仮想3次元空間における統合UI構築ソフトウェアの開発

Implementation of User Interface Composition System with 3D Visualization Toolkits

門田 昌哉
Masaya Kadota
慶応義塾大学 政策・メディア研究科
E-mail: masaya@ht.sfc.keio.ac.jp

ABSTRACT: This paper proposes vARC:Visual Aids for Remote Control, that enables end-users easy to control services in the ubiquitous computing environment. vARC contains the sowtwares for user interface composition, location aware directory service wrapper and 3D visualization of the domain. This paper also proposes CUIL:Composable User Interface Language, the user interface description language for composing user interface.

1 背景

ユビキタスコンピューティング環境は情報家電機器, AV 機器，センサ，アクチュエータ等の多様な計算機群から構成される．本稿では，これらの計算機上で動作するソフトウェアをサービスと呼ぶ．同環境において，ユーザは携帯電話や PDA 等の小型携帯端末，および環境に埋め込まれた端末を用いて遠隔地にあるサービスを制御できる．既存の遠隔制御モデルとしては, HTML 等で記述されたユーザインタフェースを動的に端末上にダウンロードするモデルが一般的である．

また，近年の分散コンポーネントミドルウェアの発達により，サービスは単体で動作するだけでなく，複数のサービスが協調動作して動作可能になりつつある．たとえば，複数のサービスに共通するする電源管理機能を集約して仮想的な統合電源管理サービスを構築したり，DVD プレーヤの映像を複数のディスプレイに表示するサービスを構築するような協調動作が挙げられる．サービスを単体で用いるだけでなく，複数のサービスを協調動作させることにより，複数のサービス群を仮想的に単一のサービスとして利用可能になる．

しかし，既存の遠隔制御モデルおよび分散コンポーネントミドルウェアにおいて，サービスは IP や特定のディレクトリサービス上での識別子によって管理されるため，ユーザが制御対象となるサービスを指定しにくい問題点がある．そのため，類似した識別子を持つサービスが同一ネットワーク上に複数ある場合，それらを区別することが難しい．たとえば，複数のディスプレイがある環境において，それぞれのディスプレイを区別するためには，ユーザが独自の命名規則を用いて静的に名前づけを行う必要がある．そのため，より直感的にサービスを指定するための基盤ソフトウェアが必要である．

また，既存の分散コンポーネントミドルウェアでは, サービス間の関係を動的に変更できず，サービス間の関係定義にも個々の分散コンポーネントミドルウェアに固有な文法の理解が求められるため，エンドユーザが動的にサービス間の関係を定義する環境には適さない問題点がある．これを解決するためには，直感的にサービス間の協調動作関係を定義可能なフロントエンドの開発が必要である．

同様に，生成された仮想的なサービスを制御可能なユーザインタフェースを動的に構築する機構がないため，従来のユーザインタフェース生成機構では，協調動作するサービス群を統合的に扱えない問題点がある．そのため，動的に定義されるサービス間の関係を反映してユーザインタフェースを動的に再構成するための機構が必要である．

2 目的

本稿では，前章の問題点を解決するため，視覚的にサービス間の協調動作を定義し，協調動作するサービス群を制御可能なユーザインタフェースを動的に構築する機構 **vARC: Visual Aids for Remote Control** を提案する．vARC は，ホームネットワーク等のシステム管理者が介在しない環境において，専門知識を持たないエンドユーザの簡易なサービス制御の支援を目的とする．

vARC は，サービス間の関係を視覚的に定義可能にするカードのメタファを用いたユーザインタフェースマネージャ，動的にユーザインタフェースを合成するユーザインタフェース合成機構，および視覚的にサービスを指定するための 3 次元空間作成ツールキット群から構成される．vARC において，ユーザは 3 次元空間上に表示されたサービスを選択することにより，視覚的に制御対象となるサービスを指定できる．また，サービスを制御するユーザインタフェースはカードとして表示され，ユーザはカードを重ねることによってサービス間の協調動作を定義できる．動的に定義されたサービス間の関係から，vARC は個々のサービスに対するユーザインタフェースを動的に合成することにより，協調動作するサービス群を単一のサービスのように制御可能になる．

本稿の構成は，第 3 章で vARC の概要および設計を述べ，第 4 章で vARC におけるユーザインタフェース記述言語を述べる．その後，第 5 章で vARC が提案するユーザインタフェース合成機構について述べ，第 6 章では vARC システムの実装，および基本性能の評価を行う．第 7 章で未踏ユース事業における本提案の開発体制について述べ，第 8 章で本稿をまとめる．

3 vARC の概要

本章では，はじめに想定環境と本稿での用語定義を行い，vARC の概要について述べる．

3.1 想定環境と用語定義

本論文では，**サービス**を複数の計算機上に分散して動作可能な**分散コンポーネント**の集合として定義する．分散コンポーネントは，環境に遍在するネットワーク接続性を持ったソフトウェアコンポーネントである．サービスが提供する不可分な機能が分散コンポーネントとして実装される．サービスを分散コンポーネントの集合として実装することにより，ネットワークを介した柔軟なサービス間の協調動作を実現可能になる．

本機構が対象とする**ホームネットワーク**は，情報家電機器や AV 機器等の計算機で構成されるため，特定の計算機上でサービスが動作することが多い．そのため，本論文では分散コンポーネントはあらかじめ特定のサービスの構成要素であると想定する．単体で動作するメディア変換等の分散コンポーネントは，それ単体を構成要素とするサービスである．

図 1 に本稿が想定するサービスの例を示す．たとえば，DVD プレーヤは電源管理コンポーネント，映像出力コンポーネント，音声出力コンポーネント，および再生，停止等のストリーミングを制御するコンポーネントから構成される．また，プラズマディスプレイは電源管理コンポーネント，映像入力コンポーネント，音声入力コンポーネントから構成される．

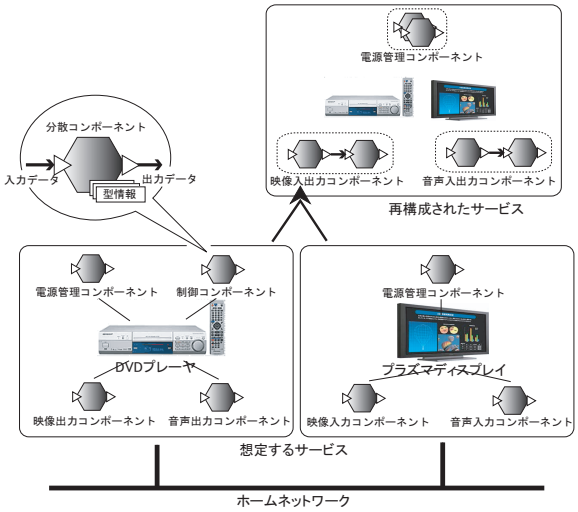


図 1: vARC が想定する環境と背景となる技術

分散コンポーネントミドルウェアは，複数のサービスの構成要素から，論理的なサービスを再構成する機構である．図 1 では，DVD プレーヤとプラズマディスプレイの電源管理コンポーネントの集約を実現すると同時に，映像入出力コンポーネントおよび音声入出力コンポーネントの接続を実現している．異なるサービスを構成する分散コンポーネントを合成し，サービスを再構成することを**サービス合成**と呼ぶ．vARC では，Jini[6] や CORBA[3] 等の分散ミドルウェアに加え，VNA[2]，Ninja Path[1]，SWORD[4] 等の機構をサービス合成機構として想定する．ホームネットワークは，これらのサービス合成機構が混在する環境である．

3.2 vARC の構成

図 2 に vARC のシステムの概要を示す．vARC は，vARC は，部屋を 3 次元表示する Domain Browser，カードのメタファを用いてユーザインタフェースの動的合成を実現する CUICs:Card-oriented User Interface Composition System, 既存のディレクトリサービスを位置情報に基づいてサービス検索可能にするための Domain Server, サービス間のデータをブリッジする Media Splitter モジュールから構成される．Domain Broser および CUICs がクライアントモジュールであり，ユーザが利用する端末上で動作する．また，Domain Server および Media Splitter がフレームワークモジュールであり，これらのモジュールは同一ネットワークセグメント内の任意の計算機上で動作している必要がある．サービス自身が持つ情報としては，ユーザインタフェース記述ファイル，およびサービスが動作する計算機の形状を記述した形状定義ファイルがある．以下に個々のモジュールの簡潔な機能概要を述べる．

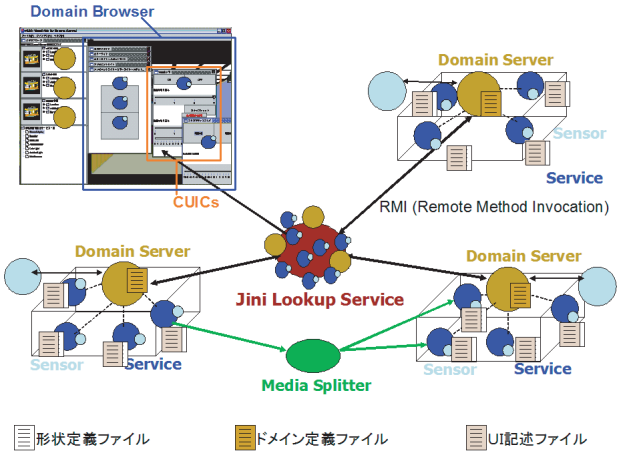


図 2: vARC の概要

Domain Browser

Domain Browser は，Domain Server に登録されたサービスから形状定義ファイルを取得し，部屋単位での 3 次元空間を作成するモジュールである．3 次元空

間におけるサービスの座標は，環境内に位置情報センサを管理するサーバが存在する場合は動的にサービスの位置情報を Domain Server から取得し，センサシステムの座標系と Domain Browser 内での座標系に変換する処理を行う．

Domain Server

Domain Server は，マルチキャストディスカバリによって地理的に離れた場所にあるサービスを検索する問題点を解決し，部屋単位でサービスを管理するためのディレクトリサービスラッパである．Domain Server 自身はディレクトリサービスの機能を持たず，Jini や CORBA 等の分散コンポーネントミドルウェアが提供するディレクトリサービスを拡張する．Domain Server は，サービスが追加されたイベントを受け取り，自身と同様な ID を持つサービスを保持する．また，Domain Server に追加されたサービスが特定のセンサシステムから座標を取得可能である場合，Domain Server は指定されたセンサシステムとコネクションを張り，センサの座標を管理する．利用可能なセンサシステムがない場合，サービスは自身の座標系を静的に持つ必要がある．

CUICs (Card-oriented User Interface Composition System)

CUICs はユーザインタフェースの生成および合成を行うモジュールである．CUICs ではユーザインタフェースはカードに抽象化され，ユーザはカードを重ねることによって動的にユーザインタフェースの合成を行う．vARC では，ユーザインタフェース記述言語として独自言語である CUIL (Composable User Interface Language) を用いる．CUIL において，サービスは分散コンポーネントの集合として定義され，分散コンポーネント単位で役割や入出力データ形式等のメタ情報を記述する．また，ユーザインタフェースも分散コンポーネント単位で，コマンドを発行する GUI コンポーネントとコマンド引数を設定する GUI コンポーネントに分けて記述される．vARC におけるユーザインタフェースの合成は，これらの分散コンポーネント単位で記述された GUI コンポーネント群を同様な役割の機能を制御する GUI コンポーネント群の集約，および入出力データ形式が一致する機能の組み合わせを同時に制御可能なユーザインタフェースの構築を指す．

Media Splitter

CUICs モジュールでは，入出力データ型の一致に基づくユーザインタフェースの合成機構を提供する．しかし，制御用端末上に配置される CUICs を介してデータ入出力をブリッジするアーキテクチャは，制御用端末が無線や Bluetooth などの比較的帯域の狭いネットワークによって接続されることを考慮すると現実的でない．そのため，サービス間の入出力を分散コンポーネント間でブリッジするためのフレームワークが必要である． Media Splitter モジュールはデジタルカメラで取り込んだ画像や，音声ファイル等を任意のサービスへブリッジするための機構を提供する．

4 ユーザインタフェース記述言語の設計

本章では，vARC で用いるユーザインタフェース記述言語 CUIL (Composable User Interface Language) について述べる．なお，CUIL の DTD およびタグセットの詳細な説明は以下の URL から参照可能である．

http://www.ht.sfc.keio.ac.jp/~masaya/varc/desc/

4.1 CUIL の特徴と他の言語との相異点

現在，ユーザインタフェースを多様な端末上で動作させるためのユーザインタフェース記述言語が多く提案されている．これらの言語では，画面の解像度や色数等の端末依存な情報を外部化し，GUI ウィジェットをユーザからの入力値を用いて抽象的に記述することにより，端末非依存なユーザインタフェースの記述を実現している．CUIL では，これらの言語と同様な手法を用いるが，ユーザインタフェース合成に必要な以下の要件を考慮する．これらの要件を実現するためには，既存の言語仕様を大きく改編する必要があるため，vARC は独自言語を設計した．

- 分散コンポーネント単位でのユーザインタフェースの記述
- 分散コンポーネント単位でのメタ情報の記述
- 実行コマンドの有無による GUI ウィジェットの階層化

分散コンポーネント単位で記述されるメタ情報として，CUIL では分散コンポーネントの役割の型を定義する．役割の型は，サービスが提供すべき機能の標準化と同義である．サービスの標準化は，現在 HAVi[5] や UPnP[7] で行われているが，現状では一般的な仕様が存在しないため，CUIL ではこれらの仕様を参考にして役割の型を定義する．

4.2 全体の構成

図 3 に CUIL の概要を示す．<service> には，ベンダ名やシリアルナンバー等のサービス固有の情報を記述し，<component> には，分散コンポーネント単位でのユーザインタフェースを記述する．<layout> には，GUI ウィジェットの画面上での配置を記述する．CUIL では，画面を網状に区切り，その上に GUI ウィジェットを配置する．これにより，GUI ウィジェットのレイアウトを壊さずに，端末ごとの解像度を考慮して GUI ウィジェットのサイズを動的に変更可能になる．

4.3 GUI ウィジェットの分類

CUIL では GUI ウィジェットを実行コマンドの有無によって mainWidget と subWidget に分類する．mainWidget はユーザの入力があった際に分散コンポーネントに対して実行コマンドを発行する GUI ウィジェットであり，subWidget は実行コマンドに必要な引数を設定するための GUI ウィジェットである．たとえば，エアコンの温度設定に必要な GUI ウィジェットの組み合

```
<?xml version="1.0" encoding="UTF-8"?>
<cuil>
   <service>...</service>
   <component>...</component>
              ⋮
   <component>...</component>
   <layout>...</layout>
</cuil>
```

図 3: CUIL の構成

わせとして，(1) 単一の Slider，(2)Slider と Button の組み合わせが考えられる．前者はスライドバーを操作した際に実行コマンドが発行され，後者はボタンを押すまで実行コマンドが発行されない．GUI ウィジェットの分類により，合成されたユーザインタフェースを構成する GUI ウィジェットがそれぞれ別の分散コンポーネントに対して実行コマンドを発行する不整合を防げる．

4.4 ユーザインタフェースの記述

ユーザインタフェースの記述は，<component> 内に分散コンポーネント単位で行う．<component> は <meta>，<mainWidget>，<command> のタグセットから構成され，<subWidget> は <mainWidget> の実行コマンドの引数を設定する GUI ウィジェットであるため，記述は任意である．図 4 に DigitalVideoCamera の Capture 機能の記述例を示す．

Capture 機能を実装する分散コンポーネントは，メタ情報として DigitalVideoCamera の Capture 型を持ち，出力データ型として MIME で規定された動画形式を持つ．また，GUI ウィジェット群は，時間を設定するための Slider と実行コマンドを発行するための Button から構成される．GUI ウィジェットの種類は，特定の GUI ツールキットに依存しないため，ユーザの入力値に基づいて抽象的に記述される．また，本機構では，実行コマンドの発行に RPC(Remote Procedure Call) を用いるため，実行コマンドとして，<mainWidget> が実行された場合に呼び出すメソッド名を <command> 内に記述する．

5 ユーザインタフェース合成機構

本章では，CUICs におけるユーザインタフェース合成機構について述べる．vARC におけるユーザインタフェース合成機構の成果は，情報処理学会 OS 研究会への論文 [8] としてまとめた．

5.1 概要

ユーザインタフェース合成機構は，既存の分散コンポーネントミドルウェア上で動作し，再構成されたサービスを制御可能なユーザインタフェースを動的に生成する機構である．ディレクトリサービス，および分散コンポーネント間の通信パスの生成は下位のサービス合成機構が提供する機能に依存する．ユーザインタフェースの合成は，ユーザが CUICs 上で定義す

```
<component id="arbitraryID">
   <desc>動画をキャプチャします</desc>
   <meta include="DigitalVideoCamera">
      <role>Capture</role>
      <inputType>null</inputType>
      <outputType>audio/x-pn-realaudio ra rm ram</outputType>
   </meta>
   <mainWidget id="button" type="boolean">
      <label>http://www.ht.sfc.keio.ac.jp/~masaya/foo.gif</label>
   </mainWidget>
   <subWidget id="slider" type="int">
      <max>30</max>
      <min>0</min>
      <unit>1</unit>
      <default>15</default>
      <label>時間設定</label>
   </subWidget>
   <command>
      <method>capture</method>
      <parameter>$slider</parameter>
   </command>
</component>
```

図 4: DigitalVideoCamera のユーザインタフェースの記述例

るサービス間の関係に従って協調動作可能な分散コンポーネントの組を求め，その組の要素となる分散コンポーネントのユーザインタフェース定義を合成することによって実現する．そのため，CUICs におけるユーザインタフェース合成は以下の二つのフェーズに分類できる．

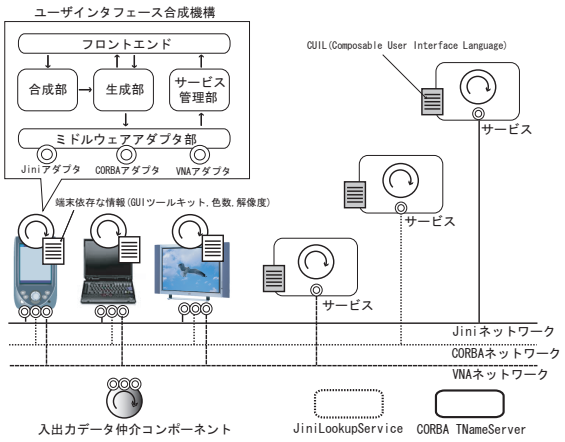


図 5: ユーザインタフェース合成機構の設計

5.1.1 分散コンポーネント間の関係定義

本機構において，協調動作可能な分散コンポーネントの関係は，分散コンポーネントごとに定義されたメタ情報の一致によって求める．利用するメタ情報として，役割の型と入出力データ型を用いる．メタ情報の一致は前向き推論であるため，事後条件となる条件を設定する必要がある．そのため，本機構では，事後条件として，同様な役割を持つ分散コンポーネントを集約する**集約合成規則**，入出力データ型が一致する分散コンポーネントを接続する**接続合成規則**の 2 つの合成

規則を用いる．これら 2 つの合成規則を用いることにより，関連性のない分散コンポーネントの接続や集約を実現できない反面，自動的に分散コンポーネント間の関係を定義できる．自動的に関係を求めることにより，ユーザはより簡易に協調動作可能な分散コンポーネントの組を求めることができる．

5.1.2 ユーザインタフェースの合成

本機構において，ユーザインタフェースの記述は分散コンポーネント単位で行う．これにより，本機構は再構成されたサービスに必要なユーザインタフェースの情報を認識できる．本機構では，前項で述べた集約合成規則，接続合成規則に一致する分散コンポーネントの組に対して，動的にユーザインタフェースを合成する．ユーザインタフェースの合成アルゴリズムは，UI 集約合成規則と UI 接続合成規則がある．UI 集約合成規則は，同様な役割を持つ GUI ウィジェット群を集約し，複数の分散コンポーネントを一括制御可能なユーザインタフェースを構築する合成規則である．また，UI 接続合成規則は，入出力データ型が一致する分散コンポーネントを連続的に制御可能なユーザインタフェースを構築する合成規則である．

図 6 にユーザインタフェース合成の概念図を示す．白色の図形が subWidget を示し，灰色の図形が mainWidget を示す．図形の形は，入力値による GUI ウィジェットの型を示し，同一の図形は同様な型を持つ GUI ウィジェットを示している．以下にそれぞれの合成規則適用時の GUI ウィジェットの合成，および実行コマンドの合成について述べる．

UI 集約合成規則

図 6(a) に UI 集約合成規則を適用した際のユーザインタフェース合成アルゴリズムの概念図を示す．UI 集約合成規則では，異なる分散コンポーネントを制御する GUI ウィジェット群から，同様な入力値を受け取る GUI ウィジェットの論理積をとることにより，GUI ウィジェットを集約する．異なる入力値を受け取る GUI ウィジェットがある場合には論理和をとる．図では，灰色の丸で示す mainWidget が同様な入力値をとる GUI ウィジェットであるため集約されている．

集約可能と判断する GUI ウィジェットは端末ごとに異なり，たとえば int 型と float 型の入力のために同様に Slider を利用する端末の場合，2 つの Slider は集約可能である．しかし，実行コマンドの引数に必要な値は異なるため，入力値の差異を考慮して値を変換する処理が必要である．たとえば，集約された Slider の割合から小数値を割り出し，実行に必要なコマンドを生成する処理を行う．

また，2 つの mainWidget が同一の GUI ウィジェットとして集約可能でない場合，実行コマンドを発行するタイミングを考慮する必要がある．mainWidget が集約可能な場合は，実行コマンドの発行は集約された mainWidget に従う．しかし，GUI ウィジェットが異なる場合，異なる GUI ウィジェットがそれぞれの分散コンポーネントに対して実行コマンドを発行してしまう．そのため，ユーザインタフェースの合成には優先度を設け，優先度が高い分散コンポーネントを制御するユーザインタフェースの mainWidget のコマンド発行にコマンド実行のタイミングを合わせる．

UI 接続合成規則

図 6(b) に接続合成規則を適用した際のユーザインタフェース合成の概念図を示す．接続合成規則では，集約合成規則ど同様にユーザからの入力値が同様な GUI ウィジェットの論理積をとるが，実行コマンドの発行のタイミングが異なる．実行コマンドは，分散コンポーネントに対して順次発行され，仲介コンポーネントに対しても同様に実行コマンドを発行する．

GUI ウィジェットの合成としては，同様な白丸で示された同様な型を持つ GUI ウィジェットの論理積，および灰色の丸で示された GUI ウィジェットの論理積，他の GUI ウィジェットの論理和をとる GUI ウィジェットの集約を行っている．入出力データ型が一致する分散コンポーネントを接続する mainWidget を集約する際には，本機構が GUIWidget を変更し，GUI ウィジェット間のラベルを矢印で結ぶ．

5.2 入出力データの接続

異なるサービス合成機構上で動作する分散コンポーネント間の相互運用性を実現する方法として，SOAP 等のアダプタを各分散コンポーネントに配置する方法，および端末上に配置された本機構を介して入出力データを仲介する方法が考えられる．しかし，前者は分散コンポーネントの実装コストが増加する問題があり，情報家電機器や AV 機器等のリソース制約の大きい機器には適さない．また，後者は端末が無線等の帯域の狭いメディアで接続されることを考慮すると，ストリーミングメディアを複数仲介することが困難なため現実的でない．そのため，本機構では，ホームネットワーク上に既知の入出力データを仲介する分散コンポーネントを配置する．5 に示す仲介コンポーネントがこれにあたる．これにより，異なるサービス合成機構上で動作する分散コンポーネント間の相互運用性を実現する．

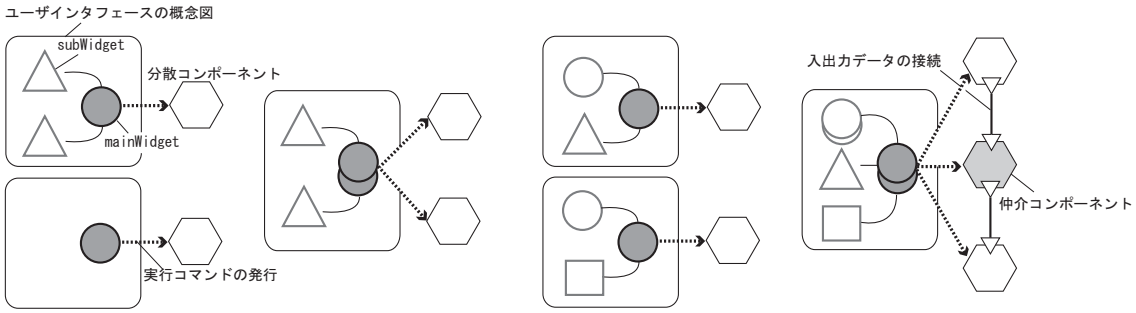
6 vARC の実装と基本性能評価

本章では，vARC の実装と各モジュール群の基本性能の評価を行う．なお，vARC の現時点での実装は以下の URL より取得可能である．

http://www.ht.sfc.keio.ac.jp/˜masaya/varc/download/

6.1 実装環境

表 1 に本システムの実装環境を挙げる．vARC の実装には，多様なプラットフォーム上で動作するため Java 言語を用いた．また，分散コンポーネントミドルウェアとして，Jini1.2.1 を用いた．また，利用した計算機環境ととして，計算機環境 (1) と計算機環境 (2) を



(a) 集約合成規則
同様な役割を持つ分散コンポーネントの集合に対するsubWidgetおよびmainWidgetを集約し，実行コマンドをマルチキャストする．

(b) 接続合成規則
入出力データ形式の一致する分散コンポーネントの集合に対するmainWidgetを順次実行し，仲介コンポーネントに対してコマンドを発行する．

図 6: GUI ウィジェットの合成と実行コマンドの合成

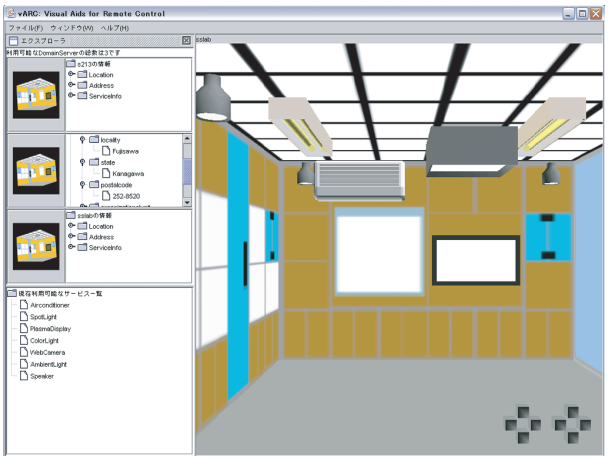


図 7: Domain Browser の実装画面

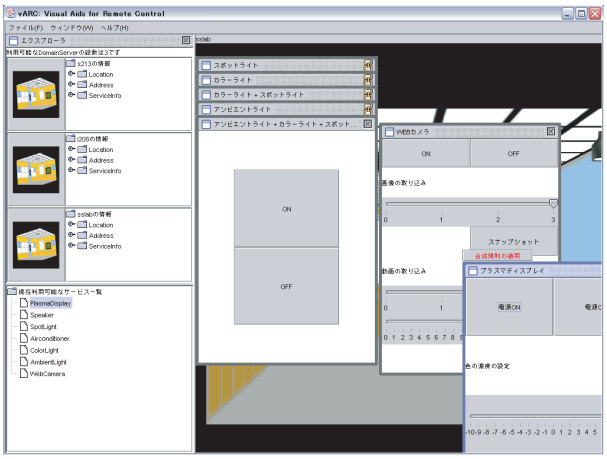


図 8: CUICs の実装画面

用いた．(1) を CUICs および Domain Server の評価に用い，(2) を Domain Browser の開発および評価に用いた．これらの計算機群は 100Mbps のイーサネットで接続される．3 次元空間の作成には，SunMicroSystems 社が提供する Java3D ライブラリ (version 1.3.1) を用いた．

表 1: 実装環境

	計算機 (1)	計算機 (2)
CPU	Mobile PentiumIII 750MHz	PentiumIV 2.4GHz
OS	VineLinux2.6	Windows2K
メモリ	256MB	512MB

6.2 実装例

図 7 に Domain Browser の実装画面を示し，図 8 に CUICs の実装画面を示す．画面左に示す Window が同一ネットワークセグメント上にある Domain Server の一覧であり，画面右に示す Window が Domain Browser および CUICs の動作例である．

6.3 基本性能の評価

本節では，本機構のプロトタイプシステムの基本性能評価について述べる．以下の性能評価では，表 2 に示す CUIL を用い，性能評価の数値はそれぞれ 1000 回のサンプルから，ガベージコレクション等によって出た外れ値を除外した値の平均値である．

表 2: 基本性能評価に用いた CUIL の概要

サービス名	サイズ	機能数	GUI 数
Light	1.27KB	2	2
PlasmaDisplay	2.57KB	4	5
Airconditioner	3.38KB	5	8
WebCamera	3.31KB	4	9
DVDPlayer	5.17KB	9	12

ユーザインタフェース生成時間の測定

表 3 にユーザインタフェース生成時間の測定結果を示す．評価には表 1(1) を用いた．ここでは，表 2 に示すユーザインタフェース記述文書からの，CUIL のダウンロード時間，構文解析時間，生成時間，フロントエンド上にユーザインタフェースが表示されるまでの時間を示している．ダウンロード時間は，Jini 上で RMI を用いて行った場合の時間である．

以上の測定結果より，プロトタイプシステムにおいて，ユーザインタフェースは概ね 50ms 程度で表示さ

れることが分かる．したがって，ユーザインタフェースとしての応答性を満たす処理速度でユーザインタフェースを生成可能である．しかし，プロトタイプシステムでは，GUI ウィジェットのラベルに画像を用いていないことや，評価に用いた GUI ウィジェットの総数が少ないことが挙げられる．そのため，今後の機能拡張によって処理時間が増加する可能性がある．

ユーザインタフェース合成時間の測定

表 4 にユーザインタフェース合成時間の測定結果を示す．評価には表 1 を用いた．ここでは，集約合成規則および接続合成規則の適用による GUI ウィジェットの減少数，および合成規則適用にかかる時間，フロントエンドにユーザインタフェースが表示されるまでの時間を示している．

測定結果より，ユーザインタフェースの生成時間と合成時間の総時間を比較すると，(1) は生成時間とほぼ同じ時間で合成規則が適用され，(3) では WebCamera の生成時間よりも少ない時間でユーザインタフェースが合成されることが分かる．理由として，ユーザインタフェースの合成が構文解析されたオブジェクト間で行われる点，CUIL のダウンロードがすでになされている点が考えられる．これにより，プロトタイプシステムにおいて，ユーザインタフェースの合成は，ユーザインタフェースの生成時間のほぼ等価に処理可能であることが分かる．

利用可能なサービスの一覧取得時間の測定

図 9 にサービス一覧の取得に必要な処理時間を示す．評価は Jini におけるサービスの参照取得に必要な時間である．横軸が利用可能なサービス数を表し，縦軸が処理時間を示す．評価には 1 計算機環境 (1) を用いた．本機構では，異なるディレクトリサービスに対して利用可能なサービスの取得要求を出すため，システムのボトルネックになる可能性がある．しかし，クライアント起動時に行う利用可能なサービスの取得要求には 15ms 程度の時間を要するが，単一のサービスの参照の取得に必要な処理時間は 6ms 程度である．そのため，バックグラウンドでサービスへの参照を更新しつづける処理はボトルネックにならないことが分かる．

3 次元空間の描画時間の測定

Domain Browser は Java3DAPI を用いて，Java から OpenGL 等のネイティブ API を呼び出すため，パフォーマンスに問題があることが想定される．そのため，3 次元空間構築および平均レンダリング画像生成時間の測定を行った．レンダリング画像の生成の評価は，3 次元空間の構築を環境にある比較的高性能な計算機上で行い，レンダリングされた画像を端末に送ることを想定しているため，画像を生成するために必要な時間を測定する必要がある．評価に用いた計算機環境は表 1 に挙げた計算機環境 (2) を用いた．

図 10 より，3 次元空間の作成は平均して 5 秒から 6 秒の時間を要することが分かる．表示する 3D オブジェクトの増加数に応じて，3 次元空間の初期化には約 250 ミリ秒程度の増加がみられる．数値のばらつきは，生成時間が 3D オブジェクトの複雑度に依存するためである．3 次元空間の構築には計算コストが高く，現状の実装ではエンドユーザの端末上では動作できない．

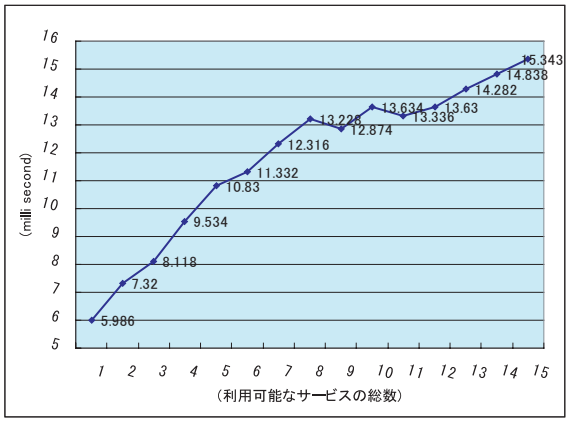


図 9: 利用可能なサービスの一覧取得時間

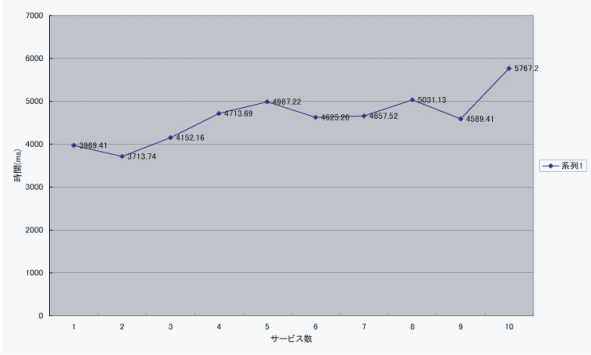


図 10: 3 次元空間構築に必要な時間の測定

図 11 は，Domain Browser におけるレンダリング画像生成の平均時間の測定結果である．測定結果より，1 枚の画像の生成に必要な時間は約 7 秒程度であり，毎秒 7 フレーム弱のレンダリングが可能である．

以上の測定結果より，現在の Domain Browser の実装は，端末上で動作させるには計算コストが高いが，画像を送信して利用する際には，ユーザの利用に支障をもたらさないといえる．

7 まとめ

本稿では，未踏ユース事業で開発した仮想 3 次元空間における統合 UI 構築ソフトウェア vARC の設計と実装，および評価について述べた．vARC は，ホームネットワークなどのシステム管理者が存在しない環境において，環境に遍在するサービスをより簡易に制御可能にするためのソフトウェア群である．vARC は，カードを用いたユーザインタフェースマネージャ，部

表 3: ユーザインタフェース生成時間の測定結果 (単位:milli second)

サービス名	DL 時間	構文解析時間	生成時間	表示時間	総時間
Light(Ambient,Spot,Color)	4.40ms	12.16ms	16.52ms	6.30ms	39.38ms
Plasma Display	4.23ms	9.14ms	15.31ms	10.94ms	39.62ms
Airconditioner	5.25ms	7.19ms	18.15ms	29.00ms	59.59ms
DVDPlayer	9.61ms	7.14ms	18.48ms	25.49ms	60.72ms
WebCamera	5.17ms	7.64ms	24.70ms	30.21ms	67.72ms

表 4: ユーザインタフェース合成時間の測定結果 (単位:milli second)

組み合わせ	GUI 数	合成規則適用時間	表示時間	総時間
(1) Spot Light + Ambient Light	4 → 2	16.70ms	22.11ms	38.81ms
(2) (1) + Color Light	4 → 2	14.38ms	21.76ms	36.14ms
(3) WebCamera + Plasma Display	14 → 3	26.23ms	22.63ms	48.86ms

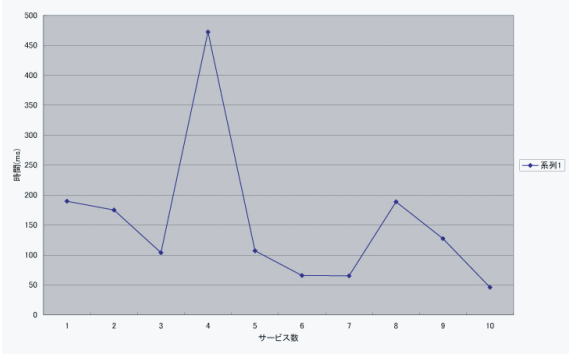


図 11: 平均レンダリング画像生成時間

屋単位でのサービス検索を実現するディレクトリサービスラッパ，実世界空間を 3 次元空間として描画する 3 次元空間作成ツールキットから構成される．vARC を用いることにより，ユーザは専門知識を必要とせずに，協調動作するサービス群を単一のユーザインタフェースを用いて利用可能になる．

8 謝辞

vARC の開発にはプロジェクト管理組織として，株式会社メディアフロント社に尽力していただいた．また，開発にあたって慶應義塾大学徳田研究室の支援をいただいた．この場を借りて深く感謝する．

参考文献

[1] Sirish Chandrasekaran, Samuel Madden, and Mihut Ionescu. Ninja paths: An architecture for composing services over wide area networks. http://ninja.cs.berkeley.edu/.

[2] Jin Nakazawa, Yoshito Tobe, and Hideyuki Tokuda. On dynamic service integration in vna architecture. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. 7, No. E84-A, pp. 1610–1623, July 2001.

[3] Object Management Group. The Common Object Request Broker Architecture, February 1998.

[4] Shankar R. Ponnekanti and Armando Fox. Sword: A developer toolkit for building composite web services. In *The Eleventh International World Wide Web Conference (Web E ngineering Track)*, 2002.

[5] Sony, Matsushita, Philips, Thomson, Hitachi, Toshiba, Sharp, and Grundig. Specification of the Home Audio/Video Interoperability (HAVi) Architecture, May 1998.

[6] Sun Microsystems Inc. *Jini Architecture Specification*, December 2001. http://wwws.sun.com/software/jini/specs/jini1_2.pdf.

[7] Universal Plug and Play Forum. Universal Plug and Play (UPnP), 1999.

[8] 門田昌哉, 松宮健太, 由良淳一, 徳田英幸. サービス横断的な制御を実現するユーザインタフェース合成機構の設計と実装. システムソフトウェアとオペレーティングシステム研究会, 2003.