VNA構築用ライブラリの設計と実装

4B－2

中澤 仁[1]　大越 匡[1]　望月 祐洋[1]　徳田 英幸[1,2,3]

[1]慶應義塾大学政策・メディア研究科　[2]慶應義塾大学SFC研究所　[3]慶應義塾大学環境情報学部

VNAアーキテクチャでは，情報家電ネットワークに接続された情報家電機器の機能，あるいはその部分機能を組み合わせた，仮想情報家電機器（VNA，Virtual Network Appliance）の構築が可能である．VNAは，情報家電機器の機能をネットワークに提供するSerdgetと呼ばれるソフトウエアコンポーネントの組み合わせとして定義される，分散オブジェクト指向アプリケーションである．本稿では，VNAを構築するためのソフトウエアライブラリを提案し，ポゼッションモデルを用いて構築したプロトタイプシステムの実装について述べる．

1　はじめに

VNAアーキテクチャ[1]では，情報家電ネットワークに接続された情報家電機器の機能，あるいはその部分機能を組み合わせた，仮想情報家電機器（VNA，Virtual Network Appliance．以下VNAと呼ぶ）の構築が可能である．VNAは，以下に示す機器やソフトウエアコンポーネントを組み合わせて，アプリケーションソフトウエアとして提供される．

- 情報家電ネットワークに接続された情報家電機器
- 情報家電ネットワークに接続された各種センサ
- VNAアーキテクチャに基づいて構築されたフィルタプログラムをはじめとする通常のソフトウエアコンポーネント

本アーキテクチャでは，上記を総称してVNAコンポーネントと呼び，これらの組み合わせによって構築された，VNAとしての機能を提供するアプリケーションソフトウエアをVNAアプリケーション†と呼ぶ．VNAコンポーネントの持つ機能は，Serdgetと呼ぶソフトウエアモジュールによって情報家電ネットワークに公開され，VNAアプリケーションはSerdgetのクライアント側スタブオブジェクトの組み合わせとして構築される．

一方，情報家電機器が接続されるネットワークは様々であることから，本システムではSerdget間において，VNAコンポーネント間通信支援機構を用いた特定の物理媒体や下位層プロトコルに依存しない通信を提供する．

図1に，本アーキテクチャを用いて実際に構築可能なVNAとして，VVCR(Virtual VCR)を例示した．VVCRは，VCR，スピーカ，およびディスプレイがそれぞれ，データ読み込み機能，音声再生機能，および映像表示機能を提供することにより，VNAコンポーネントとして構築される．VVCRの利用者は，VCRを遠隔から再生し，再生データをネットワークを通して利用者側のスピーカおよびディスプレイを用いて鑑賞できる．さらに，VVCRにフィルタプログラムをVNAコンポーネントとして新たに追加することにより，再生映像の柔軟な加工処理なども可能である．

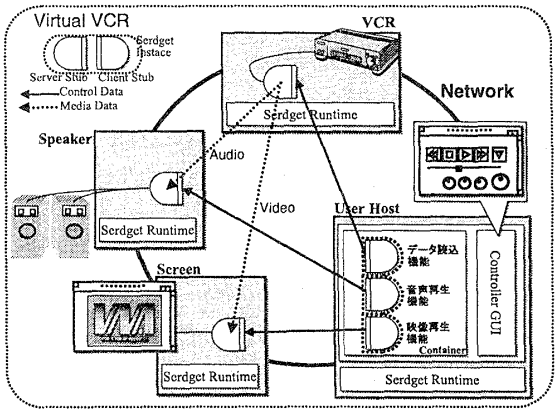

図 1: Virtual VCR(VVCR) VNAの構成

本稿では，VNAアプリケーションを構築するためのソフトウエアライブラリを提案し，ポゼッションモデル[2]を用いて構築したプロトタイプシステムの実装について述べる．第二章ではVNA構築ライブラリを概観し，第三章以降の各章で，同ライブラリを構成する各サブシステムの設計と実装について詳述する．

2　VNAアーキテクチャ

VNAアーキテクチャは，一つ以上の情報家電機器内あるいは計算機内で動作するソフトウエアコンポーネントを，容易に組み合わせ可能な分散オブジェクト指向システムである．本アーキテクチャでは，ネットワークオブジェクト[3]の概念を用いて，上記のソフトウエアコンポーネントを利用者側計算機上でマウス操作によって組み合わせることにより，VNAアプリケーションを構築できる．VNAアプリケーションは，利用者の要求に対する

Design and Implementation of VNA Toolkit Library

[1]Graduate School of Media and Governance, Keio University
5322, Endo, Fujisawa, Kanagawa 252, Japan
E-Mail: <jin@ht.sfc.keio.ac.jp>
[2]Keio Research Institute at SFC
[3]Faculty of Environmental Information, Keio University

本研究は，情報処理振興協会（IPA）次世代アプリケーション事業「次世代アプリケーション共通基盤RT-HDI(Real-Time Human Device Interaction)プラットホームの研究開発」にて行われている．

†本稿では，アプリケーションソフトウエアとして仮想的に構築された情報家電機器を指す用語としてVNAを，またそのアプリケーションソフトウエア自体を指す用語としてVNAアプリケーションを用いる．

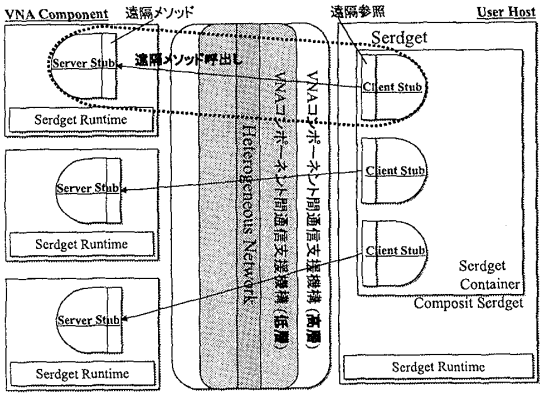

図 2: VNA アーキテクチャ構成

適応性を持ち，さらにそれを構成するコンポーネント間ではネットワーク透過的な通信が可能である．

本章では，本アーキテクチャを構築する上での問題点を整理し，具体例を示すと共に VNA アーキテクチャを構成する各サブシステムを概観する．

2.1 情報家電ネットワークにおける問題点

本アーキテクチャでは，複数の利用者側計算機上で動作する VNA アプリケーションから，情報家電ネットワークに接続された情報家電機器を制御できる．このため本アーキテクチャでは，以下を始めとする問題点を解決する必要がある．

2.1.1 利用者および計算機環境に対する適応性

本アーキテクチャを用いて構築された VNA アプリケーションは，様々な利用者側計算機上で動作し，従って利用者も様々である．本アーキテクチャでは，利用者側計算機の環境情報や，当該 VNA アプリケーションに対する利用者の習熟度に対する適応機構が必要となる．

2.1.2 異機種混在ネットワークへの対応

情報家電機器は，様々な物理媒体や下位層プロトコルを用いて情報家電ネットワークに接続している．これらの情報家電機器を，本アーキテクチャで統一的に扱うためには，VNA アプリケーションに対して物理媒体や下位層プロトコルを隠蔽する必要がある．

2.1.3 処理の競合

複数の利用者側計算機上で動作する VNA アプリケーションから同一の情報家電機器を制御する際には，当該情報家電機器に対する制御が競合する可能性がある．例えば，VCR に対する早送りと巻き戻し要求が，複数の VNA アプリケーションから同時に発生した場合がこれにあたる．本アーキテクチャでは，競合の回避および競合発生時の対処等について考慮する必要がある．

2.2 アーキテクチャの構成概要

図 2は，図 1を例にとって本アーキテクチャの構成を示したものである．本アーキテクチャは，複数の VNA コンポーネントの集合として構築された情報家電ネットワーク上で動作する．各 VNA コンポーネントの持つ機能および部分機能は，Serdget と呼ばれるソフトウエアモジュールを介して，VNA アプリケーションおよび他の VNA コンポーネントから制御できる．

Serdget はクライアント側スタブとサーバ側スタブから構成される．クライアント側スタブは利用者側計算機に移送されるネットワークオブジェクトであり，他のクライアント側スタブと組み合わされて動作する．Serdget のクライアント側スタブはユーザインタフェースを伴っており，各ユーザインタフェースオブジェクトは，対応するサーバ側スタブの遠隔参照オブジェクトを持つ．このとき，クライアント側スタブとサーバ側スタブ間，および複数の異なる Serdget 間での通信は，VNA コンポーネント間通信支援機構を用いた遠隔メソッド呼出しによって行われる．

Serdget のクライアント側スタブは，利用者側計算機で動作する VNA サービスインタフェースを用いて他と組み合わされ，コンポジット Serdget となる．コンポジット Serdget は VNA アプリケーションのソフトウエア的インスタンスであり，それ自身も VNA コンポーネントとして再帰的に他の Serdget と組み合わせられる．

以下の各節に，本アーキテクチャの構成要素について詳述すると共に，本アーキテクチャにおける上述した問題点の解決手法について述べる．

2.3 Serdget

Serdget は，VNA コンポーネントの持つ機能をネットワークに提供し，利用者から VNA アプリケーションを通じて制御可能とするためのソフトウエアモジュールである．Serdget はサーバ側スタブとクライアント側スタブから構成される．サーバ側スタブは各 VNA コンポーネント内あるいは VNA コンポーネントが接続されている計算機上の Serdget 動作環境上で，またクライアント側スタブは，上記の VNA コンポーネントから利用者側計算機に移送されて動作する．以下の各項に，Serdget 動作環境，サーバ側スタブおよびクライアント側スタブの構成を詳述する．

2.3.1 Serdget 動作環境

Serdget 動作環境は，Serdget のサーバ側スタブあるいはクライアント側スタブが動作するための環境を提供するサーバプログラムと，それに付随するライブラリの集合である．

同環境の最下層は，後述する VNA コンポーネント間通信支援機構である．同環境はまた，移送対象となるクライアント側スタブに対して，利用者側計算機の環境情

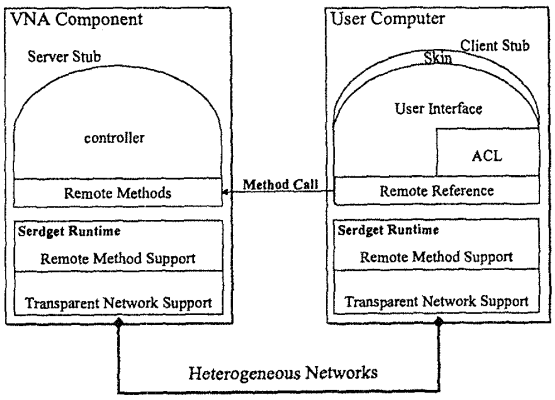

図 3: Serdget の構成

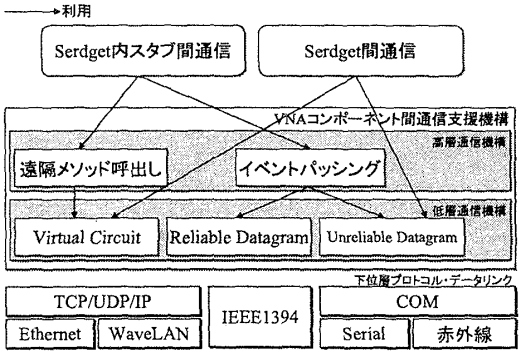

図 4: VNA コンポーネント間通信支援機構

報を提供するライブラリを含む．クライアント側スタブオブジェクトは，Serdget 動作環境から得られる以下を始めとする環境情報を用いて，利用者側計算機に適応可能である．

ディスプレイ情報： クライアント側スタブは，ディスプレイ情報として解像度や色数を取得できる．この情報を用いて，利用者側計算機のディスプレイ解像度に対するユーザインタフェースの適応処理が可能となる．

ネットワーク情報： クライアント側スタブは，ネットワーク情報としてデータリンク種別や帯域などを取得できる．この情報を用いて，ネットワークの帯域に適応した Serdget 間通信が可能となる．

言語環境情報： クライアント側スタブは，言語環境情報として言語種別や通貨表示単位，時刻表示方法などを取得可能である．この情報を用いて，特定の言語環境に依存しない Serdget の構築が可能となる．

2.3.2 サーバ側スタブ

サーバ側スタブの最下層には，VNA コンポーネントのハードウエアやソフトウエアモジュールの持つ機能を直接制御するコントローラが位置する．これらの機能ごとに制御メソッドが対応し，同メソッドは次項に記述するクライアント側スタブから遠隔呼出し可能である．

各 VNA コンポーネント，あるいは VNA コンポーネントが接続されている計算機上では，Serdget 動作環境が起動しており，サーバ側スタブは同環境上で動作するサーバプログラムである．

2.3.3 クライアント側スタブ

クライアント側スタブは図 3に示す構成を取る．図 3において，Serdget のクライアント側スタブは以下に詳述するオブジェクトを含む．

ユーザインタフェースオブジェクト： Serdget コンテナに当該 Serdget クライアント側スタブが追加された際に表示される．ユーザインタフェースオブジェクトは VNA コンポーネントごとに存在し，ACL オブジェクトに基づいてユーザインタフェース部品の表示／非表示を切り替えられる．

スキンオブジェクト： 本オブジェクトを用いてユーザインタフェースの表示のみを容易にカスタマイズできる．例えば，当該 VNA コンポーネントに対する利用者の習熟度によって，アイコンの変更を始めとする利用者への適応が可能である．

ACL(Access Control List) オブジェクト： サーバ側スタブ内の遠隔メソッドに対して，複数のクライアント側スタブから同時に呼出しが発生した場合には，当該 VNA コンポーネント上で競合が発生する．本アーキテクチャでは，各 Serdget クライアント側スタブが ACL(Access Control List) オブジェクトを用いて利用可能な遠隔メソッドを監視し，競合の発生を防止する．

通信オブジェクト： サーバ側スタブへの遠隔メソッド呼出し機能，および他の Serdget との通信機能を提供する．本オブジェクトは，後述する VNA コンポーネント間通信支援機構を利用して上記の各通信を行う．

2.4 VNA コンポーネント間通信

VNA コンポーネント間通信においては，Serdget のクライアント側スタブとサーバ側スタブ間および複数の異なる Serdget 間の通信が，VNA コンポーネント間通信機構によって実現される．図 4に VNA コンポーネント間通信支援機構の概念図を示す．VNA コンポーネント間通信支援機構は，Serdget 動作環境の最下部として動作し，Serdget 動作環境間の到達性とトランスポート機能を提供する低層機構と，その上で遠隔メソッド呼出し等高レベルな通信機能を提供する高層機構から構成される．本機構によって異機種混在ネットワークに対応し，多様なデータリンクおよび下位層プロトコルを用いる VNA コンポーネント間での通信が可能となる．以下の各項で，内

部機構構成およびVNAコンポーネントからの利用形態について述べる．

2.4.1 低層通信機構

低層通信機構は，本アーキテクチャで想定するネットワーク接続環境におけるEthernet，IEEE1394，赤外線等の多様な物理層／データリンク層媒体，およびTCP/IP，IEEE1394トランスポート等のネットワーク層／トランスポート層プロトコルの差異を隠蔽し，これらの下位層プロトコルに依存しないSerdget間の通信の到達性，およびトランスポート機能を提供する機構である．

エンドポイント識別子： Serdgetスタブ間およびSerdget間通信におけるエンドポイントの識別には，Serdget動作環境におけるSerdget(スタブ)オブジェクト識別子を用いる．各Serdget動作環境はオブジェクト識別子の経路情報を交換し，オブジェクト識別子にもとづいた各Serdgetスタブ間およびSerdget間の経路の到達性を実現し，機構利用側から下位層プロトコルを隠蔽する．Serdgetは通信先への接続時に接続先オブジェクト識別子を用いてVNAコンポーネント間通信支援機構を利用する．

トランスポート： 低層通信機構が提供するend-to-endのトランスポートは，データの信頼性および順序を保証する(1)Virtual Circuit，データの信頼性のみを保証する(2)Reliable Datagram，いずれも保証しない最も軽量なトランスポートである(3)Unreliable Datagramの3機能である．

2.4.2 高層通信機構

高層通信機構は，下層通信機構の上で動作し，下層通信機構が提供する各トランスポート機能を利用して，遠隔メソッド呼出し機能およびイベントパッシング機能といった高レベルの通信機能を提供する機構である．

遠隔メソッド呼出し機能： 低層通信機構の提供するVirtual Circuitトランスポートを利用し，Serdgetスタブ間およびSerdget間の遠隔メソッド呼出しを実現する．

イベントパッシング機能： 低層通信機構の提供するReliable DatagramおよびUnreliable Datagramを利用し，Serdgetスタブ間およびSerdget間でのイベント通信を実現する．

2.4.3 VNAコンポーネント間通信利用形態

VNAコンポーネント間での通信は，Serdget内でのサーバ側スタブ／クライアント側スタブ間の通信，および異なるSerdget間での通信の2種類である．

Serdget内スタブ間通信においては，高層通信機構の提供する遠隔メソッド呼出し機能，およびイベントパッシング機能を利用する．一方Serdget間通信においては，通信されるメディアデータの種類によってその特性に応じた通信機能を利用する．ファイルなどのデータの信頼性を要求するメディアデータの通信には，Virtual Circuitトランスポートを，音声や動画といった連続メディアには，軽量なUnreliable Datagramトランスポートを利用する．

2.5 VNAサービスインタフェース

本アーキテクチャにおいては，コンポジットSerdgetの構築，および利用時のユーザインタフェースをVNAサービスインタフェースと呼ぶ．ただし，利用時のユーザインタフェースに関しては，Serdgetのクライアント側スタブが持つユーザインタフェースオブジェクトの集合とは別に，スクリプト言語による処理を始めとして，コンポジットSerdgetを隠蔽する目的のものを指す．

2.5.1 コンポジットSerdget構築インタフェース

利用者は，本アーキテクチャが提供するグラフィカルユーザインタフェースを用いて，簡単なマウス操作などでコンポジットSerdgetを構築できる．本インタフェースは以下に示す3つのエディタから構成される．

- 構造エディタ
- リンクエディタ
- インタフェースエディタ

利用者はまず，構造エディタを用いてコンポジットSerdgetにクライアント側スタブを包含するVNAコンポーネントを選択し，当該クライアント側スタブの配置を決定する．リンクエディタはこれらのクライアント側スタブにおける，各データの集約や転送に関する設定を行う．最後にインタフェースエディタを用いて，構築したコンポジットSerdgetに対して自動処理や一括処理を定義するスクリプトの編集を行う．また，インタフェースエディタを用いて，構築したコンポジットSerdgetに対して新たなGUIコンポーネントの割り当てや，スキンオブジェクトの編集が可能である．

2.5.2 スクリプト言語インタフェース

コンポジットSerdgetに対しては，スクリプト言語を用いた高度な処理記述が可能である．本インタフェースは，コンポジットSerdgetの持つスクリプト言語へのインタフェースを指す．実際に利用されるスクリプト言語の種別は実装時に決定され，次章以降に記述するプロトタイプシステムではJPython[4]による記述が可能である．本インタフェースを用いて，複数のコンポジットSerdgetに対する一括処理や，処理の自動化などが可能となる．

3 プロトタイプシステム

コンセプトの実証を目的に，ポゼッションシステムを利用して，VNA構築用ライブラリのプロトタイプの実装を行った．ポゼッションシステムでは，魂(Soul)が肉体(Body)に憑依する(possess)アナロジに基づくコンポーネントモデルを採用している．ポゼッションシステムにおけるBody-Soulに基づくコンポーネントモデルは，本アーキテクチャにおけるSerdget-コンポジットSerdget

に基づくコンポーネントモデルに容易に応用可能であった．また，Soulコンポーネントでは，スクリプト言語を用いた処理の自動化を提供しており，本アーキテクチャにおけるVNAサービスインタフェースの構築が容易であった．さらに，ポゼッションシステムのランタイムであるFieldは，本アーキテクチャにおけるSerdgetランタイムに相当し，VNAコンポーネント間通信支援機構およびSerdget動作環境通知機構を容易に追加できた．以上の観点から本プロトタイプは，ポゼッションシステムのアプリケーションとして実装した．まず，ポゼッションシステムにおけるコンポーネントモデルについて概説した後に，VNAアーキテクチャのコンポーネントモデルとの対応づけに関して記述する．

Soulコンポーネント： あるシステムが提供するサービスを中心とする視点で処理内容が記述される能動的なコンポーネント．Soulコンポーネントはスレッドが付随しており，後述するポゼッション操作によって登録されるBodyコンポーネントを制御することでサービスを提供する．

Bodyコンポーネント： 何らかのソフトウェアモジュールや，ハードウェアおよびその部分機能を被覆するコンポーネント．Bodyコンポーネント自体は受動的実体であり，Soulコンポーネントによる制御を前提として定義され，制御用インタフェースを外部にエキスポートする．

ポゼッション操作： Soulコンポーネントが，制御対象となるBodyコンポーネントを確定する操作．ポゼッション操作の実行以後，ユーザがSoulコンポーネントに対して行うメソッド呼出しが，Soulコンポーネントがポゼッション中のBodyコンポーネントに対して伝達され，そのメソッド呼出しを行う．SoulコンポーネントがBodyコンポーネントをポゼッションするためには 両者が共通インタフェースを実装している必要がある．ポゼッション操作の実行時に，Bodyコンポーネントへのアクセス権のチェックおよびインタフェースの適合性のチェックが行われる．

3.1 Serdgetの実装

ポゼッションシステムのコンポーネントモデルをVNAアーキテクチャに適用するため，Serdgetのサーバ側スタブにBodyコンポーネントを，クライアント側スタブにSoulコンポーネントを対応づけた．本プロトタイプでは，BodyおよびSoulコンポーネント同士で通信を行う前提条件として両コンポーネントでの共通インタフェースの実装を課している．したがって，Serdgetの実装について以下で(1) 共通インタフェースの定義，(2) Serdgetサーバ側スタブ(Bodyコンポーネント)の実装 (3) Serdgetクライアント側スタブ(Soulコンポーネント)の実装という手順で説明を行う．

共通インタフェースの定義 SoulコンポーネントがBodyコンポーネントを制御するために使用するメソッドを，共通インタフェースとして定義する．基本的には，制御対象であるBodyコンポーネントの動作内容に合わせてインタフェースを設計する．次の例では，PlayerInterfaceインタフェースを定義している．

```
public interface PlayerInterface {
    void play();
    void stop();
    void forward();
    void rewind();
        ⋮
}
```

Serdget サーバ側スタブの実装 PlayerInterfaceインタフェースを実装するBodyコンポーネントとして，BodyPlayerクラスを定義する．全てのBodyコンポーネントは，BodyComponentクラスの派生クラスとして定義される．

```
public class BodyPlayer extends BodyComponent
    implements PlayerInterface {
    public void play() {
        /* omit detailed description */
    }
    public void stop() {
        /* omit detailed description */
    }
                    ⋮
}
```

Serdget クライアント側スタブの実装 Soulコンポーネントの実装に先立ち，次の要領で共通インタフェースを元にSoulコンポーネントのテンプレートクラスを作成する．

```
% javac PlayerInterface.java ↵
% tempgen PlayerInterface ↵
```

tempgenコマンドの結果，Soul_comp_PlayerInterfaceクラスが生成される．tempgenコマンドの引数には，Soulコンポーネントの制御対象となるBodyコンポーネントが実現している共通インタフェースを全て指定する．新たなSoulコンポーネントを，生成されたテンプレートクラスの派生クラスとして定義し，Soulコンポーネントが Bodyコンポーネントを連携させて実行する具体的なサービス処理内容を通常runメソッド内に記述する．

```
public class SoulPlayer extends
    Soul_comp_PlayerInterface {
    void void run() {
        /* omit detailed description */
    }
}
```

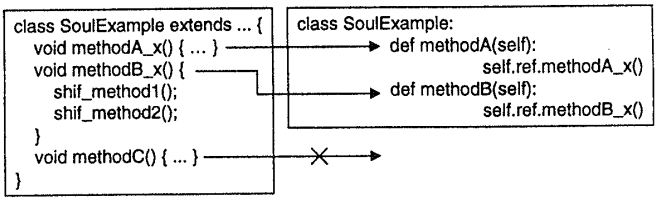

図 5: サービスインタフェースの生成

3.2 VNA コンポーネント間通信の実装

プロトタイプシステムでは，VNA コンポーネント間でのメディアデータの通信機構 (低層通信機構) は，Field と呼ぶポゼッションシステムのランタイムで提供される．前述した通り，Serdget サーバ側スタブに Body コンポーネントを，Serdget クライアント側スタブに Soul コンポーネントを対応づけており，VNA コンポーネントに対する通信の指示は，Soul コンポーネントからポゼッション中の各 Body コンポーネントへ制御データとして伝達され，Body コンポーネント経由で Field の低層通信機構によって処理される．

Soul コンポーネントから Body コンポーネントへは，メソッド呼出し形式でデータを渡す．すなわち，Soul コンポーネントのメソッド呼出し時に指定される引数が，対応する Body コンポーネントへの遠隔メソッド呼出しの引数として渡される．また，Body コンポーネントから Soul コンポーネントへの情報伝達には分散イベント通知機構を利用する．このため，Soul コンポーネントはポゼッション操作実行時に Body コンポーネントのイベントリスナとして登録される．

なお，現状では実験環境上の制約から，便宜的に制御データ用の高層通信機構をメディアデータの転送に併用している．低層通信機構は別途開発中であり，実験の進捗に応じて統合の予定である．

3.3 VNA サービスインタフェースの実装

本プロトタイプシステムでは，ユーザがコマンドインタプリタ経由で VNA アプリケーションを制御する利用形態を前提に，スクリプト言語で記述された VNA サービスインタフェースを実現した．

具体的には，スクリプト言語として JPython を採用し，コンポジット Serdget へのラッパーとして定義される JPython のクラスのインスタンスメソッドとして VNA サービスインタフェースを実現した．

スクリプト言語によるサービスインタフェースは，Soul コンポーネントのメソッドセット中の外部エキスポート指定されたメソッドについて，Soul コンポーネントの実体生成時に自動変換することで作成される．

Soul コンポーネントのメソッドセットは，(1) Body コンポーネントの制御に必要な共通インタフェースの実装，(2) VNA サービスインタフェースとして外部エキスポートすることを前提に作成されるメソッド，および (3) その他のユーテリティメソッドから成る．メソッドサブセット (2) は (1) や (3) を利用して定義され，意味的に高レベルな抽象を提供する．図 5 に例示するように，Soul コンポーネントのクラス定義の際に作成者が VNA サービスインタフェースとして外部エキスポートするメソッド名の接尾辞として "_x" を追加することで (methodA_x および methodB_x)，他のメソッドとの区別を行う．また，この例で methodB_x は，低レベルの制御インタフェースである shif_method1 および 2 を利用して記述されている (低レベルの制御インタフェースを隠蔽している)．

4 VNA 構築例

本節では，プロトタイプにおける VNA の構築例として，デスクトップテレビを取り上げる．デスクトップテレビは，図 1に示した VVCR を，計算機に接続するデバイスを用いて構築したものである．プロトタイプの開発には，Linux カーネル 2.0.37 上の Java 開発環境 (JDK1.2pre-v2) を用いた．また，TV チューナ機能には IBM Smart Capture Card II およびその接続用 TV チューナを利用した．応用事例で利用される Body コンポーネントの一部は，現状の実験環境では，Linux 上のデバイスファイルへの被覆オブジェクトとして実現されており，ネイティブコードで記述されたライブラリを必要とする．今後，組み込み機器や家電機器への JavaVM の搭載が進むことで，実験用に独自実装した部分の割合を減少させることが期待される．

デスクトップテレビの事例では，TV チューナ部品，TV リモコン部品，および TV ウィンドウ GUI 部品が Body コンポーネントとして実現されており，デスクトップテレビ VNA としての統合的な機能を実現する Soul コンポーネントが，各部品をポゼッションすることによってサービスを提供する．この場合，デスクトップテレビ機能を実現するために必要となる Body コンポーネントの種類は，Soul コンポーネントの定義時に作成者によって予め設定されており，具体的にどの Body コンポーネントインスタンスをポゼッションするのかは，デスクトップテレビ VNA が構成される時点での環境に応じて決定される．

図 6 は，デスクトップテレビ VNA の構成を模式的に示している．図中の矩形はホスト境界を意味し，矩形内に TV Window UI と Controller UI を表示している．また，アプリケーションの構成に利用される Body コンポー

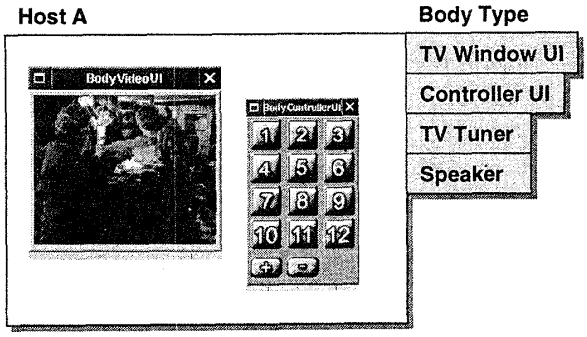

図 6: デスクトップテレビ VNA の構成

ネントの種類を矩形の横に一覧表示している．Soul コンポーネントは，ディレクトリサーバへの問い合わせによって必要な部品を発見し，ポゼッションする．複数のエントリが返された場合，結果のフィルタリング方針もプロパティで設定する．

4.1 記述例

デスクトップテレビ VNA の記述例を次に示す．

```
1 m = cmd.cm() ··· Medium の作成
2 s = cmd.cs('comp.SoulDeskTV') ··· Soul 作成
3 b1 = cmd.cb(m, 'comp.BodyTuner') ··· Body 作成
4 b2 = cmd.cb(m, 'comp.BodyControllerUI') ··· 同上
5 b3 = cmd.cb(m, 'comp.BodyTVWindowUI') ··· 同上
6 b4 = cmd.cb(m, 'comp.BodySpeaker') ··· 同上
7 s.possessIn(b1) ··· b1 を入力用にポゼッション
8 s.possessIn(b2) ··· b2 を入力用にポゼッション
9 s.possessOut(b3) ··· b3 を出力用にポゼッション
10 s.possessOut(b4) ··· b4 を出力用にポゼッション
11 s.start() ··· Soul 実行開始
```

3〜6 行目で Body コンポーネントインスタンスを作成し，7〜10 行目で Soul コンポーネントが入力用，出力用に Body コンポーネントをポゼッションした後に実行を開始する．

4.2 利用手順

ユーザがインタラクティブにアプリケーションを構成するための手段として，VNA シェルと呼ぶコマンドインタプリタを提供している．VNA シェルは，それ自体 Body コンポーネントとして実現されており，通常の Body コンポーネントと同様に Soul コンポーネントがポゼッション操作を行うことで制御可能である．また，ユーザがスクリプトを記述してバッチ処理で VNA を構成する，あるいは Soul コンポーネントにスクリプトを設定し，各種イベントに対応づけてスクリプトを自動実行することで VNA の構成変更を行うなどの処理が可能である．

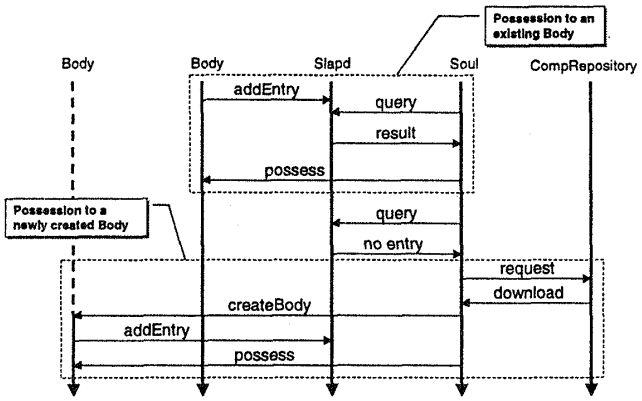

図 7: VNA の構成から動作までのインタラクション

4.3 動作手順

VNA が構成されて動作するまでの手順をまとめた結果を図 7 のインタラクションダイアグラムで示している．

Body コンポーネントはインスタンス化される際に，addEntry メソッドによって LDAP[5] ディレクトリサーバ (slapd) に属性情報の登録を行う．Body コンポーネントと slapd との通信には JNDI(Java Naming and Directory Interface) を利用する．Soul コンポーネントは slapd に問い合わせて検索を行い，エントリが返ってくれば対象 Body コンポーネントに possess メソッドを実行する (既存 Body コンポーネントのポゼッション部分)．問い合わせの結果，エントリが返ってこない場合には createBody メソッドによって新たな Body コンポーネントインスタンスを作成する．この際，必要となるクラスファイルは CompRepository と呼ぶクラスデータ管理サブシステムからダウンロードする (新規作成 Body コンポーネントのポゼッション部分)．

5 関連研究

本アーキテクチャに関連するものとして，HAVi(Home Audio/Video Interoperability)[6]，Jini(Java Intelligent Network Intfastracture)[7]，および UPnP(Universal Plug and Play)[8] が挙げられる．これらはいずれも情報家電機器の相互接続性を定める仕様であり，いずれも「機器」を視点において，プラグアンドプレイによる機器同士のスムーズな接続，連携および共通コマンドの実行に重点を置いている．これらはいずれも本研究が可能とする，利用者ごとの「仮想情報家電機器」の構築を行えない．また，本アーキテクチャは情報家電機器の開発や分散オブジェクト指向アプリケーション構築にも適用可能である点が他と異なる．一方，本アーキテクチャは機器のプラグアンドプレイについては規定しておらず，上記の各仕様とは補完関係にある．

また，遠隔メソッド呼出し機構として Java RMI[9] や Voyager[10] 等が挙げられる．これらは，本アーキテクチャ

においてVNAコンポーネント間高層通信機構にあたるものであるが，本アーキテクチャではこれに加えて下位層プロトコルを隠蔽する低層通信機構も含めて提供している点が異なる．

分散する情報家電機器の統御について，ユーザが携帯型コンピュータを用いて，遍在する機器を統合的に制御するシステムCUES (Control for Ubiquitous Embedded Systems)[11]が提案されている．CUESでは，移動コードや移動エージェントを含む複数の通信形態をサポートし，このために制御機器上にSMAF (Simple Mobile Agent Facility)と呼ぶ実行環境が動作することを前提としている．遍在機器の統合的制御の方式として，自動サービス統合(Automatic Service Integration)およびユーザ補助によるサービス統合(User Assisted Service Integration)の2つを想定しており，実現例としてプリントサービスの統合を取り上げている．これに対して本研究では，自動／ユーザ補助によるサービス統合に相当する分類に加え，既存の家電機器を機能の集合として仮想化するボトムアップと，実世界に存在する既存の家電機器にとらわれず，新しい家電機器としてVNAアプリケーションを作り出すトップダウンアプローチによる統合という分類の次元に基づくサービス統合(すなわちVNAアプリケーションの構築)を支援している点に特徴がある．

6 おわりに

本稿では，既存の情報家電機器やソフトウエアモジュールの機能を組み合わせ，新たな情報家電機器を仮想的に構築可能な，VNAアーキテクチャを提案し，プロトタイプシステムの設計と実装に関して述べた．

プロトタイプシステムは，ポゼッションシステムを用いてJava言語によって実装した．本プロトタイプシステムでは，VNAアーキテクチャにおけるSerdgetのクライアント側スタブとポゼッションシステムにおけるSoulコンポーネントが，またサーバ側スタブとBodyコンポーネントがそれぞれ対応している．本アーキテクチャの大部分はプロトタイプシステムとして実装したが，現在，クライアント側スタブ中のACLオブジェクトとスキンオブジェクト，およびサービスインタフェースの実装を進行中である．

本アーキテクチャは，各種デバイスや既存のソフトウエアモジュールを用いた分散システムの構築に応用可能であるほか，利用者指向の新たな情報家電機器開発にも適用できる．すなわち，本アーキテクチャでは既存デバイスやソフトウエアモジュールの組み合わせによって新たな情報家電機器を仮想的に構築するため，特定の組み合わせ情報を様々な場所に応用できる．さらにスキンオブジェクトや利用者側計算機の環境情報取得機能により，利用者適応性および機器適応性を持つ情報家電機器を開発できる．

一方今後の課題として，実際に本アーキテクチャを搭載した情報家電機器の試作が挙げられる他，これらを用いた情報家電ネットワークの構築，本アーキテクチャの試用が望まれる．これによって本アーキテクチャの有効性を確認し，問題点のフィードバックが必要である．

参考文献

[1] 大越匡, 中澤仁, 田村陽介, 望月 祐洋, 戸辺義人, 西尾信彦, 徳田英幸: VNA：仮想情報家電の実現へ向けて, 第59回情報処理学会全国大会 (1999).

[2] Mochizuki, M. and Tokuda, H.: Possession System: Middleware for Adaptive Multiuser Applications in a Mobile Environment, *International Conference on Distributed Computing Systems* (1999).

[3] Birrell, A., Nelson, G., Owicki, S. and Wobber, E.: Network Objects, *14th ACM Symposium on Operating System Principles*, pp. 217–230 (1993).

[4] Hugunin, J.: Python and Java – The Best of Both Worlds, *Proceedings of the 6th International Python Conference* (1997).

[5] Smith, M. and Howes, T.: *LDAP – Programming Directory-Enabled Applications with Lightweight Directory Access Protocol*, Macmillan Technical Publishing (1997).

[6] Sony, Matsushita, Philips, Thomson, Hitachi, Toshiba, Sharp and Grundig: Specification of the Home Audio/Video Interoperability (HAVi) Architecture (1998). http://www.havi.org/home.html.

[7] Sun Microsystems, Inc.: Jini Architecture Specification (1998). http://www.javasoft.com/products/jini/specs/jini-spec.pdf.

[8] Universal Plug and Play Forum: Universal Plug and Play (UPnP) (1999). http://www.upnp.org.

[9] Sun Microsystems Inc.: Remote Method Invocation Specification (1996).

[10] Glass, G.: ObjectSpace Voyager: the agent ORB for Java, *2nd International Conference on World-Wide Computing and its Applications*, pp. 38–55 (1998).

[11] Kangas, K. and Röning, J.: Using Mobile Code for Service Integration in Ubiquitous Computing, *Proceedings of the 5th Mobile Object Systems Workshop* (1999).