UIを分離した複合文書による情報共有

羽根 秀宜 , 河村 元夫 , 横田 実
NEC C&C 研究所

インタラクティブな情報共有を実現する複合文書アーキテクチャについて提案する。本アーキテクチャではテキストや図形などの複合文書の要素オブジェクトがUI部 (Shapeと呼ぶ) と本処理部 (Bodyと呼ぶ) から構成され、Shapeのみが各ユーザの端末上で動作しBodyをShapeで共有することで文書共有を実現する。さらに、ユーザが文書の同じページを見る / 異なるページを見るといった2種類の共有形態が必要であると考え、Shapeのコンテキスト情報を分離し、コンテキストの共有を切り替えることで実現する。

Information Sharing by the Compound Document based on UI Separation

This paper describes the compound document architecture for interactive information sharing. A component of the compound document such as text editor or figure editor consists the UI part (called Shape) and the essential function one (called Body). The Shape only runs on user terminal, and a shared document is realized by sharing the Body with the Shapes of each users. And further, the context information of the Shape is separated. Therefore two types of sharing, for example users read the same page or the different page of a document, is available by switching the context.

1 はじめに

インタネットに代表されるネットワークの普及に伴い、インタラクティブに情報を共有し利用するための枠組が求められている。例えばWWWはインタネットでの情報発信に盛んに用いられているが、HTMLの表現力はまだ不十分であり、CGIを用いた情報検索や掲示板などのインタラクティブサービスもユーザへのリアルタイムな変更の反映などの点で根本的な問題点がある。また分散オブジェクト技術を用いて複合文書技術であるOLE2[1]やOpenDoc[2]をネットワーク対応させる動きもあるが、文書共有の面ではまだこれからという状況である。

筆者らは、ネットワークを介したインタラクティブな情報共有を特徴とするパーソナル端末アーキテクチャとして、ユーザインタフェース (UI) に特化したシステムアーキテクチャの確立を目指している。本論文では、そのシステムの基本となる、UI分離を特徴とする複合文書アーキテクチャについて述べる。

まず目標とする情報共有の特徴について述べる。次にそれを実現する方法として、UI分離によりオブジェクトの分散共有を実現するShaped Object Model[3]と、それを用いた複合文書アーキテクチャについて述べる。

2 目標とアプローチ

ここでは情報を文書とし、情報のインタラクションを文書共有であると捉えることにする。そこで文書共有をベースとしたインタラクティブな情報共有のために、次に示す特徴を持つ文書共有を目標とする。

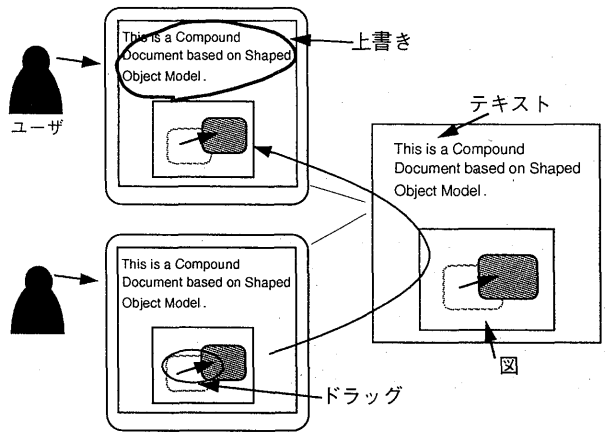

図 1: 目標とする文書共有

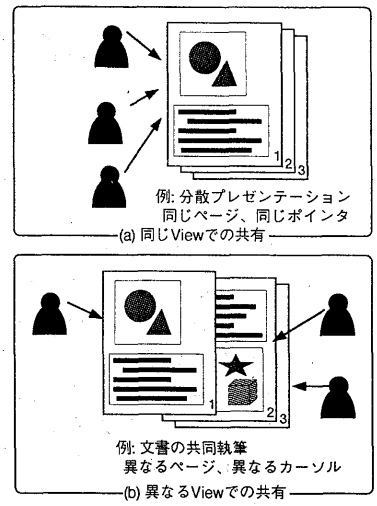

図 2: 同じ View/ 異なる View での共有

アクティブ文書の共有 ここでは、時事刻々と内容の変化する文書をアクティブ文書と呼ぶ。アクティブ文書の共有では、内容の変化が他のユーザにリアルタイムに反映される (図 1: 図形のドラッグ)。これにより、分散プレゼンテーションや文書の共同執筆など文書を介したインタラクションを実現することができる。

様々なメディアを含む文書 文書の共有による多様な応用を可能にするには、テキストや図形、表、動画といった様々なメディアを含む文書を取り扱うことができる必要がある (図 1: テキストや図を含む文書)。

文書情報の加工 共有文書を有効に利用するには、例えば紙の文書のように切抜きをスクラップブックに貼り付けたりコメントを上書きするといった、自然で使い勝手の良い情報の加工が必要である (図 1: 上書き)。

同じ View/ 異なる View での共有 例えば複数ページの文書を共有する場合、分散プレゼンテーションなどが目的であれば各ユーザが同じページ、つまり同じ View を見る必要があり、文書の共同執筆などが目的であればユーザ毎に別々のページ、つまり異なる View を見て編集できる必要がある (図 2)。どの文書についてもこれら 2 種類の共有形態を実現できるようにする。

上記の文書共有を実現するために、次の 2 つのアプローチをとった。

UI の分離 アクティブ文書を共有するには、ファイル共有などの静的な共有ではなく、CSCW における分散アプリケーション的な枠組が必要である。そこで、UI 分離を特徴とするオブジェクトの分散共有方式として筆者らが提案している Shaped Object Model を用いてアクティブ文書の共有を実現する。

複合文書アーキテクチャ 様々なメディアを含む文書を実現するためには、メディアそれぞれを文書の構成要素 (コンポーネント) とし、その複合体を文書と捉える OLE2 や OpenDoc に代表される複合文書アーキテクチャが適している。そこで、Shaped Object Model を複合文書へ拡張し、分散共有された複合文書を実現する。切り貼りや上書きなどの情報の加工は、それらをオブジェクトとして動的に文書に埋め込むことで実現可能である。また、同じ View/ 異なる View での共有を実現するために Shaped Object Model を拡張する。

3 Shaped Object Model

3.1 UI 分離によるアクティブな情報共有

Shaped Object Model の基本的なアイデアはオブジェクトの UI と本処理の分離である。UI 部分を Shape と呼び、描画 / イベント処理を行う。本処理部分を Body と呼び、UI 以外のそのオブジェクトの本質的な機能の処理を行う (図 3)。

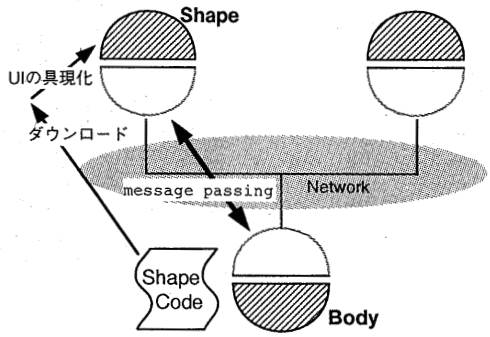

図 3: Shaped Object Model

オブジェクトがリモートマシンからアクセスされると Shape を返送する。リモートマシン上で Shape はそのオブジェクトの UI を具現化し、Body と通信路を張る。さらに、ユーザの操作に従い Body へメッセージを送り、Body 側で必要な処理が行われるとともに、Body の変更はリアルタイムに Shape に通知される。また、同じ Body を複数の Shape で共有することで、オブジェクトの分散共有を実現できる。

このように、オブジェクトの本処理である Body を共有し、その変更をリアルタイムに Shape に通知することで、アクティブな情報共有を実現する。

3.2 簡単なテキストエディタの実現例

以下の仕様を持つ簡単なテキストエディタを Shaped Object Model を用いて実現する例について説明する (図 4)。

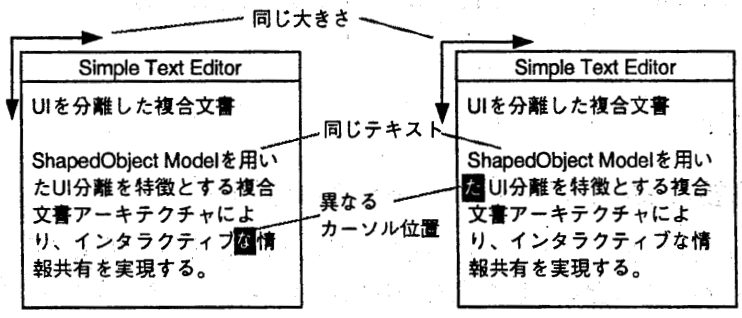

図 4: 簡単なテキストエディタの例

- ユーザは、テキスト情報とテキストエディタの大きさを共有する。
- テキストエディタのカーソル位置はユーザ毎に持つ (非共有)。
- あるユーザからの文字入力 / 削除は、即座に他のユーザの画面にも反映される。
- テキストのフォントの種類と大きさは 1 種類で固定。また Cut,Copy,Paste 等は考えない。

このテキストエディタを実現するために必要なデータと機能を図5(a)に示す。レイアウトがテキストとエディタの大きさから計算され、それに基づきテキストが描画される。またユーザからのキー入力に従いカーソル位置の変更と描画やテキストの変更が行われ、マウス操作によりエディタの大きさの変更と設定が行われる。

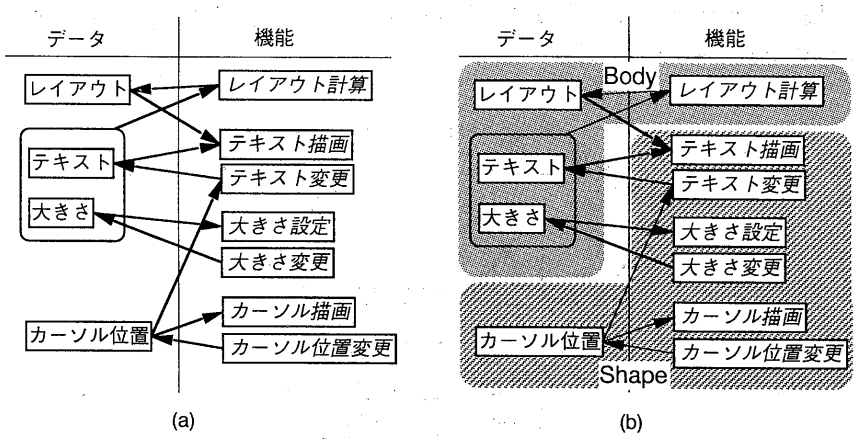

図 5: テキストエディタの実現

Shaped Object Model では Shape が描画 / イベント処理などの UI 処理を行い、Body を複数の Shape で共有することで分散共有アプリケーションを実現する。したがって、Shape でユーザ毎のデータや描画 / イベント処理の機能を、Body で共有されるデータと機能を持つように機能分割する。この例では次のようになる。

- Body は共有されるデータであるテキストとエディタの大きさのデータを管理する。更に、レイアウト計算は共有されるデータのみを用いるため、Body に配置することができる。
- Shape はユーザ毎のデータであるカーソルデータの管理と、表示 / イベント処理に関する機能を持つ。

これを図5(b)に示す。この図で Body-Shape 間を跨る線は Body-Shape 間で必要なメッセージ通信に対応すると考えることができる。

また、Body はデータが変更されるとそれを即座に全 Shape に通知することで表示内容の同期を行う。一般に CSCW では内容同期の方法として、大別して pessimistic および optimistic の2つの方法がある [4] が、Shaped Object Model では内容同期の方法は規定せず、アプリケーションの実装に委ねている。

4 複合文書への拡張

4.1 複合文書の概要と Shaped Object Model への適用

複合文書は従来のアプリケーション中心から文書中心の利用環境を提供する。具体的には、従来のアプリケーションに相当する機能単位を文書の構成要素 (コンポーネント) としその集合を文書とする。つまり、まずアプリケーションがあってそれにより作成された文書があるのではなく、まず文書があってそれに様々なアプリケーション機能を含めることができる枠組である。

複合文書の本質は、コンテナとコンポーネントの2つの機能要素である (図6)。コンテナは埋め込むコンポーネントの表示領域を管理し、コンポーネントはコンテナが与える表示領域に内容を表示しユーザからのイベントを処理する。このように、コンテナとコンポーネントは表示領域にかかわる部分についてのみ関連しており、それ以外は互いにブラックボックス的な関係にある。これが多様なコンポーネントを複合できる理由でもある。

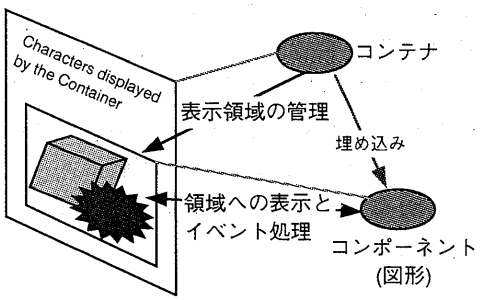

図 6: 複合ドキュメントの基本構造

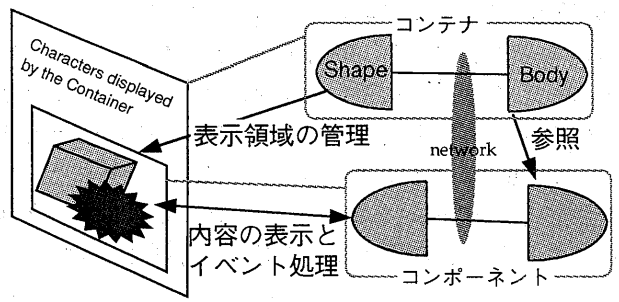

図 7: Shaped Object Model による複合文書

そこで Shaped Object Model による複合文書では、Shape における描画の枠組を前述のコンテナ / コンポーネントの枠組に拡張する (図 7)。具体的には次のようになる。

1. コンテナの Shape は、コンポーネントの Shape の描画領域の管理を行う。
2. コンポーネントの Shape は、コンテナの Shape が与える表示領域内に描画し、その領域でのユーザからのイベントを処理する。

なおコンテナ / コンポーネントの Body 間では、コンテナの Body がコンポーネントの Body への参照を保持することで、埋め込みの論理的な関係を維持する。

4.2 本アーキテクチャの利点

Shaped Object Model による複合文書の利点を以下に挙げる。

共有コンポーネントの実現 上書きは埋め込むコンポーネントの表示領域上に上書き情報を重ねて表示するコンテナにより実現できる (図 8)。この場合、上書きコンテナ自体を共有することで、共有するユーザが同時に書き込めるいわゆる白板機能を実現でき、上書きコンテナを各ユーザが独自に持つと、個人的なメモ書きの実現できるという特徴を持つ。

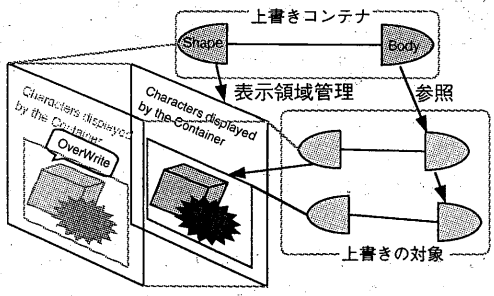

図 8: 複合文書による上書きの実現

従来の複合文書では埋め込みとリンクの 2 種類の複合方法があり、例えばあるコンポーネントを別の文書に複合する場合、埋め込みではそのコンポーネントのコピーが組み込まれ、リンクでは参照のみが組み込まれる。従って、前述の上書きの例のようなコンポーネントの共有にはリンクが用いられるが、この方法では元コンポーネントの変更が即座に反映されなかったり、特別な実装が必要である。これはコンポーネントの基本的な枠組として共有を考慮していないからである。

本アーキテクチャでは、Shaped Object Model により基本的な枠組として共有を考慮しているため、埋め込みとリンクの区別は無く、どのコンポーネントも複数のコンテナで共有することができる。

文書の再現性 複合文書で新たなメディアを利用したい場合には、そのメディアのコンポーネントを実装することになる。従来の複合文書アーキテクチャでは、そのメディアのコンポーネントが各ユーザのマシンに無いと表示 / 編集を行えないという文書の再現性の問題があった。これは多様なメディアを含む文書を共有する場合に顕著な問題となってくる。しかし、Shaped Object Model による複合文書では、Shape をダウンロードするのであらかじめユーザ側に各メディアに応じた Shape をインストールしておく必要が無い。したがって、新たなメディアが実現されても、それを表示 / 編集できない事態を避けることができる。

4.3 ViewContext の導入

第 2 節で述べたような同じ View/ 異なる View での共有は、各コンポーネントの実装に委ねることも可能であるが、本アーキテクチャにおける標準の枠組として積極的に利用するために、Shape の表示に必要な情報、つまりコンテキストを Shape から分離することにした。これを ViewContext と呼ぶ。

図 9は、最上位のコンテナの ViewContext にページ番号が存在する例を示しており、上の 2 ユーザは同じ ViewContext の Shape を見ることで同じページを、一番下のユーザは他のユーザとは異なる ViewContext の Shape を見ることで異なるページを見ている。このように、ViewContext を切替えることで同じ View/ 異なる View を容易に実現できる。

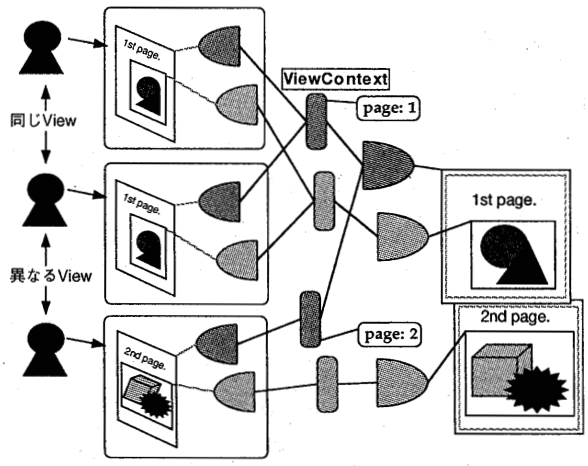

図 9: ViewContext による同じ View/ 異なる View での共有

ViewContext とユーザの対応には、さまざまな方法が考えられる。例えば、MS-Windows などの従来システムでは、同じファイルを開いた各ユーザはファイルのコンテンツが同じだけであり、アプリケーションの状態はそれぞれのユーザ毎に異なる。これと同じアナロジーで ViewContext を捉えた場合、一つの Body をアクセスするユーザ毎に ViewContext を生成する方法が考えられる。また分散プレゼンテーションなどは、どのユーザも同じ ViewContext で Body をアクセスする方法で実現できる。更にこの場合、ある文書について分散プレゼンテーション用の Shape と、各ユーザ毎の ViewContext における Shape を同時にアクセスすることで、説明者の画面を見ながら自分の手元の同じ資料を先読みするといったような利用方法も可能となる。このように、ViewContext とユーザの対応を変化させることで、さまざまな応用を実現できる。

5 関連技術

Body-Shape の UI 分離と似た考え方として、MVC[5] の Model-View があり、オブジェクトのある部分機能を Body に割り振るか Shape に割り振るかという機能分離も同じ視点で捉えることがで

きる。

しかしShaped Object Modelでは、Shapeをダウンロードし実行することを前提としている点、ViewContextを導入することで表示コンテキストを分離し同じView/異なるViewの動的な切替えを可能としている点が異なる。特に後者については、Model-Viewでも例えばデータを動的にModelとViewのどちらかに配置するような複雑な設計を採ることでさまざまな形態の共有を実現できるが、それはModel-Viewの枠組でなくModel-Viewの実現の一形態となってしまい、統一的に捉えることができない。ここでは、同じView/異なるViewという2つの共有形態を前提に、そのための枠組としてViewContextの導入を提案している。

OLE2やOpenDocなどの既存の複合文書技術は分散オブジェクト技術を用いてコンポーネントの分散化への拡張が行われている。しかしコンポーネントの基本的なアーキテクチャにおいて共有を考慮していないため、第2節で述べたアクティブ文書の共有や、切り貼り / 上書き等の情報の加工を実現することは困難と考えられる。

6 まとめ

UI分離を特徴とするShaped Object Modelを用いた分散共有可能な複合文書アーキテクチャについて述べた。このアーキテクチャでは、複合文書の各コンポーネントをShaped Object Modelにより分散共有することで、様々なメディアを含むアクティブ文書を実現でき、コンポーネントの動的な埋め込みで切り貼りや上書きなどの情報の加工を可能にしている。またViewContextを導入し、同じView/異なるViewでの共有を実現している。

Shaped Object Modelによる複合ドキュメントのアイデア検証のためにJava言語[6]を用いたプロトタイプを試作中である[7]。Javaを用いた理由は、スクリプト言語よりも複雑なプログラミングが可能であり、クラスダウンロード機能を用いてShapeのダウンロードを実現できること、RMI[8]やHORB[9]などのオブジェクトのリモートメソッド呼び出しの枠組が提供されているのでBody-Shape間の通信を実現しやすいことによる。

謝辞 本研究を進めるにあたり、NEC C&C研究所ターミナルシステム研究部の携帯情報端末グループの諸氏に様々なコメントを頂いたことを感謝致します。

参考文献

[1] Microsoft Corp., "Inside OLE2", アスキー出版, 1994

[2] Apple Computer Inc., "OpenDoc Programmer's Guide", *http://opendoc.apple.com*, 1996

[3] 横田, "Shaped Objectによる情報の分散共有", 情報処理学会マルチメディア通信と分散処理研究会, 95-DPS-73, pp.69-74, 1995

[4] David A. Nichols et.al., "High-Latency, Low-Bandwidth Windowing in the Jupiter Collaboration System", *Proc. of th ACM Symposium on User Interface and Software Technology*(UIST '95), pp 111-120, 1995

[5] Glenn E. Krasner et.al., "A cookbook for using the model-view-controller user interface paradigm in Smalltalk-80", *Journal of Object-Oriented Programming*, 1(3):26-49, 1988

[6] Sun Microsystems, "The Java Language: A White Paper", *Sun Micorsystems White Paper*, 1994

[7] 羽根 他, "分散共有アクティブドキュメントの実現方式", 情報処理学会第53回全国大会, 3J-6, 1996

[8] Wollrath, Ann, et.al., "A Distributed Object Model for the Java(TM) system", *Proc. of the USENIX 2nd Conference on Object-Oriented Technologies and Systems*, pp.219-231, June 1996

[9] 平野, "HORB: The Magic Carpet for Network Computing", *http://ring.etl.go.jp/openlab/horb*,1996