デジタル・ドキュメント 34－2
（２００２．７．２６）

多チャネルコンテンツ配信における マルチテンプレート管理システムの開発

石田 和生　　矢野尾 一男　　小川 隆一

NEC インターネットシステム研究所

ブロードバンドインターネットや携帯電話の普及によりインターネットに接続する端末の多様化が進んでいる。また、放送のデジタル化に伴いインターネットと放送の連動サービスも現実化しつつある。しかし、これらの実現のためにはひとつのコンテンツを複数のチャネル向けに変更して配信することが必須であり、全てのコンテンツを手作業で変更していたのではコストがかかりすぎるという問題点がある。これに対処するため筆者らは、多チャネル対応コンテンツ変換サーバの検討を進めている。本報告では、本変換サーバの概要について説明し、特に、「ワンソースマルチユース」と呼ばれるXMLベースの一括生成方式で使用されるテンプレートの管理コストを削減するためのマルチテンプレート管理システムの動作と試作システムについて詳述する。

Development of Multi Template Management System for Multi-channel Contents

Kazuo ISHIDA, Kazuo YANOO, and Ryuichi OGAWA

Internet System Research Laboratories, NEC Corporation

As the broadband Internet and mobile phones come into wide use, types of terminals attached the Internet increase. And digital broadcast services will make cross-media services integrating the Internet and broadcasting possible. But it is necessary to convert a content to suitable one for each channel for realizing such services, and converting by hand costs a lot. To cope with this, we develop a multi-channel content authoring method. In this report, we describe an outline of our authoring method. Especially features of a multi-template manager cutting down cost to manage templates used in “one source multi-use” system based on XML, and details of an experimental system, are described.

1. はじめに

ブロードバンドインターネットの普及や携帯電話のようなモバイル機器の普及にともない、インターネットに接続する端末の多様化が進んでいる。この結果、WWW を閲覧するブラウザ環境も出力デバイスの持つ解像度や色数などは様々なものとなっている。このような状況でコンテンツを公開する場合、特定の端末環境を想定してコンテンツを作成すると、想定されていない端末環境下で不都合が生じることがある。例えば、高解像度のデスクトップ PC での閲覧を前提として表示に高い解像度を必要とする画像や表を含んだコンテンツを作成すると、解像度の低いモバイル端末や画像を表示出来ない携帯電話では画像や表を正しく表示出来ない。しかし、閲覧で利用される可能性のある端末のブラウザ全てに適合したコンテンツを個別に作成しておくことは、コストを増大させることになり現実的ではない。

コンテンツの記述言語についても多様化が進んでいる。現在、インターネット上での情報発信には HTML[1]が主に使われている。本来 HTML は文書の論理構造を記述することを目的に設計された言語であるが、広く一般に普及するに従い、見た目(スタイル)の記述能力に重点がおかれるようになった。その結果、HTML はブラウザ毎にスタイル記述の固有拡張が進められ、Internet Explorer や Netscape Navigator 専用の WWW コンテンツ、あるいは、ブラウザの種類を見て動作を切り替える WWW コンテンツが作られるようになっている。こういった、HTML の非互換性については、W3C によって一応の統一がはかられたが、依然としてブラウザ毎で動きの違う部分が残存しており、コンテンツ側での対処が必要である。また、コンテンツの表現能力に対する要望もとどまるところを知らず、動きのあるコンテンツを作成するために JavaScript のようなスクリプト言語や、時間推移を含むコンテンツを記述するための SMIL[2]といった新たな規格が作成されている。さらに、i モード対応 HTML(以下では CHTML と記述する)[3]や BS デジタル放送用の BML[4]など、ブラウザ端末固有の記述言語も次々と作成されており、複数のプラットフォームでコンテンツを提供するためにはそれぞれの環境毎にコンテンツを作成しなければならなくなっている。

端末や記述言語がまだそれほど多様化していない時点では、利用端末の種類を判別してコンテンツ側でスタイルなどを変更して対応出来たが、多様化が進むにつれ個別に対処することが困難となってきた。そこで、OpenTV の携帯端末向けコンテンツ変換サーバ[5]や、Oracle の Portal-to-GO[6]などのような「ワンソースマルチユース」方式が利用されはじめている。これらの方式は、ひとつのコンテンツ、あるいは、ソースデータを加工して、各種端末、記述言語用のコンテンツデータを生成するもので、理想的には、元データはひとつで良いためコンテンツ作成のコストを大幅に抑えることが出来る。しかし多くの場合ソースデータの作成、管理だけでは不十分で、ソースデータを変換、加工するためのテンプレート(変換スクリプトやスタイルシートなど)を別途作成、管理する必要があり、これに要するコストを低減させることが重要となってくる。

これらに関して筆者らは、放送やモバイルを含めた複数のチャネルにコンテンツを流通させるための多チャネル対応コンテンツ変換サーバの検討を行っている[7][8]。以下では、本変換サーバの概要について説明し、その中で特に、ワンソースマルチユース方式のテンプレート管理コストを削減するためのマルチテンプレート管理システムについて、具体的手法と試作したシステムを詳細に説明する。

2. 多チャネル対応コンテンツ変換サーバ

2.1 変換サーバの全体構成

放送のデジタル化技術やモバイルインターネット技術の発展にともない、同一のデータ

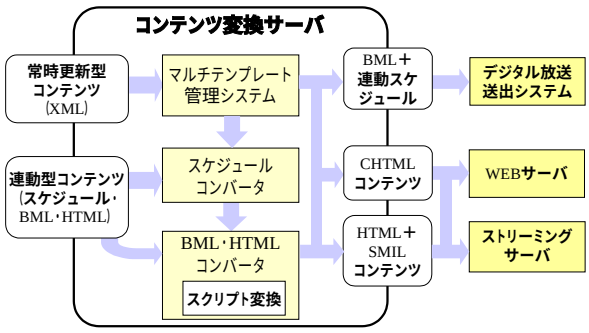

図 1　多チャネル対応コンテンツ変換サーバ

コンテンツを複数のチャネルで配信することが現実化しつつある。例えば、HTML で記述される EC コンテンツを、CHTML で記述される携帯電話向けコンテンツ、BML で記述されるデータ放送向けコンテンツに変換・配信するといったものである。しかし、MPEG2 映像の MPEG4 変換のようなシンプルなケースを除き、多チャネル化に対応した効率的なコンテンツ制作方式は確立しておらず、制作コストが多チャネル化の大きなネックとなりうる。そこで筆者らは、図 1に示すような、放送・ブロードバンド・モバイルインターネットなどの各チャネルに対し、コンテンツを連動配信するためのコンテンツ変換サーバの検討を行っている[7][8]。本変換サーバは大きく分けて、常時更新型コンテンツを入力とする一括生成と連動型コンテンツを入力とする逐次変換の 2 方式から構成されている。

2.2 テンプレートによる一括生成

ニュース・株価など、定型的な更新を常時必要とするタイプのコンテンツに有効な方式である。各チャネルに対する更新を一括して行うために、更新情報は XML[9]形式で一元管理し、チャネル別のテンプレートを XSLT[10]形式で個別に用意する手法(ワンソースマルチユース方式の一形態)が知られている(図 2)。そして運用においては、XML データを逐次更新し、特定のタイミングで XSLT を用いて各チャネル用のコンテンツデータを生成する。

本方式の課題としては、各チャネルのブラウザ仕様に合わせたテンプレートの作成とメンテナンスがある。XSLT の記法自体が難解であるのに加え、各チャネルのテンプレート間の一貫性を保ちながら修正を加えなければならない。要求されるスキルは通常のコンテンツ作成者の能力をこえるため、なんらかの技術サポートが必須である。この問題に対し

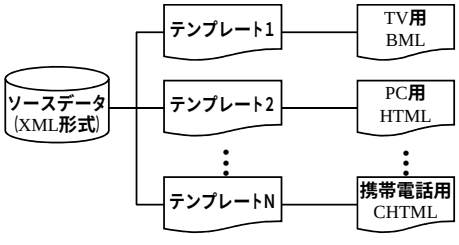

図 2　XML ベースの一括生成

筆者らは、複数テンプレート間の一貫性を保ちながら修正・メンテナンスを行うマルチテンプレート管理システムの研究開発を行っている。これについては第 3 章で詳細に説明する。

2.3 個別コンテンツの逐次変換

2003 年に予定される地上波デジタル放送では、放送とインターネットの連動サービスが本格化するであろう。このような連動型コンテンツを放送・インターネット双方で同時に制作する技術や制作体制はまだ確立していない。現時点でもっとも妥当な制作プロセスは、まず放送向けコンテンツを一定の完成度まで作成し、それをインターネット向けに変換して最終形を完成させる、あるいは、その逆を行う、といった逐次型のプロセスであろう。すなわち、HTML コンテンツ・BML コンテンツの間の相互変換技術が必須になる。これらに関し筆者らは BML で多用されるスクリプトまで含めた HTML 変換をサポートする BML-HTML 変換技術[11]と、番組スケジュールを SMIL 形式に変換するスケジュール変換技術[12]の研究開発を行っている。このような逐次変換手法では、個々のコンテンツは HTML や BML といったコンテンツ作成者が普段コンテンツ作成で使用している言語をそのまま用いることが出来るため、コンテンツ作成に関する障壁が増加しないという特徴がある。詳細については参考文献を参照のこと。

次章では、以上で説明したコンテンツ変換サーバを構成する要素のひとつであるマルチテンプレート管理システムの管理方式について説明する。

3. マルチテンプレート管理方式

3.1 ワンソースマルチユースの問題点

ワンソースマルチユース方式では個々のブラウザ環境に適応したテンプレート(スタイルシート)や変換コンポーネントを予め用意しておき、ひとつのソースデータを、利用するブラウザに対応するテンプレートや変換コンポーネントで変換することで各ブラウザ環境用のコンテンツデータを出力する(図 2)。しかし、個々の環境毎にテンプレートや変換コンポーネントを作成、管理する必要があるため、画面レイアウトの変更や表示要素の増減によりテンプレートを修正する必要が生じた時に

は、管理している全てのテンプレートを個別に変更しなければならない。このため、変更作業に要するコストは依然として少なくならない。また、複数のテンプレートに対して個別に修正を加えるため、あるテンプレートだけ修正を加えるのを忘れる、あるいは、修正内容を間違えてしまうといった人為的ミスが入り込む余地が大きいという問題点もある。

これに対し、全ての環境を一括して管理するような高機能テンプレートを作成することも考えられるが、テンプレート全体の見通しが悪くなるのと、全ての環境へ及ぼす影響を考慮しつつテンプレートに修正を加えていかなければならないので、やはり大きな変更コストが必要となる。さらに、このようなテンプレートを設計するためには通常、予め対象とする端末環境を想定しておかなければならないため、新たな環境への適応が必要となった場合には、最悪テンプレート全体の再設計を行わなければならなくなり、非常に大きなコストが発生してしまう。

筆者らは、このような問題を解決するための手法として、ワンソースマルチユース方式で利用される複数のテンプレートを一括して変更可能なマルチテンプレート管理方式を提案している[8]。本方式では、個々のテンプレートは環境毎に別々に存在しているため、ある環境用のテンプレートを修正する場合に他の環境への影響を考えながらテンプレートを修正する必要はなく、かつ、ひとつのテンプレートを修正するだけで他の環境用のテンプレートも自動的に修正されるため、テンプレート修正コストの削減も実現される。次節で、本管理方式の手法について説明する。

3.2 マルチテンプレート管理方式

前節で述べたような問題点を解決するための手法として、テンプレートを一括して変更出来る管理方式を提案する。すなわち、あるひとつのテンプレートを修正するとそこから修正部分を抽出して、他のテンプレートにもその修正部分を自動的に反映させる。このようにすると、個々のテンプレートを個別に修正する必要がないので修正コストの削減が出来、かつ、自動で修正部分の反映を行うので修正洩れや修正ミスが発生する可能性も低く出来るという利点がある。

このような管理方式を実現するために必要な機能としては

- テンプレートの変更点の抽出
- テンプレート内での反映場所の探索
- テンプレートへの変更内容の適用

の 3 つが挙げられる。以下で、上記 3 つの内容と、それぞれを実行するために必要なツリー変換の手法について順番に説明する。なお、本報告書では主に XML と XSLT を用いたマルチテンプレート管理システムを対象に説明を行うものとする。

3.2.1 ツリー変換

あるひとつのテンプレートが変更された時に全てのテンプレートを一括修正するためには、まず、その変更内容を抽出する必要がある。このための方法としては大きく分けて、変更前後のテンプレートを直接比較する方法と、別形式に変換して比較する方法の 2 つが考えられる。

テンプレートを直接比較する方法は、例えばファイルの差分生成に使われる diff コマンドのように、変更前後のテンプレートから共通する部分を削除して、残った部分を変更内容として抽出する。ただし、単純にテンプレート同士を比較してしまうと形式的な相違点(テンプレート上、意味のない空白の数など)も変更内容として抽出されてしまうという問題がある。特に、XML ベースの XSLT で記述されたテンプレートなどでは多くの空白と改行は自由に含めることが可能であるし、要素の属性なども順序に非依存であるため、文献[13]で述べられているような正規化を行ってから比較をする必要がある。

一方、別形式に変換してから比較する方法は、対象となるテンプレートから特徴的な情報(ここではキー情報と呼ぶ)を抜き出して別の形式に変換し、変換結果同士で比較を行うものである。これは例えば、テンプレートフォーマットのバージョンの違いやコメントの変更など、あまり本質的でない部分の変更に影響されずに比較したい時などに有効である。本管理方式では、変更点抽出のロバスト性を高めることと、3.2.3 項で説明する反映場所の探索での処理を考慮して後者の別形式に変換する方法を選択した。具体的には、テンプレートからキー情報を抽出してツリー構造へと変換する。ここでキー情報としては、テンプレートに入力するソースデータと出力されるコンテンツデータに強く依存した情報である

```
・・・
<link>
<description>これは1番目のリンクです</description>
<data>
<target>http://www.sample.ne.jp/</target>
<name>リンクサンプル1</name>
</data>
</link>
・・・
```

(a) ソースデータ

```
・・・
<xsl:template match="link">
 <P><xsl:value-of select="description"/><BR/>
  <A>
   <xsl:attribute name="HREF">
    <xsl:value-of select="data/target"/>
   </xsl:attribute>
   <xsl:value-of select="data/name"/>
  </A>
 </P>
</xsl:template>
・・・
```

(b) テンプレート

図 3　ソースデータとテンプレートの例

- ソースデータに含まれるタグ名、属性名
- 出力データに含まれるタグ名、属性名

を選択することとした。例えば、図 3 (a), (b)に示されるような XML 形式のソースデータと HTML 形式のコンテンツデータを生成する XSLT 形式のテンプレートがあったとする。このときのキー情報としては

- ソースデータに関係: link, description など
- 出力データに関係: P, A, HREF など

があげられる。

ツリーへの変換は、基本的に、元のテンプレート上での親子、兄弟関係と矛盾しないようにキー情報をツリーに構成することで行う。すなわち、図 3 (b)のテンプレート中で、キー情報 HREF を含むタグがキー情報 A を

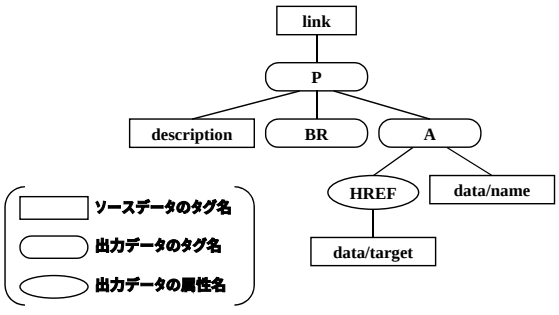

図 4　ツリー変換結果

```
・・・
<link>
<description>これは1番目のリンクです</description>
<data>
<target>http://www.sample.ne.jp/</target>
<name>リンクサンプル1</name>
<image>/img/sample.png</image>
</data>
</link>
・・・
```

図 5　変更されたソースデータ

含むタグの子供になっていることから、キー情報 HREF をキー情報 A の子供としてツリーを構成する。これを繰り返すと図 3 (b)のテンプレートは最終的に図 4に示されるようなツリーに変換される。

以上の説明で、キー情報にテンプレート固有の情報(xsl:value-of などのタグ名)を含めていないのは、変換されたツリーがテンプレートの表現形式になるべく依存しないようにするためである。このようにすることで、例えば、XSLT と JSP のように形式の全くことなるテンプレートが混在していても統一して扱うことが容易となるというメリットが生まれる。この詳細については3.2.3 項で説明する。

3.2.2 変更点の抽出

テンプレートの変更点の抽出は変更前後のテンプレートそれぞれから変換したツリーをもとに行う。例を用いて説明する。

図 3に示すようなソースデータとテンプレートがあったとき、ソースデータに新たにイメージ情報が追加された場合を考える(図 5の点線部分)。このイメージ情報を利用するために図 3 (b)のテンプレートを図 6のように変更したとする。この場合の変更点の抽出手順は次のようになる。

1. 変更前後のテンプレートをツリーに変換(図 4, 図 7)
2. 変換されたツリーで、ルートノードが同じツリー同士からノード数が最大の共通部分木を探索する(図 7の実線部分)
3. ツリーから共通部分木に含まれるノードを削除して残ったノードから構成されるツリーを変更点として抽出(図 7の点線部分)

共通部分木の削除により、図 4上に残ったツリー(上記の例では該当する部分なし)は、

```
・・・
<xsl:template match="link">
 <P><xsl:value-of select="description"/><BR/>
  <A>
   <xsl:attribute name="HREF">
    <xsl:value-of select="data/target"/>
   </xsl:attribute>
   <xsl:value-of select="data/name"/>
   <IMG>
    <xsl:attribute name="SRC">
     <xsl:value-of select="data/image"/>
    </xsl:attribute>
   </IMG>
  </A>
 </P>
</xsl:template>
・・・
```

図 6　変更されたテンプレート

もとのテンプレートには含まれていたが今回の修正により削除された部分、という意味を持ち、図 7上に残ったツリー(図 7の点線部分)は、今回の修正により新たに追加された部分、という意味を持つ。以下では説明のため、それぞれを削除ツリー、追加ツリーと呼ぶこととする。

3.2.3 反映場所の探索

変更内容が抽出出来ると、次はこの変更を反映させたい管理対象のテンプレートに対して、変更内容を反映させるべき場所の探索を行う。この探索も、前項と同様、テンプレートを直接探索対象とする方法と、変換した別形式を探索対象とする方法の 2 つが考えられる。しかし、テンプレートを直接探索する方法は、管理している全てのテンプレートの形式が同一である場合には問題ないが、形式が異なる(例えば、XSLT と JSP など)テンプレートが混在している場合には、両者のテンプレートの対応をとることが難しく、探索は容易ではなくなる。そこで、本管理方式では、

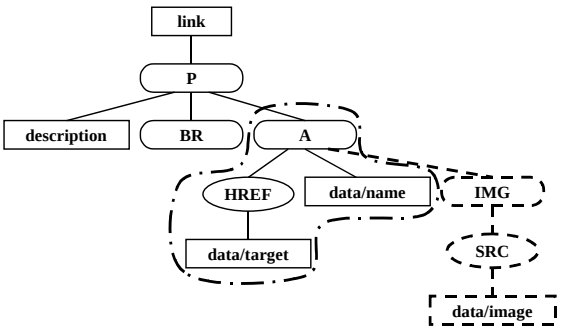

図 7　変更後テンプレートのツリー変換結果

```
・・・
<xsl:template match="link">
 <A>
  <xsl:attribute name="HREF">
   <xsl:value-of select="data/target"/>
  </xsl:attribute>
  <xsl:value-of select="data/name"/>
 </A>
 <BR/>
</xsl:template>
・・・
```

図 8　変更反映対象テンプレート

前項で変換したツリーをベースにして探索を行う。このようにすれば、全ての探索はツリー上で行われるので、形式の異なるテンプレートが混在していても対応することが可能となる。

反映場所の探索方法を前項で用いた例を使って説明する。テンプレートが図 6のように変更された時に、その変更を反映したいテンプレート(例を図 8に示す)内を探索し、変更を反映させるべき場所を決定する。手順は次の通り。

1. 変更反映対象のテンプレートをツリーに変換(図 9)
2. 変更点として抽出された削除ツリー、追加ツリーの親をルートとする部分木を抽出。ただし追加ツリー自身は除く(図 7の一点鎖線部分)
3. 抽出された部分木を変更反映対象のツリー内で探索(図 9の点線部分)

このようにして探索された部分木が変更点を反映させるための基準の場所になる。

3.2.4 変更内容の適用

変更点の抽出と反映場所の探索完了後は、変更反映対象となるテンプレートの修正を行う。まず変換されたツリーの変更を行うが、削除ツリー、追加ツリーの抽出と変更を反映

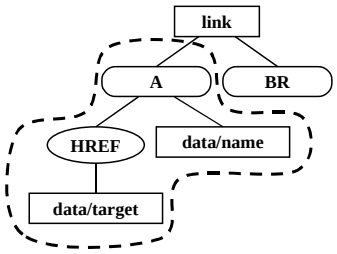

図 9　変更反映対象のツリー変換結果

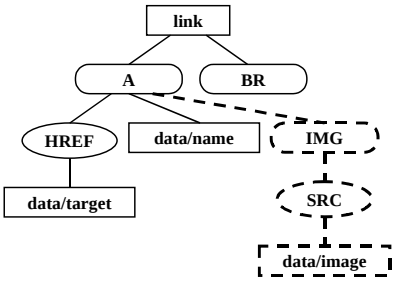

図 10　ツリー追加結果

させる基準の部分木が前項までで説明した手順で得られているので、

- 削除ツリーの場合: 基準の部分木から削除ツリーに相当するノードを削除
- 追加ツリーの場合: 基準の部分木のルートノードに追加ツリーを追加

という手順で実現出来る。前項までで用いている例でいうと、追加ツリーが図 10の点線で示されるような形で追加される。その後、3.2.1 項で説明したツリー変換と逆の手順により、ツリーをテンプレート形式に変換すれば最終的に目的の、修正されたテンプレートを得ることが出来る(図 11)。

以上をまとめると、図 3 (b)のテンプレートを図 6のように変更した時、その変更点を自動的に抽出し、図 8のテンプレートは図 11のように変更される。同様の処理を管理対象のテンプレート全てに対して実行すれば、管理している複数のテンプレートのあるひとつだけを変更することで全てのテンプレートへ同時にその変更を反映させることが出来、テンプレート修正にかかるコストの大幅な削減と修正洩れなどのミスの発生を防ぐことが可能となる。

```
・・・
<xsl:template match="link">
 <A>
  <xsl:attribute name="HREF">
   <xsl:value-of select="data/target"/>
  </xsl:attribute>
  <xsl:value-of select="data/name"/>
  <IMG>
   <xsl:attribute name="SRC">
    <xsl:value-of select="data/image"/>
   </xsl:attribute>
  </IMG>
 </A>
 <BR/>
</xsl:template>
・・・
```

図 11　自動変更後のテンプレート

4. 試作システムの構成と実行例

前章で説明した管理方式に基づきテンプレートを管理するマルチテンプレート管理システムの試作を行った。システム構成を図 12に示す。本管理システムは将来的には、複数の種類のテンプレート記述言語を同時に扱えることを目標としているが、まずは、XML ベースのワンソースマルチユース方式で利用される可能性が高い XSLT 形式のテンプレートのみを対象としている。

本試作システムの実行例を図 13と図 14に示す。図 13と図 14はクイズ番組風のコンテンツで、それぞれ PC 用の画面と携帯電話用の画面をイメージしたものになっている。クイズの問題などの基本データは XML 形式の共通データとして保持しておき、PC 用と携帯電話用の XSLT を用いて各画面を生成するようになっている。ここで図 13に示されているように、第 5 問の下に第 6 問を追加するため PC 用の XSLT を変更したとする。このとき本システムを用いてテンプレートを管理していれば、PC 用 XSLT の変更点を自動的に抽出して携帯電話用 XSLT にも同様の変更内容が反映され、最終的に図 14に示されるような変更結果が得られる。ただしこの例では、PC 用 XSLT で追加されているのはイメージで、携帯電話用 XSLT で追加されているのはテキストとなっているが、これは「PC 用 XSLT の IMG タグを携帯電話用 XSLT に持ってくる際には SPAN タグに置き換える」といった特殊ルールをシステムに組み入れる

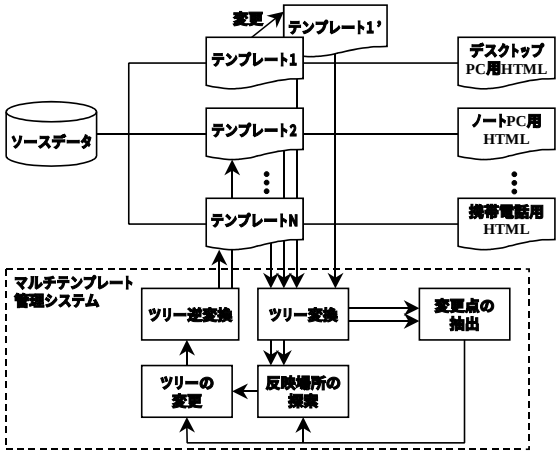

図 12　システム構成

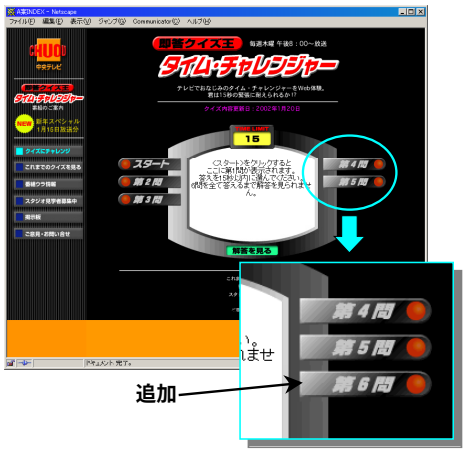

図 13　実験コンテンツ(PC 向け)

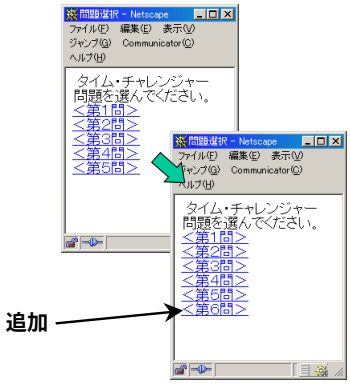

図 14　実験コンテンツ(携帯電話向け)

ことで実現されている。

5. まとめ

本報告では、放送のデジタル化やモバイル環境の発達で増加するコンテンツ配信用チャネルに対応するため、デジタル放送やモバイルを含めた多チャネル対応コンテンツ変換サーバの概要を述べ、特に複数のテンプレートの編集、管理を行うマルチテンプレート管理システムの詳細について述べた。本マルチテンプレート管理システムは、ワンソースマルチユース方式の問題点であるテンプレートの管理、修正コストを削減するもので、複数のテンプレートを一括管理し、テンプレートに修正が必要となった場合にはあるひとつのテンプレートを変更するだけで管理対象全てのテンプレートにその変更内容が反映される。これにより、修正コストの削減が可能になる他、システムで一括して全てのテンプレートを変更するため、テンプレートの修正洩れや修正ミスが発生する可能性を低くすることも可能となる。

以上で述べたマルチテンプレート管理システムと、文献[11][12]記載の BML-HTML 変換技術とスケジュール変換技術の 3 つの要素が密に連携することで、ひとつのコンテンツの多チャネル配信が実現される。今後は、各要素の評価、改良を進めるとともに、全ての要素を連携させた多チャネルコンテンツ変換サーバの構築に向け検討を行う予定である。

参考文献

[1] W3C, HyperText Markup Language Home Page, http://www.w3.org/MarkUp/, 2001.

[2] W3C, Synchronized Multimedia Home Page, http://www.w3.org/AudioVideo/, 2001.

[3] NTT ドコモ, i モード対応 HTML Version 1.0・2.0・3.0, http://www.nttdocomo.co.jp/mc-user/i/tag/index.html, 2001.

[4] 電波産業会, デジタル放送におけるデータ放送符合化方式と伝送方式, ARIB STD-B24 2.0 版, 2001.

[5] OpenTV, Spyglass Prism, http://www.opentv.com/support/ed_services/spyglass_prism.html, 2000.

[6] Oracle, Portal-to-Go, http://www.oracle.co.jp/ptg/top.html, 2000.

[7] 小川隆一, 矢野尾一男, 田口大悟, 石田和生, 多チャネル対応コンテンツ生成方式の開発 (1) ―コンテンツ変換サーバのアーキテクチャー, 第 64 回情処全大, 6Z-02, 2002.

[8] 石田和生, 矢野尾一男, 小川隆一, 多チャネル対応コンテンツ生成方式の開発 (2) ―マルチテンプレート管理システム―, 第 64 回情処全大, 6Z-03, 2002.

[9] Tim Bray et al., Extensible Markup Language (XML) 1.0 (Second Edition), http://www.w3.org/TR/REC-xml, 2000.

[10] James Clark, XSL Transformations (XSLT) Version 1.0, http://www.w3.org/TR/xslt, 1999.

[11] 矢野尾一男, 小川隆一, データ放送コンテンツの携帯端末向け配信サーバ, 第 63 回情処全大, 5V-04, 2001.

[12] 田口大悟, 矢野尾一男, 小川隆一, データ放送番組記述方式とその編集方法, 情処研報 Vol. 2001, DD-28-3, 2001.

[13] John Boyer, Canonical XML Version 1.0, http://www.w3.org/TR/xml-c14n, 2001.