グループウェアと
ネットワークサービス 45－5
（２００２．１０．２４）

# ファイル部分ロックによるウェブベースXHTML共同編集システム

櫻　裕司[†]　香川考司[††]　垂水浩幸[††]

分散環境での開発では、複数のユーザによるファイルの更新が衝突してしまう可能性がある。現在の共同開発を支援するシステムではこれを防ぐためにファイルのロック、マージといった方法を用いている。しかし、これらの方法では共同開発の作業中に問題が発生する。そこでファイルの一部分のみをロックすることによりこの問題解決するシステムを作成した。本システムではXHTMLファイルの各要素をロックの粒度とし、ユーザ間の排他制御を行っている。また、Java Servlet, Java Plug-inを用いることによりHTTP及びそのトンネリングのみの通信で制御されている。

# A Web-based Cooparative XHTML Editor by Using Partial Lock of a File

Yuji Sakura ,[†] Koji Kagawa [††] and Hiroyuki Tarumi [††]

In a distributed environment, there is a possibility of conflicts when a file is updated by multiple users. Current systems supporting distributed development employs methods such as “locking” or “merging” in order to prevent or to dissolve conflicts. However, these traditional methods are problematic. In this paper, we propose a system where a lock is applied to a part of a file in order to avoid this problem. More specifically, an entity in XHTML documents can be a unit of mutual exclusion.

Since the system is constituted of Java Servlets and Java Plug-in’s, it uses only HTTP (including HTTP tunneling) for communication between components.

## 1. はじめに

ブロードバンド化が進み、高速で、常時接続のネットワーク環境が整いつつある現在、分散環境での開発が容易に行えるようになってきた。また、社会のIT化及び、コンピュータのモバイル化により遠隔でのファイル更新が可能となってきている。しかし、こうした分散環境での開発では、複数のユーザによるファイルの更新が衝突してしまう場合がある。つまり、あるユーザの更新の結果が他のユーザの更新の操作により無くなってしまう。そこで分散環境でユーザが安全に共同作業が行えるシステムが必要となってきている。

現在、共同開発を支援するものにはWebDAV (Web-based Distributed Authoring and Versioning) のようにファイルやディレクトリ単位でロックをかけて衝突を防ぐものやCVS(Concurrent Versions System) のようにロックはかけず、ファイルの更新時衝突があれば手動で修整をするシステムなどがある。しかし、この方法では他人のロックしたファイルを編集できない、衝突時の変更が面倒であると言った短所がある。

そこでファイルに対するロックではなくファイルを論理的なブロックにわけそのブロックに対しロックをかけるという方法を用いることによりこの短所を補う。また、理解しやすい、柔軟、再利用性が高いなどの理由で注目されているXMLに注目し、特にそのなかでもなじみ深いXHTML[1),2)]に的をしぼった。そしてウェブページの共同開発および更新を安全に行うための支援システムを構築した。

また、本システムはJavaアプレット、Javaサーブレットを用いており、作業がウェブベースで行える。多くのプラットフォームでの利用が可能である。

本論文では、まず現在一般的なロックの方式の問題点をあげ、その問題を改善した本システムのロック方式を述べ、その後XHTML編集システムの構成を述べる。

† 香川大学大学院工学研究科
Graduate School of Engineering, Kagawa University
†† 香川大学工学部
Faculty of Engineering, Kagawa University

HTML

```
<!DOCTYPE HTML PUBLIC
  "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3c.org/TR/html4/loose.dtd">
<HTML>
 <HEAD>
  <TITLE> HTML サンプル </TITLE>
 </HEAD>
 <BODY>
  <HR>
  <A HREF=stwww> stwww </A>
 </BODY>
</HTML>
```

XHTML

```
<?xml version="1.0" encodind="EUC JP"?>
<!DOCTYPE html PUBLIC
  "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3c.org/TR/xhtml1/DTD/
xhtml1-strict.dtd">
<html lang="ja" xml:lang="ja"
 xmlns="http://www.w3c.org/1999/xhtml">
 <head>
  <title> XHTML サンプル </title>
 </head>
 <body>
  <hr />
  <a href="stwww"> stwww </a>
 </body>
</html>
```

**図 1** HTML と XHTML の例

## 2. XHTML について

XHTML(eXtensible HyperText Markup Language)[2] は HTML4 を XML の文法で定義しなおしたものである。したがって標準的な XML ツールを用いて処理することができる。また、現在 XHTML で書かれたウェブページは少ないが、今後は XHTML への書き換えが進むと考えられる。

また、XML は大文字小文字を区別するので XHTML では全てのタグは小文字を使用しなければならない。簡単な HTML と XHTML の例を**図 1** に示す。

## 3. 既存の制御方法

### 3.1 ファイルのロック

共同作業の衝突防止の方法として一般的なものにファイル単位でロックをかけるものがある。これは 1 人のユーザがファイルの編集を始めるとそのファイルにロックをかけるという方法である。しかし、この方法では同じファイル内の同時に編集を行っても問題の発生しないの場所への書き込みもブロックすることになり作業の効率の低下の原因になる。

### 3.2 マ ー ジ

マージ方式ではファイルのロックは行わず、ファイルの更新時に衝突がおこると自動的に編集された場所をマージする。ファイルのロックを行わないので同ファイルに対する他人の作業の停止はしない。しかし、自動的にマージが行えない場合、意味的な矛盾が発生する場合はユーザが手動での修正を必要とされ、負担が増加する。

## 4. 本システムでの制御方法

### 4.1 ロックの制御

上記の問題点を解決するため、本システムでは XML ドキュメントがサーバ上で**図 2** のように DOM ツリーとしてメモリ上に読みこまれ、各ノードをルートとする部分木を粒度として、ロックや更新を行う。ユーザがドキュメントを編集する場合、編集部分を全て含んだ 1 つのノードをロックすることにより変更の衝突をふせぐ。ただし、管理すべきノードが木構造になっているので、あるノードが編集中である場合、そのノードを子孫に持つノードの編集も許可しない。たとえば図 2 の場合、p タグ内が編集中ならば、html、body、p タグにはロックがかかり編集を行うことはできないが、それ以外の部分の編集は可能である。

このようにロックの粒度をファイルより細かくすることによって、ロックの競合を減らし、マージの修正時のにかかる労力、及びその時間の削減といった利点が生まれる。このことは 3) で検証されている。

### 4.2 更新の処理

ファイルの更新時、クライアントからロックされている部分のファイルを受け取り

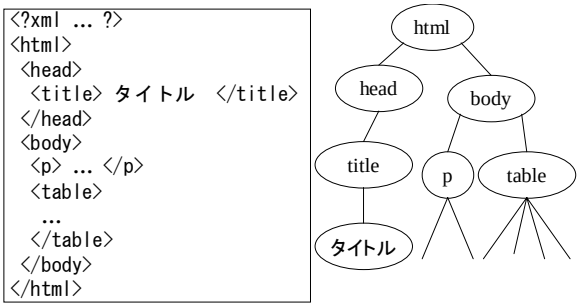


**図 2** XML と DOM

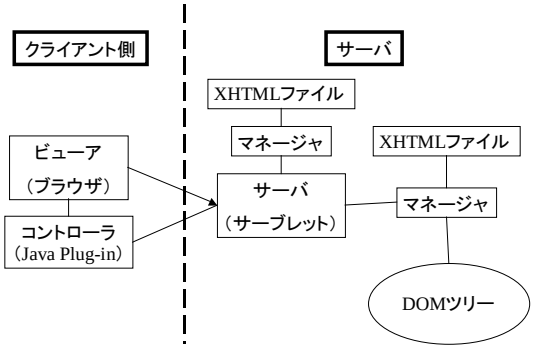


**図 3** システム構成

## 5. システム

### 5.1 システム構成

本システムは**図 3** に示すようにサーバ、クライアント、編集中の HTML ファイル閲覧用のブラウザ、XHTML のファイルを管理するマネージャで構成されている。また、ブラウザでシステムの URL にアクセスすることで起動されるよう、クライアントには Java Plug-in、サーバには Java Servlet を用いている。

#### 5.1.1 サ ー バ

サーバは Java Servlet である。サーバはリクエストによってシステムで利用可能なリストの出力、指定した編集中のドキュメントの出力、クライアントの Java Plug-in を起動させるための HTML の出力、クライアントとの通信を行う。サーバはブラウザからのリクエストが送られてきた場合、通常の Java Servlet のように HTML を送信する。しかし、クライアントの Java Plug-in からリクエストを受けた場合、レスポンスを返して接続を切るのではなく、そのまま接続したままにしておき必要な時にサーバからの通知を送る。

#### 5.1.2 クライアント

クライアント (**図 4**) はサーバと接続、ビューアであるブラウザの制御を行う。クライアントは起動時にサーバから編集するファイルの情報を受け取り XHTML ファイルをツリーとして表示する。また、ユーザがクライアントのツリー表示部分のノードを撰択すると、そのノードにあたる部分を Live Connect により JavaScript を通し、強調する ( **図 5**)。これによりユーザは直感的にコントローラ上の操作したいノードがブラウザ上のどの部分を指すのかが理解できる。さらに、自分、他のユーザによる変更があった場合はブラウザの表示を更新し、常にサーバ上のファイルとの整合性を保つ。

#### 5.1.3 ビューア

ビューアは Java Plug-in が使え、XHTML に対応しているブラウザならどんなものでもビューアとして利用できる。これはクライアントに特別なソフトウェアの導入必要が無いだけでなく、上で述べたように編集中のドキュメントの参照、編集の結果が即座に行われるという利点もある。

#### 5.1.4 マネージャ

マネージャは XHTML ファイルから DOM ツリーを作成し、クライアントからのリクエストを処理する。1 つのマネージャは 1 つの XHTML ファイルを管理する。

### 5.2 本システムでの編集作業の流れ

本システムでの編集作業の流れは次のようになる。

(1) システムの URL にブラウザでアクセスする。
(2) 現在システムが管理しているファイルの一覧が表示され、その中から編集するファイルを選択する。
(3) 撰択したファイルの内容がブラウザに表示され、クライアント Java Plug-in が起動し、編集するファイルのツリー構成が表示される。
(4) クライアントのツリーから編集するノードを撰択する (図 4)。撰択されたノードが実際ブラウザでどこに当たるかを知らせるため、ブラウザの表示が強調される (図 5)。
(5) ポップアップメニューから編集を選択する。
(6) サーバは該当ノードをロックし、その部分

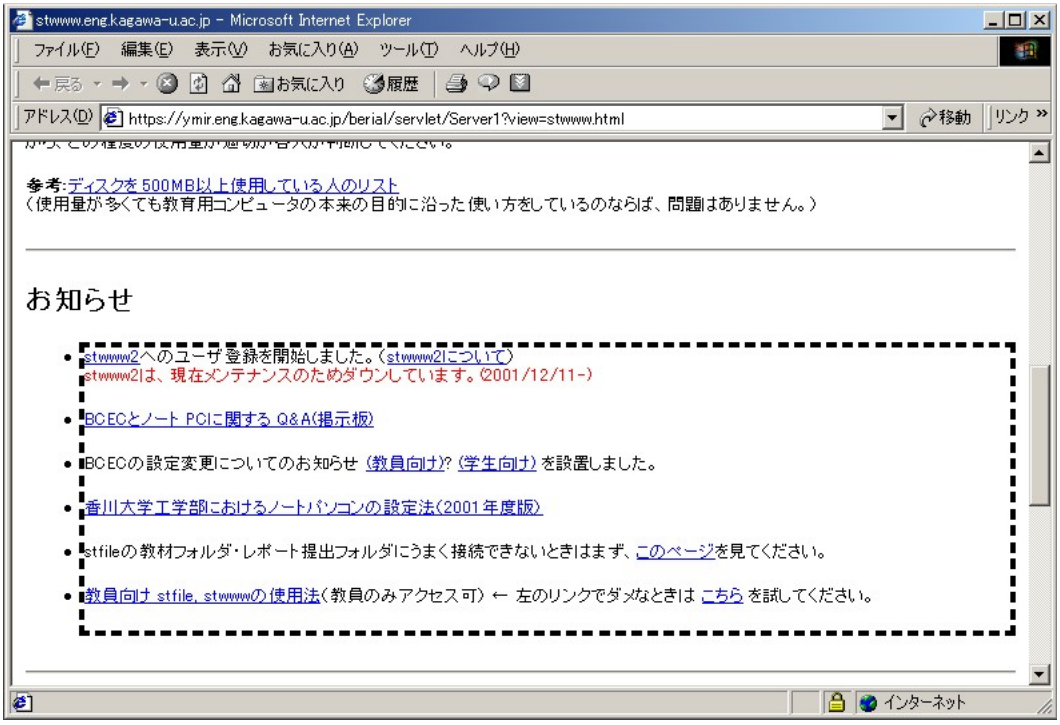


**図 5**　スタイルシートの変更

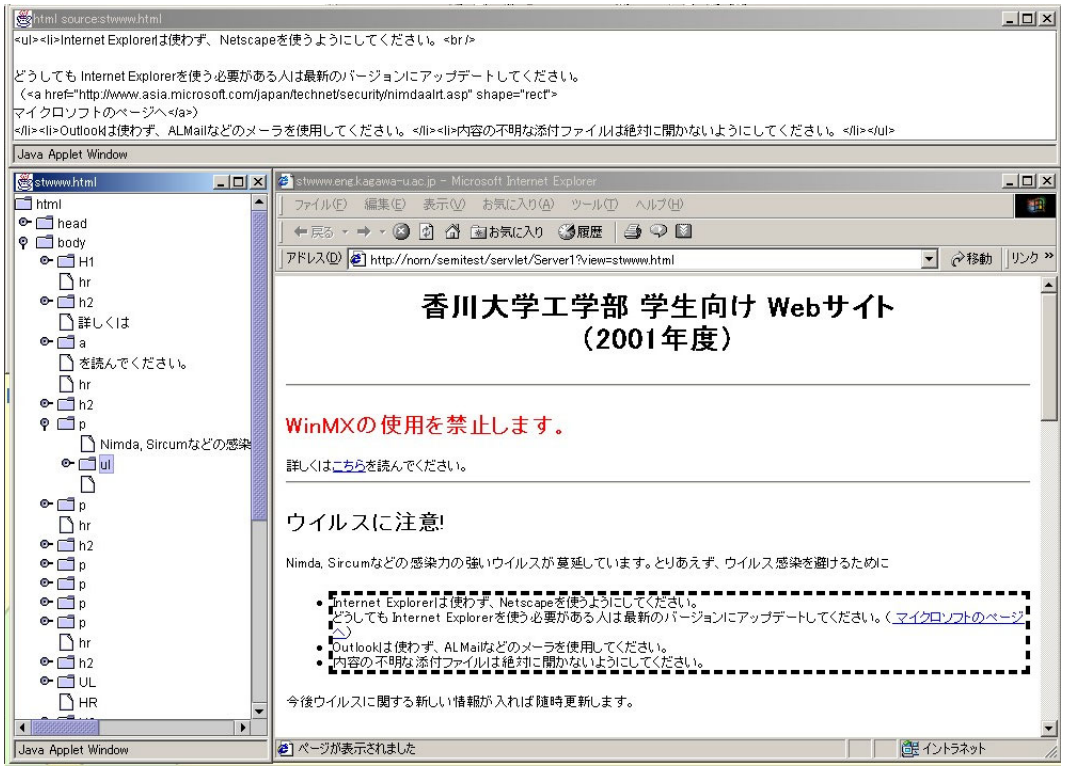


**図 6**　編集画面

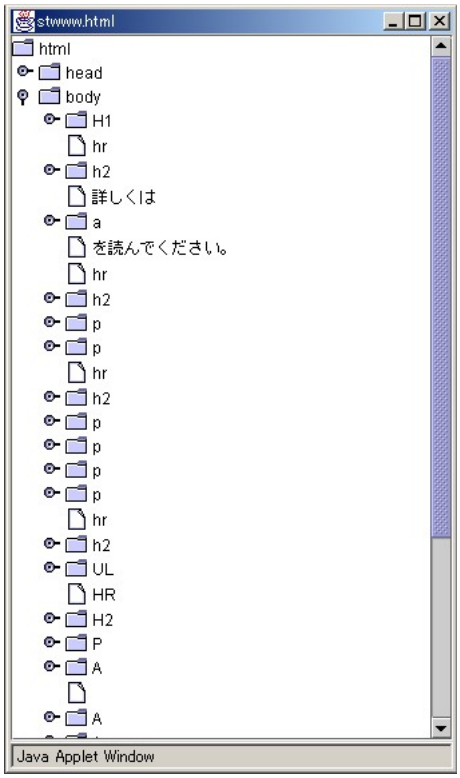


**図 4** クライアント

のソースをクライアントに送信する。クライアントがそれを表示し編集が可能となる。

(7) クライアントから更新部分をサーバに送りファイルを更新する。

(8) サーバからファイル更新の通知を受け取り、ブラウザの表示を更新する。

## 6. 考　　察

### 6.1 エディタ機能

現在本システムではエディタ部分に Java Swing 標準のテキスト入力 GUI を使用している。しかし、これは XHTML のエディタとしては非力である。そこでクライアント側で XHTML ファイルをツリー表示していることに注目し、ここで GUI を用いた直感的な操作を行うことができれば直接テキストを編集する必要性が減少し、エディタとしての機能を補えると考えられる。

### 6.2 セキュリティ

本システムはサーバのファイルへの読み込み、書き込みを行うが、全てのファイルに全てのユーザがアクセスしてよいとは限らない。しかし、本システムでは Java Plug-in と Java Servlet を用いることにより、全ての通信を HTTP(および HTTP のトンネリング) で行っている。これは既存のウェブにおけるセキュリティシステムをそのまま適用できることを意味する。つまり Basic 認証や HTTPS による通信の暗号化、SSL のクライアント認証などを容易におこなうことができ、安全にサーバへのアクセスを行うことができる。また、既存のネットワーク構成に大きな影響を与えることも無い。

### 6.3 ロック制御

現在本システムでは全てのタグでのロックを行える用に作成している。しかし、たとえば table タグ中では 1 列をロックしたいといった要求も考えられる。また、スプレッドシートのような入力フィールドを用いた方が編集の効率が上がると思われる。このようにタグに個有の編集、管理機能の追加も考えられる。

### 6.4 共同作業時の問題点

共同での作業では、他のユーザとのコミュニケーションや、アウェアネス情報が重要になってくる。これらの機能の検討を行う必要性もある。

## 7. おわりに

本システムではファイルの一部分のみをロックすることにより、編集不能状態や更新の衝突を避けることができた。これにより複数人による同時編集を容易に行うことができる。しかし、複数人での編集にはバージョン管理機能が必要不可欠である。しかし、現在本システムにはこの機能が無い。これは本システムの最も大きな課題の 1 つである。

## 参　考　文　献

1) Extensible Markup Language(XML)
http://www.w3c.org/XML/
2) W3C HTML Home Page
http://www.w3c.org/MarkUp/
3) 佐藤 友章、杉山 安洋ソフトウェアの共同開発作業における細粒度ロックの有効性の検証
日本ソフトウェア科学大会第 19 回大会論文集