卒　業　論　文

デジタルコンテンツの共同制作に於ける
バージョン管理システム CVS の活用に関する提案

福　岡　国　際　大　学
国際コミュニケーション学部
デ ジ タ ル メ デ ィ ア 学 科

張　煒

目次

第１章

はじめに

平川研究室では、エンターテイメント系デジタルコンテンツの作成を行っている。デジタルコンテンツは、基本的にコンピュータを用いて作成される。一つの作品は、最初から完成まで一人で作業をする場合もあるが、何人かで協力して共同制作を行う場合もある。このようなコンピュータを用いた複数人での作業は、デジタルコンテンツの作成だけではなく、他の場面、特に大規模プログラムの作成において頻繁に行われてきた。

多人数の協力による開発プロジェクトは、効率よく複雑で規模の大きなプログラムの作成が行えるという利点がある。しかしその一方で、問題点として次のようなことがあげられる。

1. 複数人が同時に一つのファイルを修正する場合において、自分が修正したプログラムを他の人に削除されることがある。
2. 作成しているプログラムは、すでに第三版をリリースしたのに、第二版において問題が出てきて、第二版の内容を修正する必要が出てきた場合、最新の第三版しか保存をしていなく対応ができないことがある。
3. 遠隔地の共同作業では、プログラムのやり取りもインターネット回線を使う場合が多い。
4. 一段階の修正が終わった時、古いものと入れ替えを行うが、複数人で修正を行っている場合、他の人が修正した部分を上書きしてしまう可能性がある。

以上のような問題は、プログラム作成だけでなく、デジタルコンテンツ作成においても起こることである。例えば、一つの作品は沢山の種類のファイルをもとに構成されている(.gif .png .jpg .wav .bmp .amx .txt……など)。それを、各自担当の部分を編集するために、全部のファイルを持ち、参照しながら編集することがある。よって、上記の問題を解決することは、デジタルコンテンツ作成での問題を解決することに繋がる。

本論文では、このような問題を解決するために Unix の体系内の SCCS と RCS に基づいて開発されたバージョン管理システム CVS を紹介すると共に、デジタルコンテンツ作成への活用を提案したい。

第２章

バージョン管理システム

２.１ ＣＶＳとは

ＣＶＳという言葉は、Concurrent Versions System の頭文字を取ったもので、日本語に直訳すれば、並列版管理機構である。ＣＶＳはもともとプログラム開発の為のツールとして開発された。

プログラムを作成する時は、ステップ毎に少しずつコードを書いて動かし、動いたら次へという形で作業が進んで行く。しかし、動かしてみてうまくいかなかったら、ある程度以前の状態へ戻って、また別の方法を試すことも多いので、こういうとき、今までのプログラム作成履歴が残っていなかったら不便である。勿論、普通に考えたら、途中でのファイルを別名で保存して残しておいても、以前の状態へは戻せるが、それだけで保存するファイルは多大な量になってしまい、作業が混乱してしまう恐れがある。

作業途中のファイルの状態を、通常、バージョン（version）と呼ぶ。しかし、これはソフトウェアのリリースのバージョンと紛らわしいので、Linux 上ではリビジョン（revision）と呼ぶことが多い。

ＣＶＳは複数の人と共同でファイルを編集する場合に、ファイルのバージョン（リビジョン）を管理することができるシステムである。このＣＶＳの前に、Unix 系のＳＣＣＳとＲＣＳというバージョン管理のツールがあった。ＣＶＳはそれに基づいて開発されたので、ここではまず、ＲＣＳの仕組みを説明する。

２.２ ＲＣＳとは

ＲＣＳやＳＣＣＳは、バージョン（リビジョン）間の差分だけを取り出し、一つのファイルの形で管理するツールとして開発された（図１を参照)。ＲＣＳでは、ファイルのバージョン（リビジョン）が上がる度に、修正された部分が差分として取り出され、追加保存されていく。システムの利用者から、ファイルのあるバージョン（リビジョン）を求められた時は、このファイル群から必要な差分が取り出され、求められるバージョン（リビジョン）のファイルが構成され、作業を行うことができる。

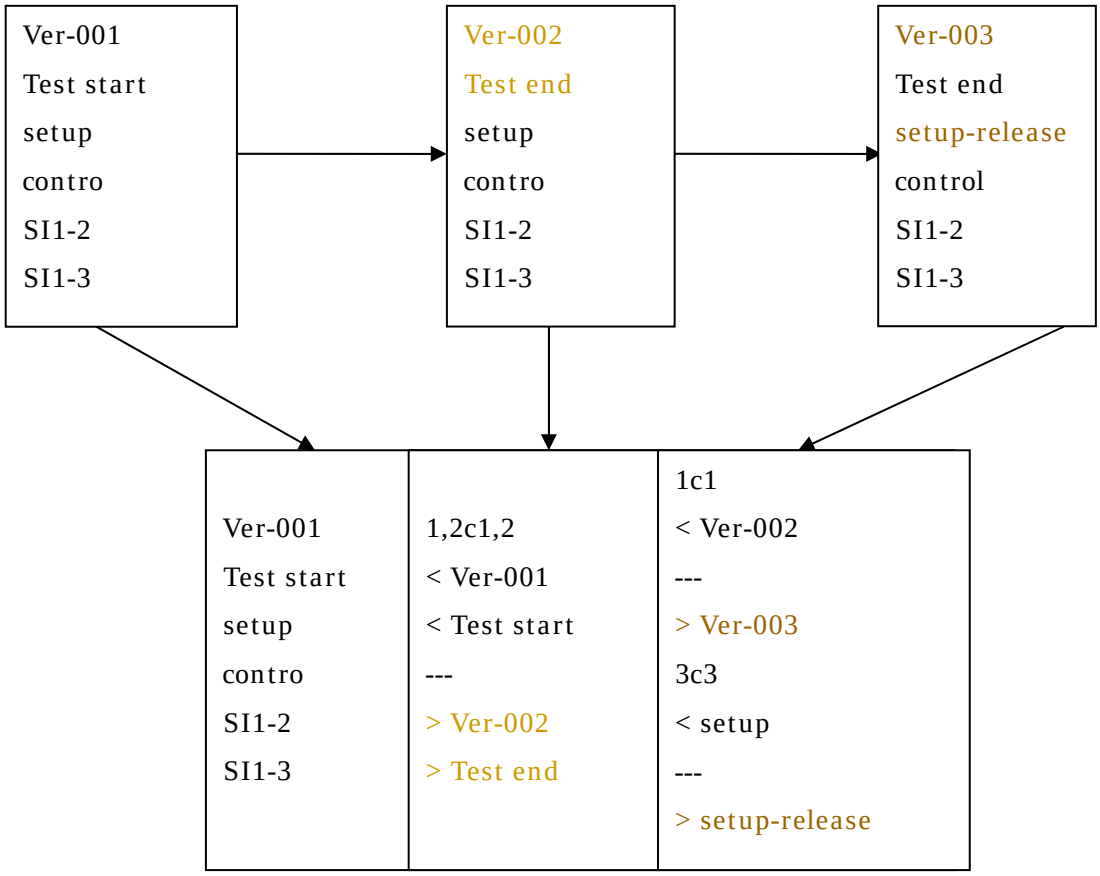

図１　ＲＣＳの仕組み

２.３ ＲＣＳの問題点

ＲＣＳでは、修正されたところの追跡やバージョン（リビジョン）の管理はできるが、プログラムが大きくなり幾つかのディレクトリに分かれた沢山のファイルから構成されるようになると、バージョン（リビジョン）を管理するファイルがばらばらになってしまう。また、一番不便なのは、このような大きなプログラムを複数の人間が協力して開発すると、ＲＣＳの機能である「ファイルの排他制御」、つまり、「ある人が編集中のファイルを他の人が編集できないようにしている」状況において、作業を行なった人が排他を解除し忘れると、他の人が作業できなくなるという問題が発生することである。例えば、図２に示すように、『誰かが本棚からファイルを持っていって、他の人がそのファイルを使えない状況』や『ファイルは返って来たのに、鍵が掛かったままで他の人が使えない状況』が発生する。これは、非常に困ったことである。

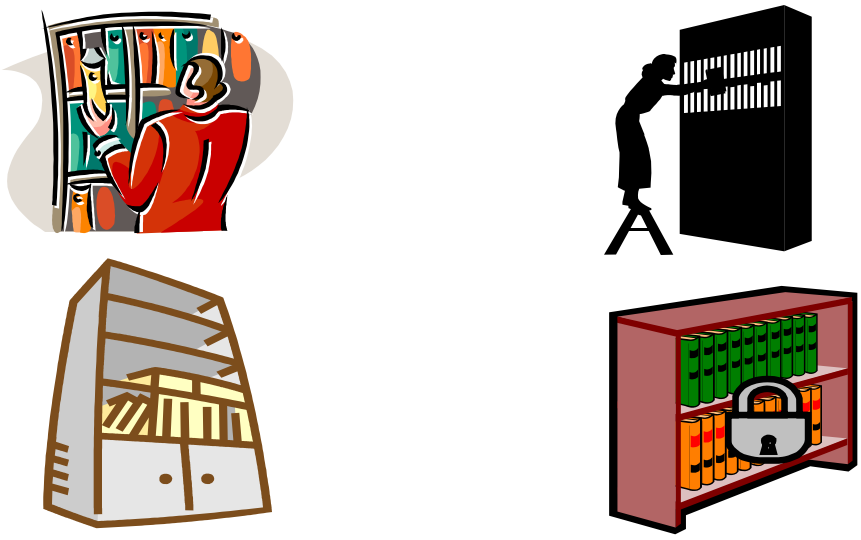

図２　ＲＣＳの問題点

２.４　ＣＶＳの仕組み

ＣＶＳの仕事の原理は、サーバーに容器（リポジトリ repository）を建てて、容器内に多くの異種の項目のソースを据え置くことができ、この容器の管理者がこれらのソースを管理する、というものである。これで、ソースを修正するのが一人だけの時と同様の状態を作り出せ、同じファイルの複数修正による衝突を避けることができる。

ＣＶＳを使ってプログラムの開発を行う場合、開発者（修正者）は、まず、リポジトリ内のソースを端末にダウンロードを行う。これはソースのコピーにあたる。次に、開発者（修正者）がそのダウンロードしたソースを使ってローカルＰＣで開発（修正）作業を行う。その後に、ＣＶＳに修正したファイルを渡すことになる。実際には、ＣＶＳサーバーにてコマンドで登録（commit）し、ＣＶＳの管理者が統一修正を行う。

これで、修正された部分を追跡することが可能になる。また、さらには上書きなどの衝突を制御することもできる。図３に、ＣＶＳ利用時の簡単な流れを示す。

1. リポジトリから、各自でコピーを取り出す（チェクアウト）。
2. そのコピーに対して作業を行う。
3. 各自の作業が終わったら、新しいバージョンとして登録する（コミット）。
4. 管理者は、差分の抽出、およびファイルの再構成を行い、リポジトリに格納する。

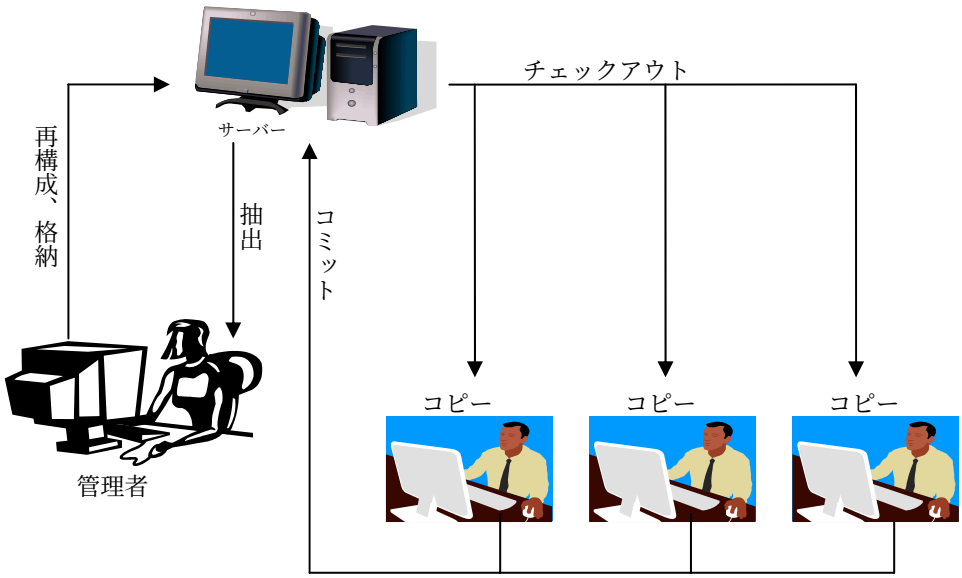

図３　ＣＶＳの仕組み

ＣＶＳでは、開発者がファイルをまとめて他の人と独立に取り出すことができ、他の人の作業とぶつかることが避けられる点である。また、排他機能を使用してない為、たまたま同じところを編集してしまった場合には、その解消は話し合いに任せるというアプローチをとっている。このことが、さらに作業の独立性を高めているといえる。

２.５　ＣＶＳでの競合

前節で述べたように、ＣＶＳでは、たまたま同じところを複数人が取り出し、修正してしまう場合が発生する。それを競合と呼ぶ。

競合の解消は、例えば、ＡさんとＢさんが同時に同じファイルをチェクアウトして、修正し、作業終了時にコミットしようとしたら、コミット前のアップデータで、「ＡさんとＢさんが修正した内容はだぶっている」と警告が出るというものである。それを受けて、ＡさんとＢさんの二人で話し会いをし、競合を解決するという形になる。これで、同時に修正した同じファイルの内容を、どちらか一方が上書きするという危険性が避けられる。

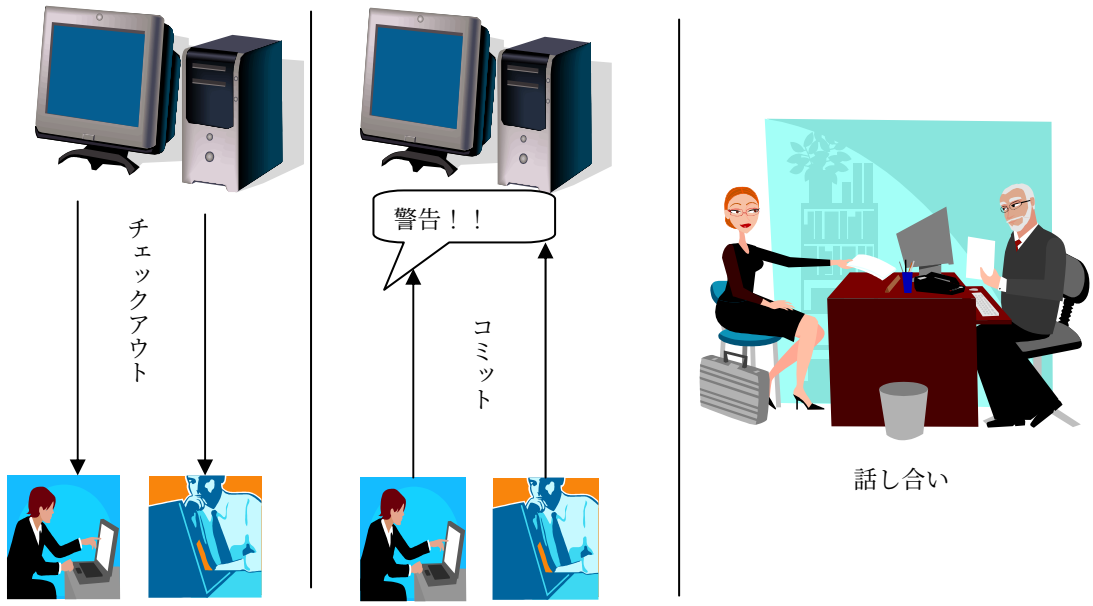

図４　ＣＶＳでの競合

２.６　ＣＶＳの利点

このように、ＣＶＳを使うと、多人数が参加する開発プロジェクトで、自分が修正したプログラムを他の人に削除されることがなくなる。確かに、ＣＶＳを使わなくても、修正し終わったファイルを持ち寄り、保存する前に皆で話し合って、二重三重の上書き保存が行われないようにすればいいという考えもあるかもしれない。しかし、二・三人ぐらいの規模のプロジェクトであれば、保存する前に常に話し合いを持ち、上書き保存の危険を回避できるかもしれないが、参加者の数が何十人にのぼる大規模プロジェクトでは難しいと思われる。

また、作っているプログラムはすでに第三版をリリースしたのに、第二版のプログラムで問題が出てきて、第二版の内容を修正することが必要になったとき、今までのプログラムの作成履歴が残っていなかったら不便である。勿論、普通に考えたら、常にプログラムの途中の状態を、別名で保存して残しておけば、前のバージョンのプログラムは出せるであろう。しかし、途中で保存されるファイル数は、考えただけでもすごい量であることは確かであり、探すだけで混乱してしまうに違いない。

もう一つの例は、遠隔地で作業する場合である。インターネットを使った協力作業の場合には、プログラムを修正した時、修正したものを送る必要が出てくる。その際、修正したプログラム全体を送るよりも、前との差分ファイル、つまり、修正したところだけを送ったほうが、時間も、手間もかからない。

このように、もともとプログラム開発のために作られたＣＶＳは、複数人で行うプログラム開発には、素晴らしく役に立っている。特にオープンソースの世界では無くてはならないツールである。企業でのバージョン管理にも使用でき、実際、多くで使われている。

ＣＶＳは、特にソースコードの共有と遠隔からのアクセスに向いている。テキストファイルの管理に利用することもでき、移動が多く移動先で仕事をする人や、家と会社とか、違う複数の場所に作業環境がある人、いくつかのコンピュータで同じファイルを編集する必要がある場合、または、一人で複数の環境を持つ人など、自分で行なった並列作業の管理をＣＶＳでバックアップとして利用ができる。

２.７ ＣＶＳの出来ない事

以上のように、ＣＶＳはとても便利なシステムであるが、できないことなのに、できるように勘違いしやすいことも述べたい。

前節で、ＣＶＳがプログラムの開発に重要な役にたっていると述べたが、ＣＶＳは構築システムではないため、複数のファイルからなるプログラムの場合、コンパイルをしてひとつのプログラムに組み上げる構築という作業が必要である。そのためにはＣＶＳ以外に別のツールが必要になる。

一般的に、プログラムの中のソースコードファイルは、別のファイルに依存している。この依存関係の端からコンパイルを行なって、マシン語に変換し、最終的に一つのプログラムに組み上げるのであるが、あるファイルが変更された場合、全てのファイルをコンパイルするのではなく、変更されたファイルとそれを組み込む部分のみをコンパイルする方法が取られる。そのために必要な作業をＣＶＳがすることはない。

また、ＣＶＳはバージョン管理のツールとして使われているが、あくまでも道具なので、ＣＶＳ自身が管理することはできない。

さらに、競合についても、ＣＶＳが「重なった変更があった」という警告と変更の内容だけは出してくれるが、どちらが正しいか、どちらをコミットすればいいかなどはＣＶＳ自身が判断できないから、関係のある人が話し合って、解決するしかない。

なお言えば、ＣＶＳは変更管理をしない。例えば、修正があった場合、誰がどの部分を変更したのか、それらの変更管理をするには、それを管理する責任者を決めておき、変更した差分を生成してその責任者にメールで送って管理する形をとる。このように、ＣＶＳの外部で行なわれる様々な作業を、Eclipse などのツールを使って、誰によりコミットされたのか調べることできるが、ＣＶＳとしてはどんな責任も持たない。

第３章

デジタルコンテンツ制作への CVS の活用提案

３．１ デジタルコンテンツの構成要素

デジタルコンテンツの構成要素は、画像、映像、サウンド、テキスト、プログラムや設定ファイル(.gif .png .jpg .wav .bmp .amx .txt……)などである。それぞれの部分も、沢山のファイルで構成されている。例えば、画像ファイルだけでも、gif、jpg、png ファイルなどがあり、写真を流すようなスライドショーでは、それらの画像ファイルを使うことになる。

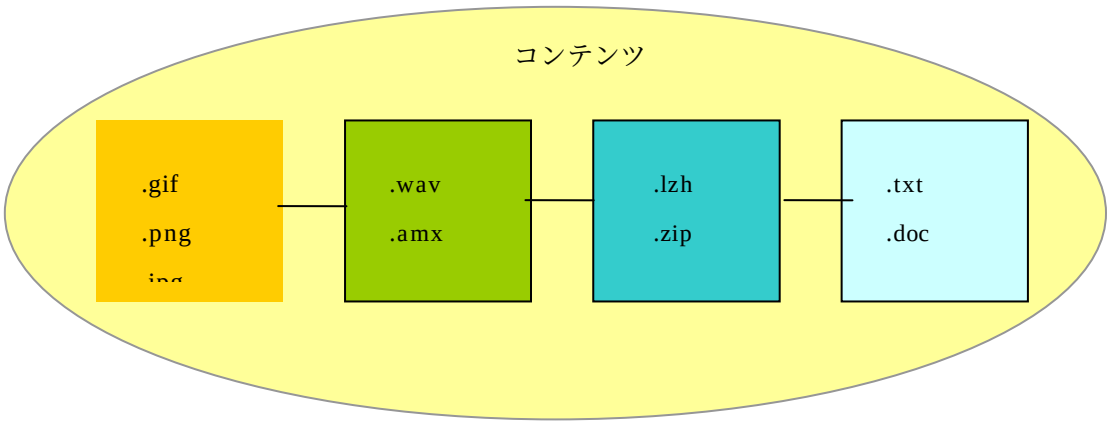

図５　デジタルコンテンツの構成要素

３．２ デジタルコンテンツの制作手順

デジタルコンテンツの制作手順としては、まず、作るものの企画を立てて、ラフスケッチを描く。それから素材を集め、編集に入る。

編集の途中でも素材を集めなおしたり、何度も何度も修正したりすることが多いうえ、このような沢山の要素で構成される一つの作品は、一人で制作することもあるが、大人数で共同制作の場合もある。つまり、この作業にＣＶＳが役に立つのではないかと思われる。

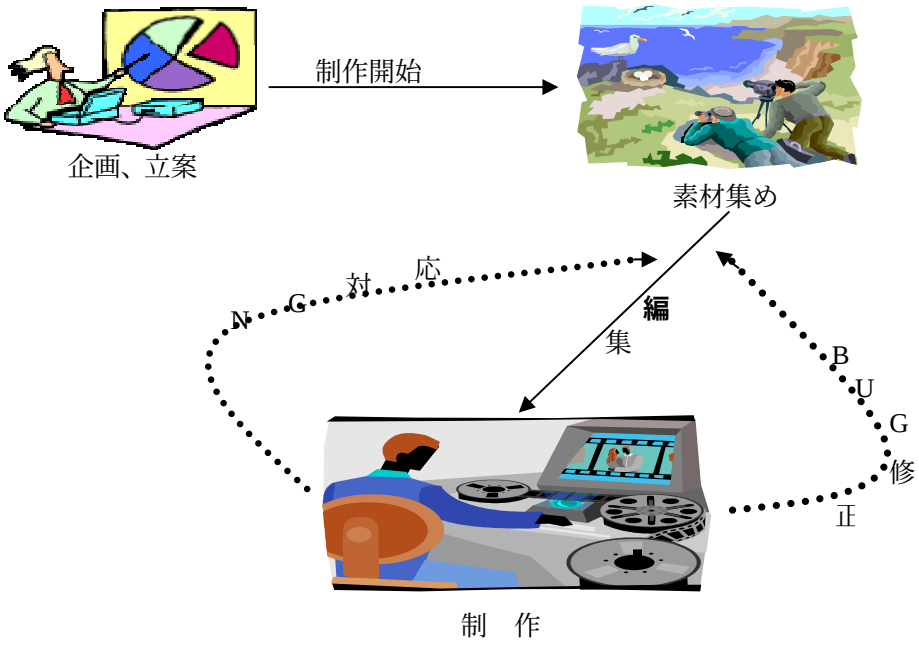

図６　デジタルコンテンツの制作手順

３.３ デジタルコンテンツ制作へのＣＶＳの活用提案

平川研究室では、エンターテイメント系デジタルコンテンツの作成をやっている。今は一人一人の個人制作だが、今後は複数の人数で制作する場合も考えられるので、CVS の利点とデジタルコンテンツ制作の実作業内容を考え、私は CVS サーバーの導入を提案する。

- 作品を構成するすべてのファイル、画像、映像、音声、テキストファイルなどを、各部分の担当者が取り出せるように、CVS サーバーに据え置く。
- 制作しながら、各自作ったものや修正したところをサーバーに入れて、協力メンバーなら誰でもすぐに出して使えるようにする。そうすることで、編集で失敗しても、いつでも、前の内容を取り出し、前の状態に戻すことができる。
- 自分の作った部分を誤って削除されないようにすることができ、他の人のものに上書きする危険性を避けられる。

CVS サーバーを立てたら、このように活用することができる。

第４章

CVS サーバーの構築に必要なハードウェアとソフトウェア

実際にＣＶＳサーバーを導入するために必要な、ハードウェアとソフトウェアを示す。ＣＶＳ自身は大きくないから、サーバーへの要求は特に高くはない。必要なＯＳがインストール出来れば十分だと考えられる。但し、ＣＶＳに入れるコンテンツの内容により、円滑に動くように、ハードディスクの容量を十分余裕があるようにした方がいいと思う。ＣＶＳは UNIX 系と Windows 系がある。UNIX 向けのＣＶＳサーバーを構築ため、必要なソフトウェアは、UNIX 系ＯＳとクライアントからＣＶＳにアクセスできるような設定が必要である。これらには、TeraTermPro、Eclipse などがある。必要な知識としては、UNIX の操作およびＣＶＳコマンドがある。

ＣＶＳコマンドや具体的な使い方は、下記を参照した。
http://nile.slis.tsukuba.ac.jp/~yuka/memo/cvs.html

ＣＶＳはフリーでダウンロードできる。
http://www.ne.jp/asahi/hishidama/home/tech/cvs/install.html

Windows 系のＣＶＳの使い方やダウンロードは、下記を参照した。
http://radiofly.to/nishi/cvs/

第５章

まとめ

最後に、ＣＶＳのデジタルコンテンツ制作への有効性をまとめる。

1. ＣＶＳサーバーを導入したら、コンテンツを構成する各種ファイルをスムーズに融合できる。
2. サーバーにリポジトリを設定し、全てのファイルを据え置いたら、簡単にファイルの管理ができる。
3. インターネットを通してサーバーにアクセスし、各担当者が対面作業ではなくても、全ての最新修正情報を手に入れることができる。
4. 制作中でバグが発生したり、緊急問題が出たりした場合、確実にすぐに対応でき、誤作業を避けられるため、時間を最も有効に効率よく利用できる。

以上のことより、デジタルコンテンツ制作にＣＶＳの活用を提案し、来年度以降の平川研究室での導入を期待する。

参考文献

参考にしたＨＰ：

①http://www.stackasterisk.jp/tech/systemConstruction/cvs02_03.jsp

②http://www.pgsqldb.org/cvstutorials.html

③http://www.chedong.com/tech/cvs_card.html

参考にした書籍：

①タイトル：『入門 CVS』
著者：大月 美佳
出版社：秀和システム
ISBN：4-7980-0105-8 C3055

②タイトル：『実用 CVS』
著者：Jennifer Vesperman
翻訳者：滝沢 徹、牧野 祐子
出版社：オライリー・ジャパン/オーム社
ISBN：978-4-87311-164-3

③タイトル：『CVS によるオープンソース開発』
著者：Karl Fogel、 Moshe Bar
翻訳者：竹内里佳、でびあんぐる
出版社：オーム社
ISBN：4-274-06473-5「入門 CVS」