2003－UBI－2 (12)
2003／11／18

動的な環境に適応可能な創発型アプリケーションのためのフレームワーク

板生知子† 山本淳† 田中聡† 須田達也‡

概要 Ja-Net は，ユビキタス環境において，時々刻々変化するネットワーク環境やユーザニーズに応じて，その場で利用可能なサービスを動的に連携することにより，適応的にアプリケーションを構築するためのフレームワークである．Ja-Net では，アプリケーションを，自律的に動作するコンポーネントである CE のグループとして構築する．CE は，実行時に，連携する他の CE を動的に検索することにより，アプリケーションを構成する CE を動的に決定する．また，他の CE とリレーションシップを生成し，アプリケーションに対するユーザ評価値を用いて，評価の高い（低い）リレーションシップを強める（弱める）ことにより，アプリケーションの構成を動的に変更する．本稿では，Ja-Net における適応的なアプリケーション実装のためのフレームワークについて述べ，アプリケーションの実装例を示し，フレームワークの有効性を考察する．

Framework for Adaptive and Emergent Network Applications

Tomoko Itao† Atsushi Yamamoto† Satoshi Tanaka† Tatsuya Suda‡

Abstract The Jack-in-the-Net Architecture (Ja-Net) provides a framework for autonomous and adaptive creation of network applications. In Ja-Net, an application is implemented by a group of autonomous service components called *cyber-entities* that are loosely coupled based on relationships among themselves. By creating and adjusting relationships through interacting with users, cyber-entities learn useful relationships and self-organize based on them to provide applications that users prefer. We describe the design and implementation of an application called AdAppli, which enables adaptive provision of commercial advertisements to customers, to show the benefit of the framework.

1 はじめに

ユビキタス環境では，ネットワークを介してモノ，情報，機能（サービス）を状況に応じて動的に組み合わせることにより，ユーザニーズに応じたアプリケーションをタイムリに提供することが重要である．ここでいうモノ，情報，サービスの例としては，情報家電，携帯電話，ウェアラブルデバイス（モノ），センサ情報，個人情報，情報コンテンツ（情報），センサやアクチュエータの制御，情報検索，データベース（サービス）などがある．ユビキタス環境をサポートするネットワークは，多種多様なモノ，情報，サービスをコンポーネントとするアプリケーションのインフラストラクチャとしてとらえることができる．このとき，ユーザやコンポーネントを取り巻く状況は動的に変化するため，状況に応じて有効なサービスの形態を創出可能な，適応能力が要求される．

筆者らが提案している創発型ネットワーキングサービスプラットフォーム Ja-Net[1][2] は，コンポーネント間の自律分散的なインタラクションを基礎として，動的に変化する環境に適応可能なアプリケーションを構築するためのフレームワークである．Ja-Net では，あらゆるシステム構成要素を，自律動作可能なコンポーネントであるサイバーエンティティ（CE）として表し，複数の CE から構成されるグループとしてアプリケーションを構築する．そして，CE 間の関係（リレーションシップ）に基づいて，各 CE が自律的に連携方法を制御することにより，アプリケーションを適応的に提供することを可能とする [3]．Ja-Net は，アプリケーション開発者に対して，CE の生成，発見，CE 間の連携，リレーションシップの生成・更新などの API と，CE の実行環境を提供する．アプリケーション開発者は，リレーションシップのセマンティクスをアプリケーショ

† NTT 未来ねっと研究所
NTT Network Innovation Laboratories
‡ カリフォルニア大学アーバイン校
University of California, Irvine

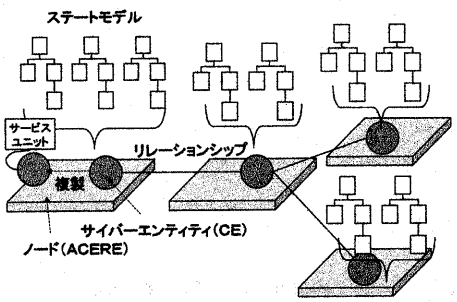

図 1: Ja-Net のシステム概要

ンごとに自由に設計することが可能であるため，CE とリレーションシップという統一的なフレームワークをベースとして個別の自律分散型アプリケーションを構築することができる．

本稿では，アプリケーション開発者の視点から見た Ja-Net フレームワークについて述べ，Ja-Net 上でのアプリケーションの設計事例について述べると供に，フレームワークに基づいて実現されるアプリケーションの適応能力について考察する．

2 Ja-Net

図 1 に，Ja-Net のシステム概要を示す．Ja-Net は，CE と CE の実行環境である ACERE (ACE Runtime Environment) から構成される．CE に共通的な振る舞いは，ACE (Abstract Cyber-Entity) と呼ばれる CE の基底クラスによって提供される．ACERE は，PC などのコンピュータ上で実行され，ACERE 同士は，peer-to-peer 型のオーバーレイネットワークを形成する．CE は，アプリケーションに関連したサービスと，移動，複製，リレーションシップ生成などの振る舞いを持ち，近隣 ACERE あるいはリレーションシップ相手の中から連携相手となる CE を動的に検索しつつ，メッセージングにより連携してアプリケーションを提供する．なお，Ja-Net は，Java 言語によって記述されており，オブジェクト間通信には RMI を用いている．

2.1 フレームワークの構成要素

以下に，CE を用いた適応的なアプリケーション構築フレームワークのキーとなる要素について説明する．

2.1.1 CE

CE は 1 つ以上のサービスと，移動，複製，リレーションシップの生成，死などの振る舞いを持ち，自己の意思決定に基づいて自律的に振舞う．そして，CE 間のメッセージングにより，他の CE と連携する．CE の粒度はアプリケーションに応じて異なる．例えば、あるアプリケーションでは，CE は飛行機予約サービスを，別のアプリケーションでは MPEG コンバータを表すかもしれない．

2.1.2 ステートモデル

CE のサービスは，ステートモデルを用いて定義する．ステートモデルを構成する各状態は，状態遷移ルールとアクションから構成される．また，ステートモデルは，これらの状態間の遷移グラフとして定義される．ステートモデルは，メッセージ（イベント）の受信を契機として起動される．そして，次の状態への遷移ルールを調べ，これが満たされる場合に次の状態に遷移し，アクションを実行する．ステートモデルの初期状態へ遷移するためのルール（すなわち、新しいステートモデルを起動する条件）は初期条件と呼ばれる．ステートモデルの起動時に，ステートモデル実行オブジェクト（スレッド）を生成することにより，1 つのステートモデルを並列多重に実行することが可能である．

2.1.3 サービスタイプとサービスユニット

サービスタイプは，CE が提供するサービスに対するメタデータであり，現状ではサービス名を表す文字列により実現している．CE のサービスは，他の CE が提供するサービスを構成要素として，階層的に定義することができる．このとき，CE のサービス間の階層関係はサービスユニットを用いて定義される．サービスユニットは，サプライタイプとディマンドタイプと呼ばれる 2 種類のサービスタイプを持つ．サプライタイプは，CE が他の CE に対して提供するサービスタイプである．ディマンドタイプは，CE のサービスの構成要素として必要な，他の CE が提供するサービスタイプ（サプライタイプ）である．サービスタイプは，特定のサービスタイプを持つ CE を検索するために用いる．

2.1.4 リレーションシップ

CE は，他の複数の CE とリレーションシップを生成することができる．リレーションシップは，ある CE から見た，他の CE に関する情報であり，相手の名前，ID，サービスタイプ，リレーションシップ名，強度などの属性を持つ．ここで，リレーションシップ名は，相手 CE と連携する目的を表す．また，強度は，相手の有効度を表す．

CE が，リレーションシップのある他の CE と連携してアプリケーションを提供した場合に，ユーザの満足度を反映した評価値をシステムに入力することによって，連携相手の CE へのリレーションシップ強度を評価値に応じて変更することができる．このとき，システムは，リレーションシップが強い（弱い）ほど，有効度が高く（低く）なるように変更される．

2.2 CE の設計

アプリケーションを実装するにあたって，開発者は，まず最初にアプリケーション論理を CE の単位に分割し，CE のクラスを抽出する．続いて、CE 間のコラボレーションと，各 CE のステートモデル（状態遷移グラフ）を設計する．これらの設計に基づいて、CE クラスを作成し，状態遷移ルールとアクションを実装し，ステートモデルに登録する．表 1 に，CE 実装クラスのコンストラクタの記述イメージを示す．CE

```
public class MyCE extends AbstractCyberEntity
{
 public MyCE() {
   //
   // サービスユニットの生成
   //
   ServiceType supplyType = new ServiceType(.., serviceName,..);
   ServiceType demandType = new ServiceType(.., serviceName,..);
   ServiceUnit su = new ServiceUnit(supplyType, demandType);

   //
   // ステートモデルの生成
   //
   StateModel stateModel = new StateModel("name", su);

   //
   // 初期状態 (s1) の登録
   // rule0 は初期遷移ルール，reaction0 は状態 1 のアクション
   stateModel.enter(s1, rule0, reaction0);

   //
   // 状態 s1, s2 から構成される遷移グラフの生成.
   // rule1〜3 は状態遷移ルール，reaction1〜3 はアクション
   // 状態 1 (s1) においてメッセージを受信すると，遷移ルール (rule2) が
   // 成立するかを調べ，成立する場合は 状態 2 (s2) に遷移し，アクション
   // (reaction1) を実行する.
   // 以下同様
   stateModel.join(s1, s2, rule1, reaction1);
   stateModel.join(s2, null, rule2, reaction2);
   stateModel.join(s2, s1, rule3, reaction3);

   //
   // AbstractCyberEntity へのステートモデルの登録
   //
   super.addStateModel(stateModel,..);

   ...

 } // end of MyCE()

} // end of class MyCE
```

表 1: CE クラス定義

は，AbstractCyberEntity クラスを継承し，サービスユニット，ステートモデルのインスタンスを生成する．ステートモデルの状態遷移グラフに，状態間の遷移順序を定義する．

2.3 CE の動作パターン

表 2 に，CE の代表的な API を示す．ステートモデルのアクションの中からこれらのメソッドを呼び出すことによって，自律分散的にアプリケーションの制御を行なう．以下に，API を活用した CE の動作パターン紹介する．

■メッセージ送信 CE のメッセージ送信方法としては、ポイント・トゥ・ポイント型 (speechAct) と，不特定多数への同報型 (emit) の 2 通りがある．

■通信相手の検索 CE は，サービスタイプまたはリレーションシップの属性を条件として，通信相手を検索する．検索の方法としては，以下の 3 通りがある．

1. Select API を用いて，近隣 ACERE 上に存在する CE の中から検索する．
2. LocalSelect API を用いて，自身の持つリレーションシップを検索する．
3. SocialSelect API を用いて，自身の持つリレーションシップを起点とし，リレーションシップを辿ることによって特性の条件を満たす CE を検索する [4].

■リレーションシップの生成 CE A が，CE B とのメッセージ送信または受信を契機として，create API を用いて CE B への片方向のリレーションシップを生成する．

■ユーザ評価に基づくリレーションシップ強度の変更 リレーションシップのある CE A, B が連携してアプリケーションを提供し，これを利用したユーザの評価値を，reward API を用いて CE A または B に入力することにより，CE A, B 間のリレーションシップ強度を変更する（双方向）．

表 2: CE の API

API	説明
speechAct	引数で指定した CE へのポイント・トゥ・ポイントメッセージの送信
emit	マルチキャストメッセージの送信
act	メッセージの受信
create	引数で指定した CE のリレーションシップを生成
delete	引数で指定した CE のリレーションシップを削除
reward	引数で指定したサービスシーケンスに沿ったリレーションシップの強度を，引数で指定した評価値に応じて更新
select	引数で指定した属性またはリレーションシップを持つ CE を ACERE ネットワーク上で検索
localSelect	引数で指定した属性またはリレーションシップを持つ CE を自身の持つリレーションシップのリストの中から検索
socialSelect	引数で指定した属性またはリレーションシップを持つ CE をリレーションシップネットワーク上で検索
migrate	引数で指定した ACERE に CE を移送する
replicate	CE を複製する

3 AdAppli:商店街におけるマスの嗜好を反映した広告配信

Ja-Net におけるアプリケーションの設計方法を，商店街において，複数の客の嗜好に応じて適応的に広告を配信するアプリケーション AdAppli を例として説明する．AdAppli では，商店街の各店舗が，客が購入した商品の組み合わせの履歴に基づいて，組み合わせ頻度の高い店舗の間でリレーションシップを生成，強化する．例えば，店舗 A，B の商品を組み合わせて購入する客が多い場合には，店舗 A, B のリレーションシップが強化される．そして，各店舗 (A, B) は，自店を訪れた客の携帯端末に，リレーションシップ相手の店舗 (B, A) の広告を提示することにより連携し，「自店の客に人気のある他店」を紹介する．これにより，商店街において、複数のユーザの嗜好に応じた広告配信が実現される．

3.1 アプリケーションロジック

■店舗による広告配布 店舗は，ユーザが入店したことを検出し，ユーザの端末上に広告を表示する．その際に，自身の広告と，リレーションシップ相手の広告(複数可) を表示する．広告は，広告をユーザに送信した店舗（送信元店舗）と，広告の商品を提供している店舗（生成元店舗）の名前を情報として保持する．

■ユーザによる商品購入 ユーザは，広告を見て，商品を購入するか否かを意思決定する．購入する場合は，購入要求を広告の生成元店舗に送信する．その際に，クーポンを持っている場合は，クーポンを一緒に送信する．

■店舗によるクーポン発行 店舗は，自店の商品を購入したユーザに対して，商店街で共通に使えるクーポンを発行し，ユーザ端末に送付する．クーポンは，クーポン発行元の店舗（すなわち，生成元店舗）の名前を情報として保持する．

■ユーザ評価 店舗は，ユーザ端末に表示した広告に対するユーザ評価値をユーザからインタラクティブに取得する．ユーザ評価値は，1 (好き)，0 (中立)，-1 嫌い) の 3 段階を設ける．ユーザが商品を購入した場合は，ユーザ評価 1 を，ユーザが明示的に嫌いであることを意思表示した場合は，ユーザ評価-1 を割り当てる．評価値は，広告の送信元店舗名の情報と共に，広告の生成元店舗に通知される．

■リレーションシップ生成 広告の生成元店舗は，ユーザがクーポンを使って商品を購入する際にユーザから通知される，過去の生成元店舗（過去に商品を購入した店舗）とリレーションシップを生成する．例えば，店舗 A，B の間にリレーションシップが未生成で，ユーザ X が店舗 A を訪れて商品を購入し，クーポンをもらう．次に，店舗 B を訪れて，クーポンを使って商品を購入する．この結果，店舗 B はクーポンの発行元である店舗 A とリレーションシップを生成する．

■リレーションシップ強度の変更 店舗は，自身の広告に対するユーザ評価値と共に通知される，該広告の送付元店舗（ユーザに広告を配布した店舗）のリレーションシップ強度を，ユーザ評価値に基づいて変更する．例えば、店舗 A と店舗 B がリレーションシップを持っており，店舗 A が店舗 B の広告をユーザ Y に配布し，ユーザ Y が店舗 B に評価値 1 を返した場合，店舗 B は，広告の送信元である店舗 A に対するリレーションシップを強化する．

3.2 設計

AdAppli では，店舗，ユーザ，広告，ユーザ端末の 4 つを CE として抽出した．広告は，ユーザ端末に表示するコンテンツをサービスとして提供する．ユーザ端末は，任意のコンテンツを画面に表示する機能と，画面からの入力機能をサービスとして提供する．

図 2 に，(広告を送信するための CE のコラボレーションを示し，以下に説明する．

1. ユーザ CE は店舗 CE に広告を要求する

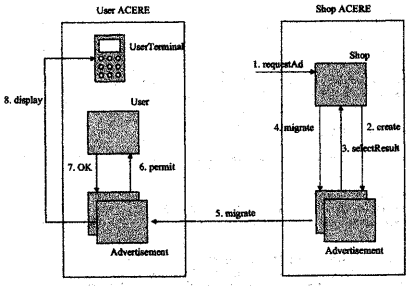

図 2: 広告送信のコラボレーション

2. 店舗は広告を生成する
3. 店舗は生成した広告を検出する
4. 店舗は，広告をユーザのユーザ端末上に移動させる
5. 広告は，ユーザ端末上に移動する
6. 広告は，ユーザに，広告の表示許可を求める
7. ユーザは，広告の表示を許可する
8. 広告は，広告コンテンツをユーザ端末に送付し，表示を依頼する

図 3 に，店舗間のリレーションシップ生成のコラボレーションを示し，以下に説明する．

1. ユーザが商品を購入する（広告画面において “Buy” ボタンを押す) と，広告がこれを検出し，ユーザに通知する．
2. ユーザは広告を削除する．
3. ユーザは，店舗 1 に商品購入要求を通知する．その際に，店舗 2 によって発行されたクーポンも店舗 1 に送付する．
4. 店舗 1 は，クーポンを発行し，ユーザに送付する．
5. ユーザはクーポンを画面に表示する．
6. 店舗 1 は，店舗 2 にリレーションシップの生成を依頼する．
7. 店舗 2 は合意し，店舗 1 のリレーションシップを生成する．
8. 店舗 1 は，店舗 2 のリレーションシップを生成する．

3.3 実装結果

AdAppli の動作環境およびウィンドウイメージを図 4 に示す．無線 LAN のアクセスポイントごとに店舗を設定し，2 つの店舗を動作させた．ユーザ端末に表示される広告画面には，“商品購入 (Buy)”，“広告消去 (Close)”のボタンと，“この商品は嫌い (don't like)” というチェックボックスを表示する．このとき，広告を見たユーザが商品購入ボタンを押下した場合は評価値 1 を，チェックボックスをチェックしていずれかのボタンを押下した場合には評価値-1 を，それ以外のボタン押下のケースについては評価値 0 を割り当てる．

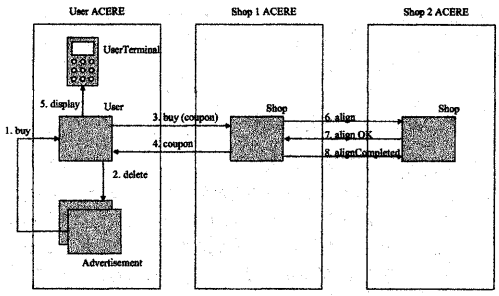

図 3: リレーションシップ生成のコラボレーション

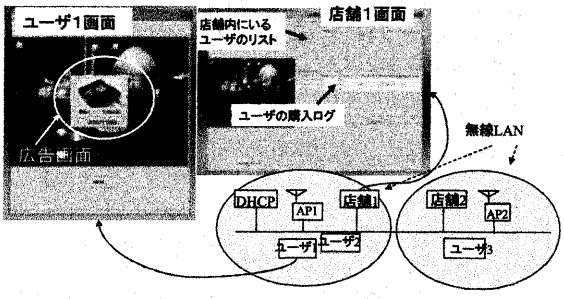

図 4: AdAppli のシステムイメージ

表 3 に，Ja-Net (ACERE, ACE のライブラリ) のプログラムサイズおよび AdAppli を構成する CE のプログラムサイズを示す．また，各 CE が持つステートモデルの数を示す．

表 3: プログラム規模の比較

	Ja-Net	AdAppli CE			
		店舗	ユーザ	広告	ユーザ端末
プログラムサイズ	388	222	52	282	194
ステートモデル数	N/A	9	9	3	3

表 4 に，図 2, 3 で示した一連のコラボレーション (ユーザに広告を送信し，購入要求に基づいて店舗間でリレーションシップを生成する) において発生したメッセージの数とサイズを示す．

3.4 適応度について

店舗間のリレーションシップを生成，変更することにより，商店街を訪れた複数のユーザの嗜好に適応して商店街の各店舗が配布する広告の内容が変化していくことを示すために，シミュレータを用いて AdAppli の適応度に関する一次測定を行なった．適応度の指標

表 4: メッセージサイズの比較

種別	数	サイズ (total)	サイズ (average)
Ja-Net	9	37 KB	14 KB (移動), 1.3 KB (移動以外)
AdAppli	2	6 KB	3.3 KB

としては，店舗に入ってきたユーザに表示する広告が，ユーザの嗜好にマッチする事象の再現率，適合率，F 尺度を用いた．

適応度の評価モデルを次に述べる．広告ごとに商品を表すキーワードを付与し，ユーザの嗜好を広告キーワードにより表現する．そして，ユーザが広告を受信した時に，同ユーザの嗜好を表す広告キーワードと，受信した広告のキーワードを比較し，両者が一致する (しない) 場合はユーザの嗜好にマッチしている (いない) 広告が配信されたと見なす．このとき，ある店舗 A において，一定期間中にユーザに広告 (複数可) を配布した場合，その期間における再現率 (R)，適合率 (P)，F 尺度 (F) は，それぞれ以下のように表すことができる．

$$R = \frac{C}{T} \cdot 100 \tag{1}$$

$$P = \frac{C}{H} \cdot 100 \tag{2}$$

$$F = \frac{2RP}{R+P} \cdot 100 \tag{3}$$

ただし，C は，店舗 A が広告を配布したユーザのうち，実際にその広告のキーワードを持っていたユーザの数，T は，店舗 A が配布した広告のキーワードを持っている全ユーザの数 (潜在顧客数)，H は，店舗 A が広告を配布したユーザの数である．再現率と適合率はトレードオフの関係にあり，F 尺度により両者を統合する．このとき，F 尺度の最大値は 100 であり，100 に近いほど適合度が高い．

本測定では，リレーションシップを用いた手法 (relational) における F 値の推移を測定する．また，比較対象として，(1) global, (2) isolated の 2 つの手法についても F 値の推移を測定する．Global, isolated では，いずれも店舗はリレーションシップを持たない．Global の場合は，各店舗は他の全ての店舗の広告を保持し，自店を訪れたユーザに対して商店街の全店舗の広告を表示する．この場合は，再現率は高いが，適合率が低くなることが予想される．一方，isolated の場合は，各店舗は自店を訪れたユーザに対して自店の広告のみを表示する．この場合は，再現率，適合率ともに低くなることが予想される．

測定は，36 店舗 (36 種類の広告キーワード)，50 人のユーザを想定したシミュレーションにより行なった．シミュレーションの単位時間の間に，各ユーザは 1 回だけ店舗間を移動し，広告を受信する．受信した広告

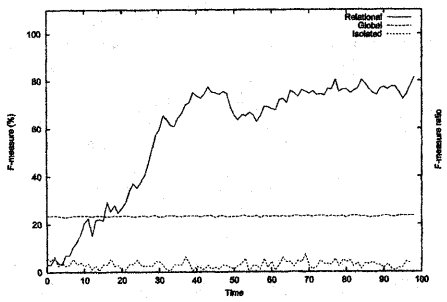

図 5: F 尺度の推移

が嗜好にマッチする場合は，0.7 の確率で，生成元店舗に移動し，商品を購入する．また，ユーザ評価は，全ての広告について返すものとする．店舗は，ユーザから受信した各イベントを逐次処理し，リレーションシップを生成，更新する．各ユーザには，5 つの異なるキーワードを設定した．シュミレーション開始時には，ユーザを店舗にランダムに配置した．また，店舗の保持しているリレーションシップ数を 0 とした．

図 5 に，シミュレーション結果を示す．グラフの横軸は，シミュレーション時刻，縦軸は，F 尺度である．提案手法 (relational) は，時間が経過すると供に，店舗間のリレーションシップが生成されるため，F 尺度が時間と供に上昇し，最高で 80 %に達するのに対し，リレーションシップを生成しない global では 20 %，isolated では数%程度となった．また，relational において，最終的に生成されたリレーションシップの数は，店舗当たり平均 5.5 本であった．

4 関連研究

Ja-Net は，CE の動的な検索，連携，およびアプリケーションのカスタマイズなど，動的な環境において適応的にアプリケーションを構築するために必要な一連の機能を内包している点で，従来の Web ベースのサービスアーキテクチャや Web サービスフレームワーク [5] と異なる．

エージェント間のインタラクションによってアプリケーションを構築するフレームワークとしては，Hive[6] や Bee-gent[7] がある．Hive では，エージェント間のインタラクションはメソッド呼び出しによって行なうため，インタラクションの柔軟性が制限される．これに対して，Ja-Net では，CE はメッセージングにより連携するため，動的なインタラクションに適している．

一方，Bee-gent では，仲介エージェントが複数のエージェント間のメッセージフローを集中管理することにより，アプリケーションの開発・管理を容易化する．これに対して，Ja-Net では，集中的な機構を排除し，CE 間の分散型のインタラクションに基づいてボトムアップ的にアプリケーションを構築することから，Bee-gent の仲介エージェントによるトップダウン的な方法では構築できない新しいアプリケーションを創発できる可能性がある．

5 まとめと今後の課題

AdAppli の一次評価結果は，Ja-Net が，オープンかつダイナミックなユビキタス環境において，自律的なサービスコンポーネントである CE と，それらの間のリレーションシップを用いて，適応的なアプリケーションを構築することが可能であることを示唆している．今後，更に詳細な評価を行う予定である．

Ja-Net の今後の課題としては，他技術に対する Ja-Net フレームワークの有効性の評価がある．例えば，リレーションシップを用いた適応技術については，データマイニングとの比較・評価などを行なう必要がある．また，携帯電話などのリソースが限られた端末や，小型デバイスなどを活用するためには，Ja-Net プラットフォームソフトウェアの軽量化や，非 IP ネットワークを想定したデバイス制御技術が必要である．

参考文献

[1] 須田達也，板生知子，中村哲也，松尾真人，“サービス創発のための適応型ネットワークアーキテクチャ，” 信学論 (B)，vol.J84-B, no.3, pp.310-320, March, 2001.

[2] 須田達也，松尾真人，板生知子，中村哲也，今田美幸，大塚卓哉，田中聡，“アプリケーション創発のための適応型ネットワーキングアーキテクチャ: Ja-Net,” 情報処理，Vol. 43, No. 6, June, 2002.

[3] 板生知子，中村哲也，松尾真人，田中聡，須田達也，青山友紀，“ユーザ嗜好に応じた動的なサービス構成のためのリレーションシップメカニズムの設計と評価，” 情処学論，vol.44, no.3, pp.812-825, March, 2003.

[4] 川西 直，板生 知子，南 正輝，森川 博之，青山 友紀，“適応型ネットワークアプリケーションのための類似性に基づいた自律分散コンポーネント発見方法，” 情処 DICOMO 2002, pp.205-208, June 2002.

[5] Web services web site, http://www.webervices.org/

[6] N. Minar, M. Gray, O. Roup, R. Krikorian and P. Maes, “Hive: Distributed Agents for Networking Things,” Proc. of the ASA/MA '99, Aug. 1999.

[7] T. Kawamura, Y. Tahara, T. Hasegawa, A. Ohsuga and S. Honiden, “Bee-gent: Bonding and Encapsulation Enhancement Agent Framework for Development of Distributed Systems,” Journal of the IEICE, D-I, vol. J82-D-I, no.9, 1999.