ネットワークコミュニティ支援システムのエージェント指向フレームワーク Shine

吉田 仙[†] 亀井剛次[†]
大黒 毅[†,☆] 桑原和宏[†]

ネットワークコミュニティを支援するコミュニケーションシステムを対象とするフレームワーク Shine について論じる．Shine は，ネットワークコミュニティを支援する各種コミュニケーションシステムに統一的な枠組みを与え，それらのシステム間での部品の共有や動作の連携を可能にする．また，Shine はエージェント指向のアーキテクチャを持っており，各ユーザに対応するエージェントが互いに直接情報交換することでそれぞれのユーザに見合ったコミュニティを形成できるようにしている．Shine のこれらの性質について述べるとともに，いくつかのシステムへの Shine の適用例を示しフレームワークとしての有効性を確かめる．

Shine: An Agent-oriented Framework of Network Community Support Systems

Sen Yoshida,[†] Koji Kamei,[†] Takeshi Ohguro[†,☆] and Kazuhiro Kuwabara[†]

A discussion is presented on Shine, a framework of communication systems that support network communities. Shine provides a unified structural design approach to various network community support systems and enables them to share software components and cooperate with each other. Furthermore, Shine has an agent-oriented architecure, which enables personal agents to form communities suitable for each user by exchanging data in peer-to-peer style. These features of Shine are described and its effectiveness as a framework is verified by applying it to several systems.

1. はじめに

ネットワークコミュニティまたはサイバーコミュニティと呼ばれる人々の集まりが，ネットワーク上に形成されている．ネットワークコミュニティのメンバは，電子メールやチャットルーム，電子掲示板などのコミュニケーションシステムを用いて活発にやりとりし，互いに親密な人間関係を築く．このようなネットワークコミュニティは，インターネットなどの公衆通信網の発展にともなって増大している．

こうした情勢を受け，電子メールなどの旧来のコミュニケーションシステムにとどまらず，マルチメディア通信や自然言語処理といったより高度な技術を用いてネットワークコミュニティを支援するコミュニケーションシステムの開発が数多く進められている[4),19)]．これらのシステムはソーシャルウェアと呼ばれる[3)]．

我々は，ソーシャルウェアの様々なアプリケーションシステムの開発において，それらに統一的な枠組みを与え，システム間での部品の共有や動作の連携を可能にするため，ソーシャルウェアのフレームワークを構築することを提唱し[22)]，実際に Shine というフレームワークの開発を進めている[21),23)]．このフレームワークの特徴は，エージェント指向のアーキテクチャを持つことにより，各ユーザに対応するエージェントが互いに直接情報交換することでそれぞれのユーザに見合ったコミュニティを形成できるようにしていることである．

本論文では，2 章においてソーシャルウェアのエージェント指向プラットフォームについて概説し，3 章で Shine の具体的な構成を説明する．4 章ではいくつ

† 日本電信電話株式会社 NTT コミュニケーション科学基礎研究所
NTT Communication Science Laboratories, Nippon Telegraph and Telephone Corporation
☆ 現在，東日本電信電話株式会社（NTT-ME）
Presently with Nippon Telegraph and Telephone East Corporation (NTT-ME)

かのアプリケーションシステムへの Shine フレームワークの適用例について述べる．5 章で関連研究に言及して考察を加え，6 章でまとめとする．

2. ソーシャルウェアのエージェント指向フレームワーク

この章では，ソーシャルウェアのフレームワークの必要性について論じ，またそのアーキテクチャとしてエージェント指向のアプローチが有効であることを説く．

2.1 ソーシャルウェアのフレームワークの必要性

ソーシャルウェアとは，ネットワークコミュニティを支援するコミュニケーションシステムの総称であり，そのアプリケーションはいろいろな種類のものが考えられる．ここで，それらアプリケーションを，コミュニティを形成することのメリットを引き出すタイプのものと，コミュニティの形成や維持，発展を支援するタイプのものに大別して以下に列挙する．

コミュニティを形成することのメリットを引き出すタイプのものは，

- 推薦システムや協調フィルタリングのような，人から人への口コミに基づく情報流通[18]，
- ブレインストーミングなどのインフォーマルで創造的な会話[9),10)]，
- オープンソースソフトウェアやデータベースなどの資源の共同開発，
- オークションやフリーマーケットなど，小規模で手軽な経済活動，

などを支援の対象とする．

またコミュニティの形成や維持，発展を支援するタイプのものは，

- 効果的な出会いの提供や，潜在的なコミュニティに気付かせること[5),12)]，
- 親密さやコミュニティ感覚の維持[14]，
- コミュニティにおける自我の確立の支援，

といったことを目的とする．

このようにソーシャルウェアには様々なアプリケーションが考えられるが，これらのアプリケーションの間にはいくつか共通する機能が存在する．すなわち，コミュニティにおける各人の情報をうまく格納し利用する機能や，メンバが流動的なコミュニティに適応する柔軟なコミュニケーション手段といったものである．しかし，これら共通の機能をまとめた統一的な枠組みは，現在のところ存在しない．その結果，たいていのソーシャルウェアアプリケーションは独立したシステムとして開発されており，アプリケーションシステム間での連携や，プログラム部品の共有，再利用といったことは行われていない．

ところで，近年，ソフトウェア開発におけるソフトウェアアーキテクチャの重要性が認識されるようになってきている[8)]．ソフトウェアアーキテクチャには以下のような利点があるといわれる．

- 設計時に統一的な枠組みを与える．
- 部品の共有や再利用，あるいはシステムの統合や協調を可能にする．
- 開発者間の意思疎通の助けになる．

ソーシャルウェアのアプリケーションシステムにおいても，それに適したソフトウェアアーキテクチャを提供できるフレームワークが存在すれば，上記のような利益を享受でき，より豊かなソーシャルウェア環境の実現が期待できる．

このような動機から，我々は，様々なソーシャルウェアアプリケーションの共通の基盤として，Shine というフレームワークを開発している．

2.2 ソーシャルウェアにおけるエージェント指向アーキテクチャの有効性

我々は，Shine を設計するにあたり，エージェント指向のアーキテクチャを採用している．本節において，エージェント指向アーキテクチャがソーシャルウェアに有効な理由を説明する．

ネットワークコミュニティは，実世界のコミュニティと比べて，地理的あるいは時間的制約を受けずにコミュニケーションができるという強みがある．しかし，制約から解放されコミュニケーションの可能性が広がるがゆえに，ともすると情報過多に陥ったり，あるいは行動をためらって孤立化したりすることにもなる．したがってソーシャルウェアでは，適切な範囲で自然なコミュニケーションが行われるようなコミュニティが構成されるようにシステムを設計することが重要である．

従来のネットワークコミュニティは，サービスプロバイダが提供する電子掲示板やチャットなどのシステムの上に存在するものが一般的である．このようなクライアントサーバ型のソーシャルウェアアプリケーションシステムでは，しかしながら，必ずしも適切な範囲で自然なコミュニケーションが行われるようなコミュニティが構成されるようになっていない．クライアントサーバ型のソーシャルウェアアプリケーションシステムでは，コミュニケーションの範囲を人々の興味によって区切り，それぞれが興味のコミュニティ（*communities of interest*）となるようにしていることが多い．しかし，興味によって区切られたからといっ

て各コミュニティの大きさが限られるわけではなく，情報過多の可能性はなくならない．また，あまり細かい興味分野で区切るとコミュニティのメンバの多様性が減り，雑多な人々の集団であるからこそ生ずるコミュニティの面白さが失われてしまう．さらに，ネットワーク上に複数のサーバがある場合には，異なるサーバ上のユーザ同士がコミュニケーションをとることが難しくなる．

一方実社会では，人々は 1 対 1 の人間関係，つまり peer-to-peer の関係をベースとし，仲介や自己紹介を通じて知らない他人と会う．その結果，人々は知人関係を柔軟に変更し，各人に見合ったコミュニティを他人と協調しながら自律的に構成していく．そして，このようにして形成されている社会は，コミュニティ内の密なつながりを保ちつつも「世界は狭い」と感じさせるような連鎖のよさもある *small-world* になっている[20]．

ネットワークコミュニティにおいても，実社会のコミュニティと同様に，各人が他人と協調しながら自律的にコミュニティを構成していくことができれば，密なつながりと連鎖のよさをあわせ持つ small-world 的ネットワーク社会が形成されることが期待できる．このようなコミュニティの構成方法を可能にするには，各ユーザに対応するエージェントが互いに直接情報交換することでそれぞれのユーザに見合ったコミュニティを形成できる peer-to-peer 型のマルチエージェントシステムが望ましい．

このことに加え，近年ファイル交換システムなどの peer-to-peer システムが普及しているが[16]，これらはクライアントサーバ型のシステムに比べ次のような利点を持つ．

- 負荷を分散でき，大規模な装置が不要になる．クライアントサーバ型のシステムでは大量の処理要求に耐える性能を持つサーバ装置が必要であるが，peer-to-peer システムではそのようなものは不要である．
- システムが頑健になる．クライアントサーバ型ではサーバが故障するとシステム全体が停止するが，peer-to-peer 型ではいくつかの peer が故障してもシステム全体の停止には至らない．

以上の理由から我々は，Shine を peer-to-peer のネットワークを構成するマルチエージェントシステムとして設計するアプローチをとる．

我々のアプローチでは，コミュニケーションの管理機構やデータは各人のもとに分散配置される．つまり，Shine フレームワークは，**図 1** に示すように，各ユーザに対しそのユーザの社会的インタラクションに関わるパーソナルエージェントを提供する．Shine のパーソナルエージェントは他のパーソナルエージェントと peer-to-peer の情報交換を行い，各ユーザに見合ったコミュニティを柔軟に構成していく．

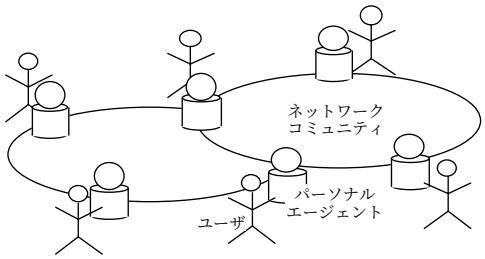

図 1 パーソナルエージェント
Fig. 1 Personal agents.

3. Shine エージェントの内部構造

この章では，Shine の具体的な構成について説明する．

2 章で述べたように，Shine ではユーザごとにパーソナルエージェントが提供される．**図 2** は Shine エージェントの内部構造の概念図である．Shine エージェントの内部には，**ひとデータベース**，**プラン実行モジュール**，および**通信モジュール**というシステムの核となるモジュール群が存在する．また，1 つのエージェント上に 1 つまたは複数のアプリケーションシステムが載る．Shine は具体的には Java☆のようにネットワークを利用できる型付き手続き型プログラミング言語で実装される．アプリケーションシステムは各モジュールの機能をアプリケーションプログラミングインタフェース（API）を通じて利用し，それぞれのサービスをユーザに提供する．

Shine では，様々なアプリケーションシステムが同一の API を利用してそれぞれのサービスを実現する．その結果，同じ API を使うソフトウェア部品をライブラリとして複数のアプリケーションシステムの間で共有，再利用することができる．たとえば，データベースに蓄積されたデータをユーザに分かりやすく表示するための可視化機構などは，多くのアプリケーションシステムが同様の機能を必要とすることから，1 つ部品を作ればそれを広く使いまわすことができる．

以下，各モジュールについて詳説する．

3.1 ひとデータベース

ひとデータベースは，人およびパーソナルエージェ

☆ Java は Sun Microsystems 社の登録商標である．

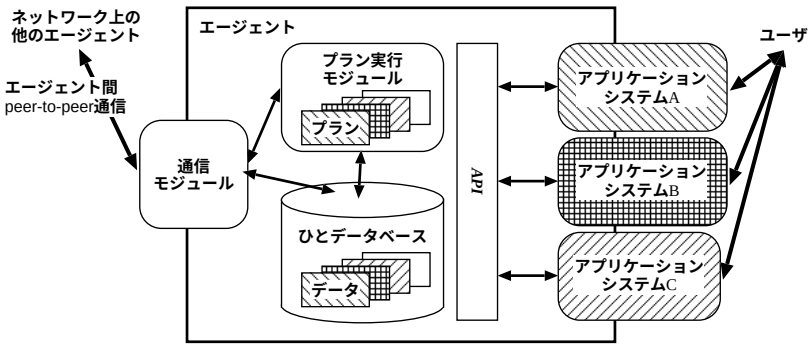

図 2 Shine エージェントの内部構造
Fig. 2 Internal structure of Shine agent.

ントに関するデータを保持する．保持されるデータには，エージェント自身やそのエージェントが対応しているユーザの情報と，そのエージェントが知っている範囲の他のエージェントおよびそれらが対応しているユーザの情報が含まれる．Shine は peer-to-peer システムであり，各エージェントが知っている範囲はそれぞれ異なる．エージェントは，そのユーザに見合ったコミュニティを形成するのに必要なデータをひとデータベースに保持し，必要に応じて他のエージェントとデータを交換する．

Shine のアーキテクチャでは，ユーザとそのユーザのパーソナルエージェントは 1 対 1 に対応する．このため，ひとデータベースには，人のデータとその人に対応するパーソナルエージェントのデータが区別なく格納される．人のデータにはたとえば名前や生年月日などがあり，パーソナルエージェントのデータにはエージェント ID やエージェント間通信のためのアドレスなどがある．各アプリケーションシステムはこれらのデータを参照してサービスを行い，また必要に応じてデータを更新する．

ひとデータベースにおいて，「名前」や「生年月日」，「エージェント ID」などはデータの属性と呼ばれる．また，各属性のそれぞれのデータは属性値と呼ばれる．以下では，属性の集合を A，ある属性 $a \in A$ の属性値を $x_a \in X_a$ と記す．

Shine のひとデータベースは，人と属性の組から属性値へのマッピングを行う表を持つ．**図 3** にこの表の例を示す．表の各行は，1 人の人間，およびその人に対応しているエージェントを表す．また各列は 1 つの属性を表す．つまり，ひとデータベースに格納されている人の集合を P とすると，この表の 1 つの列は P から X_a へのマッピング $f_a : P \to X_a$ である．このマッピングを用いて任意の人 $p \in P$ の属性 a の属性値 $f_a(p) \in X_a$ を求める機能が API として提供される．

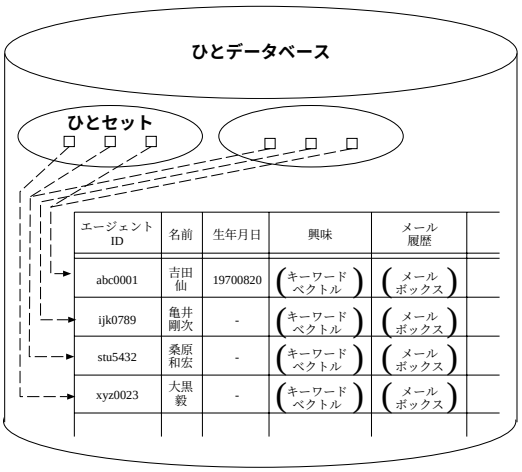

図 3 ひとデータベース
Fig. 3 Person database.

ひとデータベースに格納されるデータの形態は様々である．エージェント ID やユーザの名前などの単純な数値や文字列のデータのほかに，複雑な形態を持つものも含まれる．たとえば，情報検索の分野で人の興味を概括的に表すのに用いられるベクトル空間モデルでは，データはキーワードベクトル，すなわちキーワード文字列と数値のペアの集合という形態をとる．すなわち

$$\{\langle \text{旅行}, 0.7\rangle, \langle \text{食べ物}, 0.3\rangle, \cdots\}$$

というようなものである．また，ある人から受け取ったメールを溜めておく「メールボックス」なども，人に関するデータとしてデータベースに格納することが考えられる．Shine のひとデータベースは，こうした様々な形態のデータも，構造化された属性値として表に格納できる．

表に格納される属性のうち，値から人もしくはその人に対応するエージェントを特定することができるも

のはキー属性と呼ばれる．つまり，

$$X_{\mathrm{key}} = \{f_{\mathrm{key}}(p) \mid p \in P\}$$

に対し，$f_{\mathrm{key}}^{-1} : X_{\mathrm{key}} \to P$ という逆のマッピングが存在し，

$$x_{\mathrm{key}} = f_{\mathrm{key}}(p) \Leftrightarrow p = f_{\mathrm{key}}^{-1}(x_{\mathrm{key}})$$

というように属性値と人が 1 対 1 に対応するような属性 key がキー属性である．たとえば「エージェント ID」はキー属性であり，その値は，エージェント間通信における送信者，受信者の特定や，ひとデータベースの内容のファイルへの書き出しおよびファイルからの読み込みにおける識別子などに用いられる．API としては，属性値 $x_{\mathrm{key}} \in X_{\mathrm{key}}$ を持つ人 $f_{\mathrm{key}}^{-1}(x_{\mathrm{key}}) \in P$ を検索する機能が提供される．

Shine では，人から属性値へのマッピング $f_a : P \to X_a$ も，キー属性値から人への逆マッピング $f_{\mathrm{key}}^{-1} : X_{\mathrm{key}} \to P$ も，ハッシュを用いて高速に検索できるようにする．これにより，たとえば「名前」というキー属性の値が吉田である人の「興味」を引く，つまり

$$f_{\text{興味}}(f_{\text{名前}}^{-1}(\text{吉田}))$$

を求めるといったことが簡単かつ高速にできる API が提供される．

ひとデータベースはイベント発生機能を持つ．ある人の属性値が追加，変更，または削除されたときには，イベントが発生する．プラン実行モジュールなどの他のモジュールがこれらイベントの発生を監視し，それらに呼応して動作する．このイベント通知機構は，複数のモジュール間での協調や，さらには複数のアプリケーション間の連携を可能にする．たとえば，あるエージェントからそのユーザの興味の変化の通知を受け，それに相当するキーワードベクトルが変更されると，その変更を知らせるイベントが発生し，それを検出したアプリケーションシステムが表示画面を更新するといった具合である．

ソーシャルウェアはコミュニティを対象とするので，Shine フレームワークは，個々のエージェントだけでなく複数のエージェントが形成するコミュニティも明示的に扱えるような仕組みを必要とする．コミュニティを扱うために，Shine のひとデータベースは各コミュニティに対し図 3 に示すような「ひとセット」を定義し，個人情報の集合を対象にした操作を容易にする．つまり，あるコミュニティのメンバの集合 $S \subseteq P$ を，そのコミュニティのメンバであるかどうかを判定する関数 $m : X_a \times X_b \times \cdots \to \{\mathrm{true}, \mathrm{false}\}$ を用いて，

$$S = \{p \mid m(f_a(p), f_b(p), \ldots) = \mathrm{true}\}$$

のように取り出せる仕組みを用意する．ひとセットは，メンバの判定に影響を与える属性 $a, b, \cdots$ の属性値が更新されたことを知らせるイベントを受けてメンバを更新する．この機構はコミュニティの構成の動的な変化への追随を可能にする．

ソーシャルウェアのアプリケーションシステムの多くは，メッセージをコミュニティのメンバに同報する機能を必要とする．ユーザ集合が画一的に決まっているクライアントサーバ型のアーキテクチャと異なり，コミュニケーション範囲を分散管理する peer-to-peer 型のアーキテクチャでは，同報の範囲の決定が自明ではない．Shine エージェントは，同報における宛先のリストにひとデータベースのひとセットを用いることにより，同報の範囲の決定を柔軟に行える．

3.2 通信モジュール

通信モジュールは他のエージェントの通信モジュールと peer-to-peer で接続され，互いにメッセージを交換する．

ソーシャルウェアの様々なアプリケーションシステムでは，それぞれが前提としている通信環境も多種多様である．たとえば常時接続かダイアルアップか，途中にファイアウォールはないか，その他様々なことが考慮されるため，下位層通信プロトコルとして用いることのできるものが HTTP であったり SMTP であったりといろいろであるし，また下位層におけるネットワークのトポロジも必ずしも peer-to-peer ではなくルータを介した集中型かもしれない．

このような通信環境の違いを吸収し，通信モジュール以外のモジュールやアプリケーションシステムに影響を与えないようにするために，Shine の通信モジュールは**図 4** に示すように下位層の通信プロトコルに依存する部分としない部分に分かれている．下位層通信プロトコルに依存する部分をチャネルと呼ぶ．チャネルは，送受するメッセージの，Shine 内での形式と下位層通信プロトコルに合った形式との相互変換を行い，また接続の確立や切断を管理する．

チャネル間の通信は，下位層のネットワークのトポロジによっては，必ずしも peer-to-peer ではなくルータなどを介した集中型となる場合もありうる．しかしこれはチャネル層での物理的なネットワークトポロジが集中型であるということであり，Shine エージェント間通信のレベルの論理的なトポロジはあくまで peer-to-peer の構成をしている．つまり，他エージェントのアドレスの管理などは各エージェントに任されており，また，ネットワークに接続されている不特定の全エージェントにメッセージをブロードキャストするといった下位層のネットワーク構成に依存する通信はできない．

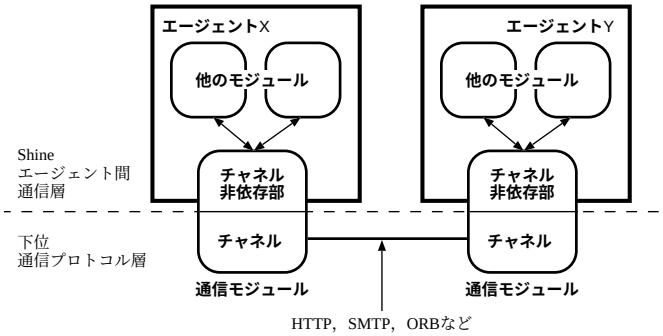

図 4 チャネル
Fig. 4 Channel.

他のエージェントのアドレスは，ひとデータベースに保持されている．プラン実行モジュールなどの他モジュールは，送信したいメッセージを作成し，ひとデータベースに格納されている人 $p \in P$ をその宛先としてメッセージに付加する．通信モジュールのチャネルは p からその下位層通信プロトコルにおけるアドレスをひとデータベースを使って $f_{\text{アドレス}}(p)$ のようにして調べ，得られたアドレスに送信する．そのとき，自身のアドレス $f_{\text{アドレス}}$(自分) を送信者アドレスとしてメッセージに付加する．メッセージを受信したエージェントのチャネルは，送信者アドレスからそのアドレスに対応するエージェントのひとデータベースにおけるエントリを $f^{-1}_{\text{アドレス}}$(送信者アドレス) のようにして引き，それを付加したメッセージをプラン実行モジュールに渡す．

アドレスの形式は下位層通信プロトコルによって異なるので，チャネルがアドレスを格納するのに使う属性もそれにともなって変わる．たとえば下位層通信プロトコルとして HTTP を用いるチャネルは URL アドレスという属性を使い，SMTP を用いるチャネルはメールアドレスという属性を使う．

チャネルを用いたメッセージの送受の具体的な手順を説明する．送信については，各チャネルは以下のような手続き *send*(*Message*) を実装しなければならない．ここで *Message* はメッセージを表すオブジェクトの型であり，メッセージの宛先を取り出す *getTo*() や中身を取り出す *getContent*() などの手続きを持つ．手続き *getContent*() によって取り出されたメッセージの中身は，文字列に変換でき，また文字列から復元できるものとする．これにより SMTP のように文字列しか送受できないような通信プロトコルにも対応できる．

procedure *send*(*Message* m) **{**
　/* メッセージの宛先を取り出す．
　　Person はひとデータベース内の表における人の列への参照の型． */
　Person $p = m.getTo()$;
　/* このチャネルで用いる形式のアドレスを
　　ひとデータベースで引く． */
　Address $a = db.get(p, \texttt{address})$;
　/* メッセージの中身を文字列などの
　　下位層プロトコルで送信できる形式に変換． */
　Data $d = convert(m.getContent())$;
　/* 下位層プロトコルを用いて実際に送信． */
　$send\_internal(a, d)$;
}

受信については，個々のメッセージは以下のような手順で受信される．この手順は，アドレス b からデータ d を受信したときにそれらを引数にして呼び出される．下位層通信プロトコルによって，メッセージの受信を検知するとこのような手続きが自動的に呼ばれるようなチャネルや，定期的にポーリングを行って受信メッセージがあればこのような手続きを呼び出すようなチャネルがありうる．

procedure *receive*(*Address* b, *Data* d) **{**
　/* メッセージを復元するために
　　新たなメッセージオブジェクトを宣言． */
　Message m;
　/* 送信者アドレスに対応するエージェントの
　　ひとデータベースにおけるエントリを
　　逆引き． */
　Person $q = db.identify(b, \texttt{address})$;
　/* 送信者を設定． */
　$m.setFrom(q)$;
　/* メッセージの中身を復元し設定． */
　$m.setContent(restore(d))$;
　/* 復元したメッセージを
　　プラン実行モジュールに渡す． */

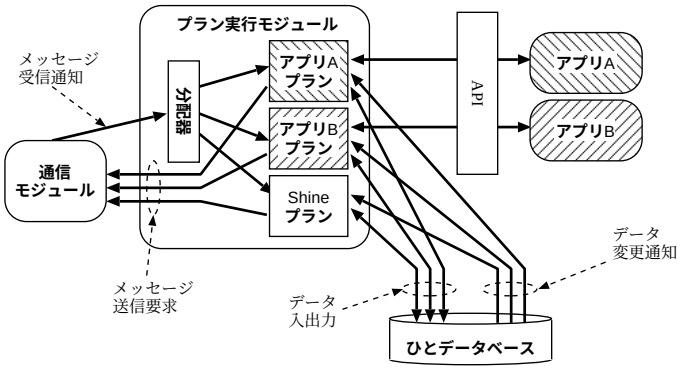

図 5 プラン実行モジュール
Fig. 5 Plan execution module.

$planExecution.put(m)$;
}

3.3 プラン実行モジュール

エージェントの行動の規則を記述したものをプランと呼ぶ．Shine では，複数のプランが，プラン実行モジュール内において並行に動作する．これらのプランの中には，各アプリケーションがそれぞれのサービスを実行するために独自に用意するプランと，Shine がフレームワークとして提供する，基盤的なタスクを行うプランがある．**図 5** にプラン実行モジュールの構成を示す．

各プランは，いろいろなイベントをきっかけとして，それに呼応するように行動する．動作のきっかけとなるイベントには，

- 他のエージェントからのメッセージの受信，
- ひとデータベースに保持されている値の更新，
- ユーザからの入力

の 3 種類ある．また，行動の内容は，Shine が実装されるプログラミング言語で直接記述される．この記述には Shine が提供する API が利用される．提供される API には，メッセージ送信手続きや，3.1 節で述べたひとデータベースのためのものが含まれる．

プラン実行モジュールは，通信モジュールから渡される受信メッセージを，適切なプランに分配する機能を持つ．各メッセージが，その内容の記述のほかにメッセージタイプを指定するフィールドを持っており，メッセージの分配はそのメッセージタイプに従って行われる．つまり，Shine エージェントは，KQML[1] やその他のエージェント通信言語に則って通信するエージェントシステムと同じく，typed-message エージェント[17] である．このメッセージ分配機構は，各プランが特定のメッセージタイプのメッセージが受信されたときに起動されるメッセージハンドラをプラン実行モジュールに登録することによってなされる．登録のための API は，具体的には

$addHandler(MessageType, MessageHandler)$

という手続きの呼び出しである．ここで，$MessageType$ はメッセージタイプを表すオブジェクトの型であり，$MessageHandler$ は $handle(Message)$ という手続きを実装するメッセージハンドラオブジェクトの型である．エージェントの振舞いは手続き $handle$ の中身として記述される．

ひとデータベースに格納されている値の更新は，その更新を伝えるイベントを各プランが受け取ることによって，エージェントの動作に反映される．この仕組みを用いて複数のプランが連携することも可能である．すなわち，あるプランがある人の属性値を変えると，別のプランは，それが変わったことによって発生するイベントを受け取ることによって，属性値の変化に対応する．このイベントは，具体的には $PersonDBEvent$ という型のオブジェクトである．この型のオブジェクトは，更新された後の属性値，更新される前の属性値，その属性値が格納されている属性を表すオブジェクト，およびその属性値を持つ人を表すオブジェクトを取得するための手続きを実装する．

ユーザからの入力の受付に関しては，各アプリケーションシステムがそれぞれのユーザインタフェースを用意し，それをアプリケーション独自のプランと結び付けることで成される．

4. アプリケーションシステムへの Shine の適用例

この章では，ソーシャルウェアのアプリケーションシステムへの Shine フレームワークの適用について，具体例を用いて説明する．題材として我々が開発している 2 つのシステム Community Organizer と「ひと

のあかり」を取り上げる．我々はこれら 2 つのシステムおよび Shine をプログラミング言語 Java を用いて実装している．これら 2 つのシステムの実装において Shine の枠組みが上手くあてはまりフレームワークとして有効に機能することを示す．

4.1 Community Organizer

4.1.1 Community Organizer の概要

Community Organizer は，新たなネットワークコミュニティの形成を支援するためのコミュニケーションシステムである[5),6)]．このシステムは，興味を共有する人々を，興味のコミュニティ（communities of interest）の潜在的なメンバと見なし，それらの人々が出会い会話することができる場を提供する．

図 6 は Community Organizer が各ユーザに提供する画面の例である．図中の表示パネルは，コミュニティの潜在的なメンバが出会い会話するための場として提供されるユーザインタフェースである．このパネルには，ベクトル空間モデルにおける各キーワードを軸とする多次元空間を，ある手法により 2 次元平面に射影したものが表示される．ユーザはこの表示パネル上にアイコンを置くことができ，それは他のユーザの表示パネル上にも現れる．各アイコンにはキーワードベクトルが付与され，その値に従って計算されるパネル上の点にアイコンが表示される．このとき，パネル上でのアイコン間の相対的な距離は，余弦尺度（cosine measure）によって計算されたそれぞれのキーワードベクトル間の類似度をもとに，似たキーワードベクトルを持つアイコンどうしは近くなるように決定される．

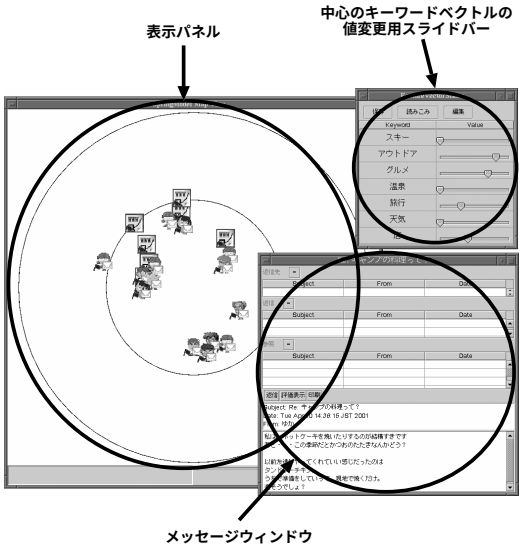

図 6 Community Organizer の画面表示例
Fig. 6 Sample screenshot of Community Organizer.

表示パネルにおける中心点は，それに対応するキーワードベクトルを持っており，それは表示パネルの横に設置された一連のスライドバーの位置で示される．ユーザはこのスライドバーを操作することによって中心のキーワードベクトルを変えることができる．中心のキーワードベクトルが変わるとそれにつれて表示パネル全体のアイコンの配置も移動する．また，ユーザが新たに置くアイコンに付与されるキーワードベクトルには，中心のキーワードベクトルの値が設定される．

ユーザは，潜在的なコミュニティを見つけるために，中心のキーワードベクトルをスライドバーで変化させてベクトル空間内を「見てまわる」．そして，たとえば “旅行” と “食べ物” の値が大きくなるようスライドバーを動かし，そのようなキーワードベクトルを持つアイコンを新たに置くと，「私は旅行と食べ物に興味がある」ということを他人に表明することになる．つまり，Community Organizer の表示パネルは，興味の表明を表す「興味アイコン」が詰まったベクトル空間を可視化するメディアである．

Community Organizer は，このベクトル空間可視化メディアに，テキストベースのコミュニケーションメディアを統合したものとなっている．この 2 つのメディアの統合は，ユーザがメッセージを書き，それを興味アイコンに添付できるようなインタフェースを提供することによって実現されている．他のユーザはアイコンをクリックすることでそのアイコンに添付されたメッセージを読むことができ，それに対する返信を添付した新しいアイコンをそのアイコンの近くに置くことによって返事をすることができる．このように，Community Organizer はベクトル空間可視化メディアとテキストコミュニケーションメディアをシームレスに統合したマルチメディアコミュニケーションシステムである．

4.1.2 Shine 上の Community Organizer

我々は当初，Community Organizer を Shine フレームワークには依存しない独立したシステムとして開発した．この節では，Community Organizer を Shine フレームワークに合わせて再構築する方法について述べる．この再構築は，Community Organizer に以下のような利点をもたらす．

- Shine の統一された枠組みに則り共通の API を利用して実装することにより，プログラム部品をライブラリ化できる．Community Organizer 向けに作ったユーザインタフェースなどのプログラム部品を他のアプリケーションシステムが再利用したり，逆に他のシステム用の部品を Community

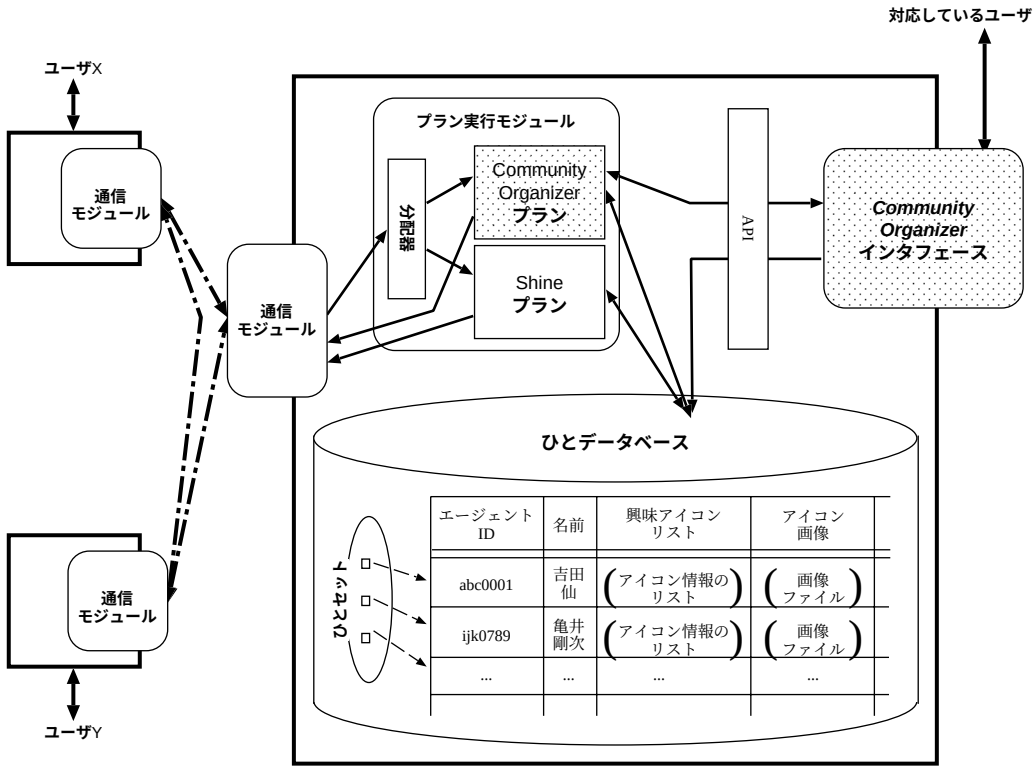

図 7 Shine に基づく Community Organizer
Fig. 7 Shine-based Community Organizer.

Organizer に援用したりできるようになる．

- 人々に関するデータを他のソーシャルウェアアプリケーションと共有でき，その結果それらのアプリケーションと連携することが可能になる．
- 元々の Community Organizer はクライアントサーバ型のアーキテクチャで設計されていた．Shine への移植にともない，システムのアーキテクチャは peer-to-peer の情報交換が行われるエージェント指向アーキテクチャへと変更される．クライアントサーバ型のアーキテクチャでは興味の近さだけで潜在的コミュニティを決めるので，全体のユーザ数が多い場合には，潜在的コミュニティの人数を適度なものに抑えようとすると興味範囲が細かくなりすぎる．その点 peer-to-peer のアーキテクチャでは，small-world における人同士のつながりを使って他人の情報を紹介しあうことにより，各ユーザに見合う範囲の潜在的コミュニティを提供することができる．

Shine に基づく Community Organizer のアーキテクチャを**図 7** に示す．

Community Organizer における興味アイコンは，それを置いたユーザ，それが持つキーワードベクトル，および添付されたメッセージという情報を持っている．これを Shine のひとデータベースに格納するときには，各属性値がリスト構造になっている「興味アイコンリスト」という属性を用意する．そしてキーワードベクトルと添付されたメッセージをひとまとめにした形態のデータを，「興味アイコンリスト」属性の列の，そのアイコンを置いたユーザに相当する行にあるリストの中に格納する．

興味アイコンを画面に表示するときには，各ユーザ p の興味アイコンのリスト $f_{\text{興味アイコンリスト}}(p)$ の要素を，そのユーザを表すアイコンの画像ファイル $f_{\text{アイコン画像}}(p)$ を用いて表示する．

Community Organizer の表示パネルには，データベースに格納されているすべての興味アイコンのうち，中心のキーワードベクトルに近いキーワードベクトルを持つものだけが表示される．この選択に，ひとデータベースのひとセットを用いることができる．すなわち，メンバを判定する関数が，中心のキーワードベクトルと各興味アイコンのキーワードベクトルの間の余弦尺度の値のうち上位に来るものだけを選ぶものであるようなひとセットを用意し，そのメンバをパネルに表示する．

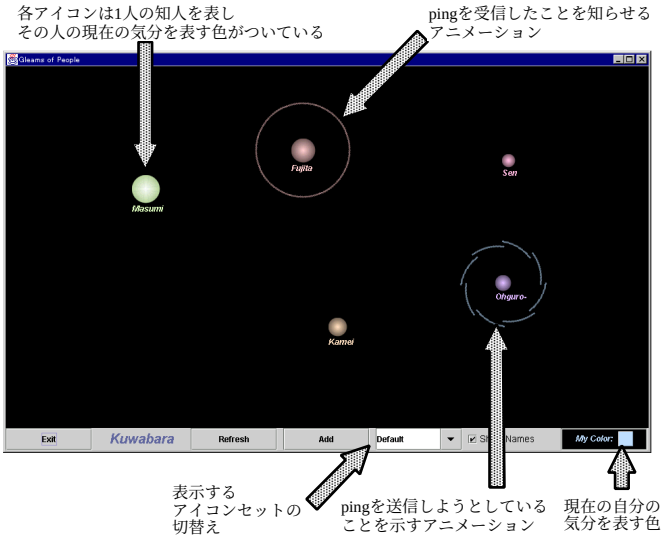

図 8 「ひとのあかり」の画面例
Fig. 8 Sample screenshot of Gleams of People.

2.2 節で述べたように，Shine はエージェント指向アーキテクチャを採用している．Shine フレームワーク適用後の Community Organizer においては，サーバは存在せず，各クライアントプログラムはサーバが持っていた機能の一部が組み込まれエージェント化される．それらエージェントは peer-to-peer のネットワークを構成し，Community Organizer の動作に必要な個人情報を他のエージェントと仲介しあう．これによって，密なつながりと連鎖のよさをあわせ持つ small-world の利点を生かし，各エージェントがそれぞれのユーザに対し適切な範囲で自然なコミュニケーションが行われるような出会いと会話の場を提供できるようになる．

プラン実行モジュール内の Community Organizer プランは，ユーザが新たに作成した興味アイコンを他のエージェントに送信し，また他のエージェントから送られてきた興味アイコンをひとデータベースに格納するなどの動作をする．

4.2 ひとのあかり

「ひとのあかり」は，コミュニティのメンバが互いに存在や気分を伝えあうことにより，コミュニティ感覚を維持し発達させるための，従来になくシンプルなメディアによるコミュニケーションシステムである[13),14)]．

「ひとのあかり」の機能を直感的に説明すれば，人間を対象とする `ping` コマンドであるといえる．通常の `ping` コマンドは，ある計算機に向けて別の計算機からパケットを送信し返信パケットを受け取るもので，その計算機が動作しているかどうかを確かめるのに使われる．これを計算機どうしではなく人同士で行うことにより互いの存在を伝えあうのが「ひとのあかり」である．また，単なる存在伝達に加えて，「ひとのあかり」で交わされるメッセージは送信者の現在の気分を色で表現した情報も伝える．**図 8** は「ひとのあかり」の画面表示例である．

このシステムの動作は以下のようになる．まずユーザが自身の現在の気分を表す色を選択する．また，対象とする人を数名指定する．指定された人は画面内に球状のアイコンとして表示される．あるアイコンをダブルクリックすると，アイコンに向かって同心円が縮んでいくアニメーションが行われ，次いでそのアイコンに対応する人のパーソナルエージェントに ping が送られる．パケットには，送信者の現在の気分を表す色の情報が付加される．逆に，ある人から ping が送られてくると，今度はその人を表すアイコンから同心円が拡がっていくようなアニメーションが行われ，またそのアイコンの色が ping が伝えてきたものに変わる．同時に，送られてきた ping への返信の ping が，今の自分の気分として設定されている色とともに自動的に送られる．

「ひとのあかり」も Community Organizer と同様に Shine とは独立に作られた．元のシステムは，個々のユーザに対応するパーソナルエージェント群と，リピータと呼ばれるメッセージ交換装置からなっており，メッセージはすべてリピータを介して送られる集中型のネットワークトポロジとなっている．リピータは，あるエージェントがメッセージを送ろうとした際，宛

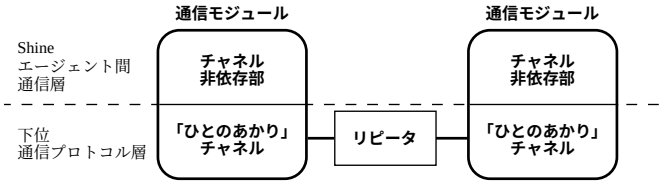

図 9 リピータのチャネル層への隠蔽
Fig. 9 Hiding a repeater under the channel layer.

先のエージェントがメッセージを受け取れる状況でなかった場合に，送られるべきメッセージを一時的に溜めておき，後にメッセージを受け取れるようになった時点で配送する機能を持つ．

このような「ひとのあかり」のメッセージ交換機構を Shine フレームワークにあてはめた場合，リピータは，**図 9** に示されるように，チャネルに隠蔽される．エージェント間のネットワークの論理的なトポロジは，下位通信プロトコル層ではリピータを中心とする集中型，エージェント間通信層では peer-to-peer 型となる．また，チャネルを，リピータを介さない peer-to-peer 型のものに置き換えることも容易にできる．

ひとデータベースには，名前，色，メールボックスといった属性を持たせる．あるユーザが別のユーザへ ping を送信したときの，それぞれのひとデータベースにおける値の更新は次のように行われる．送信者エージェントのひとデータベースにおける送信者のエントリを p，宛先のエントリを q，宛先エージェントのひとデータベースにおける送信者のエントリを $p\prime$，宛先のエントリを $q\prime$ とする．送信者が，画面上の宛先のアイコンをクリックすると，ユーザインタフェースはそのイベントを受けて送信者の気分を表す色 $c = f_{色}(p)$ を宛先 q に送るよう通信モジュールに依頼する．通信モジュールは，それを宛先エージェントの通信モジュールに届ける．これを受け取った宛先エージェントの通信モジュールは，送信者のアドレスから，宛先エージェント内のひとデータベースにおける送信者のエントリ $p\prime$ を割り出し，c とともにプラン実行モジュール内の「ひとのあかり」プランに伝える．プランは，$p\prime$ の色 $f\prime_{色}(p\prime)$ を c に更新する．この更新はユーザインタフェースに伝わりアイコンの色が変えられる．同時に，送られてきた ping メッセージに対する返事が，$q\prime$ の気分を表す色 $f_{色}(q\prime)$ とともに $p\prime$ に送られる．

「ひとのあかり」では，表示するアイコンを何個かずつセットにして切り替えることができる．アイコンのセットは，ひとデータベースのひとセットをそのまま対応させることができる．

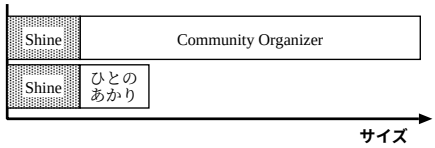

図 10 Java クラスファイルの合計サイズ
Fig. 10 Total size of the Java class files.

4.3 2 つのアプリケーションシステムの Shine 上での実現に関する考察

我々の実装では，Shine フレームワークの Java クラスファイルの合計，Shine 部分とその他に既存のクラスライブラリを利用している部分を除いた Community Organizer 自体の Java クラスファイルの合計，および Shine 部分を除いた「ひとのあかり」自体の Java クラスファイルの合計を比較すると**図 10** のようになる．Community Organizer では全体の約 19%が，「ひとのあかり」では約 53%が Shine フレームワークによって提供されている．

ところで，ソーシャルウェアのためのフレームワークを提供することの目的の 1 つに，アプリケーションシステム間で連携を可能にすることがある．Community Organizer と「ひとのあかり」でも，1 つの Shine エージェント上で両者を並行に動作させることができる．このとき，ひとデータベース，通信モジュール，プラン実行モジュールは 1 つのものが共有される．ひとデータベース内のデータに関しては，各アプリケーションシステムが独自に属性を持つほか，共通の属性を持つことによって両システムを連携させることができる．たとえば「ひとのあかり」の ping の頻度を「親しさ」のパラメータとして Community Organizer と共有し，Community Organizer はそれを表示パネルにおけるアイコンの配置に利用するといったことが考えられる．通信モジュールについてはチャネルも含めて共有される．交換されるメッセージの各アプリケーションシステムへの振り分けは，両者が別々のメッセージタイプをメッセージに付加することによりなされる．

5. 関連研究

この章では，Shine に関連するシステムをいくつか取り上げ Shine と比較することにより，Shine に考察を加える．

ADIPS フレームワーク[7] は「やわらかい」分散システムのためのフレームワークである．ビデオ会議システム，非同期メッセージングシステム，およびネットワーク上に仮想的な仕事場を提供するシステムなどのコミュニケーションシステムが ADIPS フレームワークの上に構築されている．ADIPS フレームワークの主な特徴は，ADIPS 上のシステムにおける各ノードが，QoS やユーザ要求などの環境要因の変化に適応するために，柔軟かつ動的に内部構成を変化させられることである．この適応性を獲得するために，ADIPS システムのノード内に存在する各種モジュールがエージェント化されており，それらのモジュールエージェント間の通信が typed-message を用いて行われる．一方 Shine では，各ノードを，それらが個人化されていて他のパーソナルエージェントと通信しながらユーザの社会的状況の変化に適応するという理由でエージェントと呼び，typed-message を用いた通信を行うようにしている．

KQML[1] はエージェント間で知識を共有するのに適した typed-message の形式と一連のメッセージ処理プロトコルを規定する仕様である．この仕様を実装したマルチエージェントフレームワークはいくつか存在する．しかし，KQML はエージェント間通信について仕様を決めているだけで，コミュニケーションシステムを構築するための，データベースやプランについての実装の方針が与えられるわけではない．Shine はエージェント間通信の機能だけでなくエージェントの内部構造まで立ち入ってソーシャルウェアに適した枠組みを提供する．ただし Shine でもエージェント間通信のプロトコルを KQML などの標準化された仕様に合わせることは検討に値する．

MINDS[11] は協調文献検索のためのマルチエージェントシステムである．このシステムでは，各パーソナルエージェントはそのユーザの計算機上に蓄積されている文献集合を知っており，また，だれがどんなトピックに関する文献を持っていそうかというメタ知識も持っている．ユーザがその人のエージェントにあるトピックに関する文献の検索を頼むと，エージェントはこのメタ知識を上手く使って他のエージェントに質問式を送る．MINDS は協調パーソナルエージェントシステムの先駆的存在であるが，その適応範囲は情報検索に限られる．ただし，メタ知識を上手く使って他のエージェントに質問式を送る方法については，それを Shine のプラン実行モジュール内で動作するプランとして実装し，他のソーシャルウェアアプリケーションシステムでも利用できるようになると有益であろう．

Plangent[15] などのエージェントの自律性に主眼を置いたエージェントシステムでは，アプリケーションシステム開発者がエージェントの目標や行動規則を宣言的に記述すると，エージェント自身がそれに従って自らの行動をプランニングする．この機能は，たとえば旅行計画の作成のように，あるタスクを達成するという形でエージェントが行動するようなアプリケーションの場合には有効である．一方，Shine のエージェントは，ユーザ同士の社会的インタラクションを支援するという目的で存在するので，タスク達成型のアプリケーションと異なり，エージェントの目標や行動規則を宣言的に記述するというアプリケーションシステム開発のスタイルは馴染まない．このため Shine では，エージェントの行動はアプリケーションシステム開発者が具体的な手続きとして記述する．

Yenta[2] は，パーソナルエージェント間の peer-to-peer 通信によるマッチメイキング機構である．マッチメイキングは各ユーザの興味に基づいて行われる．Yenta は，ユーザの興味というプライベートな情報が必要以上に他のユーザに伝わらないようにするためのプライバシ管理機構を持つ．Shine の上に構築されるソーシャルウェアのアプリケーションシステムも Yenta と同じくユーザの個人情報を扱うものがほとんどであり，そのことを考えると Shine にもプライバシ管理機構があることが望ましい．つまり，どの情報はどの範囲まで公開してよいかということを指定でき，情報流通はその指定に従って管理されるような機能が求められる．

6. おわりに

本論文では，ソーシャルウェアのエージェント指向フレームワークについて論じ，我々が開発しているフレームワーク Shine の具体的な構成について述べた．また，応用例として，Community Organizer と「ひとのあかり」への Shine の適用について述べた．これら 2 つのシステムの実装において Shine の枠組みがうまくあてはまり，フレームワークとして有効に機能することを示した．

Shine には多くの課題が残されている．プラン実行モジュールにおけるプランの動作については，いくつかのプランを実際に作成することを通じ，より詳細

な機能要件を明らかにしていく必要がある．特に，コミュニティの構成を適切に保つためには，ひとデータベースにどういう情報が必要で，また他のエージェントとどのように情報交換すればよいのかということを考えプランとして実現することが重要である．また，プライバシ管理機構についても取り組まなければならない．さらに，Community Organizer と「ひとのあかり」の間のデータ共有による連携を実装し，アプリケーションシステム間連携を実証したい．

参 考 文 献

1) Finin, T., Fritzson, R., McKay, D. and McEntire, R.: KQML — A Language and Protocol for Knowledge and Information Exchange, *Knowledge Building and Knowledge Sharing*, Fuchi, K. and Yokoi, T. (Eds.), pp.249–259, Ohmsha and IOS Press (1994).
2) Foner, L.N.: Community Formation via a Distributed, Privacy-Protecting Matchmaking System, *Community Computing and Support Systems: Social Interaction in Networked Communities*, Ishida, T. (Ed.), Lecture Notes in Computer Science, Vol.1519, pp.359–376, Springer-Verlag (1998).
3) Hattori, F., Ohguro, T., Yokoo, M., Matsubara, S. and Yoshida, S.: Socialware: Multiagent Systems for Supporting Network Communities, *Comm. ACM*, Vol.42, No.3, pp.55–61 (1999).
4) Ishida, T. (Ed.): *Community Computing — Collaboration over Global Information Networks*, John Wiley & Sons (1998).
5) Kamei, K., Jettmar, E., Fujita, K., Yoshida, S. and Kuwabara, K.: Community Organizer: Supporting the Formation of Network Communities through Spatial Representation, *Proc. 1st SAINT 2001*, pp.207–214, IEEE Computer Society (2001).
6) 亀井剛次, 吉田 仙, 大黒 毅, 服部文夫：Community Organizer：ネットワークコミュニティの形成支援，ヒューマンインタフェースシンポジウム '99 論文集，pp.333–336 (1999).
7) Kinoshita, T. and Suganuma, K.: ADIPS Framework for Flexible Distributed Systems, *Proc. 1st PRIMA 1998*, LNAI, Vol.1599, pp.18–32, Springer-Verlag (1998).
8) 岸 知二，野田夏子：ソフトウェアアーキテクチャ，コンピュータソフトウェア，Vol.18, No.2, pp.68–77 (2001).
9) Mase, K., Sumi, Y. and Nishimoto, K.: Informal Conversation Environment for Collaborative Concept Formation, In Ishida [4], chapter 6, pp.165–205.
10) Matsubara, S., Ohguro, T. and Hattori, F.: CommunityBoard 2: Mediating between Speakers and an Audience in Computer Network Discussions, *Proc. 3rd Agents 1999*, pp.370–371, ACM Press (1999).
11) Mukhopadhyay, U., Stephens, L., Huhns, M. and Bonnell, R.: An Intelligent System for Document Retrieval in Distributed Office Environments, *J. Am. Soc. Inf. Sci.*, Vol.37, pp.123–135 (1987).
12) Nishibe, Y., Morihara, I., Hattori, F., Nishimura, T., Yamaki, H., Ishida, T., Maeda, H. and Nishida, T.: Mobile Digital Assistants for International Conferences, In Ishida [4], chapter 8, pp.245–284.
13) Ohguro, T.: Toward Agents which are Suggestive of "Awareness of Connectedness", *Trans. IEICE*, Vol.E84–D, No.8, pp.957–967 (2001).
14) Ohguro, T., Yoshida, S. and Kuwabara, K.: Gleams of People: Monitoring the presence of people with multi-agent architecture, *Approaches to Intelligent Agents — Proc. 2nd PRIMA 1999*, LNAI, Vol.1733, pp.170–182, Springer-Verlag (1999).
15) Ohsuga, A., Nagai, Y., Irie, Y., Hattori, M. and Honiden, S.: PLANGENT: An Approach to Making Mobile Agents Intelligent, *IEEE Internet Computing*, Vol.1, No.4, pp.50–57 (1997).
16) Oram, A. (Ed.): *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, O'Reilly (2001).
17) Petrie, C.J.: Agent-Based Engineering, the Web, and Intelligence, *IEEE Expert*, Vol.11, No.6, pp.24–29 (1996).
18) Shardanand, U. and Maes, P.: Social Information Filtering: Algorithms for Automating "Word of Mouth", *Proc. CHI 1995: Mosaic of Creativity*, pp.210–217 (1995).
19) 梅木秀雄：ネットワークコミュニティ形成支援技術，人工知能学会誌，Vol.14, No.6, pp.943–950 (1999).
20) Watts, D.J. and Strogatz, S.H.: Collective Dynamics of 'Small-World' Networks, *Nature*, Vol.393, pp.440–442 (1998).
21) Yoshida, S., Kamei, K., Ohguro, T., Kuwabara, K. and Funakoshi, K.: Building a Network Community Support System on the Multi-Agent Platform Shine, *Design and Applications of Intelligent Agents — Proc. 3rd PRIMA 2000*, LNAI, Vol.1881, Springer-Verlag, pp.88–100 (2000).
22) 吉田 仙, 大黒 毅, 亀井剛次, 船越 要, 桑原

和宏：サイバーコミュニティのアプリケーションのためのプラットフォーム Shine の提案，1999年度人工知能学会全国大会（第 13 回）論文集，pp.461–462 (1999).
23) Yoshida, S., Ohguro, T., Kamei, K., Funakoshi, K. and Kuwabara, K.: A Platform for Making Network Community Support Systems in a Cooperative Distributed Architecture, *Proc. 7th ICPADS 2000: Workshops*, pp.441–446, IEEE Computer Society (2000).

(平成 13 年 5 月 31 日受付)
(平成 13 年 11 月 14 日採録)

吉田　仙（正会員）

平成 7 年東北大学大学院情報科学研究科情報基礎科学専攻博士前期課程修了．同年日本電信電話（株）入社．分散協調システムの研究に従事．人工知能学会，日本ソフトウェア科学会各会員．

亀井　剛次

平成 9 年京都大学大学院工学研究科電子通信工学専攻修士課程修了．同年日本電信電話（株）入社．集団におけるコミュニケーションの支援に向けた協調システムの研究に従事．電子情報通信学会，人工知能学会各会員．

大黒　毅（正会員）

平成 2 年東北大学大学院工学研究科情報工学専攻博士前期課程修了．同年日本電信電話（株）入社．ソーシャルウェア，新たなネットワークコミュニケーションメディアの研究等に従事．現在東日本電信電話（株）．電子情報通信学会会員．

桑原　和宏（正会員）

昭和 59 年東京大学大学院電子工学専門課程修士課程修了．同年日本電信電話公社入社．知識ベースシステム，分散協調問題解決，マルチエージェントシステム等の研究に従事．昭和 63 年から平成元年にかけて米国マサチューセッツ工科大学アムハースト校客員研究員．博士（工学）．電子情報通信学会，人工知能学会各会員．