Versionnage : garder facilement trace des versions successives d'un document

Exemples avec un outil de contrôle de versions (CVS)

Stéphane Bortzmeyer — Olivier Perret

Institut Pasteur
Service d'Informatique Scientifique
25-28 rue du docteur Roux
75724 PARIS CEDEX 15

bortzmeyer@netaktiv.com
perret@pasteur.fr

RÉSUMÉ. *Un document numérique, plus malléable qu'un document papier, connait plusieurs versions au cours de son existence. Si, dans certaines applications, seule la dernière version compte, il est souvent intéressant d'accéder aux versions précédentes. Cet accès doit être simple et permettre de répondre à des questions comme « Qu'est-ce qui a changé en janvier 1999 ? », « quelle est la somme des changements depuis mars 2000 ? » Après avoir rappelé les concepts de la gestion de configuration auxquels répond l'outil CVS, ce document étudie les avantages et les limites d'un tel outil dont le but est a priori plus large que l'archivage de documents, mais dont la conception a été très influencée par les informaticiens. On y trouvera des exemples d'utilisations très puissantes inspirées des phases classiques du développement informatique, avec quelques limites plus ou moins prévues par les auteurs, certaines renvoyant au problème plus ouvert de la définition de ce qu'est une « modification significative » dans un document.*

ABSTRACT. *A digital paper is easier to reuse than a printed text. It often has many versions, and the last is not always the only one to be used. Sometimes the previous versions are interesting and should stay available to the writing team. One could ask for « How was the the january 1999 version ? », « What are the changes since march 2000 ? ». Considering the general goals of configuration management, we suggest to use CVS to manage an archive. Many examples are given by the programmers.*

MOTS-CLÉS : *CVS, archivage de documents, gestion de configuration, versionnage*

KEYWORDS: *CVS, configuration management, version control*

1. Le besoin du versionnage

1.1. *L'exception informatique*

Le concept de versionnage est en très ancien en milieu industriel. Sans attendre l'apparition de l'informatique, il suffisait d'éplucher le catalogue d'un constructeur automobile, et en particulier la rubrique « pièce détachées » pour en deviner l'interêt.

Des méthodes normalisées de gestion de projet, comme CALS [1] mettent en pratique ce concept, gage d'évolution et de réutilisabilité des développements, dans le cadre plus large de la qualité industrielle.

Dans ce monde normalisé idéal parfois un peu déconnecté des réalités, l'industrie informatique fait figure d'exception qui enrichit la règle du fait de plusieurs caractéristiques propres :

– l'ouverture : depuis l'avènement des systèmes ouverts, tous les grands développements sont généralement prévus pour que leurs produits (logiciels et données) soient réutilisables sur des plate-formes différentes par d'autres utilisateurs[2] ;

– tout est volatile : une virgule de trop peut transformer un produit opérationnel en un vulgaire programme buggué et inutilisable ;

– tout est virtuel : les développements et leur documentation ont la même nature physique ; numérique devrait-on dire.

1.2. *La standardisation des formats*

Un préalable au développement d'outils de gestion de configuration a été l'adoption à tous les niveaux de l'informatique, que ce soit les réseaux, leurs couches basses ou leurs applicatifs, de formats de représentation des données standards et lisibles par tous.

– au niveau des textes, qui est le contexte d'utilisation de CVS, le plus célèbre est l'ASCII, fortement limité mais adopté de fait par tous les programmeurs, car assez riche pour décrire tous les langages informatiques conçus à ce jour ;

– pour des textes écrits dans des langues européenes, les utilisateurs se contentent souvent des différents déclinaisons du standard iso-8859, très utilisés pour le courrier électronique ;

– le standard le plus complet, au prix d'un allongement de la taille des caractères, et qui sert maintenant de référence dans le domaine de la description de documents est Unicode[3].

---

1. Continuous Acquisition and Life-cycle Support, soutenue par la marine américaine.
2. A quelques exceptions près comme les suites logicielles de Microsoft, dont l'usage est déconseillé à ceux qui souhaitent apporter un minimum de pérennité à leurs documents
3. C'est entre autre le format par défaut de la norme XML, désigné alors par le sigle UTF-8

Une fois adopté le standard sur la représentation des chaînes de caractères, le besoin suivant a été une information structurée, répondant à plusieurs nécessités de la documentation :

– décharger l'auteur de la documentation des préoccupations de mise en page tout en lui permettant de l'éditer lui-même et d'éviter ainsi des erreurs de recopie. Le langage le plus représentatif de ce concept est sans doute TEX, qui a beaucoup de succès dans la communauté scientifique et qui a été utilisé pour écrire ce document par l'intermédiaire de sa surcouche LATEX.

– faciliter la recherche automatique sur le contenu, activité dans laquelle le langage qui a eu le plus de succès est certainement HTML, grâce au World Wide Web ;

– d'une manière plus générale, séparer l'information et les règles de présentation, considérant que la présentation d'un document est d'autant plus difficile à soigner qu'elle est mélangée avec son contenu [4]. Le langage créé pour répondre spécifiquement à ce besoin s'appelle SGML (Standard Generalized Markup Language), et est caractérisé comme son nom l'indique par l'usage intensif de « balises ».

Le format qui fait l'unanimité pour unifier ces trois points existe ; il a pour nom XML(eXtensible Markup Language[BRA 00]), qui fait la synthèse des qualités de SGML et HTML. Coïncidence amusante : l'expérience montre qu'il est généralement très simple de faire des passerelles entre XML et le troisième format présenté auparavant, LATEX.

On notera l'absence dans cette liste de tous les formats dit « propriétaires », parce que liés à une logiciel dont les caractéristiques sont tenues secrètes par son éditeur, lequel, même quand il en publie les spécifications fonctionnelles se réserve toujours le droit de les modifier ou d'interrompre le support sans préavis. [5].

1.3. *Définition de la différence*

Une fois convenus le codage des caractères et le format de représentation des textes, on peut commencer à envisager d'effectuer des comparaisons entre les textes numérisés, autrement dit des traitements informatisés.

Même en appliquant les précautions précédentes, il reste un problème de fond que l'on peut désigner par le terme : « caractérisation de la différence », qui consiste à définir ce qu'on reconnait comme étant une modification notable d'un document, autrement dit répondre à la question : « *A partir de quelle quantité d'information modifiée numériquement admet-on qu'on a effectivement une nouvelle version d'un document ?* »

---

4. problème rencontré par tous les utilisateurs de logiciels dits « WISIWYG », qui en facilitant à première vue le travail des utilisateurs dans la phase de saisie, complique gravement tous leurs traitements ultérieurs

5. Quand on lit attentivement les licences, on constate que l'autorisation même d'utiliser ces logiciel reste le privilège de l'éditeur, au nom du droit d'auteur.

Exemples de cas litigieux :

– espace significatifs ou non selon le langage de programmation, indentation automatique des programmes ;

– indentation d'un texte sans modification de son contenu ;

– suite de caractères différents aboutissant au même résultat écrit.

Ainsi, en prenant comme exemples des documents XML [6], deux documents peuvent être sémantiquement équivalents alors que leur représentation en texte diffère. Par exemple :

```
<article>
   <p>Bref, mais précis.</p>
</article>
```

a exactement la même sémantique[7] que :

```
<article>

<p>Bref, mais précis.</p>

</article>
```

alors que les deux fichiers sont distincts. Un outil qui ne travaille qu'au niveau du texte, comme CVS, verra un changement du document, alors que l'auteur avait juste changé la présentation du fichier source. Le rendu du document, effectué par des systèmes de feuille de style comme XSLT, ne sera pas affecté.

Il n'y a pas de solution générale (indépendante du langage) au problème. Pour le cas spécifique de XML, on peut tout au plus réécrire les programmes de comparaison de documents en tenant compte de la définition de XML. Celle-ci précise les règles selon lesquelles espaces et sauts de ligne doivent être considérés significatifs ou pas *(2.10 White Space Handling)*.

Un autre facteur de problème, qui n'est pas spécifique à XML, vient de l'utilisation du jeu de caractères Unicode. Ce dernier permet au même texte d'avoir plusieurs représentations. S'il est rare qu'une même personne utilise alternativement deux représentations différentes des caractères, il faut parfois penser à *canonicaliser* les textes avant toute comparaison, s'ils proviennent de sources différentes.

---

6. XML est de plus en plus répandu, car il est indépendant de toute entreprise et techniquement très réussi.
7. Les connaisseurs noteront que cela dépend en fait de la DTD de ce document.

1.4. *Les outils de base : SCCS et RCS*

Profitant des considérations précédentes, et une fois fait le choix de la caractérisation de la «*différence*» il est possible de stocker les différentes versions d'un même texte non pas comme autant de fichiers qu'il y a de versions, mais comme un document unique original, suivi de la succession de ses modifications, inventoriées et datées. L'un des premiers exemples en est SCCS *(Source Code Control System)*, outil créé et utilisé par des programmeurs sur des textes de programmes en langage C.

Pour détecter les différences entre deux versions, SCCS fait une comparaison ligne à ligne pour :

– retrouver dans la nouvelle version chaque ligne de la précédente ;

– noter le numéro des lignes disparues ;

– enregistrer les nouvelles lignes dans la nouvelle numérotation.

Il garde une trace de toutes les mises à jour dans un répertoire particulier appelé «*référentiel*».

SCCS était déja fourni avec une batterie d'outils suffisante pour la majorité des traitements évoqués dans cet article, utilisables comme des fonctions, la plus utilisée étant la restitution du texte courant à partir des données du référentiel.

SCCS souffrait cependant d'un grave défaut d'ouverture puisqu'il était pratiquement impossible a un non-initié de reconstituter un texte géré par SCCS, (un comble pour un outil de programmeur Unix). A part quelques grands projets spécifiques à une équipe permanente ou à un constructeur, comme le noyau SunOS, ceux-ci l'ont donc spontanément abandonné pour RCS (Revision Control System)

RCS est une évolution astucieuse de SCCS qui prend pour référence le fichier courant, et enregistre la trace des modifications précédentes, partant du principe plein de bon sens qu'on a plus souvent besoin de la dernière version que de la première. RCS sert depuis de référence à tous les outils de gestion de documents isolés, et fut même pendant longtemps la couche basse de CVS [FOG 00].

L'expérience de SCCS puis de RCS a donc abouti à une standardisation vers des outils qui manipulent des fichiers de données assimilables à du texte, comparés ligne à ligne. Tous les outils gèrent aussi pour chaque fichier un référentiel, généralement assez proche du fichier manipulé effectivement par l'utilisateur, appelé «*copie de travail*». Les copies de travail sont des fichiers tout à fait classiques.

2. CVS et la gestion de configuration

CVS (Concurrent Version System), [CVS 00] quand à lui, renouvelle le genre, en incluant la notion de branche de développement, qui permet un véritable développement en concurrence, chaque branche pouvant à son tour devenir la branche de référence, tout en conservant son versionnage propre. Aux capacités multi-utilisateurs de

RCS, CVS ajoute aussi des possibilités de développement en réseau, qui en font un outil de gestion de configuration, sur laquelle il est utile de faire quelques rappels.

2.1. *La gestion de configuration et ses outils*

Depuis les débuts de l'informatique, la gestion de configuration automatisée est le doux rêve de tous les acteurs de l'industrie informatique, qu'ils soient constructeurs, développeurs ou simple utilisateurs.

D'abords privilège des constructeurs qui fournissaient sur les premiers ordinateurs des configurations rigides déléguables au mieux à un administrateur système, la gestion de configuration est devenue le problème de tous avec l'apparition de machines multi-utilisateurs, des systèmes ouverts, des systèmes micros, et des applications qui vont avec.

Si on peut dire que l'évolution des systèmes est désormais aux mains des développeurs, qui enrichissent les systèmes avec des applications, ceux-ci sont loin d'avoir résolu les nombreux problèmes de gestion de configuration qu'ils posent au reste de la communauté informatique et que l'on peut résumer rapidement en cette liste :

– les problèmes de structure : modélisation, interfaces, relation, consistence...

– les problèmes de construction : génération, snapshot, optimisation, régénération...

– les problème de comptabilité et d'audit : statistiques, rapports, historique, traçabilité...

– les problèmes de sécurité : contrôle d'accès, cloisonnement, suivi des bugs...

– les problèmes de version : version du logiciel, version de la configuration, contexte, référentiel...

Tous ces problèmes sont passionants et certains font déjà l'objet de recherches abondantes dans le monde de la science informatique [MOL 97] ; ceux de la dernière catégorie ont la particularité de s'appliquer aussi à des documents numériques ne correspondant pas forcément à des projets informatiques. C'est ce qui a motivé la présente étude.

En effet, depuis qu'ils ont été mis en relief, ces problèmes de gestion de versions ont suscité le développement d'outils spécifiques. Dans la partie technique de cet article, ce sont les services rendus par ces outils que nous associerons au terme de « versionnage ».

2.2. *CVS et le travail en groupe*

On peut résumer les services de CVS en 4 grands besoins :

– travailler à plusieurs sur les mêmes fichiers au même moment ;

– gérer les versions des fichiers en cours de rédaction ;

– suivre les versions de documents externes ;

– gérer plusieurs branches de développement.

Pour cela CVS utilise une zone de stockage spécifique appelé référentiel, dans lequel sont stockées toutes les données nécessaires au fonctionnement et suffisante pour retrouver toutes les données qui lui ont été confiées.

Il fait aussi un grand usage des ressources du système, comme la notion d'utilisateurs multiples, la structure arborescente des fichiers, et les possibilités client/serveur. C'est pourquoi ces fonctions-là ne seront pas détaillées.

2.3. *Utilisation du référentiel*

Un référentiel CVS est un aggrégat d'objets ayant chacun leur version. L'évolution des objets et de leurs versions se fait par :

– extraction du référentiel (checkout/update) ;

– modification en environnement de travail standard ;

– repose dans le référentiel pour mise à jour (commit).

On dit que CVS utilise un modèle par composition. Pour la gestion de la concurrence, il s'inspire du modèle des transactions longues, qui a la caractéristique de supporter autant de divergences que possible entre les différentes versions, la seule contrainte de cohérence étant exigée au moment de la réunification des versions.

Autrement dit une session de travail avec CVS peut se résumer ainsi :

1. cvs update : Je synchronise ma copie de travail avec le référentiel. CVS me signale les mises à jour.

2. à la fin de ma session, je valide mon travail. CVS me confirme si mes modifications sont compatibles avec les autres travaux en cours.

3. si elles ne le sont pas, j'ai le choix de conserver mon autonomie ; je ne profiterai plus des mises à jour concernant cet objet. Ce choix se répercutera sur les tentatives de mises à jour futures (*warning*), à moins que je ne crée ma propre branche de développement.

4. si je choisis la dissidence, et que je veux en faire profiter les autres, je dois créer une nouvelle branche.

Chaque transaction est enregistrée et peut faire l'objet d'une action (envoi d'un message électronique à l'équipe de développement). Les conséquences les plus importantes sont qu'il est possible à tout moment :

– pour chacun de connaitre l'historique des actions et en particulier des modifications apportées à chaque objet [8]

– pour l'administrateur de savoir qui en est où.

Remarque : Il est courant d'ajuster les droits pour mettre tous les développeurs à égalité. En revanche, il ne peut y avoir à tout moment qu'une branche principale, chaque branche pouvant reprendre ce titre sur commande, d'où le besoin d'une certaine concertation dans l'équipe.

2.4. *Evolutions*

Le succès de CVS s'explique par le fait qu'il effectue des tâches compliquées faciles contrôler, du fait de l'accessibilité des textes et la simplicité de l'outil[PUR 00].

On constate à l'usage que :

– les conflits sont rares et généralement faciles à résoudre ;

– la gestion des versions est rassurante pour les auteurs qui peuvent retrouver à tout moment une version satisfaisante à leur goût ;

– le fait d'associer aux validations des échanges de messages ou d'autres actions aide à mesurer l'activité réelle de développement ;

D'une manière générale, l'utilisation d'un mécanisme automatique de contrôle responsabilise les participants. L'un des spécialistes français de CVS [MOL 97] y voit même une explication au faible nombre de conflits.

2.5. *Limites*

On a vu plus haut que CVS ne couvre pas tous les besoins de ses utilisateurs en matière de gestion de configuration. On peut classer aussi dans cette catégorie les problèmes organisationnels qui sont laissés à la liberté des gestionnaires de projets car ils sont d'un autre niveau comme :

– quand une version peut-elle être validée ?

– qui a le droit de metre à jour la branche courante ?

– quand doit-on créer une nouvelle branche ?

Ce sont des décisions concertées que doivent prendre les développeurs au moment où ils conviennent de versionner un projet. Contrairement à d'autres tâches de spécification des projets informatiques, il est très facile d'intégrer dans un référentiel CVS un projet en cours.

---

8. Cette fonction comprend la possibilité de récupérer n'importe quelle trouvaille perdue par mégarde entre deux mises à jour, sans avoir recours aux sauvegardes. Les spécialistes de la recherche de texte sous Unix (grep) y arrivent en une ligne de commande.

CVS souffre aussi de quelques faiblesses qu'il est bon de connaitre si on doit passer brutalement d'un développement anarchique à un développement coopératif :

– le renommage des fichiers dans le référentiel est assez pénible ; il vaut mieux avoir défini une bonne politique de nommage avant de confier le projet à CVS ;

– celui de l'arborescence est encore pire. De même, il est délicat de mélanger deux arborescences ayant des référentiels différents ;

– l'utilisation de fichiers spéciaux au sens du système. Cette limite de CVS le rend impropre à la sauvegarde (mais qui n'empêche en rien la sauvegarde du repository, au contraire).

3. Exemples de tâches

Voyons donc quelques tâches typiques des professionnels de l'information, tâches qui peuvent justifier un système de versionnage. [BOR 00] Les exemples sont montrés avec la ligne de commande de CVS. CVS a aussi des interfaces graphiques.

3.1. *Retrouver un serveur web dans l'état où il était le 10 février*

Le contenu des serveurs web évolue très vite, souvent sans tenir compte des problèmes d'archivage. On supprime ou on modifie des pages qu'on ne garde pas. Cela rend très difficile toute analyse de l'évolution d'un serveur, par exemple.

Souvent, on éprouve le besoin de garder une copie du serveur à une date donnée. Mais, sans outil de gestion des versions, il est très difficile, après coup, de retrouver la copie pertinente.

On voit ainsi des serveurs web dont le répertoire contient `index.html`, `index.html.old`, `index.html.very-old`, `links.html`, `links.html.old`, etc. Sans que l'on puisse dire facilement quel `links.html` va avec quel `index.html`.

Un bon système de versionnage permet de récupérer l'état d'un ensemble de pages à une date donnée.

Commande CVS pour récupérer le répertoire `htdocs`, avec tous ses fichiers, dans son état du 10 février :

```
cvs checkout -D '2001-02-10' htdocs
```

Un système de gestion de versions peut permettre de marquer un ensemble de fichiers d'un *label*, qui facilitera l'extraction ultérieure.

Commande CVS pour marquer le répertoire `htdocs` :

```
cvs tag VERSION_DURAND_MODIFIEE_DUPONT
```

et pour extraire le répertoire tel qu'il était lors du marquage :

```
cvs checkout -r VERSION_DURAND_MODIFIEE_DUPONT htdocs
```

Figure 1. *L'interface CVS-Web de SourceForge*

3.2. *Trouver ce qui a changé en juin (c'était mieux avant)*

Avant tout, un système de versionnage stocke des versions successives du document[9]. Il peut donc facilement indiquer qu'est-ce qui a changé entre deux versions, sans avoir à tout relire ligne par ligne.

Commande CVS pour afficher la différence entre deux versions, ici données par leurs dates :

```
cvs diff -D '2000-06-01' -D '2001-06-30' article_epidemio_moustique.tex
```

9. Pour gagner de la place, CVS ne stocke en fait que les différences entre versions successives.

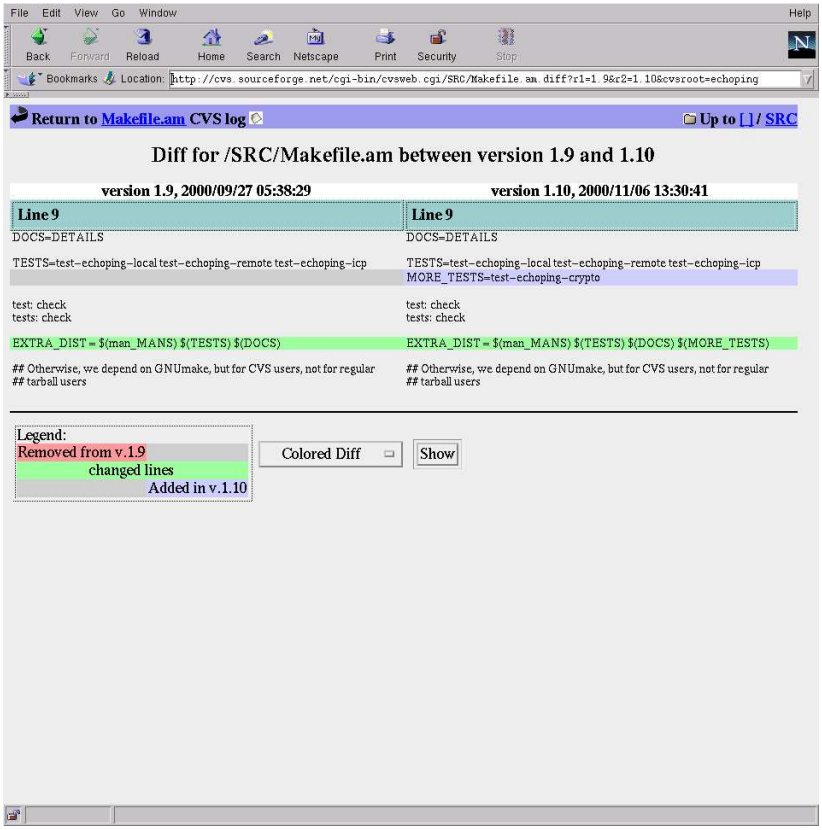

Figure 2. *L'interface CVS-Web de SourceForge : différence entre deux versions*

3.3. *Travailler à deux sur la même documentation sans avoir besoin de se téléphoner toutes les cinq minutes pour se coordonner*

L'internet a permis de créer des équipes travaillant en étroite collaboration (par exemple pour écrire un article scientifique), sans que ces équipes n'aient à être concentrées en un point.

Pour éviter de passer tout son temps en coordination, l'aide d'outils informatiques est nécessaire.Un outil de gestion de versions permet de revenir facilement en arrière (si on n'approuve pas les modifications, toutes les versions antérieures sont facilement accessibles).

Certains outils de gestion de versions permettent de fusionner les modifications réalisées indépendamment par deux auteurs. Ainsi, il n'est pas nécessaire de *réserver*, de bloquer toute modification parce qu'on a un paragraphe à ajouter.

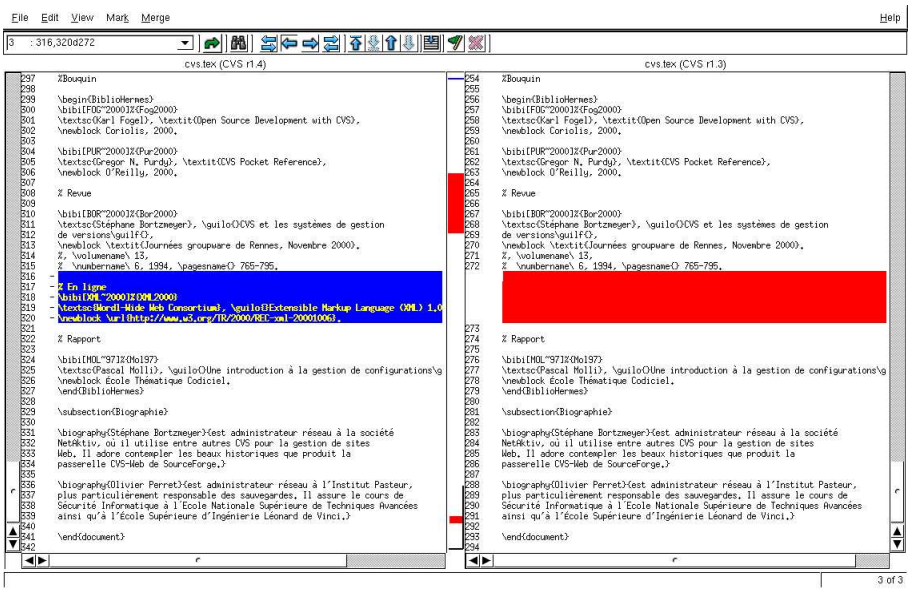

Figure 3. *L'interface graphique Tk CVS : différence entre deux versions*

Commande CVS pour afficher l'historique des modifications, avec le texte qu'a tapé chaque auteur au moment de la confirmation :

```
cvs log article_epidemio_moustique.tex
```

et son résultat :

```
revision 1.2
date: 2001/03/19 09:54:20;  author: bortz;  state: Exp;  lines: +1 -1
ça compile
----------------------------
revision 1.1
date: 2001/03/19 02:51:15;  author: perret;  state: Exp;
pemière version apres la mise au point
```

4. Bibliographie

[BOR 00] BORTZMEYER S., « CVS et les systèmes de gestion de versions », *Journées groupware de Rennes, Novembre 2000*, Rennes, 2000, Comité Réseaux des Universités.

[BRA 00] BRAUN T., DIOT C., HOGLANDER A., ROCA V., « Extensible Markup Language (XML) 1.0 (Second Edition) », Standard n° 1.0, 2000, World-Wide Web Consortium.

[CVS 00] CVS, « Concurrent Version System, the open standard for version control », rapport, 2000.

[FOG 00] FOGEL K., *Open Source Development with CVS,* Coriolis, 2000.

[MOL 97] MOLLI P., « Une introduction à la gestion de configuration », *École Thématique Codiciel*, Nancy, 1997, Loria.

[PUR 00] PURDY G. N., *CVS Pocket Reference,* O'Reilly, 2000.