電子通信普及財団 研究調査報告書 No.18
pp.579-586, (Dec. 2003)

01-01054

Web サービス連携のためのプロトコルとアプリケーション開発技法

中 所 武 司　明治大学理工学部情報科学科教授

1 はじめに

近年，電子商取引に代表される web アプリケーション間の連携や EAI（Enterprise Application Integration）の分野では，頻繁な機能変更を伴うものが増加している。このような絶えざる変化に対応することが求められるアプリケーションは，業務の専門家が自ら開発し，自ら保守を行なうのが望ましい。本研究では，このような視点からインターネットの普及とエンドユーザサイドのブロードバンド化を背景とした Web アプリケーションの開発に着目し，その基本技術である Web サービス連携のためのプロトコル設計と迅速な開発と変化に対応する保守を実現する開発環境について研究開発を行なった。

第１の Web サービス連携のためのプロトコルについては，ソフトウェアのサービス化の進展と共にサービス連携のためのXMLベースの SOAP（Simple Object Access Protocol）[1]，UDDI（Universal Description, Discovery and Integration）[2]，WSDL（Web Service Description Language）などの技術が出現している。しかしながら，これらの技術は IT 専門家が利用するものであり，エンドユーザ主導の開発／保守を実現するには至っていない。本研究では，Web サービスの登録や検索および仲介を扱う UDDI の基本機能であるホワイトページ（サービス提供者情報），イエローページ（サービス情報），グリーンページ（サービス授受のインタフェース情報）よりもエンドユーザ向きに簡潔な電子フォーム主体の方式をベースとし，基本項目を依頼先，依頼内容，依頼方法の３種類に限定する。

第２の開発技法に関しては，その共通プラットフォームとして，Java 言語，EJB（Enterprise JavaBeans）ベースの製品が普及しはじめているが，従来から注目されてきたオブジェクト指向プログラミングの部品化・再利用技術の観点では，抽象データ型のクラスライブラリの部品としての粒度が小さすぎるという欠点があった。そこで，最近ではアプリケーションと部品の間のギャップを埋めるためのコンポーネント関連の研究が始まっている。しかしながら，コンポーネントモデルの構成法やその組み合わせ技術の視点に欠けるため，Web アプリケーション構築技法が確立していない。本研究では，このような課題に対する解法として，アプリケーションのフロントエンド・サブシステムはフレームワーク主体で開発し，業務ロジック中心のバックエンド・サブシステムはビジュアルモデリング主体で開発する研究アプローチをとる。

2. サービス連携プロトコル

2.1 メタファーとしての窓口業務

本研究では，窓口業務をサービス授受のメタファーとして位置付け，窓口業務アプリケーションの典型的な利用手順である，窓口とフォームの検索，フォームへの記入，書類の提出と処理状況の確認をサービス連携プロトコルのメタファーとみなす。

窓口業務のアプリケーションは，Webを利用したオンラインショッピングや銀行・証券取引，旅行予約等のシステムが既にインターネット上に次々と実用化されている。また最近では，行政サービスの受付等を電子化する電子政府の実現が注目されている。しかし，現状のアプリケーションの窓口検索では，窓口依頼者がインターネットを経由して申込書を提出する場合，その窓口を呼び出すためにハイパーリンクによる検索や全文検索を行うことになるが，ハイパーリンクを利用する場合はリンク構造が複雑で目的のところに簡単にたどり着けないことが多い。全文検索システムではその文書に含まれる語句で検索するため，検索条件を工夫しなければ結果件数が多くなり，目的の窓口を見つけるのが難しくなる。

2.2 マルチエージェントシステムの導入

本研究で対象とする，マルチエージェントモデルに基づくフォームベースシステムの構成を図1に示す。受付窓口サーバは，窓口業務の担当者の端末である。窓口依頼者端末は，従来は窓口へサービスの依頼に訪れる一般の人の端末である。図の枠内に示すシステム共通のサーバは，以下のようなものである。

・ディレクトリサーバ：受付窓口のアドレスと業務（サービス）のディレクトリを管理
・フォームサーバ：各種の提出書類のフォーム（書式）を管理
・トランザクションサーバ：提出された書類とその識別番号を管理
・認証サーバ：書類の提出者の認証の管理

図1には以下の4種類のソフトウェアエージェントが示されている。

(1) エキスパートエージェント

受付窓口サーバに存在し，業務の専門家のノウハウを学習して，フォームへの記入の誘導，ヘルプメッセージの表示，記入内容のチェック等を行なう。

(2) ユーザエージェント

依頼者端末に存在し，個々のユーザに固有の情報を学習し，電子フォームの名前や住所などの欄に自動記入する。

(3) モバイルエージェント

フォームサーバに存在し，クライアント側からの検索に応じて依頼者端末へ移動し，エキスパートエージェントの機能を果たす。実際はエキスパートエージェントとモバイルエージェントは同一であり，電子フォームの中に組み込まれている。

(4) ブローカエージェント

ディレクトリサーバに存在し，クライアント側からの検索に応じて適切と思われるフォームを提示する。

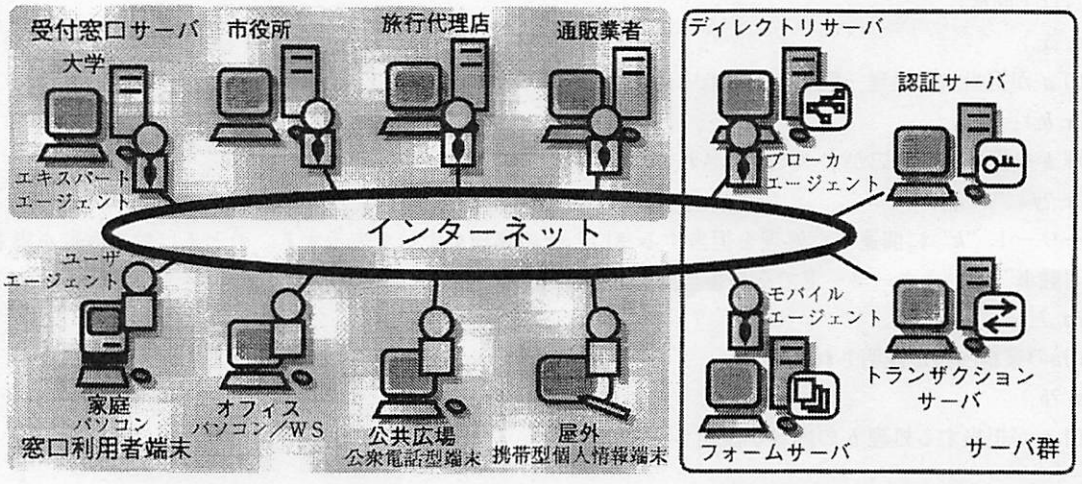

図1　マルチエージェントに基づくフォームベースシステムの構成例

2.3 プロトコル設計

このようなマルチエージェントモデルでは，エージェント間コミュニケーション言語（ACL；Agent Communication Language）が必要である。その代表的なものとしてFIPA ACL[3]があるが，われわれはFACL（a Form-based ACL）を開発した。主な相違点は，FIPA ACLがあらかじめ23個の限定された対話行為（Communicative Act）を用意しておくのに対して，FACLでは，「1サービス＝1フォーム」という考えから，対話行為数を限定しないフォーム単位の対話を実現した。

そして，オブジェクト指向のメッセージ駆動型の分散協調モデルをベースにしたわかりやすい対話インタフェースとして，「誰に，何を，どのように，頼む」というメッセージにその識別番号（どれ）を加えた4項目

のパラメータを有する以下の基本形式を設定した。

(Who, What, How, Which)

4つのパラメータは，オブジェクト指向プログラミングの概念では，メッセージの送信先，メソッド名，メソッドの実引数，メッセージ識別番号に対応する。エンドユーザ向けの概念としては，Who は窓口すなわち依頼先あるいは書類の提出先，What は依頼業務の種別あるいは提出書類の名称，How は依頼業務の内容あるいは提出書類の書式，Which は依頼または提出書類を識別するための受付番号またはコードである。

2.4 対話言語の使用例

システムの使用例を示すことで受付窓口業務に関連したほとんどの依頼内容を表現できることを示す。なお，簡潔に説明するために，外部仕様の表示形式ではなく，(Who, What, How, Which) の基本形式を用いる。パラメータの *a, b* は定数または値のバインドされた変数を意味する。*x, y* は値が未定義の変数を意味する。？は値の問い合わせを意味する。

以下の(1)から(3)は，業務依頼の例である。

(1) (*a, b, c, x*) ：

窓口 *a* に処理 *b* を依頼するために書類 *c* を提出し，受け取った受付番号を *x* に記入する。入力形式は窓口依頼者端末に依存する（以下同様)。

(2) (*a, b, , ?f*)

窓口 *a* に依頼済みの処理 *b*（受付番号 *f*）の状況が表示される。

(3) (*a, b, , -f*)

窓口 *a* に依頼済みの処理 *b*（受付番号 *f*）に対して取消を依頼する。

次の(4)から(9)は依頼先，業務種別，書式の問い合わせ例である。

(4) (*a, b, ?x,*)

窓口 *a* に処理 *b* を依頼するための書式が表示され，入力が誘導される。表示形式は窓口依頼者端末に依存する（以下同様)。

(5) (*a, ?x,,*)

窓口 *a* が担当する処理一覧が表示される。

(6) (*?x, b,,*)

処理 *b* を担当する窓口がすべて表示される。

(7) (*?x, ?y= "k" ,,*)

キーワード "*k*" に関連した処理を担当する窓口とその処理をすべて表示する。たとえば駐車許可を得るために "駐車" というキーワードでその担当部署と手続きを調べる。

(8) (*?a,,,*)

窓口aの業務内容が説明される。

(9) (*a, ?b,,*)

窓口 *a* が担当する処理 *b* の内容が説明される。

2.5 FACL の実装

インターネット上で先に定義した対話インタフェースを実現するために，FACL の外部形式として，以下のような RDF ベースのメッセージ記述言語を開発した。FACL では，メッセージを <Message>要素を用いて定義し，Who, What, How, Which に対応する，<who>，<what>，<how>，<which> 要素が含まれる。その値は以下の4種類である。

(1) 定数 *a, b*

値をパラメタ要素内に記述する。(例：<what>図書貸出</what>)

(2) 値のバインドされた変数（値の問い合わせ）*?a,?b*

値の問い合わせを示す<Inquiry>と入力値をパラメタ要素内に記述する。

（例：<what><Inquiry/> 図書貸出 </what>)

(3) 未定義の変数（値の問い合わせ）*?x,?y,?y="k"*

値の問い合わせを示す <Inquiry> をパラメタ要素内に記述する。

(?xの例：<what><Inquiry/></what>)

(?y="図書"の例：<what><Inquiry keyword="図書"/></what>)

(4) 空

要素を省略する。

「図書管理」窓口に「図書貸出」の申請書類を提出するメッセージ例は以下のように記述できる。

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
<SOAP-ENV:Header>
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:w="http://wwhww.org/1.0/">
  <w:Message rdf:about="">
    <w:who>/明治大学/理工学部/情報科学科/ソフトウェア工学研究室/図書管理</w:who>
    <w:what>図書貸出</w:what>
    <w:how rdf:resource="" />
    <w:which/>
  </w:Message>
  </rdf:RDF>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
  <takeout
    xmlns="http://wwhww.org/schemas/lifecycle/takeout">
    <itemid>19991220</itemid>
    <user>fujiwara</user>
  </takeout>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

使用例の（a, $?x_n$）に対応する「明治大学ソフトウェア工学研究室の図書管理窓口」の担当する処理一覧を要求するメッセージは以下のように記述される。

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:w="http://wwhww.org/1.0/">
  <w:Message rdf:about="">
  <w:who>/明治大学/理工学部/情報科学科/ソフトウェア工学研究室/図書管理</w:who>
  <w:what><w:Inquiry/></w:what>
</w:Message>
</rdf:RDF>
```

3 フロントエンドサブシステム開発技法

3.1 フレームワーク

最近の情報システム構築では，オープンなアーキテクチャとフレームワーク，デザインパターン，コンポーネントなどの構成要素からビジュアルツールを用いてアプリケーションを再帰的に構築していくことが追求されている。そこで本研究では，予約業務フレームワークをとりあげた。最初にアプリケーションの具体例として会議室予約システムと座席予約システムを実際に構築し，共通部分の分析を行ない，フレームワークの抽出・構築を行なった。予約業務の予約単位は，会議室予約が空間（会議室）と時間（時限）に対し，座席予約は時間（イベ

ント）と空間（座席）となる。このように予約単位の違う２つの業務をプログラム上で比較することで，フレームワークの適用範囲の検討を行なう。

今回の実験では，使いながら機能拡張していくという開発方法をとるために，頻繁な変更に対応できる保守性が重要となってくる。そこで，本システムにおいては，図２に示すように，Java クラスをモデル，サーブレットをコントローラ，JSP をビューとした MVC モデルに基づいた構成[4]を適用した。

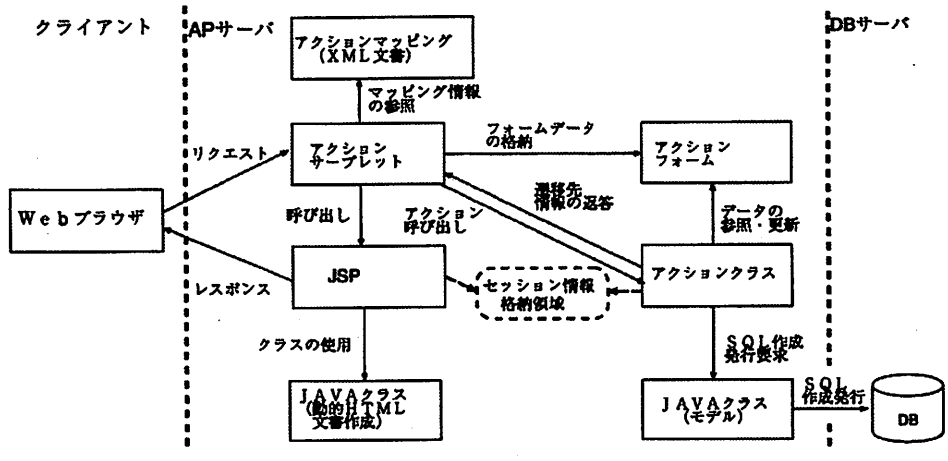

図２　MVC モデルに基づくシステムアーキテクチャ

3.2　比較実験

会議室予約は，従来の事務室のスケジュール表への書き込みという方式を Web ブラウザ上で実現するために構築したものである。概要は以下の通りである。

・対象ユーザ：教員（一般利用者）と事務員（運用管理者）
・対象資源：情報科学科で使用される会議室（現在４室）
・予約単位：１日の中で区切られた１～６限までの枠単位

一方，座席予約は，以下のような状況を仮定して構築した。

・対象ユーザ：インターネットを利用する人
・対象資源：ある施設の中で行なわれる行事（以下イベントと呼ぶ）
・予約単位：３×３に区切られた座席

両システムの一般利用者が利用する機能を比較すると以下のようになる。

・共通機能：予約・取消・予約状況照会
・会議室予約固有機能：認証情報の変更，個人別予約状況照会，定期予約
・座席予約に固有の機能：なし

3.3　抽出されたフレームワークの再利用性

抽出された予約業務フレームワーク再利用性について述べる。今回抽出したフレームワークが，開発した２つのシステムにおいてどの程度の割合を占めるかをステップ数によって検証した。比較は，予約・取消・照会機能で使用したクラスのステップ数の合計を，各サブシステムごとに計算した。ただし，クラスの中で予約・取消・照会処理に直接関係しない処理は，ステップ数に含めていない。結果を表１に示す。

表1　抽出したフレームワークの割合

	サブシステム	Javaクラス（モデル）	アクションクラス	アクションフォーム	JSP	Javaクラス（動的HTML文書生成）	アクションマッピング	合計
会議室予約	全体	740	370	170	300	640	100	2320
	フレームワーク	460	280	140	60	0	60	1000
	業務に固有	280	90	30	240	640	40	1320
	フレームワークが占める割合	62%	75%	82%	20%	0%	60%	43%
座席予約	全体	550	300	90	260	130	90	1420
	フレームワーク	480	270	90	60	0	60	960
	業務に固有	70	30	0	200	130	30	460
	フレームワークが占める割合	87%	90%	100%	23%	0%	67%	68%

会議室予約では，Java クラス（動的 HTML 文書生成）のカレンダー作成処理と Java クラス（モデル）の休日管理や時間の計算処理などのステップ数が大きいため，フレームワークの占める割合が全体として低くなってしまった。また，予約機能の操作性向上のための処理や予約確認画面の挿入に伴う処理などをアクション・クラスに記述したため，アクション・クラスの割合も少し低めとなった。一方，座席予約では，Java クラス（モデル）とアクション・クラスとアクション・フォームの部分は，平均して90%近い部分がフレームワークによって構築されている。

全体の再利用性を大きく下げてしまったのは，ビューの役割を果たす JSP と Java クラス（動的 HTML 文書生成）である。JSP はエンドユーザのページデザインなどにも依存するので，この箇所はエンドユーザ自身に構築してもらうためである。

4　バックエンドサブシステム開発技法

4.1　モデリングプロセス

グループウェアやワークフローシステムなどを対象に，オフィスでのバックエンドの業務の開発をエンドユーザ主導で行なう方式を開発した。基本的コンセプトは，「ドメインモデル＝計算モデル」および「分析＝設計＝プログラミング」である。これは，問題領域を分析してドメインモデルを構築した時点でソフトウェアの開発を完了させようというものである。即ち，「ソフト開発＝モデリング＋シミュレーション」という図式で表現される方式により，稼働後の機能変更（モデル変更）も容易となる。

アプリケーションアーキテクチャはユーザインタフェース，モデル，コンポーネントの3要素からなる。モデルはアプリケーションに固有の処理を行う本体で，インスタンスベースのドメインモデル（動的モデル）をモデリング&シミュレーションツールを用いて作成する。オブジェクトとメッセージフローで構成される動的モデルの構築は業務コンポーネントの組合せが基本となる。ドメインモデルはクラスベースの設計モデル（静的モデル）に変換される。ユーザインタフェースはそのアプリケーションのユーザとの対話処理部分であり，その構築ツールとして UI ビルダがある。コンポーネントには分野に共通の基本部品と特定業務分野向きの部品がある。後者が充実してくるとエンドユーザによるアプリケーション構築が容易になるので，コンポーネント化を支援するコンポーネントビルダを用意する。実際のアプリケーション開発は，業務仕様の詳細化，ドメインモデルの作成，ユーザインタフェースの構築，シミュレーション実行による検証の順で行われる。

4.2　具体例

ここでは例題として，情報処理学会ソフトウェア工学研究会要求工学ワーキンググループにおいて共通問題とされている「国際会議のプログラム委員長の業務」[5]を取り上げ，実際の開発手順を説明する。

(1)　業務仕様の詳細化

まず，システムの利用者を抽出し，機能概要（ユースケース）を記述する。例題では，スケジュールの決定，CFP の作成・配布，プログラム委員選出，投稿論文登録，などのユースケースを抽出して業務の詳細化を行った。

(2)　ドメインモデルの作成

次にシステムの動的な振舞いをインスタンスベースのメッセージのやりとりとしてモデリングしていく。「1業務＝1オブジェクト」という原則で，オブジェクトやメッセージの抽出を行う。国際会議のプログラム委員長業務のドメインモデルの一部分を図3に示す。

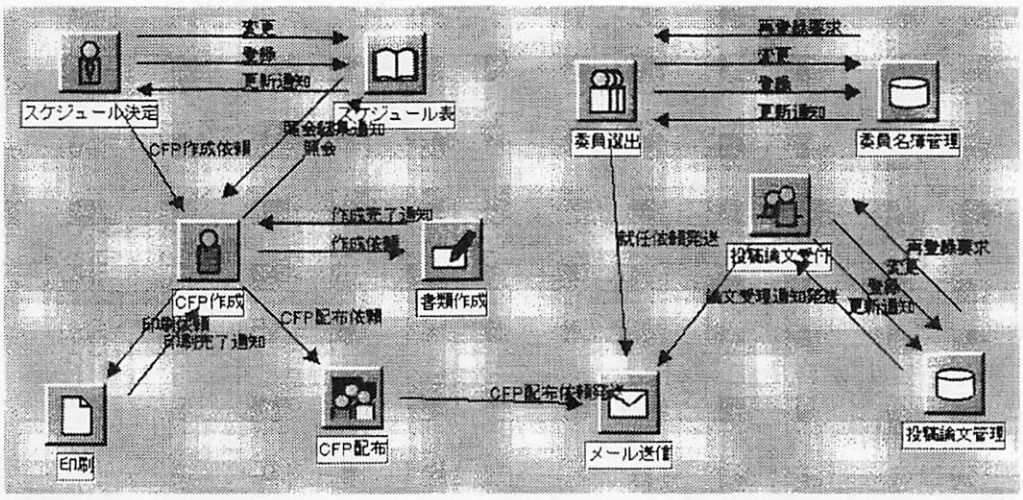

図3　国際会議のプログラム委員長業務のドメインモデル

(3)　ユーザインタフェースの構築

ユーザインタフェースはシステムから自動生成されるが，エンドユーザは必要に応じてカスタマイズすることで，使い勝手の向上を図ることが可能である。この段階で UI 画面の遷移図を表示し，業務フローの概要を早期に検証できるようにしている。

(4)　シミュレーション実行による検証

シミュレーションは，作成した業務フローの動作確認のためにユースケース単位で行う。業務フローの各メッセージをステップ実行し，起動したメソッドの内容をひとつずつビジュアルにチェックできる。

5　おわりに

以上，エンドユーザ主導型アプリケーション開発技法の研究ついて述べた。実用化のためには個々の技術をできるだけオープンシステムとして構築することが重要である。

参考文献

[1]　Simple Object Access Protocol （SOAP） 1.1 W3C Note, World Wide Web Consortium （2000).

[2]　UDDI: Universal Description, Discovery, and Integration of Business for the Web. http://www.uddi.org/specification.html

[3]　FIPA, Agent Communication Language, FIPA Spec 2-1999, Draft ver.0.1, Apr. 1999.

[4]　The Ja-JakartaProject, http://www.ingrid.org/jajakarta/struts/

[5]　大西淳：要求工学ワーキンググループとその活動，情報処理学会 ウィンターワークショップ・イン・高知，論文集，pp.21-24（Jan. 1999).

〈発 表 資 料〉

題　　名	掲 載 誌 ・ 学 会 名 等	発 表 年 月
電子フォーム自動記入エージェントの実現方式と評価	情報処理学会ソフトウェア工学研究会	2002年5月
窓口業務アプリケーションフレームワーク wwHww におけるルール生成を自動化した自動記入エージェントの実現方式	情報処理学会論文誌 vo. 43, No.6	2002年6月
3層 Web アプリケーションの実用試験による変更容易性の評価	第1回情報科学技術フォーラム (FIT2002)	2002年9月
要求品質の数量化—モデル駆動型分析と UI 駆動型分析の場合	情報処理学会ソフトウェア工学研究会要求工学 WG 第11回ワークショップ	2002年10月
要求分析における業務モデルの表現に関する考察—モデル駆動型と UI 駆動型—	情報処理学会ソフトウェア工学研究会要求工学 WG 第11回ワークショップ	2002年10月
Web アプリケーションにおける予約業務フレームワークの抽出実験と再利用性の検証	情報処理学会ソフトウェア工学研究会	2002年10月
エンドユーザ向き分散アプリケーションフレームワーク wwHww における分散協調型自動記入エージェントの実現方式	コンピュータソフトウェア，19，6，日本ソフトウェア科学会	2002年11月
ルール生成を自動化した Web フォーム自動記入エージェントの開発と評価	平成14年度第1回情報処理学会東北支部研究会	2002年12月
Automatic Filling in a Form by an Agent for Web Applications	APSEC2002, IEEE Computer Society	2002年12月
フォーム自動記入のための Web サービス変換フレームワーク実現方式	情報処理学会ウインターワークショップ・イン・神戸	2003年1月
モデル駆動型分析と UI 駆動型分析に基づく要求仕様からの最終品質の定量的予測方法	情報処理学会ウインターワークショップ・イン・神戸	2003年1月
コンポーネントベースアプリケーション開発技法の研究	明治大学科学技術研究年報　第43号	2003年2月
電子自治体向けフォームベースシステムと検索・記入・提出用ポータルサイトの構築法	情報処理学会第65回全国大会特別トラック (10)「e-Japan の進展」	2003年3月
Web アプリケーションにおける UI 駆動型要求分析に関する考察	情報処理学会ソフトウェア工学研究会要求工学 WG 第12回ワークショップ	2003年5月