グループや役割を考慮したXSLTに基づく共有文書のカスタマイズ機構

Role-based Customizing Functions for Shared Documents using XSLT

村上隆治† 谷垣美沙子††
横田裕介†† 上林彌彦††

組織の中での共有文書も使用される作業や使用者の役割に応じて手を加えて利用したい。こうした要望に応えるため、環境ごとに共有文書に対して個別にコメントの追加などのカスタマイズを行え、またこれらの環境を階層構造にすることで更に同じ作業グループ内での役割に応じたカスタマイズにも対応できるような、環境モデルを用いた協調作業支援システムをこれまでに提案してきた。本稿では現在進めている XML に基づくこのシステムの構築について述べる。本システムでは共有文書として XML、XHTML を想定しており、XHTML 形式の WWW 上の文書も利用できる。またカスタマイズは多段階の変換に対応できる形で XSLT を用い、その内容は XSL 文書で管理される。システム内の環境やユーザの情報も XML 文書で管理し、この中で環境の構造や XSL 文書の管理、ユーザの各環境内での権限の管理などを行う。

1. はじめに

近年の計算機ネットワークの発展にはめざましいものがあり、空間的に広い範囲に分散した人同士が共同で作業を行うといった作業形態は決して珍しいものではない情勢となってきている。また計算機そのものの普及もここ数年でかなり進んでおり、多くの業務で計算機が使用され、各人が計算機を使用した作業を行っている。このような情勢の中では遂行される業務に関係する人員も多数に上る。こうした人たちは業務の内容や規模に応じていくつかのグループを構成し、そうしたグループ同士が共同して作業を分担して行うといった作業形式が多くとられることになる。このような分散協調作業に対しては計算機による作業支援システムが業務の遂行に際して重要な問題となりうる。この場合の支援システムでは共有資料をどのように管理及び利用するか、利用者の作業状態をどう管理するかといったことが重要な問題と考えられる。

このような分散協調作業を支援するものとしては利用者の所属する作業空間を定義し、各作業空間に対して共有資料を提供するシステムが多く研究されている。しかし利用者にはそれぞれの役割に応じて資料にコメントを追加するなどしてより使いやすい形で利用したいという要望があり、これに応えるため、我々はこれまでに VIEW Media という協調作業支援システムを設計し開発を行ってきた。本システムでは作業空間として環境モデルを導入している。環境ごとに共有資料のビューを利用者の役割に応じてカスタマイズできる。また環境を階層構造とすることにより、作業グループでカスタマイズされたビューに対してさらに利用者独自のカスタマイズを加えるといったことが可能になる。現実の利用者の立場や役割も階層構造に決められていることが多く、VIEW Media システムのこの階層構造の環境構成にはこれを反映させ、利用者の環境へのアクセス権限を設定することができる。このような環境構成にすることで、利用者個人向けのビューに作業グループでのカスタマイズの内容を反映させ利用することが可能となる。

これまでなされてきた VIEW Media の開発、実装はシステムの各要素をオブジェクトとして定義し, Java 上で HORB を用いて行われてきた。そこでは環境や共有資料もオブジェクトとして管理される。これに対して本研究ではシステムのもつ共有資料や環境の管理する情報に永続性を持つ文書として管理し、またこうした文書の形式やその変換方法に一般性を持たせるため、共有資料及び環境の管理情報の形式として XML を導入する。共有資料を XML 文書としたことによって文書のカスタマイズ操作の実現に XSLT を利用することができる。本稿では XML に基づいて本システムの構築を行うにあたり、共有資料をグループや役割に応じてカスタマイズするための機構について述べる。

† 京都大学工学部
†† 京都大学大学院情報学研究科

2. 背　　景

2.1 関 連 研 究

システムの利用者に対して作業空間を定義して共有資料や利用者の権限の管理を行い、協調作業を支援するシステムは数多く研究されている。

CBE[4)5)] や Orbit[6)7)] はそれぞれ room や locale の概念を用いた作業空間を用いた協調作業支援システムである。それぞれの作業空間ごとに作業に用いるデータやコミュニケーション機能が提供される。これらのシステムでは利用者が必要な資料は所属する作業空間の組み合わせにより得られる。こうしたシステムでは各文書ごとの変更権限などの細かな権限管理が難しい。またそれぞれの作業空間の関連性を扱う機能が乏しく、利用者の役割や立場に対応した柔軟なビューを提供するためには機能が不十分である。

一方 Suite[12)] では 2 次元配列の各行各列に共有資料とユーザを対応させ詳細な権限の設定、管理が可能である。しかしその反面、複雑な設定が必要となりその効率化のためのオブジェクト管理の階層化において多重継承による矛盾などの問題を抱えている。

また Digestor[8)] や WebSplitter[10)] などのように利用者に応じて表示を変化させるものもある。しかし利用者の使用するデバイスの多様化に対応し、それに合わせて表示を変えるためのものが多い。そのため、利用者の細かな役割の違い等に柔軟に対応し、協調作業に用いるためには不十分だと言える。

2.2 VIEW Media の概要

本節では本研究の基盤となる協調作業支援システム VIEW Media[1)2)] について述べる。作業を行う人にはそれぞれ別の役割が存在し、その役割に応じて参照したい資料が異なる。また作業の内容に応じて資料に手を加えて利用したいこともある。この考えのもと本システムでは、利用者それぞれの役割に応じて異なった資料のビューを提供する。VIEW Media では作業空間の構成に環境の概念を導入することでこれに対応する。

2.2.1 環　　境

本システムでは協調作業空間を構成するために環境モデルを導入している。これらの環境ごとに参照できる共有資料やその環境における利用者の権限が定められる。こうした環境それぞれにおいて参照される資料のビューを考えると、どの資料に対しても同一の環境にアクセスする利用者には同じカスタマイズが施されたビューを提供することが保証される。ただし利用者がそれぞれどの文書のどの部分を参照するかについては自由である。

システムで定義されたこれらの環境は階層構造を持っており、全体として木構造に構成される。ある環境での共有資料のビューの内容は基本的にその環境の子環境でも利用できる。子孫の環境でのビューはこれに順次環境ごとのカスタマイズを施して生成されるので、子孫となる環境にはその内容が反映される。

2.2.2 資料の共有とそのカスタマイズ

協調作業を行う利用者にはそれぞれ、作業の内容や役割、立場などに違いが存在する。こうした利用者の役割の違いは組織内で共有された資料をどう利用するかといった点にも違いを生じる。この問題への対応として本システムでは利用者の役割や立場を環境として表している。そして共有資料のビューに対してはこの環境ごとにカスタマイズを行い、その環境独自のビューを利用者に提供することで対応している。

しかし協調作業を行う利用者の間には作業内容に関連がある、共通の立場を持つなどといった関係が存在する。こうした利用者間の立場や役割に応じて環境にも関連をもたせることが必要である。このため本システムでは環境に階層構造を持たせ、各環境は木構造を構成する。ある環境で共有文書を参照すると、その環境の先祖にあたる環境で行われたカスタマイズ操作を順次適用し、さらにその環境独自のカスタマイズを加えてその環境独自のビューを生成し、利用者に提供される。こうしたメカニズムを利用するためには、各利用者独自のカスタマイズを行う環境をその利用者の所属するグループの環境の子環境として構成する。これによってそのグループ内部での資料へのカスタマイズ内容を保持したまま、各利用者が独自に柔軟なカスタマイズを加えることができる。

図 1 に利用者の立場に応じて資料をカスタマイズした例を示す。これは会議を行う場合を想定したものである。会議の参加者で共通に利用される資料は図 1(a) に示したように用意されているとする。しかしこの会議で何らかの発言を行う予定の者に対しては、図 1(b) のように発言時に参考にする情報をその資料に付けるといったことを可能にする。同様に図 1(c) のように会議の議長が議事進行に役立つメモを付けるといったことも考えられる。こうしたカスタマイズは環境を図 2 のように構成することによって実現することが可能である。この場合には利用者 U1 と利用者 U2 には資料そのままのビューを提供する。利用者 U3 や利用者 U4 はそれぞれの環境においてこの共通環境のビューに対するカスタマイズを行うことが可能である。

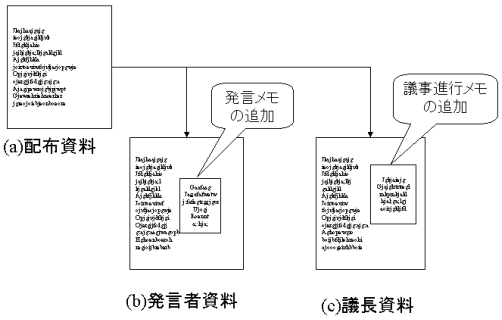

図 1 会議における資料のカスタマイズ

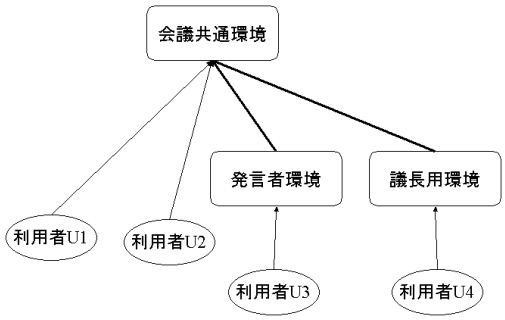

図 2 会議における環境構成例

2.2.3 利用者の権限管理

利用者の権限には各環境へのアクセス権限、文書のカスタマイズ権限、環境設定の変更権限がある。環境へのアクセス権限は、その環境で参照可能な共有文書に対しての環境独自のビューを参照できる権限である。各利用者の役割や立場に応じてアクセス可能な環境を設定することにより、文書のセキュリティー保持を図ることが可能である。

文書のカスタマイズ権限は参照する文書のビューにその環境独自のカスタマイズを行う、あるいはそのカスタマイズの変更を行う権限である。その環境にアクセスする利用者に応じて、文書のビューにコメントやリンクを追加するなどのカスタマイズを行うことができる。各環境で行われたそれぞれのカスタマイズ操作は対象となる共有文書自体を変更するものではなく、カスタマイズの内容は各環境に付属するものとして管理される。これによって文書のカスタマイズ操作による情報の変更は、その環境の中での処理だけで済ませることが可能となる。

環境設定の変更権限はそれぞれの環境の設定を変更する権限である。参照できる文書やその環境にアクセスできる利用者などを変更する。

以上に述べたような利用者の権限に関する情報は環境ごとにまとめて管理される。これによって各環境の動的な構造の変化などに柔軟な対応が可能となるという利点を得ることができる。

図 2 の例において各利用者の環境の利用を考慮し、それぞれの環境における利用者の権限を考える。会議共通環境については、この環境におけるビューが一般に公開されていることを考えればこの会議に関係する利用者のアクセスを妨げる必要性は無く、各利用者のアクセスを認めれば良い。しかし一般の多くの利用者がこの環境でのビューを変更することを認めるべきではないと言えよう。議長用環境へのアクセスはもちろん議長のみに許可されるべきである。これは議長の役職上使用される重要な情報が資料に付加されていることも考えられるためである。このように環境を利用するためにはカスタマイズ権限や環境設定の変更権限を議長が保持している必要がある。発言者環境についても議長用環境と全く同様の議論を行うことができる。発言者が複数の場合にはそれぞれの発言者に対して環境を用意することが、情報のセキュリティー保持の面から見ても必要であると言える。

3. XML に基づく文書共有モデル

VIEW Media の共有資料及び環境の管理情報の形式として XML を導入する。本章ではまずここで用いる基本的事項について簡単に説明し、続いてここで定められるシステムのフレームワーク [9] について説明する。また本システムで使用される情報は XML 文書として管理されており、以下に説明する環境 XML はそのひとつである。利用者がシステムを利用する際に認証に利用するパスワードも XML 文書として管理される。その文書は利用者名とそのパスワードを属性とするノードを列挙した形式で構成される。

3.1 共有資料

システムで利用される共有資料の形式については資料作成の容易さ、利用可能な資料の豊富さが求められる。これ加えて本システムで行われる資料のカスタマイズの実現が容易であることが重要である。こうした点を考慮して本システムで用いられる共有資料は XHTML 形式を基本とする。

現在 WWW 上の文書も次第に HTML 形式から XHTML 形式に移行しつつある。これにより WWW

上のこうした多くの文書も本システムで利用可能になると考えている。また自ら共有資料を作成することもHTML 文書の作成と同程度に容易である。

また XHTML 文書は XSLT を用いることによって文書の内容や構造の変換を行うことが可能である。この技術を用いることにより元の資料自身を変更することなくカスタマイズされたビューを生成し、利用者に提供することが可能となる。本システムではこれを実現するためにビューのカスタマイズに XSLT を用いる。この場合は、各環境での資料へのカスタマイズの内容は XSL 文書として表現される。

3.2 環境モデルの定義

環境の管理情報はそれぞれの環境ごとに環境 XML 文書でまとめて管理される。その主な情報としては、環境名、その環境にアクセス可能な利用者名、その環境の親環境、子環境の名前、参照できる共有文書名及びそれに対応する XSL 文書名などがある。この文書中の各項目について以下に説明する。

環　境　名

これはその環境 XML 文書で規定される環境の名前を示している。システムが環境ごとに管理される情報を探す時には、この名前で探すことになる。利用者に示される環境名もこの名前である。

アクセス可能な利用者

この環境にアクセスする権限を有する利用者を定める。ひとつの環境内では全ての利用者が得る資料のビューは同一である。環境へのアクセス権限はそれらのビューを得る権限と同義である。環境 XML 文書にはアクセス権限を持つ利用者名のリストが示される。

環境の管理者

その環境にアクセスできる利用者や参照できる共有資料を変更するなど、環境の管理情報を変更する権限を持つ利用者である。環境の管理情報の変更とはすなわちその環境 XML 文書自体の変更を行うことである。

親環境、子環境

VIEW Media システムにおいて環境は木構造に構成される。よって各環境は最大 1 つの親環境を持ち、0 個以上の子環境を持つ。共有資料のビューは親の環境で行われたカスタマイズ内容を継承する。この際この環境における資料のビューは環境 XML 文書定められた親環境のビューを元にカスタマイズを行い生成される。

参照可能な文書

その環境で参照可能な共有文書が規定される。つまりここにリストされた文書に対してカスタマイズされたビューが得られる。それぞれの文書ごとに文書名、対応する XSL 文書の名前、及びその XSL 文書の変更権限を持つユーザの名前を示している。環境独自のビューを生成する際にはここに示された XSL 文書が使われる。資料のビューに対するカスタマイズ内容の変更はこの XSL 文書を変更することで行う。ここで XSL 文書の変更権限は環境へのアクセス権限とは別に指定される。これは環境にアクセス可能な利用者が勝手に文書のカスタマイズを行う行為を制限する。これは上位の環境のビューは公開性が高く、それ故に多くの人に共通の内容を保持するのが望ましいが、そのビューの変更は影響が大きいため自由にカスタマイズされるのは避けたいといった状況に対応するアクセスコントロールが可能になる。

ここではまた資料のビューの継承体系に関する 2 つの属性を定めている。属性 top はその資料のビューが親の環境から継承されないことを示す。属性 top が false の時は親の環境から継承されたビューの対してカスタマイズを加えて、その環境独自のビューを生成する。また属性 top が false の時には元の共有文書を親環境から継承されたビューと同様に扱い、それに対してカスタマイズを行うことでその環境のビューを生成する。もうひとつの属性 leaf はその環境におけるビューを子孫の環境で継承しないことを示す。つまりこの属性が false の時にはこの環境の子環境においてビューを生成する際に、この環境のビューをもとにしてカスタマイズを行うことが出来る。しかし属性 leaf が true の時にはこの環境におけるビューを継承して利用することはできない。これは子孫の環境の利用者に対して見せたくない文書を隠すといった用途への対応に、環境へのアクセス権限と合わせて用いることができる。

しかしこれだけではビューの中のある部分を隠蔽するといった用途には不十分である。そのためその環境のビューを利用者の要望に合わせてカスタマイズする際に、その親の環境のビューに対して情報の一部削除などのある程度の操作を加える必要がある場合も考えられる。そのための XSL 文書を style-lim として定める。この環境独自のビューへのカスタマイズは、その親の環境におけるビューにこの style-lim で定められた XSL 文書を適用したものを基準として行われる。これに対して、必要な要素やリンクの追加などの操作を記述する XSL 文書を style-add に規定する。当然 style-lim に定める XSL 文書に対する変更権限は style-add に定める XSL 文書に対する権限より厳しく定められるべきである。

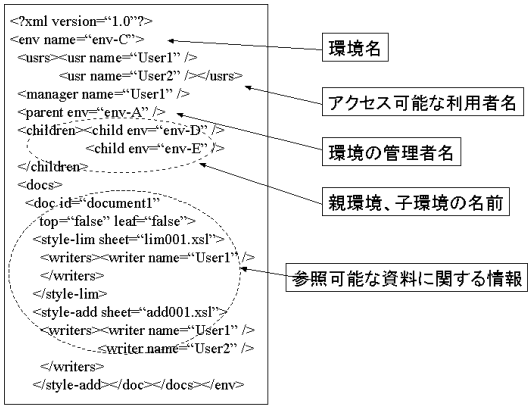

図 3 環境 XML 文書の例

3.3 ビュー生成のメカニズム

前節までに定義したモデルに基づいて、共有資料に対する環境ごとのビューを生成するメカニズムを簡単に説明する。利用者が資料の参照を要求すると、その時点で利用者が所属していた環境のビューが生成される。システムはまずビュー生成に必要な XSL 文書を得るために、環境 XML 文書の該当する資料に関する部分を参照する。この時利用者が所属する環境だけでなくその先祖の環境の環境 XML 文書も参照され、利用者の所属する環境におけるビューを生成するために必要な XSL 文書名を全て得る。この中でより上位の環境の環境 XML 文書に記述された XSL 文書から順に該当の共有資料に対して適用され、利用者の所属する環境独自のカスタマイズがなされたビューを利用者に提供する。

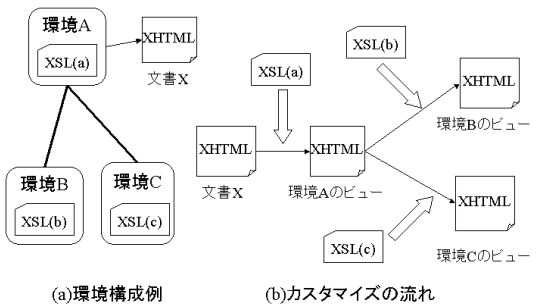

図 4 ビュー生成のメカニズム

図 4 にビュー生成の例を示す。(a) に示した環境構成の元での各環境のビューを生成する。(a) の図中の各環境に示した XSL 文書は、その環境における文書 X に対するカスタマイズを行うためのものである。各環境のビューは (b) に示したように上位の環境のビューから順次生成される。この図を見ると分かるように環境 B、C のビューを生成する際には環境 A の XSL 文書である XSL(a) も適用される。これによって環境 A で行われたカスタマイズの内容がその子環境 B、C におけるビューに反映される。また環境 B、C の間では互いの環境の XSL 文書は使われていない。これによって先祖子孫の関係の無い環境に対してそのカスタマイズ内容が漏れることを防ぐことができる。

4. システムの構成

4.1 システムの構成

XML 技術を用いた VIEW Media システムの構築にあたり、共有資料に対する環境ごとの独自のビューを提供する機能を持つシステムをクライアントサーバモデルで構築する。システムで永続的に管理される情報は、前の章で述べたようにそれぞれの環境の環境 XML 文書、利用者のパスワードを管理するユーザ XML 文書、システムで利用される XHTML 形式の共有資料である。これから利用者が文書への参照要求に対応して動的にビューを生成して利用者に提供する。

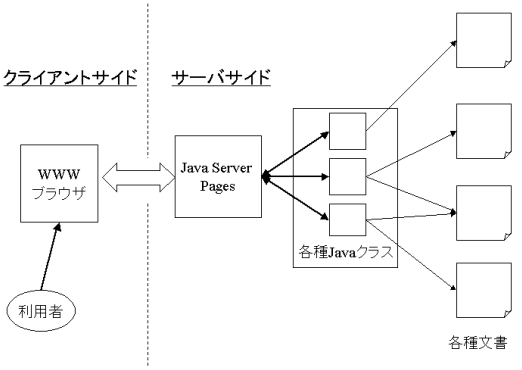

図 5 システムの構成

4.1.1 クライアント

システムの利用者は WWW ブラウザを使用する。利用者は HTML のフォームを利用して必要な情報をサーバサイドに送る。処理結果は XHTML 形式の文書として受け取り WWW ブラウザに表示する。よってクライアント側のマシンには WWW ブラウザが必要であるが、その他には特に必要なプログラムは無い。資料のビューを得るまでの利用者の操作は以下のよう

に示すことができる。

利用者はシステムの利用にあたってまず利用者の ID とパスワードを入力し、システムに送信する。利用者の確認がなされるとシステムからその利用者がアクセス可能な環境のリストが返される。利用者はその中から所属する環境を選択する。するとシステムから参照可能な共有資料のリストが示されるので利用者は参照したい文書を選択する。それに対してシステムは利用者の所属環境における参照文書のビューを生成してクライアントに送信する。以上によって利用者は要求の資料のビューを得ることができる。

4.1.2 サ ー バ

サーバの構成はクライアントとの情報のやりとりを行う部分とそこから情報を受け取り実際に文書の処理を行う部分からなる。

実際に環境 XML 文書の参照や、ビュー生成のための文書変換といった処理を行う機能は後者にあり、そえぞれの機能ごとに Java クラスとして実装されている。これらのクラスは必要に応じて前者の部分から呼び出される。この部分で用意されている Java クラスには利用者のパスワードをチェックするクラスや、環境 XML 文書を参照してアクセス可能な環境のリストを作成するクラス、環境に応じて共有資料をカスタマイズするクラスなどがある。

前者はクライアントから送信されたフォームの情報を取得し、実際に環境 XML などの文書の参照やビューの生成のための処理を行う Java クラスに渡す働きを持つ。またそれぞれの Java クラスの処理により得られた結果を XHTML 文書としてクライアントに送信するのもこの部分の働きである。この部分は JSP によって記述されている。

4.2 システムの動作

システム内部で実際にビューを生成する過程は、以下に示すような動作によって行われる。

ビューの生成に必要な XSL 文書名を積むための空のスタックを用意する。その上で利用者の現在所属している環境の環境 XML 文書をパースする。この環境 XML 文書からその環境の親環境の名前と参照要求のあった文書に対応する XSL 文書の名前を得る。得られた XSL 文書は style-add に指定された XSL 文書、style-lim に指定された XSL 文書の順にスタックに積まれる。この際に資料の top 属性を確認し、これが true の場合には作業を終了する。もし top 属性が false の場合は、先ほど得た親環境の環境 XML 文書を先ほどの説明と同様にしてパースし、XSL 文書とその親環境の名前を得る。以下同じようにして top 属性が true となる環境まで順に遡り XSL 文書の名前をスタックに積んでいく。こうして環境ごとにカスタマイズされたビューを生成するために必要な XSL 文書を全て得ることができ、それらの XSL 文書は先に適用されるものほどスタックの上に積まれている (図 6)。

こうして得られたスタックから取り出された XSL 文書を、順に元の共有資料に適用し XSLT による変換を行う。スタックが空になった段階で得られた文書が得るべきビューとなる。

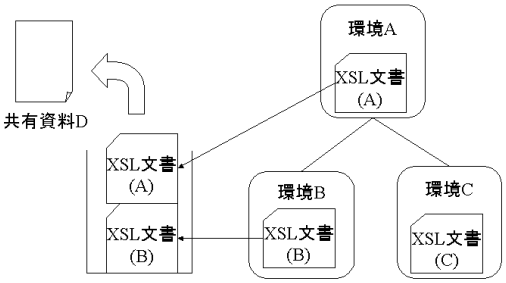

図 6 環境 B におけるビューの生成

5. 実装上の問題点及び今後の課題

前節までに述べたようにしてビューのカスタマイズを行う機構を実装する上での問題点について述べる。

カスタマイズの変更に関する問題

まず XSLT による変換に関して起こる問題点が挙げられる。それぞれの環境において行われるカスタマイズは、全てその環境の先祖の環境で行われたカスタマイズの結果として生成されたビューを元に指定される。そのため先祖の環境で行われるカスタマイズの内容や文書の構造が変化すると、その環境で文書中にコメントを追加するなどの操作を行う位置を指定する XPath、XPointer にずれが生じる恐れがある。この問題に対しては、ビューとして示される文書中の各ノードに対して一意に ID をつけるといったことでほぼ対処は可能であると思われる。この場合にはカスタマイズを行う際に追加されるノードについても同様に ID を付加する必要がある。しかし WWW 上の外部資料をもとにカスタマイズを行っていた場合には、各タグに固有の ID を付けることは難しく対応は困難であるという問題がある。

```
<?xml version="1.0" encoding="shift_jis" ?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output encoding="shift_jis" />
 <xsl:template match="@*|node()">
  <xsl:copy>
   <xsl:apply-templates select="@*|node()" />
  </xsl:copy>
 </xsl:template>
 <xsl:template match="*[@id='b001']">
  <xsl:element name="a">
   <xsl:attribute name="href">http://alpha.c.oka-pu.ac.jp/</xsl:attribute>
   <xsl:copy>
    <xsl:apply-templates select="@*|node()" />
   </xsl:copy>
  </xsl:element>
 </xsl:template>
</xsl:stylesheet>
```

図 7　XSL 文書の例:リンクの追加

XSL 文書の形式に関する問題

またコメントを追加していたノード自体が消去された場合には、そこに付加されていたコメントも自動的にビューから消えてしまうといった問題もある。この場合にはコメントを追加するために用いられた XSL 文書中にはその操作が残ることになるが、XSL 文書だけからビューのどの部分に関連のある内容であるか判断することは難しいといえよう。また XSL 文書中にこうした使用されない記述があることを利用者が検知するためにも XSL 文書を逐次検査することが必要である。

また XSL 文書に記述される操作には幾通りかの記述の仕方が存在する場合もある。この場合、他人の書いた XSL 文書や自分で書いたものでも時間が経過すると分かり難いものとなってしまう可能性がある。さらに XSL は多機能であり、そのゆえに大きな構造変換を行うことも可能である。しかしその子孫の環境への影響は前述したようなカスタマイズ位置の問題を引き起こす。他にも XSL の書き方によっては見易さに難のあるビューになってしまうこともある。

ここに述べたような問題に対処するために本システムで用いる XSLT の機能を制限し、XSL 文書の形式を定めることで対処することを考えている。しかしこれは外部で用いられている XSL 文書を本システムで利用する際に大きな障害となりうる。

また現在のところ XSL 文書は通常のエディターを用いて作成している。しかしこれではビューのカスタマイズを行う利用者は XSLT の知識をもっている必要があり、利用者が簡単にカスタマイズを行えるようなインタフェースは必要であろう。

ビュー統合機能

利用者が複数の立場を持つ場合や複数のグループの管理者の立場にある場合には、利用者の指定した複数の環境のビューをまとめて見たいといった場合も考えられる。この場合、文書へのカスタマイズ操作を内容の隠蔽と追加に分けて考え、それぞれ別の XSL 文書として管理し、そのうち内容を追加する XSL 文書を適用することで対応を図っている。しかしここでは統合対象の環境のどこかで見られるビューの内容を全て集めて示すことを考えており、それらの内容の表示の仕方についてまでは考えていない。このため、上で述べたような統合ビューは XSL 文書を選択して適用することで得られている。この場合、利用者の要望に応えられる内容となっているか、結果の表示の見易さはどうかといったことは問題となる。検討を行う必要があるだろう。

環境の動的変更

実際の現場で協調作業を行上では利用者の立場や役割が変化することや、共有資料の重要性が変化することがある。その場合には状況に合わせて利用者の権限を変更する必要が生じる。グループ作業にゲスト参加者が発生するような場合や、会社組織内で公開されている文書を一時公開停止とするような場合がこれに相当する。また作業の進行上環境構成を変更するといったことも考えられる。グループ作業を行っている中で新たな作業が発生した時にそれを処理するためにグループを分割する場合や、会議を行うために通常の業務とは違った役割構成となる場合などがこれに相当する。

利用者の役割や環境の構成などの変更は一時的な変更とそうでないものに分けれられる。変更が可逆的でない変更については環境の管理情報自体を書き換えるのが適当であり、該当する環境及び必要に応じてその親環境、子環境の環境 XML 文書を書き換えれば良い。しかし一時的な変更に対応するには新たに機能を用意する必要がある。こうした環境の動的な変更に対応できるモデルの導入が必要である。

システムが環境の情報を取得する際には各環境 XML 文書を直接解析すれば良い。しかし上に述べたような環境の変化は可逆的なものであり環境 XML 文書自体を変更することはできないため、各状態に対応した環境 XML 文書を動的に選択できることが必要である。これには各状態に対応した環境 XML 文書をそれぞれ用意しておく方法と、各状態に対応した形に変換する XSL 文書を用意し必要に応じて XSLT で変換する方法が考えられる。しかしどういった手法を採るにせよ作業の状態を管理する機構が必要である。これには作業進行の過程で必要な状況における各環境の情報を作

業状態として定義し、状態オブジェクトを用いて管理する方法 [3] がある。これはアクティブデータベースにおける技術である ECA ルールを用いて状態遷移に対応するものである。こうした環境の変化に対応する機能を現在のシステムに取り入れることは今後の課題である。

6. ま と め

本稿ではまず協調作業支援システム VIEW Media で用いる共有資料を XHTML 形式とし、またシステム内部で使用する環境の管理情報といったデータを XML で表現した。さらに環境ごとに共有資料のビューをカスタマイズする手段として XSLT を導入した。これにより各環境で行われるビューに対するカスタマイズは、XSL 文書として管理される。これにより本システムで利用される共有資料は高い一般性を持ち、WWW 上の XHTML 文書もそのまま協調作業に利用することが可能になる。また共有資料や環境の情報を文書の形で管理することでデータの永続性も実現でき、長期の協調作業への対応も可能である。

作業中には利用者の立場や役割の変更等により、それぞれの環境で作業に参加する利用者の変更や参照する共有資料の追加などといった環境の管理情報を変更することもある。しかしこの影響は、親環境と子環境に対して多少ある程度でありあまり他の環境への影響を考えずに行うことができる。これは利用者のアクセス権限や参照可能な資料情報の管理、その資料のビューへのカスタマイズのための XSL 文書などといった各環境の情報を、それぞれの環境ごとに管理していることによりもたらされる利点である。こうした環境の動的な変更を行う機能を実際にシステムに取り入れることは重要な課題である。

現在のモデル及び実装においては、こうした機能を実用的なものとするためにはまだまだ多くの問題が存在することを示した。上位環境におけるビューのカスタマイズの変更が下位環境で不都合を生じる場合がある点については対応が必要である。またビューのカスタマイズを簡単に行えるようなユーザインタフェースの導入も同時に考慮する必要があるだろう。

協調作業を支援するシステムとしては、共有文書の管理や利用者の権限管理といった機能以外にも重要な機能がある。資料や環境の変更を利用者に通知する能動性や他の利用者の状況を提供するアウェアネス機能が挙げられる。こうした機能に対応したシステムの構築が今後の課題といえる。

参 考 文 献

1) Y. Yokota: Design and Implementation of Cooperative Hypermedia System VIEW Media with Non-WYSIWIS Personalizing function for Cooperative Work. Master Thesis, Department of Information Science, Kyoto University, 1998
2) Y. Yokota, K. Sugiyama, H. Tarumi and Y. Kambayashi: Evaluation of Non-WYSIWIS Functions of VIEW Media. International Jarnal of Cooperative Information Systems, Vol. 9, Nos. 1 & 2 (2000), pp29-51, 2000
3) 杉山圭司: 協調作業における作業状態管理機構. 修士論文, 京都大学大学院情報学研究科, 2001
4) J. H. Lee,A. Prakash,T. Jaeger and G. Wu: Supporting Multi-User, Multi-Applet Workspaces in CBE. Proc. of CSCW '96, pp.334-353, 1996
5) A. Prakash, H. S. Shim and J. H. Lee: Data Management Issues and Trade-Offs in CSCW Systems. IEEE Transactions on Knowledge and Data Engineering, Vol. 11, No. 1, pp213-227, 1999
6) G. Fitzpatrick, S. Kaplan and T. Mansfield: Physical Spadces, Virtual Places and Social Worlds: A study of work in the virtual. CSCW '96, pp334-343, 1996
7) T. Mansfield, S. Kaplan, G. Fitzpatrik, T. Phelips, M. Fitzpatrick and R. Tayler: Evolving Orbit: a progress report on building locales. Proc. of Group97, pp241-250, 1997
8) T. W. Bickmore and B. N. Schilit: Digestor: Device-Independent Access to the World Wide Web. Sixth International World Wide Web Conference (WWW6), Pager117-TEC116, 1997
9) M. Tanigaki, Y. Yokota and Y. Kambayashi: An XML-Based Workspace and Document Model for Flexible Cooperative Work. The 20th IASTED International Multi-Conference Applied Informatics (AI2002), 2002
10) R. Han, V. Perret and M. Naghshinech: WebSplitter: a unified XML Framework for multi-device collaborative Web browsing. Proc. of the ACM 2000 Conf. on CSCW, pp221-230, 2000
11) M. Stefik, D. G. Bobrow, S. Lannning and D. Tatar: WYSIWIS Revised: Early Experiences with Multi-user interfaces. Proc. of CSCW'86, 1986
12) H.-H. Shen and P. Dewan: Access Control for Collaborative Environments. Proc. of ACM 1992 Conf. on CSCW, 1992