資源適合型アプリケーション統合開発環境の提案

A development system for resource-oriented applications

金指 文明 [1)] 水野 業介 [2)] 石川 貴士 [3)] 小森 聡 [4)] 谷沢 智史 [5)]
Fumiaki KANEZASHI Gyousuke MIZUNO Takashi ISHIKAWA Satoshi KOMORI Satoshi YAZAWA

1) 有限会社カラビナシステムズ（〒430-0917 静岡県浜松市常磐町 288 番 (パークアベニュー 2-A)
E-mail: bunmei@carabiner-systems.com）

2) 有限会社カラビナシステムズ（〒430-0917 静岡県浜松市常磐町 288 番 (パークアベニュー 2-A)
E-mail: mizuno@carabiner-systems.com）

3) 有限会社カラビナシステムズ（〒430-0917 静岡県浜松市常磐町 288 番 (パークアベニュー 2-A)
E-mail: takashi@carabiner-systems.com）

4) 有限会社カラビナシステムズ（〒430-0917 静岡県浜松市常磐町 288 番 (パークアベニュー 2-A)
E-mail: satoshi@carabiner-systems.com）

5) 静岡大学情報学部（〒432-8561 静岡県浜松市城北 3 丁目 5 番 1 号
E-mail: t-minazuki@e-mail.ne.jp）

ABSTRACT. In this papaer, we propose a development environment for resource-oriented application based on component-oriented design. This development environment is ISEN application system. ISEN is a extended Web-based application runtime environment to execute EJB, Axe application, Servlet and JSP. The platform includes Application Server(SpindleSE), Client Application Browser(SpindleCE) and Component Repository Server(Vessel).

1 背景

近年の情報処理技術 (IT) 推進の流れにより，多くの分野において情報処理システム構築の需要が高まってきている．しかし，情報処理システムの導入を検討した場合，その構築・管理・運用に大きなコストがかかる．そのため，特に中小企業では新たな投資に高いリスクが伴うため，情報化を望むが，欲している情報システムを導入するだけの投資ができず，結論としてはあきらめざるを得ない場合がある．しかし，現実的に企業間取引が情報化され始めており，早急に導入を検討しないと淘汰される可能性もある．このような背景から，現在のシステム構築のコスト・保守管理費用削減に効果があり，中小企業でも導入しやすいシステムが実現できる，ソフトウェア開発方法・流通のインフラストラクチャの構築が最重要課題となっている．

2 目的

本論文におけるシステム (以下，本システムとする．) の目的は，「Service Anywhere.」ということばを実現することである．どのような場所でも，どのような端末であっても，同じサービスならば，端末に合わせて最大のサービスを提供できる情報システムを提供するための手段を構築することである．

現在，インターネットは既に水道の水のように常時接続で利用できるような状況にある．それにともない，アプリケーション自体がインターネットに接続されていることを要求していることも多くなっている．さらに，携帯電話は，アプリケーションをダウンロードして動作させることができるなど，様々なネットワーク端末上で様々なサービスを受けられるようになってきている．しかしながら，次のようなサービスを構築できれば，この情報世界をさらにすばらしい世界に変えることができると考える．この目指す世界を一言でいうならば，

「どのような場所でもサービスを受けられる」
Service Anywhere

である．この世界の実現がこれからの情報化社会では非常に重要となる．実際，はじめて電子メールが携帯電話で読めるようになった時に，非常に便利であると感じたが，その後，様々なサービスが全ての携帯電話でできれば，さらに便利になると感じた方も多いのではないかと思う．実際，携帯電話を利用したサービスは数多く提供されている．今後，情報端末自体がさらに高機能化することが予測できる．これにより高度なサービスが様々な場所で得られるようになるが，携帯電話でも，PDA(Palm, WindowsCE, PocketPC など) や PC などのどのような端末でも同じサービスを受けられるシステムを構築できればと思うのである．もちろん，それぞれに資源の違うマシンであるため，全く同じサービスを各端末で提供するのは難しいだろう．しかし，各端末の資源を知っていれば，その端末における提供可能な最大限のサービスを構築できると考えている．

現在，またこれからも数多くの情報機器が生まれるだろう．それに伴いそれらの機器ごとに動作するシステムの需要も非常に高まっている．情報機器はそれぞれ特徴があるため，それぞれの機器に合ったシステムを構築する必要がある．そのために，ソフトウェアを部品のように組み合わせることで，各情報機器に合わせたシステムを作り出すことが本論文のアプリケーション構築方法である．実行時に各機器が必要なコンポーネントを集めて結合し実行するという方式を取る．本論文はコンポーネント指向型のシステ

ム開発方法を利用しているが，コンポーネント指向開発方法によって大きな効果を得るためには，非常にたくさんの部品が流通する世界が必要である．そこで，コンポーネント指向開発のために多くの部品が流通するようなインフラストラクチャを考える．インフラを整えることにより，より多くのコンポーネントを利用して開発ができるようになる．結果，アプリケーションの開発向上にもつながり，またコンポーネントウェアの需要も高まる．

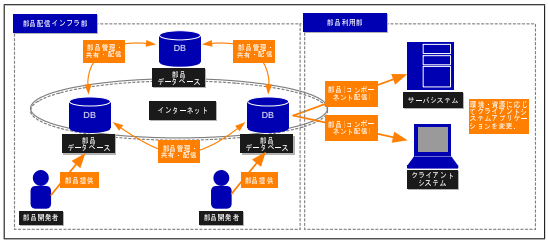

図 1: 基本アイデア

本論文では図 1 に示すような 2 つの世界を構築する．

- **部品配信インフラ部：**コンポーネントウェアの流通を行う世界である．部品開発者は部品サーバにコンポーネントを登録することにより，世界中に部品を公開することができる．
- **部品利用部：**部品配信インフラ部で提供されている部品から必要な部品を選択して利用する世界である．

このような 2 つの世界を構築することで，比較的実装の難しい部品の作成部分を上級技術者が行い，それらを組み立てる作業を一般の人間もしくは技術レベルの低い技術者が行うようになる．本システムを利用することにより，部品の流通およびアプリケーション開発のコミュニティの成長を促すことにもつながる．

3 資源適合型アプリケーション

本節では，資源適合型アプリケーションを配信する方法について述べる．資源適合型アプリケーションとは，アプリケーションが動作する環境に適合して，その環境に合った動作を行うアプリケーションことである．このような資源適合を行うために本システムでは，環境に合わせて，アプリケーションを構成するコンポーネント (部品) を入れ替えるという方法をとる．一般的なアプリケーションは，開発時間にコンポーネントの結合を行い，結合された状態でリリースされる．しかし，本システムのアプリケーションは，開発時間にコンポーネントの結合を行って動作テストなどを行うが，リリース時には，コンポーネントを切り離した状態にする．実際に動作させる場合には，環境の資源を調べて，その資源に合ったコンポーネントを選択して結合する．ここで，リリース時にコンポーネントが切り離された状態のアプリケーションを XAD(eXtended Application Descriptor) と呼ぶ．XAD 自体は，コンポーネントが無い状態のアプリケーションであるため，そのままでは動作しない．言い換えると，XAD は，環境に関係なく動作する部分，要するに環境に依存しない動作部分から成り立っているといえる．この XAD と環境資源を参考にコンポーネントを選択して結合することにより初めて動作するアプリケーションとなる．このように，資源適合型アプリケーションはコンポーネントを取捨選択して環境にあったアプリケーションに変換して実行する特徴がある．

図.2 は資源適合型アプリケーションの概念図である．アプリケーションサーバからは XAD が配信される．通常，これらの XAD はアプリケーションを動作させたい端末からの要求により配信される．この時点では当然コンポーネントが選択されていないため，XAD 自体は動作しない．

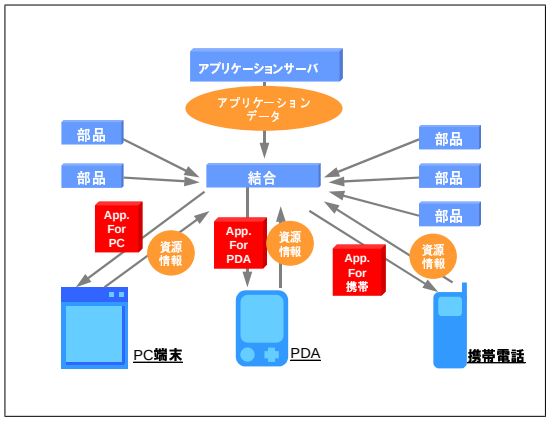

図 2: 資源適合型アプリケーション

各端末はアプリケーション利用の要求を出すと同時に自端末の資源情報を取得する．この配信された XAD と端末資源情報を参考に必要な部品をインターネット上から収集し結合を行う．そして，結合が終了すれば実行可能アプリケーションとして各端末上で動作する．

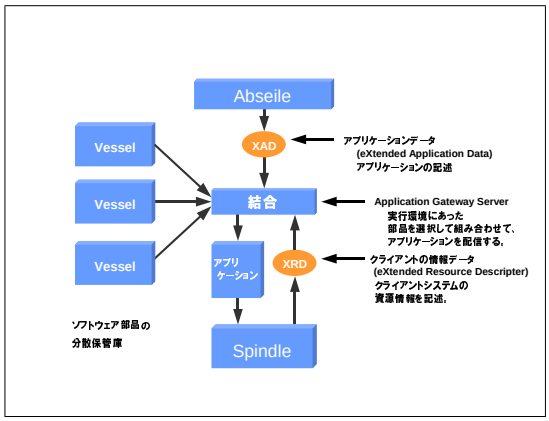

図 3: 資源適合型アプリケーション配信のしくみ

図 5 は，図 2 の概念図を実際に実装したシステムで置き換えた概要図である．Spindle(クライアントシステム) からはアプリケーション要求と XRD(eXtended Resoruce Descriptor:端末資源情報) を出す．アプリケーション要求を受けた Abseiler(アプリケーションサーバ) からは XAD が配信される．XAD と XRD を参考にインターネット上に散在する Vessel(部品配信サーバ) から必要な部品を取得する．そして，取得した部品と XAD を利用してアプリケーションを結合し Spindle に結合後のアプリケーションを配信する．

4 システム概要

本節では前節で述べた資源適合型アプリケーションを配信するための仕組みを実現しているシステムの概要を説明する．目的を実現するにあたって，今回の開発では，具体的に図 4 のシステム概念図で表されるシステムの開発に取り組んだ．実際に以下の 4 つの構成要素の実装を行った．

- アプリケーションサーバ：スピンドル SE(Spindle Server Edition)
- EJB アプリケーションマウント：アブザイラ (Abseiler)
- アプリケーション実行環境：スピンドル CE(Spindle Client Edition)
- 部品配信サーバ：ベッセル (Vessel)

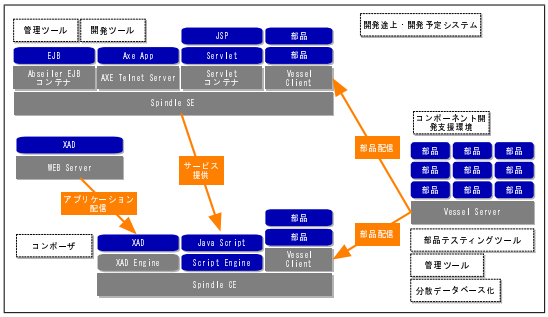

図 4: システム概念図

各構成要素の詳しい説明は後に示す．なお，本システムでの資源適合アプリケーションは，様々な端末環境に対応するクライアントアプリケーションと，会社の性質に合わせて組み合わせが変わるビジネスロジックサーバアプリケーションの 2 種類をターゲットとしている．前者をクライアント型，後者をサーバ型アプリケーションと呼ぶことにする．図 4 の矩形で囲まれた構成要素は今後実装が必要となるツール群であるため，本論文での提案対象には含まれていない．しかしながら，実際のシステム開発に本システムを利用することを考えた場合に必ず必要となる構成要素であるため，今後の課題となる．

4.1 アプリケーションサーバ

本システムにおけるアプリケーションサーバを Spindle(スピンドル) と呼ぶ．そしてアプリケーションサーバ部であるため，SE(Server Edition) をつける．

SpindleSE は様々なサーバアプリケーションコンポーネント (ビジネスロジックと呼ぶ) を管理するサーバである．ビジネスロジックは一般に再利用できる場合が多い．そのようなビジネスロジックをコンポーネント化した部品を管理・動作するシステムである．ビジネスロジック自体も動作する状況 (会社状況・状態，ハードウェア環境) によって変化することが考えられるためサーバ型資源適合型アプリケーションと見ることができる．そのため，その環境に応じてビジネスロジックコンポーネントを入れ替える機能を有している点が，他のアプリケーションサーバに無い特徴である．

SpindleSE は一般的な WEB アプリケーションサーバと同様，クライアントからの指示に従って，サーバ側の指定されたサーバサイドアプリケーション (Servlet, JSP アプリケーション) に処理を委託する．その処理結果はテキスト型返信データまたは RPC(Remote Procedure Call) 方式でクライアントに返信される．通常の WEB アプリケーションでは，このサーバ側からの実行結果は HTML を利用して配信されるが，これがテキスト型返信データであり，RPC 型は EJB 等のサーバ側で動作する処理結果を取得する方式である．

本システムでは，GUI 表示情報と簡単なプログラムからなる独自の XML 形式である XAD によりアプリケーションが配信されるが，この形状をサーバ側で変更させて配信させたい場合はテキスト型返信データを利用し，単純にサーバ側で動作しているアプリケーションに処理を委託したい場合は RPC 型を利用する．RPC 型の場合は様々なプロトコルで要求が出され，返信が期待されるが，これはサーバ側で動作するアプリケーションの仕様に依存する．要するに様々なプロトコルに対応したアプリケーションを動作させることで様々なサービスを提供するアプリケーションサーバを構成することができる．

本システムは，Jakarta Project(http://jakarta.apache.org/) の Tomcat を拡張することで実装されている．以下は，SpindleSE の特徴と主な機能である．

- ポータブルである．:設計・実装が非常にシンプルであり，Personal Java 仕様の VM 上でも動作するアプリケーションサーバである．そのため，Personal Java が提供されている比較的資源の少ない機器 (家電製品，組み込み機器) 上でも動作させることが可能である．
- スケーラビリティが高い．:アプリケーションマウントと呼ばれる独自の拡張機構により，様々な枠組みのサーバアプリケーションを動作管理することが可能である．そのため，システム拡張性に優れている．
- 資源情報 (XRD データ) にしたがって，Vessel サーバと連携して，サーバアプリケーションコンポーネントの入れ替えが可能である．
- セキュリティ Role による各種サービス呼び出しの制限機能を持つ．
- Servlet, JSP エンジンを搭載 (Tomcat の機能を利用) している．
- Servlet, JSP コンポーネントの部品置き換え機能を搭載している．これは Vessel サーバとの連携により，資源や要求が変更した場合に，アプリケーションサーバ内のアプリケーションを動的に入れ替える機能および動作中のコンポーネントを停止させ，新しいコンポーネントに入れ替え動作を開始する運用中におけるコンポーネント管理の機能である．
- ミラーリング機能：ミラー元サーバに存在するアプリケーションを，ミラー先サーバにコピーする機能である．運用時，開発時に同じシステムが必要となったときに，容易に同じシステムを構築することができる．
- サーバアプリケーションとして
  - EJB(Enterprise Java Beans version 1.0 以上)
  - Axe:独自のサーバサイドアプリケーション形式である．EJB ほど高機能ではないが簡単にサーバ側にサービスを配備したい場合に利用できる．Telnet プロトコルを利用してサービスを提供することが可能である．

  等の各種アプリケーションフレームワークを利用することができる．SpindleSE では複数のアプリケーションフレームワークのサーバアプリケーションを動作・管理するためにアプリケーションマウントを呼ばれる管理コンポーネントを利用する．アプリケーションマウントとは，SpindleSE のカーネル部と実際に動作するアプリケーションの間に位置するコンポーネントであり，各種アプリケーションの動作方式および管理方式の違いを吸収する働きがある．現状では，EJB, Axe, Servlet, JSP の各アプリケーションマウントが実装されているが，この他に新しいアプリケーションフレームワークが提案されたとしても，それに対応するアプリケーションマウントを実装することにより，新しいアプリケーションフレームワークのアプリケーションを動作・管理することが可能となる．

4.2 EJB アプリケーションマウント

本システムにおけるアプリケーションサーバを Spindle(スピンドル) と呼ぶが，その中で動作する EJB 用のアプリケーションマウントを Abseiler(アブザイラ) と呼ぶ．

- EJB version1.0 相当の EJB コンポーネントが動作可能である．
- 資源情報 (XRD データ) にしたがって，Vessel サーバと連携して，ビジネスコンテナ内のコンポーネントの入れ替えが可能である．(SpindleSE の機能と連携)
- サーバを停止させずに，ビジネスロジックコンポーネントの入れ替えが可能である．(SpindleSE の機能と連携，実行時更新機能)

- ミラーリング機能：ミラー元サーバに存在するアプリケーションをミラー先サーバにコピーする機能である．

4.3 アプリケーション実行環境

本システムにおけるクライアント側のアプリケーション実行環境を SpindleCE(スピンドル CE) と呼ぶ．

クライアント型資源適合アプリケーションの実行環境である．クライアントの環境は，現状を考えても非常にたくさんの環境が存在している．例えば，携帯電話 PDA 端末，パーソナルコンピュータ，家電製品，カーナビゲーションシステムなどである．このような様々な資源を持つクライアント環境に合わせてアプリケーションを動作させることが SpindleCE の目的である．本システム主な働きは，SpindleSE へ必要なアプリケーションの要求を出すこととクライアント資源を XRD にまとめて送信することである．

- アプリケーションブラウザという新しいアプリケーション領域を作り出すシステムである．
- PC(Windows, Linux の Java2 1.3 以上の環境), WindowsCE(Parsonal Java 1.0 以上の環境) で動作可能である．
- XAD 解釈系を持っているため，HTML ライクにアプリケーションの開発が可能である．

4.4 部品配信サーバ

本システムにおける部品 (コンポーネント) を配信するサーバのことを Vessel(ベッセル) と呼ぶ．

アプリケーションを資源適合させるために，本システムでは，クライアントに合ったコンポーネントを配信することで実現する．そのため，多くのコンポーネントを管理して配信するようなシステムが必要不可欠となる．Vessel は，コンポーネントの配信や管理を行うサーバである．

- 要求に従ってコンポーネントを配信する機能を持つ．
- コンポーネントのバージョン管理機能，各種情報管理機能 (開発者や利用実績等) を持つ．
- コンポーネント利用に関する制約条件機能により，ユーザごとによりコンポーネント利用権および利用数制限などが可能となる．この機能はコンポーネント配信に関するセキュリティ機能の実現に利用される．

5 資源適合型アプリケーションの構成方法

本節では資源適合型アプリケーションが構築されるまでの実際の流れについて述べる．本システム上で動作するアプリケーションは，簡単に言うと，資源を調べ，それを元に必要なコンポーネントを取得し，それらを動的に組み合わせることでアプリケーションを構成する．この構成方式を以下で詳しく述べる．

図は資源適合型アプリケーションが構築されるまでの流れを表している．図を利用して，実際の流れを解説する．なお，以下ではクライアント側で動作するアプリケーションが配信されるまでの流れを説明する．

1) SpindleCE から SpindleSE 対して，アプリケーションの要求を出す．(なお，アプリケーションの要求は SpindleSE, CE の両方で出される場合がある．Spindle SE はサーバサイドのアプリケーション動作管理を行い，SpindleCE はクライアント側のアプリケーション管理を行うためである．基本的に SpindleCE(クライアントアプリケーション要求) は SpindleSE に，SpindleSE(サーバアプリケーション要求) は Vessel に対して要求を出す．) SpindleSE は要求の応答として XAD を返す．
2) SpindleCE は取得した XAD 内のコンポーネント情報を取得し資源分析を行う．資源分析では，現在のクライアント環境資源を XRD として変換する作業を行う．
3) SpindleCE の環境内に既に存在するコンポーネントから利用可能なコンポーネントを調査する．なお，既に存在しているコンポーネントとは，これまでのアプリケーション実行で使用したコンポーネントを SpindleCE の環境内でキャッシュとして保持しているコンポーネントのことである．コンポーネントによってはキャッシュできないものも存在する．この調査により，Vessel に対して，配信要求を出さなければならないコンポーネントを決定する．
4) 作成した XRD 情報を基に，Vessel に対して必要なコンポーネントを要求する．
5) 要求を受けた Vessel は，指定の SpindleCE にコンポーネントを配信する．
6) SpindleCE は XAD と取得したコンポーネントを組み合わせる処理を行う．
7) 結合したアプリケーションを実行する．実行後に環境が変更された場合には，XRD 情報を作り直して再度コンポーネントの取得を行い再結合をする．

以上が，SpindleCE 側のアプリケーション構成までの流れである．なお，SpindleSE 側のアプリケーション構成もほぼ同様の流れで構築される．

5.1 サーバ側アプリケーションコンポーネント構築方法

本節ではサーバ側アプリケーションコンポーネント構築方法について述べる．サーバ側には現在次の 3 種類のアプリケーションを動作させることができる．

1) EJB(Enterprise Java Beans) version 1.0 相当
2) Axe アプリケーション
3) Servlet, JSP アプリケーション

(3) については Tomcat の機能により実現しているため，(1), (2) について述べる．

(1) は J2EE の枠組みで実装された EJB である．そのため，基本的には EJB version 1.0 相当であれば動作させることは可能である．本システムにおける EJB の作成方法は，付属のツールを利用して以下の手順で作成することができる．

1) EJB の作成：EJB で必要な各インターフェースおよびクラスを作成し，デプロイメントデスクリプタを記述し，jar ファイルにパッケージする．この作業は，XML 定義から EJB を生成するツールを利用して行なうことも可能である．
2) EJB の配備：EJB を格納した jar ファイルから，Abseiler 用の詳細な実装クラス，設定ファイルを作成する．
3) EJB の実行：配備で作成したクラスファイルを Abseiler で実行する．これで，EJB サービスは利用可能な状態になる．
4) EJB の使用：EJB で提供されるサービスを利用するクライアントを記述し，サービスを実際に利用する．

ここでは，開発の一連の流れについて，サンプルを用いて説明する．サンプルとして，String getMessage() という形式で，”Hello, Abseiler!”と返すメソッドを持つ，セッション Bean の作成を取り上げる．

1) EJB の作成：EJB の作成する場合，XML による EJB 記述 (デプロイメントデスクリプタ) を利用する．この記述法では，サンプルで挙げたセッション Bean は以下のように記述される．

```
<?xml version="1.0" encoding="Shift_JIS"?>
<session type="stateful"
```

```
        package="jp.carabiner.yl.ejb.sample"
        beanname="HelloEjb"
        component-name="jp.carabiner.ejb.HelloEjb.jar"
        component-version=""
        component-depends="jp.carabiner.abseiler.app.jar"
        component-description="簡単な Session Bean です。"
        default-classname="StringBuffer">

  <!-- 文字列を追加します． -->
  <method name="append"
          resultclass="void">
    <param class="String"/>
  </method>

  <!-- 現在の文字列長を返します． -->
  <method name="length"
   resultclass="int"/>
</session>
```

次に，この XML ファイルを hello.xml というファイル名で保存し，EJB-JAR ファイルを作成する．この作業には，ejbgen コマンド (本システムで提供) を利用する．

```
% ejbgen -jarfile hello.jar hello.xml
```

以上で EJB の作成は完了である．

2) EJB のデプロイ・配備: EJB-JAR ファイルはそのままでは SpindleSE に組み込むことはできない．あらかじめ EJB Deployer を用いて，必要な実装クラスを生成する必要がある．実装クラスは，ejbdp コマンド (本システムで提供) を用いて自動的に生成することが可能である．

```
% ejbdp -jarfile hello_deployed.jar
hello.xml
```

次に，実行するために jar ファイルを SpindleSE に登録する．

```
% sputil -addcomp hello_deployed.jar
```

以上で配備は完了である．XML 記述で，component-name 属性に指定した文字列をコンポーネント名として，jar ファイルが SpindleSE に登録される．

3) EJB の実行：EJB を実行する．

```
% sputil -start
    jp.carabiner.ejb.HelloEjb.jar -app
    -parent EJBContainer(nodep)
```

起動コンポーネントを jp.carabiner.ejb.HelloEjb.jar として，サーバコンテナ EJBContainer(nodep) の配下として起動する．実行が完了したら，sputil コマンドを使って実行されたことを確認する．

```
% sputil -dlist
...
以下情報が表示される．
```

以上，HelloEjbHome というバインド名で，EJB サービスが利用可能になっていることが確認できる．

4) EJB の使用：サービスを利用するクライアントを開発する．クライアントの開発に必要な jar ファイルは以下の 2 つである．
   - ejb-client.jar(JNDI + RMI)
   - hello.jar

   hello.jar については，自分で用意した EJB-JAR ファイルを利用する．また，ejb-client.jar については，Abseiler のパッケージに同梱されている．以下，サンプルプログラムである．

```
import jp.carabiner.yl.ejb.sample.*;

import jp.carabiner.yl.rmi.*;
import jp.carabiner.yl.rmi.service.*;
import java.util.*;
import javax.naming.*;
import java.rmi.*;
import javax.ejb.*;

public class Client {

  public static void main( String[] args ) {

    try{

      // Context の準備（はじめに行う）
      RMIContext ctx = new RMIContext( "irmi://localhost:1099/" );

      // EJB の取得
      // ホームオブジェクトの取得
      HelloEjbHome home = (HelloEjbHome)ctx.lookup( "HelloEjbHome" );

      // Bean の取得
      HelloEjb helloejb = home.create();

      // 実行
      // 文字列長の取得
      System.out.println( "length: " + helloejb.length() );

      // 文字列の追加
      System.out.println( "append" );
      helloejb.append( "sample" );

      // 文字列長の取得
      System.out.println( "length: " + helloejb.length() );
    }catch( Throwable e ){
      e.printStackTrace();
    }

  }
}
```

6 XAD 記述 (クライアント側アプリケーション構築方法)

本節では，クライアント側アプリケーションを記述するため言語である XAD (eXtended Application Descriptor) 言語について述べる．XAD 言語は，クライアント側で動作するアプリケーションを記述するための言語であり，主に，GUI(Graphics User Interface) を構成するために利用される言語である．GUI のレイアウト構成を指定し，GUI を構成するコンポーネントにイベントを割り当てることが大きな役割である．XAD 言語は XML 言語として設計されており，タグが，GUI コンポーネントに対応している．タグの組み合わせにより GUI を構成したりイベントを割り当てたりすることになる．なお，タグの定義集合のことをタグセットと呼ぶ．

以下は，XAD の構造を表している．

- 基本タグ集合定義ファイル：XAD の初期状態で持っているタグの定義ファイルである．これはアプリケーションを記述するために必ず必要となる定義である．
- 拡張タグセット定義ファイル (任意)：基本タグに存在しない機能を持つタグを定義するためのファイルである．
- XAD 記述 (基本・拡張タグセットを利用して記述)：以上の基本タグ，拡張タグを利用した実際の XAD 記述である．

本システムでは，クライアントの資源・環境によって動作するアプリケーションの機能を変化させられることが特徴として挙げられるが，これを実現する仕組みとしてタグに対応しているコンポーネントを入れ替えるという方式を取っている．要するに，PC 用の基本タグの集合や PDA 用の基本タグの集合を用意しておくことにより，これを XAD 読み込み時に選択することで，その環境にあったアプリケーションとして動作することになる．

6.1 タグセットファイル

タグセットファイルはタグとコンポーネントの関係を記述するファイルである．具体的には以下のような表記である．

```
<?xml version='1.0' encoding='Shift_JIS' ?>

<tagdef>
    <tag name="タグ名"
        type="クラス名（コンポーネント名）"
        walker="クラス設定 XML パーサ名"/>
    <!-- 以下 tag タグが続く -->
</tagdef>
```

このファイルは tagdef タグ以下に tag タグを利用してタグに対応するコンポーネント (Java のクラス名) とそのコンポーネントの設定を行う XML パーサを指定する．通常 XML のパーサは，XML 文書 1 つに対して 1 つのパーサを作成するが，XAD 言語のパーサは各タグ毎にパーサを指定する必要がある．そのため，各タグごとに設定用の

パーサを指定するようになっている．以下は tag タグ要素の説明である．

- **name:** タグの識別子 (XAD の中で使われるタグの名称)
- **type:** タグに対応するクラス名
- **walker:** タグ以下の構造をスキャンするためのパーサクラス

6.2 基本タグセット

本節では基本タグセットの概要を述べる．基本タグセットは，一般的な GUI アプリケーションを作成するために必要最低限のコンポーネントを利用できるタグから成り立っている．以下のそのタグセットを紹介する．但し，ここで紹介するタグは本論文作成時までのタグであり，さらにタグの整備が続けられている．

- **xad：**XAD 記述の始まりを表すタグ
- **component：**必要なコンポーネントファイルを指定する．
- **frame：**Frame(Window) コンポーネント
- **button：**ボタンコンポーネント
- **label：**ラベルコンポーネント
- **textarea：**テキストエリア (複数行入力可能) コンポーネント
- **textfield：**テキストフィールド (1 行入力可能) コンポーネント
- **progressbar：**進捗グラフコンポーネント
- **menu：**メニューコンポーネント
- **menuitem：**メニュー項目コンポーネント
- **checkbox：**チェックボックスコンポーネント
- **combobox：**コンボボックスコンポーネント
- **toolbar：**ツールバーコンポーネント
- **tree：**ツリーコンポーネント
- **table：**テーブルコンポーネント
- **list：**リストコンポーネント
- **scroll：**スクロール可能パネル
- **gridpanel：**Java の GridPanel に相当するパネル
- **tablepanel：**HTML のテーブル的にコンポーネントを配置するパネル
- **xadpanel：**他の XAD 記述を読み込んで貼り付けるパネル
- **borderpanel：**Java の BorderLayout に相当するパネル
- **tabbedpanel：**タブ付きパネル
- **event：**コンポーネントにイベントを割り当てるタグ
- **action：**Action イベント追加タグ
- **keyset：**Key イベント追加タグ
- **focuslost：**フォーカスが外れた場合のイベント追加タグ
- **focusgained：**フォーカスが当たった場合のイベント追加タグ
- **windowiconified：**Window がアイコン化したときのイベント追加タグ
- **windowclosing：**Window が閉じようとしたときのイベント追加タグ
- **windowclosed：**Window が閉じたときのイベント追加タグ
- **windowdeactivated：**Window がアクティブでないときのイベント追加タグ
- **windowopened：**Window が開いたときのイベント追加タグ
- **windowdeiconified：**Window がアイコン化から元に戻ったときのイベント追加タグ
- **mousedragged：**マウスをドラッグしたときのイベント追加タグ
- **mousemoved：**マウスが動いたときのイベント追加タグ
- **mousepressed：**マウスのボタンが押されたときのイベント追加タグ
- **mouseclicked：**マウスがクリックされたときのイベント追加タグ
- **mouseentered：**マウスがコンポーネントに入ったときのイベント追加タグ
- **mousereleased：**マウスのボタンが離されたときのイベント追加タグ
- **mouseexited：**マウスがコンポーネントから出たときのイベント追加タグ
- **xadengine：**XAD 処理エンジンタグ

6.3 XAD 記述の例

以下では XAD 記述の簡単な例を示す．

```
<?xml version='1.0' encoding='Shift_JIS' ?>
<xad>
    <component name="basic_pc.jar" version="1.0"/>

    <tagdef url="tagset.xml"/><!-- タグセット -->
    <include url="stdlib.xad"/><!-- ライブラリ -->

    <frame id="main" title="Button プログラム"
        width="400" height="200" x="0" y="0">
        <tablepanel id="main_panel">
            <tr>
                <td height="1" width="1">
                    <button label="ボタン"
                        icon="icon/Import16.gif"
                        tooltip="ボタンです"/>
                </td>
            </tr>
        </tablepanel>
    </frame>
</xad>
```

このプログラムは，Window を開き，その中にボタンと表示されたボタンコンポーネントを一つだけ表示するプログラムである．component タグは，tagset.xml で定義されているコンポーネントを Vessel サーバから調査して，現在の環境に存在しない場合はダウンロードをすることを意味している．

6.4 資源適合とタグセット

SpindleCE 側の資源適合は，タグセットの入れ替えにより実現される．例えば，PC 用のタグセット，PDA 用のタグセット，その他特殊機器用のタグセットが用意されているとし，アプリケーション起動時または構成時にタグセットを選択する．そのため，環境に合ったコンポーネントを収集し構成し実行することになるため，資源適合が可能になる．よって，資源にあわせてタグセットと各タグに対応するコンポーネントを実装する必要がある．

7 動作環境

本システムは，Sun 社から提供されている Java 言語を利用して実装されている．そのため，Java の VM が実装されている環境であれば，本システムは利用することが可能である．以下にシステム要件を示す．

- **Spindle Server Edition, Abseiler, Vessel:**
  - Java SDK version 1.3 以上:SpindleSE, Abseiler, Vessel
  - Personal Java version 1.0(for WindowsCE, PocketPC):SpindleSE
  - 開発するシステムに応じて CPU，メモリ，HD を

選択.

- **Spindle Client Edition:**
  - Java Runtime Environment version 1.3 以上 (for PC), Personal Java version 1.0(for WindowsCE, PocketPC)
  - 開発するシステムに応じてハードウェア資源を選択

8 評価

本節では，本システムで実装できる資源適合型アプリケーションの特徴を延べ，他のシステムと比較を行う．以下は，本システムによる特徴である．

- 複数プラットフォーム動作アプリケーションをコンポーネントの入れ替えで実現することにより，より環境に適合したアプリケーションを配布可能
- 資源情報，コンポーネント情報の常時交換を行うことで，最新のアプリケーションをクライアント側に提供することが可能.

このような特徴を持つアプリケーションを構築するためには，設計段階で，以上の特徴を実現する仕組みを入れる必要がある．しかし，本システムではコンポーネントの組み合わせによりシステムを設計し構築する．これらの設計，構築された情報は XAD により記述され，これがアプリケーションとなる．XAD で記述するため，完成後でもコンポーネント間の接続関係は「緩く」保たれる．そのため，完成後にコンポーネントの入れ替えが可能となる．しかし，一般のアプリケーションでは実装言語を利用して，コンポーネント間の接続を完全に記述してしまうため，完成後にプログラムの入れ替えを行うという行為は難しい．しかし，本システムは，コンポーネントの粒度が適切に保たれれば，アプリケーション全体におよぶコンポーネントの入れ替え (更新) が可能となる．

8.1 Java との比較

「Write once, run anywhere」の実現を目指して設計された実装言語として脚光をあびている言語である．様々な端末上で動作するシステムを構築するという意味では競合技術である．しかし，本システムは独自の資源管理およびコンポーネント管理を利用して資源にあったコンポーネントを選択し，アプリケーションを変化させているため，全く違うアプローチによりマルチプラットフォーム性を実現している．実際問題，Java は，通常の PC 向け，組み込み向け，携帯電話向けなど複数の実行環境の仕様が提供されている．そのためそれぞれの環境用にプログラムを書く必要がある．本システムもコンポーネント単位で各環境用のコンポーネントを実装しなければならないが，コンポーネントが用意できていれば，XAD を作成することにより，複数の環境で動作するアプリケーションが実装できてしまうため，Java に対して実システム実装向けの複数環境動作アプリケーションのソリューションを与えている．

8.2 EJB 関連技術との比較

既存の EJB ビジネスコンテナ製品と異なるのは，Vessel と呼ばれる部品サーバとの連携が挙げられる．この部品サーバとの連携により，EJB コンテナ内に格納されているコンポーネントがいつも最新のものに保つことができたり，不具合が見つかった場合に，即座に代替コンポーネントに入れ替えるといった，コンポーネント単位での管理機能が強化されている．現在，EJB コンポーネントの流通が一つのビジネスとして動き出しているが，その流通は現在のパッケージソフト販売とほとんど変わらない．しかし，Vessel サーバと連携することにより，流通管理が自動化されるため，EJB コンポーネントの流通の新しいインフラとなる可能性を持っている．

8.3 エンタープライズシステム

サーバ側に処理プログラムを置き，クライアントからサービスを受けるサーバクライアント型の典型的なシステムであるため，既存の各種アプリケーションサーバとの違いはほとんど無い．しかし，本システムでは，クライアントの資源を考え，その資源に合ったサービスを提供するため，他のシステムと異なる．(今回の開発では，ビジネスロジック部の開発は含まれておらず，あくまで，それらのロジックを開発する環境を提供したに過ぎない．)

9 まとめ

本システムには大きく 2 つの特徴にまとめることができる．

- １つのアプリケーション (XAD) を実装すると同時に資源に特化したコンポーネントが存在すれば，その資源を持つ端末のアプリケーションも同時に構築することができる．これにより，複数の端末用のサービスが提供できるため，サービスの幅が広がり，さらに開発効率も上げることができる．
- 運用時のアプリケーション管理が容易となる．本システムは，システムを配信するときに資源状況などから判断して必要となるコンポーネントをダウンロードする．このメカニズムを応用することで，アプリケーション構成部品のバージョンアップがなされた場合に，容易に部品を交換し，再構成ができるようになる．

以上の特徴は以下のような効果をもたらすと考えている．本システムを応用することで，業務システムを含む様々な WEB アプリケーション (ネットワークアプリケーション) 開発の低コスト化が実現できる．これは，近年，各業種においてＩＴ化が進められているため，ＩＴシステム構築の需要は非常に大きくなっているが，ほとんどが人間の手作業による実装であり，時間がかかる作業であるため，システムの値段が非常に高価となってしまう．そのため，現状ではコストの問題から，中小企業においてのＩＴ化は，パソコンの導入，そして，市販されている業務パッケージソフトを導入するといった状況がほとんどであると考えられる．しかし，実際に IT 機器を導入することにより大きな効果を出すためには，その業種または，その企業の特徴にあわせて業務システムを選ぶ必要がある．結果的に業務システムを各企業にあわせてカスタマイズしたり，または一から設計して実装する．実際，このような業務システムの需要は現時点においても多い．将来的にみても，以上のようなＩＴ化の状況はさらに進むことが考えられるため，業務システム開発の需要がさらに大きくなることは必然である．本提案では業務システムを含む Web アプリケーション開発を主眼に置いているため，システム開発の需要が高まれば，そのシステム開発を請け負う開発会社において開発環境の需要も高まる．また，ソフトウェアの部品化も進み，その結果，部品を販売する会社も増えるだろう．本システムの成果では部品の配信技術も考えて入り，ソフトウェア部品の流通が多くなれば，本システムの成果が有効なアプリケーション流通のインフラとなりうる．そのため，市場規模としては，ソフトウェア業界全体に関係するものとなる．

今後の課題として以下の問題に取り組む予定である．

- **コンポーネントウェア配信技術の確立：**本システムは結果的に，コンポーネントを利用した従来からある開発方法を応用している．しかし，従来方法は，あくまでコンポーネントの設計，実装方法に着目した研究成果が主である．本システムでは，コンポーネントを配信して利用することに注目することで，コンポーネントの流通を活発化させるねらいがある．しかし，現状

では，あくまでコンポーネントのインフラを実現したに過ぎず，実際のシステム実装および保守に耐えうるのかという評価ついては着手していない．

- **実業務稼働環境での実証的評価：**コア・テクノロジーの実装においては、一定の完成度を得たものと評価している。しかし、実業務稼働環境においては、あらかじめ想定し得ない事態も十分に予想される。そこで次の課題として、このような事態を見極め例外的な事象に対しても確実に対応できる実証実験を行う必要があるだろう．

参加企業及び機関

- 有限会社カラビナシステムズ
- 静岡大学情報学部

謝辞

このプロジェクトの実行にはプロジェクトマネージャの東京大学松島克守教授には，システムの応用適用分野・ビジネス化等に関して適切なアドバイスをいただいた．またサポート組織としてネイチャーランドジャパンの齋藤昌義様には事務的サポートの他，ビジネスプランの提案をしていただいた．静岡大学富樫教授には本システム開発を行うにあたり様々なアドバイスをいただいた．本研究を進めるにあたり，静岡大学情報学部の方々には議論およびシステム実装の協力をいただいた．