

Le document n'a pas été inclus à la compilation.

Chapitre 16

Outils de gestion de problèmes

16.1 Le besoin

Un outil de gestion de problèmes (*bug tracking system*) est primordial pour un projet logiciel à partir d'une certaine taille (que ce soit en termes de lignes de code ou de masse de développeurs). En effet pour de petits projets, il est possible de gérer les problèmes par une simple file d'attente, ou un tableau récapitulatif régulièrement mis à jour. Pour des projets complexes où les difficultés techniques sont très diversifiées, et où les retours d'information proviennent de sources très variées, un outil de gestion devient indispensable.

Le projet VideoLAN s'était déjà doté d'un tel système vers 2000, développé en interne, mais celui-ci était très inadapté aux réels besoins et surtout très peu convivial et rebutait les utilisateurs. Il a vite été abandonné au profit d'un simple fichier modifiable par tous les développeurs, ce qui représentait une solution acceptable tant que les problèmes n'étaient pas trop nombreux.

Avec le succès de VideoLAN, et le nombre de développeurs extérieurs (dont moi-même durant mon stage), la question d'un vrai gestionnaire de problèmes a été nécessaire. Une base à ma réflexion a été le document *Call Center, Bug Tracking and Project Management Tools for Linux* (http://linas.org/linux/pm.html).

16.2 Outils de gestion de problème

16.2.1 SourceForge

SourceForge (http://www.sourceforge.net/) est une plate-forme complète de développement coopératif incluant un système de *bug tracking*.

Ce système est malheureusement trop lourd à mettre en place, il faut installer la moitié de SourceForge, qui d'ailleurs comprend des modules pro-

priétaires. Pour information la personne s'occupant des packages de SourceForge au sein du projet Debian travaille dessus depuis plus d'un an et n'a toujours pas de version fonctionnelle.

Par ailleurs, selon les développeurs de Bugzilla, le système n'est pas très bien pensé au niveau de la sémantique de gestion ; l'impossibilité de fusionner des bugs par exemple est rhédibitoire.

16.2.2 Bugzilla

Site web : http://www.bugzilla.org/

Bugzilla est le *bug tracker* du projet Mozilla, qui se rapproche énormément de VideoLAN par certaines de ses caractéristiques intrinsèques : projet multi-plateforme, destiné au grand public, avec une masse de développeurs très disséminée sur la planète, et développement coopératif séparé en branches.

- Perl, backend MySQL ;
- tout l'interfaçage se fait par le Web ;
- un compte est nécessaire ; comptes publics possibles ;
- extrêmement complexe, mais très complet ;
- gestion de priorité, sévérité, de *target* aussi (par exemple des considérations telles que «à régler avant la version 2.0» sont possibles) ;
- tri poussé des bugs (bugs des X dernières heures, bugs comprenant des mots-clés donnés, bugs de tel responsable, etc.) ;

En résumé Bugzilla semble vraiment très adapté au logiciel ; il semble faire au moins la même chose que GNATS (voir plus loin), il conviendra de décider si ce qu'il apporte par rapport à GNATS est vraiment utile et n'apporte pas une complexité dont on pourrait se passer.

Il existe un fork RedHat avec backend PostgreSQL : http://bugzilla.redhat.com/bugzilla/

Exemples :

- Mozilla : http://bugzilla.mozilla.org/
- Landfill sur bugzilla : http://landfill.bugzilla.org/bugzilla-tip/

16.2.3 GNATS

Site web : http://www.alumni.caltech.edu/~dank/gnats.html

Fonctionnalités retenues :

- rapport de bug et consultation des bugs par le web ;
- codé en Perl ;
- très sobre, mais intuitif et facile à comprendre ;

– fonctions intéressantes : multi-modules, logins pour développeurs, bugs assignés à un responsable.

Exemples :
– GNU : http://www-gnats.gnu.org:8080/cgi-bin/wwwgnats.pl
– FreeBSD : http://www.freebsd.org/support.html
– Apache : http://www.apache.org/bug_report.html

16.2.4 Debian BTS

Site web : http://www.benham.net/debbugs/

– rapport de bug par mail, consultation par mail ou par web ;
– codé Perl, backend filesystem ;
– pas de login ou d'authentification nécessaires, tout est public mais journalisé, avec possibilité d'annuler une suite d'opérations.

Exemples :
– Debian : http ://bugs.debian.org/

16.2.5 Jitterbug

Site web : http://samba.anu.edu.au/cgi-bin/jitterbug

– administration par le web, mais rapport et notification par mail ;
– CGI en C, backend filesystem ;
– authentification, création de FAQ à la volée.

Exemples :
– Samba : http ://bugs.samba.org/
– Willows : http ://www.willows.com/cgi-bin/twintrack

16.2.6 Bugin

Site web : http://sourceforge.net/projects/bugin/

– plutôt gestionnaire de tickets que *bug tracker* ;
– PHP, backend SQL (MySQL et PostgreSQL) ;
– pas très joli mais couleurs vives et métaphores simples.

16.2.7 RT

Site web : http://www.fsck.com/projects/rt/
– plutôt gestionnaire de tickets ;
– programmé en Perl, divers backends SQL ;
– très clair, pas de fioritures inutiles.

16.2.8 ReqNG

Site web : http://reqng.sycore.net/reqng/
- email-based ;
- programmé en Perl, backend filesystem ;
- interface web très peu attractive et au design désuet.

16.3 Le choix final

Après divers essais et délibérations entre développeurs, le choix s'est porté sur *Bugzilla*, qui a d'abord été testé en interne et n'a à ce jour pas encore été mis à disposition des utilisateurs finaux. Nous n'avons donc pas d'indicateur du gain en organisation de la part de ceux-ci, mais de l'avis des développeurs un tel système est beaucoup plus pratique que la méthode utilisée auparavant.

Chapitre 17

Outils de documentation

17.1 Critères de choix

Un problème lié à la rédaction de la documentation d'un logiciel est que sa phase est souvent postérieure à celle de rédaction du code, et il est parfois nécessaire de revenir en arrière et de procéder à une nouvelle lecture du code. De plus, une modification du code nécessite souvent une modification conjointe de la documentation ; si celle-ci se trouve à un endroit très différent, une perte de temps est induite.

Ayant besoin de rédiger de la documentation pour les modifications que j'allais apporter à VLC, je me suis enquis d'outils de documentation pouvant régler les problèmes sus-cités. Il existe à ce jour des outils de documentation automatique de code ; j'ai procédé à une étude de leurs fonctionnalités, en me basant sur les critères suivants :

- l'outil devra accepter les langages C et C++ (*Objective C* serait souhaitable) ;
- l'outil devra exporter de la documentation dans un langage permettant la publication sur un site web (comme HTML) mais aussi dans un langage de mise en page (comme LaTeX).
- l'outil devra permettre l'écriture de documentation dans le code lui-même (documentation *inline*) ;
- il serait souhaitable que l'outil supporte des références croisées entre parties du code ;
- il serait souhaitable que l'outil supporte des sections supplémentaires qui soient, elles, séparées du code ;
- il serait souhaitable que l'outil permette la création de pages de `man` Unix (outil de documentation standard).

17.2 Outils étudiés

Les outils que j'ai pu tester sont les suivants (je cite aussi les outils que j'ai pu éliminer d'office ainsi que la raison de cette élimination, afin d'éviter une éventuelle duplication de travail lors de la reprise de cette recherche) :

- **adoc** : sort uniquement du LaTeX, mais d'une qualité exemplaire ; génère un index alphabétique des fonctions.
- **AutoDOC** (http://www.oche.de/~akupries/soft/autodoc/index.htm) : ne supporte que le code au format Tcl.
- **Autoduck** : ce projet ne semble plus actif.
- **Cocoon** (http://www.stratasys.com/software/cocoon/) : permet l'insertion de documentation directement dans le code, en commentaires, mais ne génère que du HTML (soit en pages pleines, soit en cadres).
- **ccdoc** (http://www.joelinoff.com/ccdoc/ : ne supporte que les langages C++ et Java.
- **CXRef** (http://www.gedanken.demon.co.uk/cxref/) : génère de la documentation aux formats LaTeX, HTML, RTF, SGML ; ne supporte que le langage C et les auteurs ne prévoient pas de supporter le C++.
- **cxxwrap** (http://www.deaven.net/~deaven/Software/cxxwrap/) : génère du HTML (en cadres), et ne supporte que le C++.
- **doc++** : génère du HTML (en pages pleines), crée une table des matières alphabétique intéressante, mais est d'aspect plutôt disgracieux.
- **doocgen** (http://www.doocgen.org/) : il s'agit d'un *fork* de **doxygen** qui ne semble plus développé.
- **doxygen** (http://www.stack.nl/~dimitri/doxygen/) : il s'agit de la référence des outils de documentation de code. Il supporte le C et le C++ (mais pas Objective C), et permet de générer des pages au format HTML et LaTeX. De plus, toute la documentation se fait en *inline*.
- **gtk-doc** : site web injoignable au moment de l'étude.
- **happydoc** : site web injoignable au moment de l'étude.
- **headerdoc** (http://developer.apple.com/darwin/projects/headerdoc/) : c'est probablement le seul outil supportant complètement le langage Objective C (très utilisé par Apple), mais il supporte aussi C et C++. Malheureusement les informations des pages HTML générées sont beau-

coup trop disséminées (il n'est pas rare d'obtenir des écrans avec moins d'une dizaine de mots), et la documentation *inline* nécessite beaucoup de mots-clés complexes que par exemple **doxygen** ne requiert pas.

- **kdoc** (http://www.ph.unimelb.edu.au/~ssk/kde/kdoc/) : ne supporte que le C++.
- **object outline** (http://www.bbeesoft.com/) : ne supporte que le C++.
- **robodoc** (http://www.xs4all.nl/~rfsber/Robo/robodoc.html) : le développement semble arrêté.
- **scandoc** (http://scandoc.sourceforge.net/) : la sortie de ce logiciel est très propre et agréable à lire, malheureusement il ne supporte que le C++.
- **sdoc** (http://www.cit.uws.edu.au/~blilburn/sdoc/) : outil très générique, adapté à l'écriture manuelle de documentation, mais bien qu'il supporte tous les langages il ne permet pas la documentation *inline* ; de plus il n'a pas de support pour les références croisées et les mises en page complexes.
- **synopsis** (http://synopsis.sourceforge.net/) : gère tous les langages de programmation communs, dont C et C++, mais la sortie HTML (très sobre et agréable par ailleurs) est obligatoirement en cadres.

Mon choix s'est finalement porté sur **doxygen**, car en plus de ses qualités techniques j'ai été rassuré par les centaines de projets qui lui avaient fait confiance pour la gestion de leur documentation.

Chapitre 18

Documentation utilisateur du plugin Mozilla

Le document n'a pas été inclus à la compilation.

Chapitre 19

Documentation développeur de VLC

Le document n'a pas été inclus à la compilation.