計測自動制御学会産業論文集
Vol.4, No.12, 81/90 (2005)

帳票のイメージデータ化と疎結合コンポーネント接続による銀行営業店後方事務処理システムの開発

染谷 治志*，森 有一*，阿部 正弘*，町田 勇**，長谷川 篤***，吉江 修****

Development of Centralized Processing System for Back-office Operations in Bank Branches using Form Image Data and Loosely Coupled Component Connection Method

Harushi Someya*, Yuichi Mori*, Masahiro Abe*, Isamu Machida**, Atsushi Hasegawa***, Osamu Yoshie****

Abstract：Due to the deregulation of financial industry, the branches in banking industry need to shift to the sales-oriented bases from operation-oriented bases by rationalization of back-office operations. It is subjects how the back-office operations systematizes and how the system development cost is reduced.

In this paper we propose a centralized processing method that directly inputs forms in a computer as image data by a non-contacting image scanner, transmits to a centralized operation center and then processes back-office operations centrally by workflow. By this method, rationalization of the back-office operations and reduction of the operational cost are able to be accomplished. Also we propose a client application architecture that has a loosely coupled component connection method, and allows developing the application by only describing the screen configuration and their transitions in XML documents. By proposed architecture, it raises the reusability of the components, can localize the customize range and allows users to easily develop applications.

By adopting proposed system measures, we developed the centralized processing system of the back-office operations in the banking branches. We checked quantitatively the rationalization effect by this system. And we verified the software productivity of the proposed client application architecture, our experiments demonstrate good performance.

Keywords：Back-office operations in the banking branch, Centralized processing, Image data, Loosely coupled component connection method, XML

1. はじめに

情報通信技術の発達により，エレクトロニックコマースや企業におけるIT化が進展してきている．また，社会基盤においても，電子政府[1)]やe-Japan戦略Ⅱ[2)]にもとづくIT基盤を活かした社会経済システムの積極的な変革が進められている．その結果，各種文書や帳票が電子化され，ペーパレスが進んできている．ところが，銀行営業店においては，自行が店舗内に備え付けている定型帳票（制定帳票）のほかに，税金や公金帳票，一般振込み帳票などの諸帳票（非制定帳票）が多く，紙ベースの現物処理（帳票事務）が依然として残り，ペーパレスが進んでいない状況にある．

銀行営業店における帳票事務は，窓口で顧客と直接やり取りをする窓口事務と，窓口の後方で顧客から受け取った帳票や書類の事務処理を行う後方事務に大別される．後方事務では，印鑑の確認や不正送金のチェックなど，正確な取引の実現と内部不正を防止する目的で，帳票およびその事務処理結果に対して二重三重のチェックが行われている．預金や為替の例では，業務プロセス全体の 70%～80%が営業と直接関係がない事務処理で占められており，後方事務要員は営業店人員の約 20%～30%を占めているという報告[(3)]がある．このことが，営業店の取引コストを，インターネットバンキングの約 8 倍[3)]と，他のダイレクトチャネルに比べて非常に高くする原因となっている．その結果，営

＊ （株）日立製作所システム開発研究所
〒244-0817 神奈川県横浜市戸塚区吉田町 292 番地
Hitachi, Ltd., Systems Development Laboratory
292 Yoshida, Totsuka, Yokohama, Kanagawa 244-0817
＊＊ 日立オムロンターミナルソリューションズ（株）
〒141-0032 東京都品川区大崎 1-6-3
Hitachi-Omron Terminal Solutions Corporation
1-6-3 Osaki, Shinagawa, Tokyo 141-0032
＊＊＊ （株）日立製作所金融システム事業部
〒212-8567 神奈川県川崎市幸区鹿島田 890 番地
Hitachi, Ltd., Financial Systems Division
890 Kashimada, Saiwai, Kawasaki, Kanagawa 212-8567
＊＊＊＊早稲田大学大学院情報生産システム研究科
〒808-0135 福岡県北九州市若松区ひびきの 2-7
Graduate school of Information, Production and Systems, Waseda University
2-7 Hibikino, Wakamatsu, Kitakyushu, Fukuoka 808-0135
(Received May 12, 2005)

業最前線のデリバリーチャネルである営業店は，「事務」主体のオペレーション拠点となっている．

日本版金融ビッグバンに代表される規制緩和などにより，金融機関を取り巻く環境が大きな変革期を迎え，銀行業界では多様化する顧客ニーズや新業務・新サービスへの迅速な対応が求められている．このため，帳票事務の徹底的な合理化により，営業店を「営業」主体のセールス拠点へと変えていく必要がある．依然として残る現物処理（帳票事務）の合理化をどのようなシステム化方式で実現するかが課題である．また，制定帳票や非制定帳票など多種多様な帳票の処理アプリケーションの開発コストをいかに低減していくかも，システム化する上での課題である．

帳票処理の分野では，電子帳票アプリケーションパッケージソフトウェアが種々実用化されている．銀行営業店窓口で扱う帳票をこのパッケージソフトウェアで実装することが考えられるが，制定帳票は可能としても，非制定帳票すべてを実装することは非現実的であり実質上不可能である．なぜなら，非制定帳票の発行側にも同様のパッケージソフトウェアを導入してもらわなければならないからである．また，銀行によって使用するパッケージソフトウェアが違えば，非制定帳票の発行側はすべてのパッケージソフトウェアに対応することになるか，あるいは顧客に同じパッケージソフトウェアを使用している銀行での取引を強いることになる．すなわち，非制定帳票の発行側のコストに見合ったメリットが見出せず，あるいは顧客サービスの低下を招くことになるからである．制定帳票だけをパッケージソフトウェアで実装した場合，制定帳票と非制定帳票とで事務が異なり，かえって事務を複雑にすることになりかねない．

一方，帳票アプリケーションの開発分野では，XML(eXtensible Markup Language)とスクリプト言語を組み合わせた帳票アプリケーション言語XFA(XML Form Architecture)[4),5)]が提案されている．また，多種多様な帳票では類似項目が多いことから，GUI(Graphical User Interface)や業務ロジックをソフトウェア部品（コンポーネント）化して，これらを組み合わせてアプリケーションを開発していくコンポーネント指向の開発方法[6)~11)]がある．しかし，スクリプト言語によるコーディングが必要であったり，コンポーネント間が密結合となるためにカスタマイズ性が悪かったりと，開発コストを低減する開発環境としては不充分である．

本稿では，銀行営業店と顧客や非制定帳票の発行者とのインタフェースやプロセスを変更することなく帳票をコンピュータ内に直接取り込み，ひとつの事務センタ（以下では，この事務センタを集中事務処理センタと記述する）に集約して集中処理するシステム化方式を提案する．また，集中事務処理センタには多種多様な帳票が集まり，これらを処理するアプリケーションの開発コストを低減するため，コンポーネント間を疎結合する接続方式を導入し，画面を構成するコンポーネントの接続関係と画面遷移をXML文書で記述することでアプリケーションを開発できる，帳票処理クライアントアプリケーションアーキテクチャを提案する．これらのシステム化方式により，各営業店で実施していた後方事務を集中事務処理センタに集約でき，事務を合理化し事務コストを低減することができる．また，疎結合コンポーネント接続方式の導入により，コンポーネントの再利用性が高まり，カスタマイズも容易でアプリケーションの開発コストを低減することができる．提案するシステム化方式を採用して，銀行営業店の後方事務の集中事務処理システムを開発した．このシステムの導入による事務合理化効果を定量的に確認するとともに，提案アーキテクチャの生産性検証評価を通じてその有効性を確認した．

2. 銀行営業店における帳票事務合理化の課題

銀行業界では，これまで顧客サービスの向上や取引コスト削減に向けた事務の合理化を図ってきた．帳票事務に関しても，銀行の公共性や社会的責任をまっとうする厳格性と正確性を保ちつつ，徹底的な事務オペレーションの合理化を進めてきた．しかし，規制緩和などによる経営環境の変化から，さらなる合理化が求められている．帳票事務における合理化ポイントは，先に述べた通り，多くの人手と時間を要している後方事務にある．**Fig.1** に為替業務における後方事務処理例を示す．銀行営業店窓口で顧客から受け

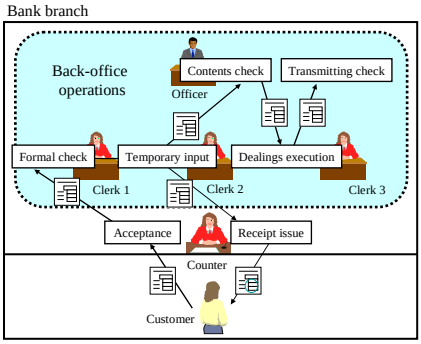

Fig.1 Example of back-office operations

取った帳票は，各事務の担当者に回付されて事務規定に則り処理されていく．この後方事務を電子化して担当者のPCでオペレーションできるようにすれば，紙ベースの帳票を担当者に回付する時間（立ち歩き時間）の削減が期待できる．また，ある営業店で後方事務量が0.5人分あるとすると，1人の担当者が必要になる．これと同じ事務量の営業店があれば同じく1人，合計2人の人員が必要になる．各営業店で分散して事務処理をしているため，合計1.0人分の事務量に対して2人の人員を配することになる．後方事務を電子化できれば，その事務処理を集中化することが可能であり，2人要していた人員が1人で済むようになり，事務合理化と取引コストの削減が期待できる．

これを実現するためには，現物の帳票をどのように電子化するかが課題である．最近実用化されている電子帳票ア

プリケーションパッケージソフトウェアの活用が考えられるが，先に述べたように実質上不可能である．また，帳票にバーコードを付けバーコードリーダによって電子化する案もあるが，パッケージソフトウェア活用案と同様，すべての非制定帳票にある決められた仕様に従ったバーコードを付けることは実質上不可能である．したがって，外部（顧客や非制定帳票の発行者）とのインタフェースやプロセスを変更することなく，帳票を電子化してコンピュータ内に取り込むシステム化方式を検討していかなければならない．

ところで，銀行営業店で扱う帳票は多種多様であり，帳票を処理するアプリケーションの開発コストが問題となってくる．帳票は多種多様であるが，種別（たとえば，振込みに関わる帳票）によって，起票項目がほとんど同じ場合が多い．帳票内の各項目には，口座番号の桁数チェックや支店コードの存在確認など各種業務ロジックが設定されている．しかし，同じ種別でも種目（たとえば，一般振込や給与振込）によって項目に設定する業務ロジックが異なったり，項目間のチェック内容が異なったりするなど項目間の関連に相違がある．これらは銀行独自の事務規定に則っているもので，銀行によっても異なる．また，新たな帳票の追加や帳票の項目追加・削除が頻繁に発生する．そのため，帳票の項目追加・削除，帳票の文言の修正やシステムメッセージの表現修正など，些細な変更はユーザ自ら行うことでコストダウンを図り，変更に早期対応したいというユーザニーズもある．したがって，集中事務処理センタの帳票処理クライアントアプリケーションの開発においては，再利用性が高く，かつカスタマイズが容易で，ユーザ自身による開発を可能とする開発環境の整備が必要である．

帳票アプリケーションの記述言語として知られているものに，XFA(XML Form Architecture)[4),5)]がある．XFAでは，GUIと業務ロジックはスクリプト言語でプログラミングする仕様となっている．このため，GUIと業務ロジックの分離設計が図られておりそれぞれの再利用性は高いものの，業務ロジックのカスタマイズ性とユーザ開発性の観点から上記課題に対して充分に応えられるものではない．多種多様な帳票の中で類似項目が多いことから，GUIや業務ロジックをコンポーネント化して，これらを組み合わせてアプリケーションを開発していくBML(Bean Markup Language)[6)~8)]やCurl[9)~11)]などのコンポーネント指向の開発方法がある．**Fig.2**に，銀行の為替振発業務をコンポーネント指向で開発した場合の一般的なコンポーネントの接続関係を示す．為替振発業務は，振込み帳票で指図された銀行口座に送金して債権債務の決済を行う業務であり，送金種目（給与振込や一般振込など）によって送金先口座への入金日が異なる．前述のように，帳票処理では項目間に複雑な依存関係があるためコンポーネント間にも依存関係が発生し，コンポーネント間の接続は密結合となる．このため，カスタマイズが発生すると変更箇所に依存するすべてのコンポーネントを変更する必要があるため，カスタマイズが容易であるとは言い難い．

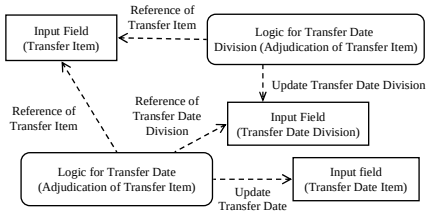

Fig.2 Example of dependency relations among components

本稿では，営業店窓口と外部とのインタフェースやプロセスを変更することなく，帳票をコンピュータ内に取り込む手段としてイメージスキャナを用い，帳票のイメージデータをワークフローで集中事務処理センタに集約して後方事務を集中処理するシステム化方式を提案する．また，画面を構成するコンポーネントの接続定義および画面遷移定義をXML文書で記述することでアプリケーションを開発することができる，帳票処理クライアントアプリケーションアーキテクチャ[14),15)]を提案する．本アーキテクチャでは，イベントアダプタ[12),13)]にデータオブジェクトを内包させた拡張イベントアダプタを用いることにより，コンポーネント間の依存関係を緩やかにしてコンポーネントの独立性を高めた接続方式を導入している．これらのシステム化方式により，営業店の後方事務の合理化と事務コストの低減を図ることができる．また，コンポーネントの高い再利用性が得られるので，カスタマイズ範囲の局所化を図ることができ，アプリケーションの開発コストを削減することができる．さらに，XML文書記述によりアプリケーションを実現しているので，ユーザでも容易にアプリケーションを開発することができる．

3. システム化方式

3.1 事務集中処理方式

事務集中処理方式の基本方針は，(1)銀行の営業店窓口と外部とのインタフェースやプロセスが変わらないこと，(2)また窓口行員に新たな手間と時間がかかるオペレーションを発生させないこと，(3)さらに帳票事務オペレーションに際して現物帳票の look and feel を大きく変えないことと，(4)立ち歩き時間を排除することである．これらの基本方針に従い，帳票は人間の業務を集約したエッセンスであることから，帳票をコンピュータに直接取り込み，その帳票を自動回付して事務オペレーションを実施する考え方に基づきシステム化する．

具体的なシステム実現方式は，(1)帳票のコンピュータへの取り込み手段として，**Fig.3** に示す非接触イメージスキャ

Fig.3 A non-contacting image scanner

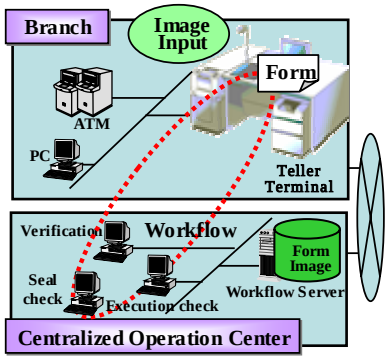

Fig.4 Overview of the centralized processing system for the back-office operations

ナで帳票をイメージデータとしてコンピュータ内に直接取り込む，(2)読み込んだ帳票イメージデータの集中事務センタへの伝送と事務規定に則り帳票を回付する手段として，ワークフローで実装する，(3)事務処理は，帳票イメージデータをもとに事務オペレーションできるよう帳票処理クライアントアプリケーション（Webクライアントアプリケーション）で実装する．Fig.3 に示す非接触イメージスキャナは，帳票を所定の位置に置くだけで自動的に読み込む仕組みであり，帳票の大きさ，厚み，皺，折り目や紙質にも対応する自由度の高い手段である．また，読み込み対象である帳票の位置や方向合わせが不要であり，帳票種別の自動認識による業務画面の自動索引も可能であるので，窓口受付事務におけるクイックオペレーションを実現することができる．帳票イメージデータのコード化の考え方を示しておく．従来は一人目の行員が帳票内容をキーボード入力（一次入力）していたオペレーションを自動認識ソフトウェア[18]で代替し，二人目の行員が帳票のイメージデータを見てその内容をキーボード入力（二次入力）する．両者の入力内容を突合し，同じであればそれを正しいコードデータとして扱う．違っていた場合，例外処理として，第三者（役席行員など）が両者の入力内容と帳票のイメージデータをもとに正しいコードデータを入力する．使用する認識ソフトウェアは，その認識アルゴリズムに方向性特徴を用いた統計的識別方式[19]を採用している．認識率は，活字で 99%，JIS規格に準拠した手書き文字で 97%である．

Fig.4 に，集中事務処理システムの概要を示す．本システムは，営業店の非接触イメージスキャナを設置した窓口端末，集中事務処理センタのワークフローサーバと事務オペレーションを行うクライアント PC から構成される．窓口端末の非接触イメージスキャナで，顧客から受け取った帳票をイメージデータとして読み込む．このとき，同時に文字認識を自動実行して，従来は人手で行っていた帳票の形式点検（記入漏れチェックで Fig.1 の Formal check）や一次入力（記入内容のコードデータ化で Fig.1 の Temporary input）を自動化する．読み込んだ帳票イメージデータと自動認識した記入内容のコードデータを，集中事務処理センタのワークフローサーバに伝送する．ワークフローサーバは，受け取ったデータを案件として管理し，ワークフロー定義に従ってクライアント PC に事務オペレーションの実行を指示し，案件の進捗を制御する．クライアント PC では，事務内容に応じた帳票処理クライアントアプリケーションが動作し，帳票精査（帳票内容の二次入力を行い，一次入力内容との突合せを行うコードデータのチェックで Fig.4 の Verification），印鑑照合（Fig.4 の Seal check）や検印（取引実行可否の判定で Fig.4 の Execution check）などの後方事務処理を行う．このように，営業店後方事務を事務集中処理センタに集約することで，事務の合理化・効率化を図っている．

事務集中処理センタのクライアント PC の操作者は事務行員や役席行員で，クライアント PC に対するオペレーションをできる限り簡略化するために，前章で述べた帳票内の各項目に設定されている業務ロジックや項目間の関連チェックなどは，帳票処理クライアントアプリケーションに実装する．このとき，帳票ごと個別にアプリケーションを開発していくと，似て非なるものを多数開発することになり，開発コストが大きくなってしまう．そこで，次節で提案するアプリケーションアーキテクチャで，開発生産性を向上させている．

3.2 帳票処理クライアントアプリケーションアーキテクチャ

3.2.1 コンポーネント接続方式

提案するコンポーネント接続方式は，①従来のようにコンポーネント間を直接接続する（コンポーネントから別のコンポーネントを呼び出す）のではなく，仲介者（データオブジェクトを持ち込んだイベントアダプタ）を介して間接的に接続する，②コンポーネントと仲介者とのインタフェース仕様を共通化することにより仲介者の汎用性を持たせる，という考え方に基づいている．従来のような直接接続では，接続元コンポーネントは接続先コンポーネントのインタフェース仕様が分っていなければならず，それが自分の仕様を決定づけている．すなわち，依存関係を持っている．これに対して提案方式では，各コンポーネントは汎

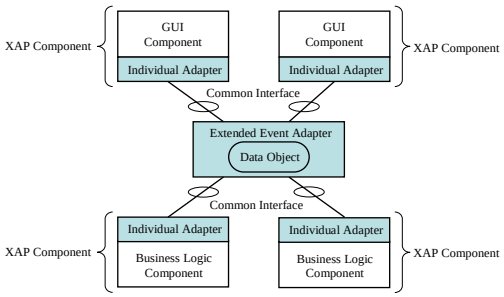

Fig.5 A proposed method of component connection

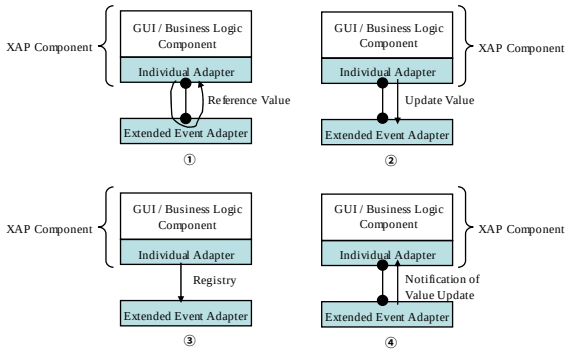

Fig.6 Interfaces between XAP component and Extended Event Adapter

用的な仲介者とのインタフェース仕様だけで自分の仕様を決定づけることができ，接続先コンポーネントの仕様とは直接的な関係がない緩やかなコンポーネント間接続を実現している．これにより，コンポーネントの独立性と再利用性を高めている．

提案するコンポーネント接続方式を，**Fig.5** に示す．本方式では，コンポーネント間を直接接続するのではなく，イベントアダプタにデータオブジェクトを内包した拡張イベントアダプタを導入し，コンポーネントと拡張イベントアダプタに内包されたデータオブジェクト（以下では，単にデータオブジェクトを記述する）との接続関係で間接的にコンポーネント間を接続する．各コンポーネント，すなわち GUI コンポーネントや業務コンポーネントの実装方法は規定しないが，各コンポーネントが有する個々のインタフェースを Fig.6 に示す共通インタフェースに変換する機能を持つ個別アダプタを用意する．この個別アダプタを有するコンポーネントを XAP(XML Applications)コンポーネントと呼ぶことにする．

Fig.6 に示すデータオブジェクトと XAP コンポーネント間の 4 つの共通インタフェースは，

①XAP コンポーネントからデータオブジェクトの値を参照するメソッド
②拡張イベントアダプタに登録された XAP コンポーネントから当該データオブジェクトに設定されたデータ値を更新するメソッド
③拡張イベントアダプタに XAP コンポーネントを登録するメソッド
④データオブジェクトのデータ値が更新された場合，当該拡張イベントアダプタに登録された XAP コンポーネントにデータ値の更新通知を行うメソッド

である．①は，XAP コンポーネント動作時に他の XAP コンポーネントにより更新されたデータオブジェクトのデータ値を参照する場合に用いられるメソッドである．②～④は，XAP コンポーネントの動作を契機に他の XAP コンポーネントを動作させたい場合に用いられるメソッド群である．従来では，このような場合，コンポーネント中に他のコンポーネントを動作させるためのイベントハンドラを記述するが，提案方式では XAP コンポーネント間で直接の依存関係を持たないため，データオブジェクトを介しての通知メカニズムとなる．Fig.2 に示した銀行の為替振発業務は，提案接続方式によれば **Fig.7** のようになる．

提案接続方式により，コンポーネント間の依存関係を緩やかにでき，コンポーネントの高い再利用性が得られる．また，カスタマイズ発生時は，カスタマイズ対象 XAP コンポーネントとデータオブジェクトとの接続関係を変更するだけでよく，カスタマイズ範囲の局所化を図ることができる．アプリケーション開発者は，画面を構成する XAP コンポーネントのデータオブジェクトを介した間接的接続関係と画面遷移フローだけを定義すればよい．この定義言語に Web 環境との親和性を考慮し，XML を適用することで提案方式のオープン性とユーザによる記述容易性を持たせている．

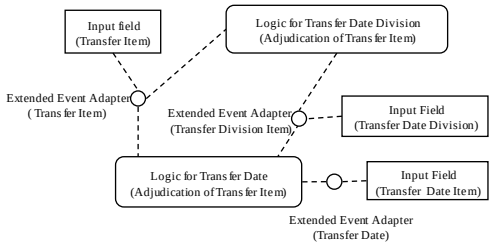

Fig.7 Example of relations among XAP components and Extended Event Adapters

3.2.2 XAP アプリケーション構成

XAP コンポーネントの組み合わせでできあがるアプリケーションを XAP アプリケーションと呼ぶことにする．XAP アプリケーションは，**Fig.8** に示すように，アプリケーションフレームワークとして提供される XAP コア，XAP コンポーネント群（GUI コンポーネントと業務ロジックコンポーネント），画面定義，フロー定義及び Web ブラウザから構成

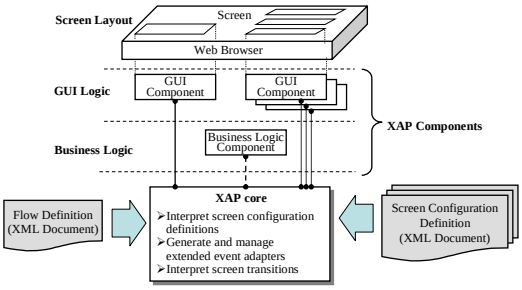

Fig.8 Configuration of XAP application

される．XAP アプリケーション開発では，GUI コンポーネントと業務ロジックコンポーネントの接続情報によって作られる画面を記述する画面定義と，画面遷移を記述するフロー定義を XML 文書でそれぞれ作成することになる．

画面定義は，XAP コンポーネントとデータオブジェクトの接続関係を XML 文書で記述したものである．例として，Fig.7 の接続関係を記述した画面定義を **Fig.9** に示す．画面定義のために用意された XML タグを，**Table 1** にまとめる．

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<XScript>
<XApplet name="振込種目フィールド">
  <XComponent name="入力フィールド" />
  <Parameter name="Text" value="EventAdapter;振込種目" />
</XApplet>
<XApplet name="振込指定日区分フィールド">
  <XComponent name="入力フィールド" />
  <Parameter name="Text" value="EventAdapter;振込指定日区分" />
</XApplet>
<XApplet name="振込指定日区分判定">
  <XComponent name="振込指定日区分判定ロジック" />
  <Parameter name="振込種目" value="EventAdapter;振込種目" />
  <Parameter name="振込指定日区分" value="EventAdapter;振込指定日区分" />
</XApplet>
<XApplet name="振込指定日フィールド">
  <XComponent name="入力フィールド" />
  <Parameter name="Text" value="EventAdapter;振込指定日" />
</XApplet>
<XApplet name="振込指定日判定">
  <XComponent name="振込指定日判定ロジック" />
  <Parameter name="振込種目" value="EventAdapter;振込種目" />
  <Parameter name="振込指定日区分" value="EventAdapter;振込指定日区分" />
  <Parameter name="振込指定日" value="EventAdapter;振込指定日" />
</XApplet>
</XScript>
```

Fig.9 Example of a screen configuration definition

Table 1 XML tags used in the screen configuration definition

Tag name	Attribute	Description
XScript		Entire screen configuration definition
XApplet		Definition of each XAP component
	name	Name of XAP component instance
XComponent		Definition of component program
	name	Name of component program
Parameter		Parameter of XAP component
	name	Parameter name
	value	Parameter value

フロー定義は，XAPアプリケーションの画面遷移フローをXML文書で記述したものである．フロー定義の各タグには，画面遷移フローに関する制御コマンドが定義されている．XAPコアが，SAX(Simple API for XML)[16]パーサの作法に従い，フロー定義のXML文書の木構造を解析しながらタグの制御コマンドを実行していく．**Fig.10** に示す画面遷移フロー例に対するフロー定義を**Fig.11** に示す．FLOWDEFタグは，フロー定義全体を意味する．XAPコアは，このルートタグであるFLOWDEFから処理を開始し，最初の子ノードであるTransactionタグを実行する．Transactionタグは画面表示の制御コマンドで，この例では，XAPコアは”MainMenueForm”という名称を持つ画面定義によって定義される画面を表示する．フロー定義のために用意されたタグ一覧を，**Table 2** に示す．

XAPコアは，XML文書で定義された画面定義の解釈，拡張イベントアダプタのインスタンス生成と管理やフロー定義に従い画面遷移を実行するアプリケーションフレームワークである．**Fig.12** にXAPコアの機能構成を示す．フロー定義を読み込み解釈するのが「フロー定義パーサ」で，フロー定義XML文書をDOM(Document Object Model)[17]の木構造としてメモリ上に展開する．展開された木構造の各ノードを実行管理するのが「フロー制御エンジン」である．ノードが画面定義（Transactionタグ）の場合は「画面定義パーサ」に読み込むべき画面定義を指示し，ブラウザから画

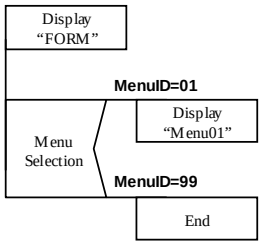

Fig.10 Example of a screen transition flow

```
<?xml version="1.0" encoding="Shift_JIS"?>
<FLOWDEF>
  <Transaction layout="MainMenuFrom"/>
  <SWITCH DID="MenuID">
    <CASE VALUE="01">
      <GOSUB NAME="Menu01"/>
    </CASE>
    <CASE VALUE="99">
      <End/>
    </CASE>
  </SWITCH>
</FLOWDEF>
```

Fig.11 Flow definition corresponding to the screen transition shown in Fig.10

Table 2(a) XML Tags used in flow definition (flow control)

Tag name	Attribute	Description
FLOWDEF		Root
SUB		Declaration of subroutine
	NAME	Name of subroutine
EXITSUB		Termination of subroutine
GOSUB		Subroutine call
	NAME	Name of called subroutine
SWITCH		Conditional branch
	DID	Conditional name of event adapter
CASE		Branch process
	VALUE	Data constant for condition
DEFAULT		Branch process
LOOP		Loop process
EXITLOOP		Termination of loop process
TRANSACTION_NODATA		Process in case of no data

Table 2(b) XML Tags used in flow definition (communication control)

Tag name	Attribute	Description
Upload		Transmission to server
	message	Name of telegram
Transaction		Communication with server
	message	Name of telegram
	layout	Name of screen configuration definition
	free	Data clear
Queue		Store in queue
	message	Name of telegram
	layout	Name of screen configuration definition
	size	Look-ahead number
Dequeue		Take off from queue
	message	Name of telegram
	layout	Name of screen configuration definition
Free		Release data
	message	Name of telegram
Error		System error
	message	Error message
End		End of system

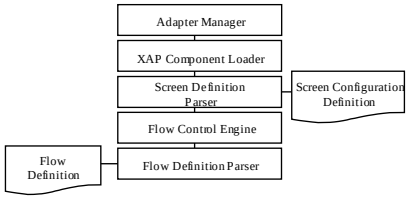

Fig.12 Configuration of the XAP-core

面終了が通知されると次のノードを実行する．「画面定義パーサ」は指示された画面定義を解釈し，画面定義内のXAppletタグ（個々のXAPコンポーネントに対する定義内容を表す）ごとにXComponentタグ（XAppletタグで定義されたXAPコンポーネントの実体としてXAPコアにより起動されるXAPコンポーネント名を表す）で指定されたXAPコンポーネントのインスタンスの設定パラメータ（Parameterタグのname属性で指定されたパラメータ名とvalue属性で指定されたパラメータ値のセット）を抽出し，「XAP部品ローダ」へ通知する．「XAP部品ローダ」は，上記パラメータに基づきXAPコンポーネントを呼び出す．

以上のように，XAP コンポーネントはユーザの GUI 操作などを契機として動作する．また，XAP コンポーネントは，データオブジェクトのデータを更新する．データオブジェクトのデータに更新があった場合，「XAP 部品ローダ」はその拡張イベントアダプタに登録されている XAP コンポーネントすべてに対して，データオブジェクトの更新イベントを通知する．更新イベントを受け取った XAP コンポーネントは，その仕様に基づき動作する．このようにして，たとえば GUI 操作→GUI コンポーネント→業務ロジックコンポーネント→他の GUI コンポーネント→…，のような状態変化により，XAP アプリケーションの動作を実現している．

Fig.13 に，XAP アプリケーションの実装概要を示す．本図は，入力必須チェックロジックの実装例であり，”Input Data” 拡張イベントアダプタのデータオブジェクトに文字列が設定されていない場合，”Message ID”拡張イベントアダプタのデータオブジェクトに設定されているエラーガイダンス ID を，”Guidance”拡張イベントアダプタのデータオブジェクトに設定する機能を示したものである．

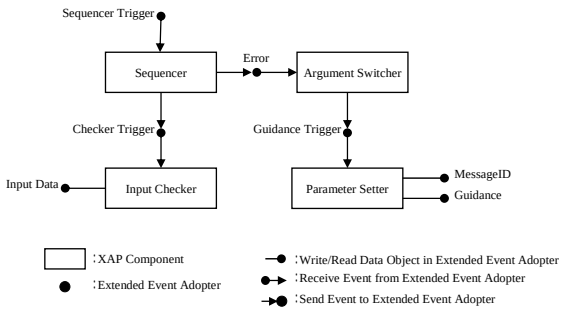

Fig.13 Example of implementation overview based on the proposed architecture

4. 適用評価

提案したシステム化方式を採用して，営業店の後方事務の集中事務処理システムを開発した．**Fig.14** は，集中事務処理システムを含む銀行営業店システム全体の概要を示したものである．3.2 節で提案したクライアントアプリケーションアーキテクチャは，集中事務センタ内の為替業務クライアントアプリケーションで採用している．その開発内容を **Table 3** に示す．アプリケーションのステップ数は，XML 文書タグを 1 ステップとしている．**Fig.15** は，為替振発業務の一次入力オペレーション画面例である．

提案システム化方式により開発したシステムの導入による営業店後方事務の集約効果として，営業店における事務量を 75%～55%に削減することができている評価結果を得た．また，事務要員も 20%程度の削減を達成している．こ

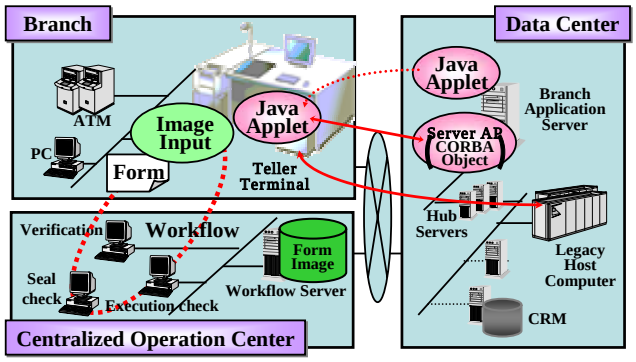

Fig.14 Overview of the banking branch system

Table 3 Contents of development

XAP Component / Business Operation	Category	GUI Parts	Cursor-control Parts	Strings Editing Parts	Date Processing Parts	Logic Control Parts	Logic Operation Parts
	Number Steps	10	1	12	4	22	4
Single Money Transfer: Temporary Input of Money Transfer	14516	used		used	used	used	used
Single Money Transfer: Verification of Temporary Input Data	12789	used		used	used	used	used
Single Money Transfer: Money Transfer Holding	9585	used		used	used	used	used
Single Money Transfer: Verification of Money Transfer	6529	used		used	used	used	used
Multiple Money Transfers: Temporary Input of Money Transfer	18405	used	used	used	used	used	used
Multiple Money Transfers: Verification of Temporary Input Data	15908	used	used	used	used	used	used
Multiple Money Transfers: Money Transfer Holding	12855	used	used	used	used	used	used
Multiple Money Transfers: Verification of Money Transfer	7914	used	used	used	used	used	used

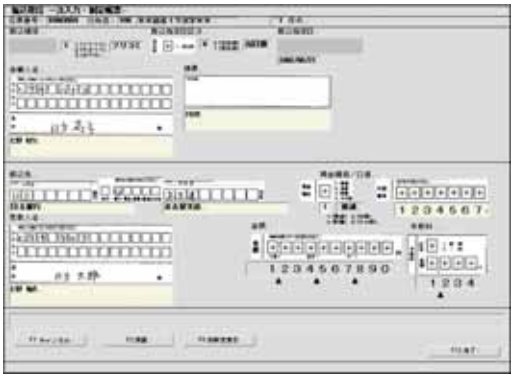

Fig.15 Example of a screen developed by using our architecture

れにより，提案システム化方式による事務の合理化効果を定量的に確認した．本システムは 7 行の銀行に導入済みであり，いずれの銀行においても同等の定量的合理化効果を得ている．

つぎに，提案した帳票処理クライアントアプリケーションアーキテクチャの有効性を検証する．

(1) まず，4GL ビジュアル開発ツール(Visual Basic)と開発生産性に関して比較実験を行なった．開発生産性の比較ではソースコードのステップ数による比較が一般的であるが，提案アーキテクチャではアプリケーションをソースコードではなく XML 文書で記述するため，ソースコードのステップ数による直接比較が困難である．このためカスタマイズ比率を評価基準として用いることにした．カスタマイズ比率とは，アプリケーションのカスタマイズに要したステップ数とその母体アプリケーションのステップ数との比率である．したがって，尺度の異なる開発言語の開発生産性比較に用いることができる．比較実験は，銀行営業店システムの標準仕様版と，ある銀行向けにカスタマイズを行った仕様版を対象に, Fig.15 に示した業務画面を取り上げて実施した．ある銀行向けに行ったカスタマイズの内容は，手数料区分や手数料現金振替など 6 つの項目追加と，手数料区分と種目（電信扱い／文書扱いなど）で整合性がとれているかなどの各項目に対するチェックロジックの追加である．

評価結果を **Table 4** に示す. ここでは 4GL ビジュアル開発ツールのステップ数はソースコードの 1 行を 1 ステップと換算し，提案アーキテクチャでは XML 文書タグを 1 ステップと換算した．Table 4 によれば，アプリケーション全体のステップ数は，提案アーキテクチャの方が一桁大きくなっている．提案アーキテクチャでは，たとえば，4GL ビジュアル開発ツールでは 1 ステップで記述できる複数演算子含む計算式などは各演算子とデータ（拡張イベントアダプタ）との接続関係をすべて XML 文書で記述する必要がある．このため，1 ステップの記述能力の違いにより，アプリケーション全体のステップ数（XML 文書量）が多くなっている．これは，提案アーキテクチャによる初期開発コストは 4GL ビジュアル開発ツールに比べ大きくなることを表している．

カスタマイズにおける提案アーキテクチャの効果を考察する．まず，アプリケーション全体ステップ数に対する，カスタマイズにより生じたコンポーネントの修正／追加／削除に要したステップ数（カスタマイズステップ数）の割合である，カスタマイズ比率を評価する．4GL ビジュアル開発ツールのカスタマイズ比率が 0.584 であるのに対して，

Table 4 Evaluation results

	4GL visual tool		Proposed architecture	
	Standard specification	Custom specification	Standard specification	Custom specification
Total number of steps	1125	1429	11369	14516
Increase in steps due to customization	1.27		1.23	
Number of customized steps	657		4242	
Number of component deletion steps	54		69	
Number of component modification steps	94		391	
Number of component addition steps	509		3782	
Customization ratio	**0.584**		**0.373**	

提案アーキテクチャではその約 64%にあたる 0.373 となっている．これは，提案アーキテクチャでは，カスタマイズに直接関連するコンポーネントと拡張イベントアダプタとの接続関係だけを変更（削除／修正／追加）すればよいのに対して，4GL ビジュアル開発ツールでは GUI と業務ロジックの分離ができていないのでカスタマイズに関わる変更部分が複雑であるためと考察できる．一方，カスタマイズコストに関してカスタマイズステップ数で評価すると，提案アーキテクチャでは，全体ステップ数が大きいのでカスタマイズステップ数も 4GL ビジュアル開発ツールに比べて大きく，カスタマイズ時のコスト削減効果は得られていない．しかし，銀行営業店システムの帳票処理アプリケーションを開発していく過程で，業務数（アプリケーション数）が増えていくほど，新規に開発しなければならない XAP コンポーネントの数が減少する傾向が見られた．これは，XAP コンポーネントの独立性から，XAP コンポーネントの再利用性が高いことを示していると考えられる．この結果，カスタマイズステップ数の多くを占めている表 4 の追加コンポーネントステップ数が減少するとともに，カスタマイズ比率も小さくなる結果を得ている．提案アーキテクチャを採用して開発した為替業務機能全体で見ると，試算ベースで，4GL ビジュアル開発ツールに比べ約 60%にカスタマイズステップ数，すなわちカスタマイズコストを抑えられると推定できる．本システムの適用銀行が増えれば，さらにカスタマイズコストを抑えられると考える．

(2) つぎに，同じコンポーネント指向の開発言語である BML と開発生産性の比較実験を行なった．実験内容は，4GL ビジュアル開発ツールと比較した場合と同じ，為替振発業務のアプリケーションのカスタマイズを対象にした．結果は，BML の方が，Table 4 に示す提案アーキテクチャの数値より約 10%小さい数値が得られた．これは，ある銀行向けに行ったカスタマイズは，プログラム上，他のコンポーネントとの依存関係がほとんどないコンポーネントの新規追加であったためと判明した．あるコンポーネントが独立で他のコンポーネントと依存関係がないものと仮定し，そのコンポーネントを別のコンポーネントと接続することを考える．提案アーキテクチャでは，拡張イベントアダプタを介して間接的にコンポーネントを接続するので，双方のコンポーネントに関しての接続記述が必要である．これに対して，BML ではその接続に関する記述のみとなるため，接続情報に関しては，BML は提案アーキテクチャの 1/2 になる．XML 定義文書の中でコンポーネントの接続情報が占める割合は 20〜30%なので，接続情報の記述量は約 10〜15%ほど BML の方が提案アーキテクチャより少なくなると分析できる．したがって，この実験で約 10%小さい結果を BML が示したことになる．

帳票業務アプリケーションのカスタマイズでは，前述のように単純に項目や業務ロジックの追加ですむ場合は少ない．多くの場合は，2 章で述べたように各項目間に複雑な関連があるものをカスタマイズすることになる．その一例に，為替の件数や金額を集計する集計照会業務がある．この業務例では，ある項目値が更新されると合計値を更新するロジックが起動されて合計値が更新される仕様を持つ．この業務アプリケーションでのこれまでと同じ銀行向けに行ったカスタマイズ（合計ロジックの仕様が変更）では，提案アーキテクチャは BML に比べて約 46%のカスタマイズステップ数ですんでいる．提案アーキテクチャでは合計ロジックコンポーネントの接続関係だけを変更すればよいのに対して，BML では合計ロジックコンポーネントのほかに関連する各項目のコンポーネントについても変更の必要があるため，提案アーキテクチャの方がカスタマイズ量は少なくなっている．

以上の生産性検証評価の結果，本稿で提案したアプリケーションアーキテクチャの有効性を確認できたと考える．

5. おわりに

本稿では，銀行営業店における事務合理化の背景と課題について述べ，2 つのシステム化方式を提案した．まず，多くの人手と時間を要している営業店の後方事務の合理化を図る目的で，非接触イメージスキャナで帳票をイメージデータとして直接コンピュータ内に取り込み，ワークフローによって集中事務処理センタに伝送して後方事務を集中処理するシステム化方式を提案した．これにより，営業店の後方事務の合理化と事務コストの低減を図ることができる．次に，集中事務処理センタにおける多種多様な帳票を処理するアプリケーションの開発コストの低減を目的に，疎結合コンポーネント接続方式を導入し，画面および画面遷移を XML 文書で記述するだけでアプリケーションを開発できる帳票処理クライアントアプリケーションアーキテクチャを提案した．提案アーキテクチャにより，コンポーネントの再利用性が高まり，カスタマイズ範囲を局所化でき，ユーザでも容易にアプリケーションを開発することができる．

提案システム化方式により，銀行営業店の後方事務の集中事務処理システムを開発した．本システムの導入による事務の合理化効果を定量的に確認した．また，提案アプリケーションアーキテクチャの生産性検証評価を実施し，その有効性を確認した．開発したシステムは，現在，性能面でも問題なく安定稼動している．

参 考 文 献

1)金融財政事情，1997.7.7 号 (1997-7)
2)http://xml.coverpages.org/xfa.html
3)http://www.w3.org/1999/05/XFA/xfa-template.html
4)青山，中所，向山編：コンポーネントソフトウェア，ソフトバンク (1998)
5)http://e-wop.jp/w/4GL.html
6)http://www.alphaworks.ibm.com/tech/bml
7)http://www.javaworld.com/javaworld/jw-08-1999/iw-08-beans.

html
8)http://www.javaworld.com/javaworld/jw-10-1999/iw-10-beans.html
9)http://www/cag.lcs.mit.edu/curl/
10)M.Hostetter, D.Kranz, C.Seed, C.Terman and S.Ward: “Curl: Gentle Slope Language for the web”, World Wide Web Journal, Vol. II, Issue 2, Spring (1997)
(http://www/w3j.com/6/s3.kranz.html)
11)N.Damle, G.Gray, B.Mount: Curl Programming Bible, Hungry Minds Inc. (2002)
12)James Gosling, Bill Joy and Guy Steele: The Java Language Specification, Sun Microsystems (1997)
13)Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides: Design Patterns, Elements of Reusable Object-Oriented Software, pp.173-182, Addison Wesley (1995)
14)http://java.sun.com/webservices/docs/1.0/tutorial/doc/JAXPSA.html
15)http://www.w3c.org/DOM/
16)染谷，森，阿部，町田，長谷川：「第 4 次銀行営業店システムにおける XML を適用した帳票処理アプリケーションアーキテクチャの開発」，電気学会情報システム研究会，IS-03-26，49/52 (2003-9)
17)Harushi Someya, Yuichi Mori, Masahiro Abe, Isamu Machida, Atsushi Hasegawa and Osamu Yoshie: Development of a Client Application Architecture Using XML in a Bank Branch System, IEEE International Symposium on Communications and Information Technologies (ISCIT 2004), 204/209 (2004-10)
18)http://www.hitachi-omron-ts.co.jp/products/gazou/003.html
19)Cheng-Lin Liu, Hiroshi Sako and Hiromichi Fujisawa: Effects of Classifier Structures and Training Regimes on Integrated Segmentation and Recognition of Handwritten Numeral Strings, IEEE Trans. Pattern Analysis and Machine Intelligence, **26**-11, 1395/1407 (2004-11)

[著 者 紹 介]

染 谷 治 志

（正会員）1962 年 3 月 1 日生。1986 年早稲田大学大学院理工学研究科電気工学専攻修士課程終了。同年，（株）日立製作所に入社。システム開発研究所にて，事象駆動型システムのモデリング，ストレージ応用ソリューション，金融情報システムの研究開発およびソフトウェア生産技術の研究開発に従事。現在，早稲田大学大学院情報生産システム研究科博士後期課程に在学中。電気学会，情報処理学会，電子情報通信学会の各会員。

森 有 一

1991 年大阪大学大学院工学研究科通信工学専攻修士課程終了。同年，（株）日立製作所に入社。同社システム開発研究所にて，DB検索インタフェースワークフローシステム，金融情報システムの研究開発に従事。情報処理学会会員。

阿 部 正 弘

1993 年大阪大学大学院基礎工学研究科情報工学専攻修士課程終了。同年，（株）日立製作所に入社。同社システム開発研究所にて，金融情報システムの研究開発に従事。

町 田 勇

1981 年日本大学理工学部電気工学科卒業。同年，（株）日立製作所に入社。現在，日立オムロンターミナルソリューションズ（株）にて，端末ソフトウェアの設計開発に従事。

長 谷 川 篤

1979 年（株）日立製作所に入社。情報通信グループ金融システム事業部チャネルソリューションセンタにて，金融機関の営業店自動機ソリューションの取り纏め業務に従事。技術士（情報工学部門）。

吉 江 修

1987 年早稲田大学大学院理工学研究科博士課程退学。早稲田大学理工学部助手，東京理科大学理学部助手・講師を経て，現在早稲田大学大学院情報生産システム研究科助教授。専門は生産情報システム，コミュニティコンピューティング。1985 年計測自動制御学会学術奨励賞受賞，1990 年電気学会学術振興論文賞受賞（小平記念賞受賞），1996 年日本設備管理学会学会賞論文賞受賞，2002 年電気学会電子・情報・システム部門特別貢献賞，同大会企画賞受賞。工学博士。