社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE
SS2002-46(2003-1)

非オブジェクト指向のオンラインシステムに対するコンポーネントウェア組込手法の検討

岡久 浩章[1] 飯田 元[2] 米光 哲哉[3] 福地 豊[3] 松本 健一[1]

1 奈良先端科学技術大学院大学 情報科学センター, 2 奈良先端科学技術大学院大学 情報科学研究科

〒630-0192 けいはんな学研都市

3 日立製作所 情報・通信グループ 生産技術本部 〒136-8632 東京都江東区新砂 1-6-27

E-mail: 1,2 {iida@itc, hiroa-o@is, matumoto@is}.aist-nara.ac.jp, 3 {t-yonemitu, fukuchi}@itg.hitachi.co.jp

あらまし 本稿では，オブジェクト指向アーキテクチャを持たないオンラインシステムに対して，アーキテクチャに大きな変更を加えずにコンポーネントウェアを組込む為のブリッジ機構と，その機構に基づいた系統的設計手法を提案する．本機構は，既存システムの制御モジュールやコンポーネント呼び出し専用モジュールのプログラムコードを改変せずに，コンポーネントの追加・変更・削除を行うことが可能である．さらに，非オブジェクト指向オンラインシステムのサンプルに本手法を用いて既存ソフトウェア部品とコンポーネントを試験的に実装し，それらの併用が容易に実現できることを確認した．

キーワード コンポーネントウェア，オンラインシステム，ソフトウェアアーキテクチャ

A Componentware Incorporation Technique for Online Systems with Non-Object-Oriented Legacy Architecture

Hiroaki OKAHISA[1] Hajimu IIDA[2] Tetsuya YONEMITSU[3] Yutaka FUKUCHI[3] and Ken'ichi MATSUMOTO [1]

1,2 {Graduate School of Information Science, Information Technology Center}, Nara Institute of Science and Technology, Kansai Science City, 630-0192 Japan

3 Information & Telecommunication Systems, Hitachi.Ltd, 1-6-27 Shinsuna, Koutou-ku, Tokyo, 136-8632 Japan

E-mail: 1,2 {iida@itc, hiroa-o@is, matumoto@is}.aist-nara.ac.jp, 3 {t-yonemitu, fukuchi}@itg.hitachi.co.jp

Abstract This paper proposes a bridge mechanism for incorporating componentware to online systems without object-oriented architecture, and also proposes the systematic design process based on the mechanism. This technique can be applied without big changes to the existing architecture. Adding, changing, and deleting components with this method don't affect the code of the main control module or the component manipulation module. The component incorporation design process extended from the conventional design process is also proposed. Experimental implementation shows that this method is actually effective in combining componentware and conventional software parts to the legacy (non-object-oriented) online systems at a time.

Keyword componentware, online system, software architecture

1. はじめに

1.1. コンポーネントウェアを用いたシステム開発

近年，再利用が簡単に行えるソフトウェア部品として，コンポーネントウェア(以下，単にコンポーネントと呼ぶ)の規格が開発され，幅広く利用されるようになってきた．コンポーネントとは，ソフトウェアの一部を機能毎に分解して保存したもので，ある特定のインタフェース規約に従った，自己完結型のソフトウェア部品である[1]．コンポーネントは通常バイナリ形式で保存されており，共通インタフェースにより動的に組込んで処理を行うことができる．この共通インタフェースの仕様は一般にコンポーネントモデルと呼ばれており，基本的に，コンポーネントモデルに準拠したソフトウェアアーキテクチャにコンポーネントを組込む形で再利用が行われる[2]．コンポーネントは単独での動的な生成が可能で，一定の内部隠蔽性を保ちながら，提供する機能を再利用できる．

コンポーネントアーキテクチャがもたらす効果と

して，以下のような利点が挙げられる[3].

- 既存コンポーネントの再利用により，ソフトウェア開発期間の短縮と，コスト削減が可能となる
- 信頼性の高いコンポーネントを組込むことにより，ソフトウェア信頼性が向上する
- コンポーネントの置換・修正作業により，要求仕様や運用環境の変化に対し，容易に対応が可能である

広く用いられているコンポーネントモデル/アーキテクチャの例として，ActiveX/COM/DCOM[4]，JavaBeans[5]，EJB[6]，CORBA/CCM[7]などが存在するが，これらは基本的にオブジェクト指向に基づいて設計されており，コンポーネントはある機能を実現する為の関連オブジェクトをまとめてカプセル化した物としてモデル化されている．したがって，オブジェクト指向アーキテクチャを持つシステムでの利用は容易であるが，それ以外のシステムでは，必ずしも容易に利用できない構造となっている．

1.2. オンラインシステムの従来アーキテクチャ

一方，ハードウェアの性能向上，ネットワークの拡大に伴い，いわゆるオンラインシステムの社会的影響度と，それを実現するソフトウェアの開発量は増大し続けており，品質，生産性に対する要求は一段と厳しくなっている[8].

しかし，膨大な量のプログラムが要求される大規模な基幹業務系のオンラインシステム開発においては，従来蓄積してきたソフトウェア部品や，開発手法などの有形・無形の資産を継承する必要があるため，既存アーキテクチャの抜本的な変更を行うことが困難である．

この為，オブジェクト指向アーキテクチャを持たない従来アーキテクチャによるオンラインシステムの開発においては，コンポーネントの利用は非常に困難であった．

2. コンポーネント利用機構の提案

そこで，本研究では，日立製作所において開発され，長年利用されてきたオンラインソフトウェアのアーキテクチャと開発法を分析し，従来のアーキテクチャに極力変更を加えずにコンポーネントの組込み・利用を可能とする為の機構を考案・実装した．さらに，その機構に基づいた系統的設計手法についても検討を行った．

2.1. 要件

非オブジェクト指向システムへのコンポーネント組込みを実現する為に我々が考慮した要件は，大きくは次の２つである．

1) 前述のように，既存オンラインシステムについては極力アーキテクチャに変更を加えないことが望ましい．つまり，既存方式でのソフトウェア部品はそのまま利用可能とした上で，コンポーネントを組み込んで利用できる仕組みが必要となる．
2) 利用するコンポーネントを追加・変更・削除する際に必要な改変は，新たに拡張される部分にのみ限定されることが望ましい．

更に，上記の要件を満たすコンポーネントの組込みを実現できたとしても，コンポーネントを利用したソフトウェア設計のガイドラインが与えられなかったり，設計プロセスが従来のものから大きくかけ離れたりしてしまうと，従来アーキテクチャによる設計法に習熟した開発者にとって開発が困難なものとなってしまうので，提案する機構に基づいた系統的な設計手法を用意することが必要である．

2.2. 設計方針

上記要件に基づき，コンポーネント組込み機構について，以下のような設計方針をたてた．

- **ソフトウェア部品自体は改変しない**：従来アーキテクチャに基づく業務用部品は，改変せずにこれまでと同様に利用できるようにする．また，コンポーネントの組み込みについても同様に，特別な改変を必要とせず，標準仕様に準拠するものをそのまま利用できるようにとする．
- **各コンポーネントの操作は専用モジュール経由に限定する**：各コンポーネントは非オブジェクト指向アーキテクチャとコンポーネントとの界面となる専用モジュールを通じて操作する．このとき，コンポーネントの追加・変更・削除を行う際にも，この専用モジュールや従来アーキテクチャ部分の制御モジュールのコード改変を必要としないようにする．
- **データ授受を実現するイベントハンドラを用意する**：従来システムとコンポーネント間のデータのやりとりは，メッセージ及び入出力フィールドの相互変換機構を組み込んだイベントハンドラを経由させる．
- **動的リンクを実現するブリッジ機構を用意する**：組込むコンポーネントのインタフェース（リンクのための情報）を参照して動的にコンポーネントを生成・リンクし，呼び出しを行うためのコンポーネントブリッジを，非オブジェクト指向アーキテクチャからの起動が可能なネイティブコードにより実装する．

2.3. 機構の概要

以上の設計方針に基づいた，コンポーネント組み込みのための拡張機構の概要を図.1 に示す．従来システムにおいて，ソフトウェア部品の呼び出しを制御する

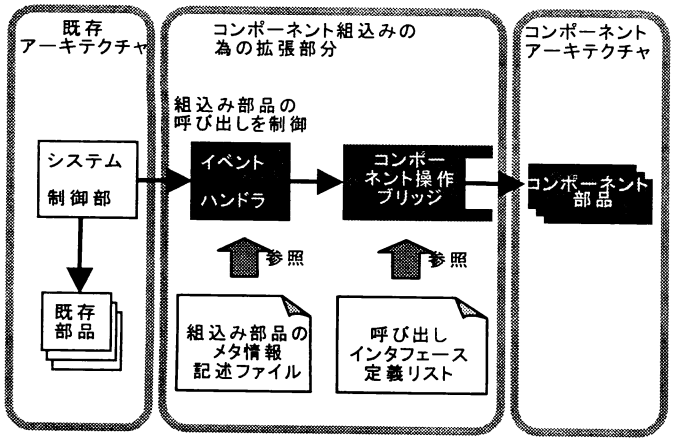

図.1 コンポーネント部品組込み機構

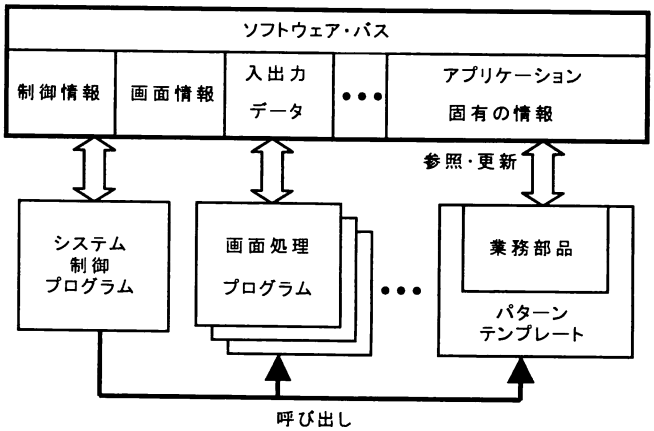

図.2 ソフトウェア・バス方式

モジュールを，ここでは**システム制御部**と呼ぶことにする．コンポーネントの呼び出しを制御する**イベントハンドラ**をシステムに追加し，イベントハンドラの呼び出しを行うように，システム制御部に対して変更を加えた．

イベントハンドラは，呼び出し時にシステム制御部から渡されたイベント内容と，組込み部品に関するメタ情報の記述されたファイルを参照し，必要に応じてコンポーネント操作用の専用ブリッジモジュールを呼び出す．

コンポーネント操作ブリッジはコンポーネントの動的生成・呼び出しに必要な情報をインタフェースの定義リストから取得し，組み込み部品を呼び出して処理を行わせる．組み込み部品の追加・変更・削除はイベントハンドラとコンポーネント操作ブリッジそれぞれが参照するデータファイルを差替えることで行われる．

以上のような拡張により，既存アーキテクチャから直接呼び出すことが困難だったコンポーネントを容易に利用することが可能となる．

3. コンポーネント利用機構の実現手法

3.1. 対象システムのアーキテクチャ

本研究では具体的な対象システムとして，ソフトウェア・バス方式に基づく画面フロー指向のオンラインシステムを取り上げた．

ソフトウェア・バス方式とは，制御プログラムや各プログラム部品に至る全てのプログラムから必要に応じて参照・変更を受け入れる共有データ領域(ソフトウェア・バス)を確保し，これを用いて各プログラム間のデータのやりとりを行う方式のことである(図.2)．画面フロー指向アーキテクチャとは，処理内容を表示画面単位で分割し，プログラムを画面ごとに部品化することで再利用と保守を行うアーキテクチャである．

日立製作所では，本アーキテクチャに基づく大規模オンラインシステムを多数開発しており，これまではCOBOL を用いた自動生成スケルトンに準拠した部品化と再利用が行われてきている[8]．

3.2. ブリッジ機構の実現

本機構が対象とするシステムは，プログラムコードを記述する言語として COBOL を採用しており，直接コンポーネントを呼び出す手段を持たないが，C 言語でコーディングされた動的リンクライブラリ(DLL)を呼び出す機能を有する．そこで，システム本体とコンポーネント間のブリッジ機構を C 言語で追加実装し，これを呼び出すようにシステム本体の基本制御プログラムを改変する事で，間接的にコンポーネント呼び出しを実現した．

具体的には，追加する機構として，

1) システム本体を構成する COBOL プログラムからコンポーネントを呼び出す為のブリッジモジュール(C 言語で記述された動的リンクライブラリ)
2) コンポーネントの呼び出しを制御するイベントハンドラ(COBOL で記述)
3) イベントハンドラから参照されるコンポーネントの情報を保持するメタ情報テーブル(COBOL で記述)

を用意した．メタ情報テーブルはイベントハンドラによって参照され，コンポーネント情報として，部品名や入出力フィールド情報，起動トリガーとなるイベントコードなどを保持する（実装の簡単化の為，現段階ではコンポーネント操作ブリッジの参照するインタフェース定義リストについては，ブリッジモジュール内に直接記述している）．

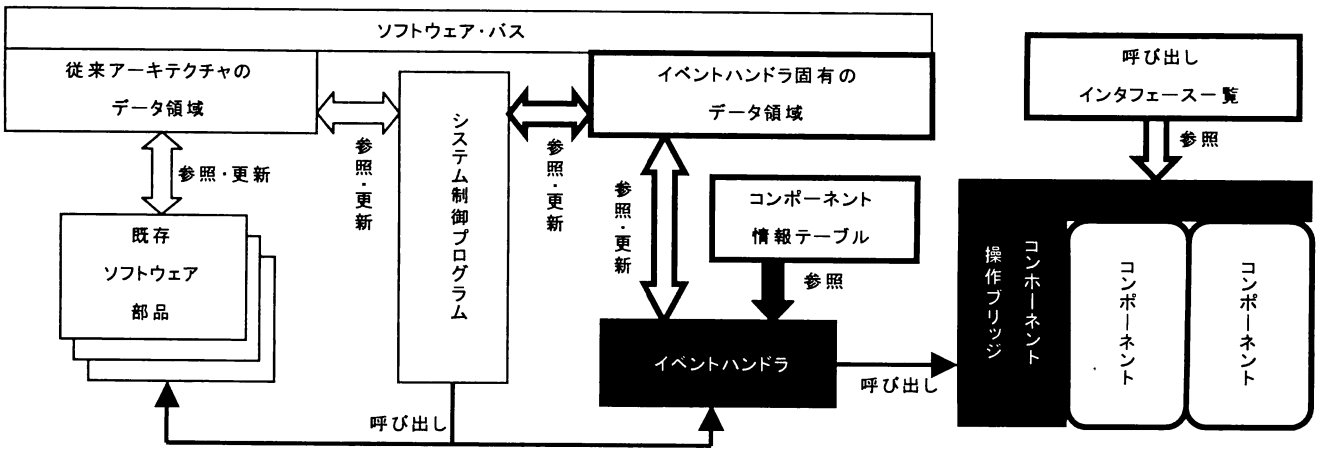

図.3 既存アーキテクチャへのコンポーネント組込み機構の実装図

ブリッジ機構検証用に実装したシステムの構成を図.3に示す．コンポーネント側と既存アーキテクチャ側との間のデータおよび制御に関するやり取りはすべてソフトウェア・バスへの読み書きとして実現され，このとき，イベントハンドラ内でデータの相互変換が行われる．このようにすることで，アーキテクチャの差異は拡張部分が吸収し，従来アーキテクチャとコンポーネント環境，双方の構造を大幅に変更する必要が無い．

3.3. 設計プロセスの拡張

次に，本機構を採用してシステムを設計する際のプロセスモデルについて説明する．本手法の設計プロセスは，従来アーキテクチャを用いる場合の設計の枠組みをもとに，追加的にコンポーネント組込み設計のプロセスを取り入れたものである．つまり，基本的な画面遷移は従来の設計方法をそのまま利用し，より粒度の細かいカスタマイズにコンポーネントを利用するという手順を踏むように定義した．

3.3.1. 従来の設計プロセスと画面遷移モデル

従来方式における業務処理プログラムの設計プロセスは以下のようになっており，これら一連の作業により，1つの画面フロー(つまり1つの業務処理)が完成する．

a) 作成する業務の画面の流れ，つまり，処理内容とその順序を決定する．
b) 各画面のGUIデザイン，イベントの配置，入出力フィールドの設定を行う．
c) b) の設定内容をもとに，画面入出力フィールドのデータテーブルを作成する．
d) 画面毎の処理を行うプログラムのスケルトンコードに具体的な処理内容を書き加えて画面処理プログラムを作成する．
e) c), d) の作業を全ての画面について行う．

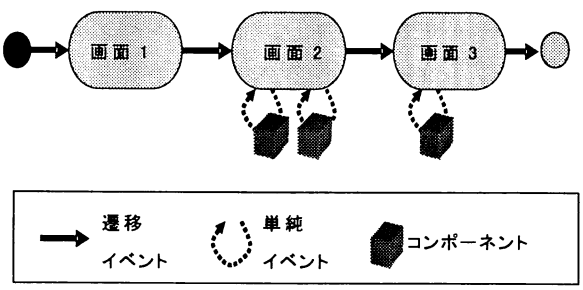

図.4 コンポーネント組込みイベントモデル

従来設計における画面遷移のモデルでは，1つの画面が1つの処理単位に対応する．ボタン押下などの入力イベントにより，データベースの参照や書き込みなどの対応する処理が行われ，次の画面，つまり次の処理へと遷移する．業務は複数の処理から成り，分岐やループ構造を構成していくことで，より大規模な処理群を構築することができる．

3.3.2. 画面遷移モデルと設計プロセスの拡張

コンポーネント利用を考慮した設計を行えるように，上で述べた画面遷移モデルの拡張を行った．図.4にコンポーネント組込みを考慮して拡張したイベントモデルの概念図を示す(ここでは簡略化の為，ループも分岐も存在しない3つの画面を持つフローを例としている)．このとき，1つの画面内で完結する，より細かな処理をコンポーネントを利用する処理として対応付ける．このような画面遷移の生じない，より細かな粒度の処理に対応するイベントを単純イベントと呼ぶ．単純イベントは遷移イベント同様に端末からのキー入力等によって発生するが，その性質上，画面遷移と切り離して設計する事が可能である．よってコンポーネントの組込みは画面の遷移設計後に検討すれば良い．

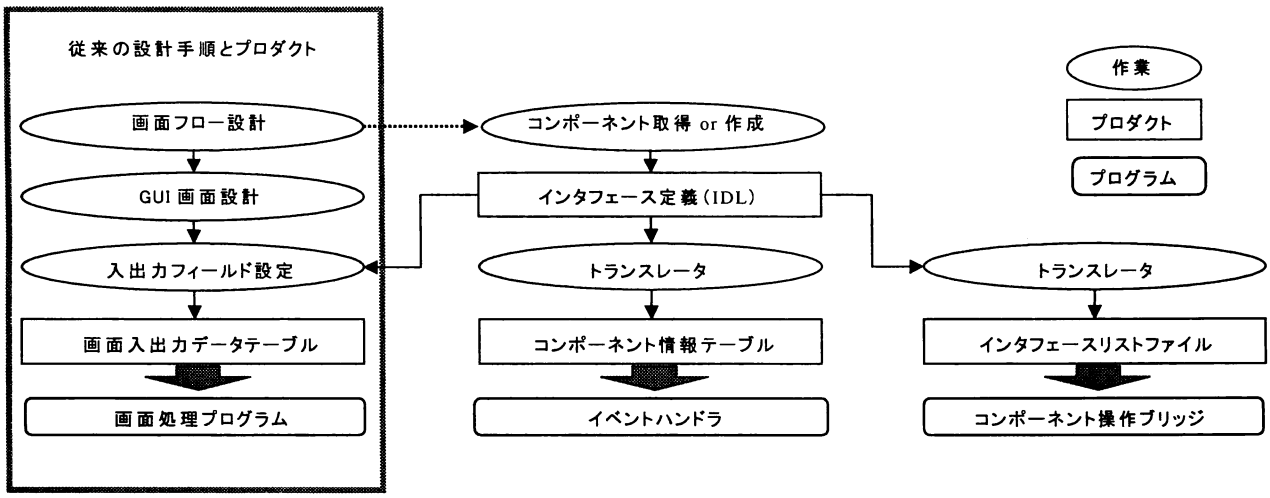

図.5 設計手順とプロダクトフロー

上記モデルに基づいた，プログラム開発プロセスは，概ね以下の通りとなる(各手順の手戻りや繰り返しが発生することもある).

a) 従来方式に従って画面フローの概要設計を行う．画面間の移動は，従来通り，遷移イベントによって起こるものとして扱う．
b) 各画面について，必要に応じて，コンポーネントの利用を前提とした単純イベントを追加する．
c) a) b) の設計内容を元に，GUI 画面のデザインを行う．デザイン作業には入出力フィールドやイベント識別コードの設定なども含まれる．
d) b) の設計内容を元に，利用するコンポーネント(バイナリ部品とインタフェース定義)の調査･取得，あるいは作成を行う．
e) 入手したコンポーネントのインタフェース定義を元にソフトウェア・バスとコンポーネント間のデータ相互変換に利用するコンポーネントデータテーブル(COBOL で記述)を作成する．この作業はトランスレータにより自動化する．
f) 同様に，コンポーネント操作ブリッジ(C/C++,Java,etc でコンポーネントアーキテクチャ毎に記述)から参照する，利用コンポーネントとインタフェースのリストファイルを作成する．

上記の手順とプロダクトの流れを図.5 に示す．

4. 提案機構を用いたシステムの試作

4.1. 既存デモシステムからのコンポーネント操作

本手法の検証実験として，日立製作所で利用されているオンラインソフトウェア開発システムの教育・デモ環境に付属するサンプルプログラムを対象に，COM コンポーネントと JavaBeans コンポーネントを利用する為の機構を実装し，既存システム上でのコンポーネント組込み機構の動作を確認した．そして，従来型の設計とコンポーネント組込みの併用を実現する拡張部分(イベントハンドラとコンポーネント操作ブリッジ)について試作実装を行った．

具体例として，ある画面中の郵便番号入力フィールドに郵便番号を入力して，あるボタンを押すと，入力した郵便番号に対応して住所入力フィールドが変更される，という機能を実現した．まず，COM 仕様に準拠した郵便番号辞書コンポーネントを作成し，これを呼び出すトリガーとなるボタンをサンプルデモプログラム上に配置した．次に，このボタンを押下するとコンポーネント組込み機構によって郵便番号辞書コンポーネントが呼び出されるように，コンポーネントデータテーブルを作成した．最後にサンプルデモシステムを再結合し，メインプログラムから郵便番号辞書コンポーネントへの呼び出しが設計どおりに行われることを確認した．

4.2. 実装上の課題

今回の試作では以下のような課題が存在する．

- **利用できるコンポーネントの情報がコードに固定されている**：部品を特定する ID(たとえば COM の場合は CLSID 及び IID)を格納した変数をコンポーネント操作ブリッジに直接コーディングしているが，これを，利用できる部品 ID の一覧をテーブルに保持するように変更する必要がある．この情報は，上流の画面設計時にあらかじめ決定される為，テーブル情報のみを別ファイルとして動的に参照することで，コンポーネント操作ブリッジの改変や再コンパイルは行わなくても良い．あるいは，インタフェースを動的に調べる機能を利用して，部品 ID はイベントハンドラがコンポーネント情報テーブルから読み出し，コンポーネント操作ブリッジに受け渡すという方法も考えられる．
- **既存システムと部品間でやりとりされるデータ型に**

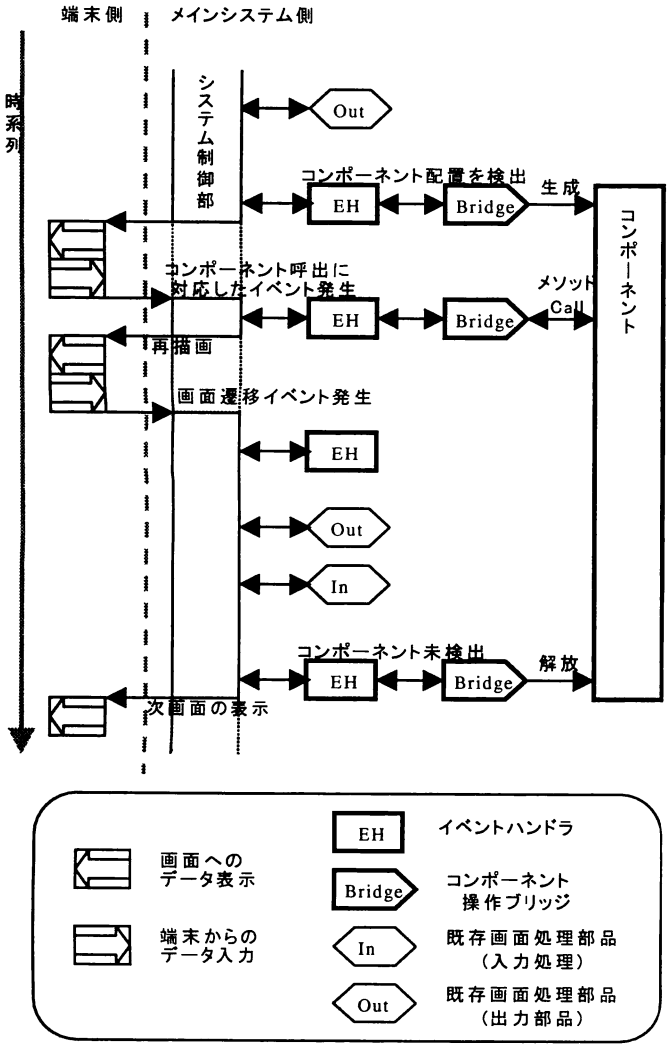

図.6 従来システムからのコンポーネント呼び出しシーケンス

制約がある：現状では，やりとりされるデータ型として数値と文字列しか扱っておらず，構造体やオブジェクト型によるパラメータは考慮していない．将来的にはビットマップや音声，場合によってはネットワークのソケットなど，様々なデータを扱う可能性があるので，COBOL が扱う型と各コンポーネントで扱う型の間の変換について，更に検討する必要がある．ただし，データ型について完全に汎用化した実装を行う事は，言い方を変えれば，オブジェクト指向の型システムを COBOL 上でエミュレートする事に等しく，そのような実装は本研究の目的からは外れるものであると考えられる．現実的な解としては扱える型にある程度制約を設けた上で，それらに対する変換ルーチンをあらかじめ用意しておくか，ユーザが用途に応じて変換ルーチンを追加できるようにするのが適当であろう．

5. おわりに

以上述べたように，コンポーネント組込みに対応しない非オブジェクト指向のアーキテクチャを利用したシステム開発においてコンポーネントウェアを利用する技術について，設計方法論と，実装機構の両面から検討を行い，制御プログラムの試作を行った．この手法により，従来からのプログラム部品をそのまま利用しつつ，新たにコンポーネントを導入することが，比較的容易に実現可能であることを確認した．現段階において試作を行ったプログラムでは，Microsoft の COM と，Sun Microsystems の JavaBeans の 2 種類のコンポーネントアーキテクチャに従う，単純な機能とインタフェースを持つ部品にのみ対応している．また，メタ情報を用いた動的なコンポーネントの生成・操作については実装途中で，現在，コンポーネント操作ブリッジにコーディングしている呼び出しコンポーネントの情報を外部ファイルに移行して，動的なコンポーネント呼び出しに対応する作業を行っている．

今後は，より複雑なインタフェースや分散コンポーネント技術への対応も検討する予定である．また，今回は特定のアーキテクチャを持つオンラインシステムを対象としたが，本手法をより一般化し，様々な既存アーキテクチャへの対応法についても検討を行いたい．

謝辞

本原稿の執筆に際し，貴重な助言をいただいた，奈良先端科学技術大学院大学・情報科学研究科の島和之助手に感謝いたします．

文　献

[1] Peter Herzum, and Oliver Sims , Business Component Factory: A Comprehensive Overview of Component-
Based Development for The Enterprise, John Wiley & Sons, Inc., New York, 2000.（長瀬嘉秀 監修，ビジネスコンポーネントファクトリ，トップスタジオ 訳・編，翔泳社，東京，2001）

[2] 青山幹雄，“コンポーネント指向ソフトウェア開発へのいざない，” FUJITSU, vol50-2, pp.62-71, Mar.1999.

[3] 早稲田大学理工学部 深澤研究室 コンポーネント指向ソフトウェア工学グループ,"オブジェクト指向からコンポーネント指向へ," (http://www.fuka.info.waseda.ac.jp/Project/CBSE/)

[4] Rogerson, D., *Inside COM*, Microsoft Press, 1997.

[5] Sun Microsystems, *JavaBeans Documentations*, (http://java.sun.com/products/javabeans/).

[6] Sun Microsystems, *EJB (EnterpriseJavaBeans) Documentations*, (http://java.sun.com/products/ejb/).

[7] Jon Siegel, CORBA Fundamental and Programming, Jon Siegel, ed., John Wiley & Sons, Inc., New York, 1996.

[8] 大野治，小室彦三，降旗由香理，渡部淳一，今城哲二，“多次元部品化方式によるソフトウェア開発の自動化—自動生成系の開発とその評価—，”信学論(D-I), Vol.J84-D-I, No.9, pp.1372-1386, Sep.2001.