社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2003－OS－93 (3)
2003／5／8

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

門田 昌哉[2] 松宮 健太[2] 由良 淳一[2] 徳田 英幸[1 2]
[1] 慶應義塾大学 環境情報学部 [2] 慶應義塾大学大学院 政策・メディア研究科

環境に遍在する分散コンポーネントを合成し，サービスを動的に再構成するサービス合成の研究が行われている．サービス合成に関する研究は，分散コンポーネントの発見や通信パスの生成に加え，視覚的に分散コンポーネント間の関係を変更するフロントエンドの研究を内包する．しかし，再構成されたサービスに対応するユーザインタフェースを動的に構築できないため，既存のサービス合成機構はホームネットワーク等のユーザが直接サービスを再構成する環境には適さない問題点がある．本稿では，サービスの再構成に必要な分散コンポーネント間の関係をメタ情報を用いて自動的に求め，再構成されたサービスを直接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 RINCS (userR INterface Composition System) を提案する．RINCS では，分散コンポーネント単位でユーザインタフェースを記述し，再構成されたサービスにおける分散コンポーネント間の関係に従ってユーザインタフェースを合成することにより，ユーザインタフェースの動的合成を実現する．これにより，ユーザは分散コンポーネントのメタ情報を意識することなく，複数のサービスを構成する分散コンポーネントを横断的に制御可能なユーザインタフェースを利用できる．

Design and Implementation of User Interface Composition System for Cross-cutting control of Services

Masaya KADOTA,[2] Kenta MATSUMIYA,[2] Junichi YURA[2] and Hideyuki TOKUDA[1 2]

[1]Faculty of Environmental Information, Keio University
[2]Graduate School of Media and Governance, Keio University

Currently many researchers are studying service composition mechanism, which compose arbitrary omnipresent distributed components in the environment to construnct a logical services dynamically. Although this mechanism covers the research area about dsicovering distributed components, building communication paths, and also covers the research area of front-end for users to manage the relationship between distributed components, it lacks the mechanism for reconstructing the user interface for users to control reconstructed services. Consequently, the existing service composition mechanism are not suited for home network in which users need to reconstruct services directly. In this paper, we propose RINCS: the dynamic user interface composition mechanism which automatically extracts relationships between distributed components, and reconstruct an user interface for users to control them. In this mechanism, user interface descriptions are defined by each distributed components and composed along with extracted relationships. Using this mechanism, users can use the user interface for cross-cutting control of services without being concious of the meta-information of distributed components.

1 はじめに

ユビキタスコンピューティング環境は，ネットワークに接続された情報家電機器，AV 機器，センサ，アクチュエータ，小型携帯端末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分散コンポーネントの集合を**サービス**と呼ぶ．また，異なるサービスの構成要素である分散コンポーネントを再構成し，複数のサービスが提供する機能を協調動作させることによって論理的なサービスを構築することを**サービス合成**と呼ぶ．サービス合成により，協調動作する分散コンポーネント群を構成要素とする論理的なサービスを動的に構築できるため，ユーザの要求に動的に適用したサービスの構築を実現できる．

しかし，ホームネットワーク等のユーザがサービスを直接再構成する環境では，より簡易に分散コンポーネント間の関係を定義し，それに合わせたユーザインタフェースを構築する機構が必要である．現在，多くのサービス合成を実現する機構が提案されているが，既存のサービス合成機構では，個々のサービス合成機構に依存した文法の理解が求められるため，専門知識を持たないユーザが動的にサービスを再構築できない問題点がある．そのため，既存のサービス合成機構の研究は，視覚的に分散コンポーネント間の関係を定義可能なフロントエンドの研究 [11] を内包するが，再構成されたサービスに対するユーザインタフェースを構築できない問題点がある．現状では，再構成されたサービスへのユーザインタフェースは，プログラマによって静的に記述されなければならない制約がある．

本稿では，以上の問題点を解決するため，再構成されたサービスに対するユーザインタフェースを動的に構築する機構 RINCS (useR INterface Composition System) を提案する．RINCS において，ユーザインタフェース記述文書は分散コンポーネント単位で定義される．サービス合成時には，RINCS は再構成されるサービスが含む分散コンポーネント間の関係を求め，その関係に従ってユーザインタフェース記述文書を動的に合成する．たとえば，複数のサービスに共通する分散コンポーネントや入出力データ形式が一致する分散コンポーネント群のユーザインタフェースを合成することにより，ユーザは一つのユーザインタフェースを使用して再構成されたサービスを利用可能になる．このように，異なるサービスに所属する複数の分散コンポーネントを制御することを，本稿ではサービス横断的制御と呼ぶ．

本稿の構成は，第 2 章で RINCS で採用する分散コンポーネントの関係定義モデルの考察を行い，第 3 章でユーザインタフェース合成機構の概要について述べる．第 4 章では，システムの設計および提案するユーザインタフェース記述言語と合成アルゴリズムを述べ，第 5 章においてプロトタイプシステムの実装および基本性能の評価を行い，第 6 章で関連する研究との性質面での比較を行う．第 7 章で本稿をまとめ，今後の研究課題について述べる．

2 既存のサービス合成機構の考察

本章では，想定するホームネットワークと既存のサービス合成機構の概要を述べ，本稿で採用する分散コンポーネント間の関係定義モデルを考察する．

2.1 想定環境と用語定義

本稿では，**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できる．

本稿が対象とする**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，本稿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．また，情報家電機器や AV 機器は消耗品であることや，小型携帯端末は頻繁にネットワークを出入りすることから，ホームネットワーク上で動作するサービスは動的に変化することが想定される．

図 1 に本稿が想定するサービスの例を示す．たとえば，DVD プレーヤ上で動作するサービス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上で動作するサービス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．

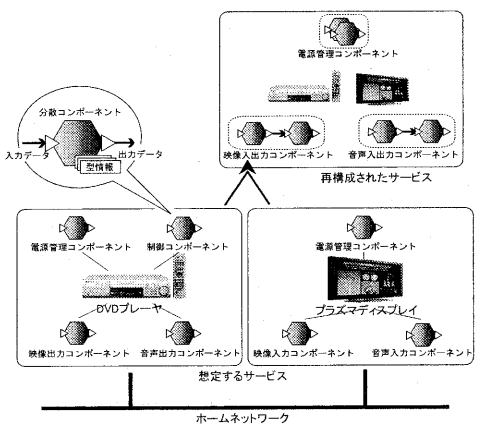

図 1: サービスとサービス合成

2.2 サービス合成機構の概要

サービス合成機構は，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上で動作するサービスの電源管理コンポーネントを集約すると同時に，映像ストリームの入出力および音声ストリームの入出力を接続している．異なるサービスの構成要素であ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．

本稿では，Jini[9] や CORBA[4] 等の分散コンポーネントミドルウェアに加え，VNA[3]，Ninja Path[1]，SWORD[5] 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．

図 2 にサービス合成機構の概念図を示す．図 2(a) に示す白色の六角形が分散コンポーネントごとに定義されたメタ情報を示し，その間の点線が分散コンポーネント間の関係を示す．サービス合成機構は，このように明示的に定義された分散コンポーネント間の関係に従い，分散コンポーネントの発見および通信パスの生成を行う．図 2(b) に示す灰色の六角形が分散コンポーネントの実体を示し，その間の実線が通信パスを示す．

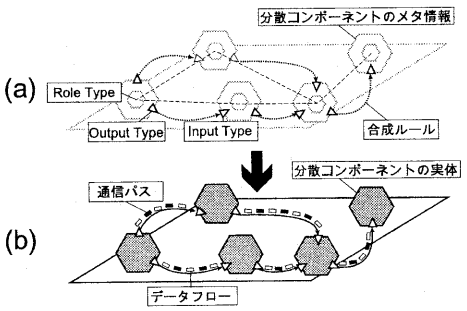

図 2: サービス合成機構

既存の分散コンポーネント間の関係定義モデルでは，個々の分散コンポーネントごとに定義されたメタ情報を用いる．メタ情報としては，分散コンポーネントの**役割の型**や**入出力データ型**を用いることが多い．役割の型は分散コンポーネントが果たすべき役割を規定するもので，サービスが提供すべき機能を標準的に定義する情報である．また，入出力データ型は，MIME 等で規定されるデータ型を用いて，分散コンポーネントの入出力データ型を定義する情報である．次節では，分散コンポーネントの関係定義モデルから，既存のサービス合成機構の分類を行う．

2.3 分散コンポーネント間の関係定義モデルの分類

本稿では，分散コンポーネント間の関係モデルを，**テンプレート主導モデル**と**ルール生成モデル**の 2 つに分類する．以下に個々のモデルの特徴を挙げる．

- **テンプレート主導モデル**
  図 2(a) に示すメタ情報の集合と関係が静的に与えられているモデルをテンプレート主導モデルと呼ぶ．テンプレート主導モデルは，分散コンポーネントの型と関係を定義するが，分散コンポーネントの実体を静的に指定しないため，特定の型を持つ分散コンポーネントが一時的に利用不可能な場合，同様な型を持つ分散コンポーネントを代替できる利点がある．しかし，テンプレートは静的に記述されるため，特定の分散コンポーネントを指定できない点に加え，ユーザの要求に合わせて分散コンポーネント間の関係を動的に定義できない問題点がある．
- **ルール生成モデル**
  図 2(a) に示すメタ情報の集合から，再構成可能な分散コンポーネントの組合せを求め，分散コンポーネント間の点線を求めるモデルをルール生成モデルと呼ぶ．ルール生成モデルは，分散コンポーネント間の関係を動的に生成するため，ユーザが指定す

る任意の分散コンポーネントの集合を構成要素とするサービスを生成可能である．しかし，自動的に関係を求めるため，無意味な関係を生成したり，与えられた分散コンポーネントのメタ情報から関係を生成できない場合がある．これらの問題点を回避するためには，テンプレート主導モデルに対して，分散コンポーネントの型情報をより詳細に定義する必要がある．たとえば，出力データ型に文字列と指定するだけでなく，人名や地名等の文字列のセマンティクスを考慮する必要がある．

2.4 採用するモデルの考察

ホームネットワークは，システム管理者が介在せず，ネットワークを構成するサービスが動的に変化する環境である．以下に，ホームネットワークにおける分散コンポーネント間の関係定義モデルに必要な要件を述べ，本稿で提案する機構で採用するモデルを考察する．

(1) **分散コンポーネントの実体指定**
ホームネットワークにおいてサービスは特定の計算機上で動作するため，サービスが特定の計算機と結び付いて存在する．そのため，ホームネットワークでは，特定のメタ情報に一致する分散コンポーネントを指定するより，特定のサービスに所属する分散コンポーネントの実体を指定する必要がある．

(2) **分散コンポーネント間の関係の可変性**
ホームネットワークを構成するサービスは動的に変化するため，静的に分散コンポーネント間の関係を定義することが難しい．そのため，再構成するために必要な分散コンポーネントの関係は動的に定義可能である必要がある．

(3) **関係定義の簡易性**
ホームネットワークはシステム管理者が介在しない環境であるため，ユーザが直接サービスを再構成する．そのため，分散コンポーネント間の関係を簡易に定義可能である必要がある．

以上のホームネットワークの要件から，本稿で提案する機構では，分散コンポーネント間の関係定義モデルとしてルール生成モデルを採用する．ルール生成モデルを採用することにより，(1) と (2) の要件を満たし，ユーザが任意に指定するサービスを構成する分散コンポーネント群から，自動的に再構成可能な分散コンポーネントの組合せを求められる．

しかし，ルール生成モデルではユーザが特定のサービスを指定する必要があるのに対して，テンプレート主導モデルはあらかじめ分散コンポーネントの組合せを定義するため，ルール生成モデルは (3) 項の簡易性の点で劣る．そのため，再構成するサービスの構成要素を分散コンポーネント単位で指定するのではなく，特定の計算機上で動作するサービスの単位で分散コンポーネント群を指定し，再構成可能な分散コンポーネントの組合わせを自動的に求め，より簡易に分差コンポーネント間の関係を定義可能な機構が求められる．

3 ユーザインタフェース合成機構

本章では，本稿で提案するユーザインタフェース合成機構 RINCS (userR INterface Composition System) の概要と機能要件を述べる．

3.1 概要

RINCS は，既存のサービス合成機構上で動作し，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構築する機構である．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．

再構成されたサービスの構成要素となる分散コンポーネント間の関係定義には，前節で述べたルール生成モデルを利用する．ルール生成モデルによって動的に定義される関係に従い，RINCS は個々の分散コンポーネントごとに定義されたユーザインタフェース記述文書を合成する．再構成されたサービスに対するユーザインタフェースを動的に構築することにより，ユーザは複雑な専門知識を必要とせずに，再構成されたサービスを利用可能になる．

RINCS において，ユーザインタフェースの合成は，(1) 分散コンポーネントの関係定義，(2)GUI ウィジェットの合成と実行コマンドの合成の 2 つのフェーズに分けられる．以下に，RINCS における各フェーズの概要を述べる．

分散コンポーネント間の関係定義

RINCS において，分散コンポーネントの関係は，第 2.3 節で述べたルール生成モデルを用いて，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，既存のサービス合成機構と同様に，役割の型と入出力データ型を用いる．

ルール生成モデルは与えられたメタ情報から特定の条件への一致を求める前向き推論アルゴリズムを用いるため，事後条件となる条件を設定する必要がある．そのため，RINCS では，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．また，これらの分散コンポーネント間の関係を再帰的に行うことにより，より複雑なサービスの再構成も実現できる．

GUI ウィジットの合成と実行コマンドの合成

RINCS 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，RINCS 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．RINCS 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では，定義された分散コンポーネントに対する GUI ウィジェットの合成，および GUI ウィジェットの実行に伴って分散コンポーネントに対して送信される実行コマンドの対応を考慮する必要がある．

3.2 機能要件

前項で述べたユーザインタフェース合成機構を実現するため，RINCS で考慮すべき要件を以下に述べる．

(1) **動的なユーザインタフェースの合成**
ユーザインタフェースの合成は，自動的に定義される分散コンポーネント間の関係に従って行うため，静的にユーザインタフェースを構成するウィジェット間の関係を定義できない．そのため，ユーザインタフェースの合成を動的に実現する必要がある．

(2) **実行コマンドと分散コンポーネントの対応**
ユーザインタフェースの合成により，複数の GUI ウィジェットが合成され，単一の GUI ウィジェットの操作によって複数の分散コンポーネントの制御を実現できる．その際，個々の分散コンポーネントと実行コマンドの対応を考慮する必要がある．

(3) **実装の多様性の考慮**
RINCS では，ユーザインタフェースの実装の多様性を考慮し，特定の分散コンポーネントに対するユーザインタフェースの実装を画一的に定義しない．そのため，異なる型の GUI ウィジェット間の合成を考慮する必要がある．

(4) 実行コマンドの多様性の考慮
ユーザインタフェースの合成により，単一の GUI ウィジェットが複数の分散コンポーネントに対応する．しかし，分散コンポーネントの実装は個々に異なるため，同様な機能を提供する分散コンポーネントであっても，実行に必要な情報が異なる場合がある．そのため，GUI ウィジェットを集約した際の実行コマンドの不整合を考慮する必要がある

(5) GUI ウィジェットの動的再配置
GUI ウィジェットの合成により，ユーザインタフェースのレイアウト情報が変化する可能性がある．そのため，他の GUI ウィジェットの配置から，動的に GUI ウィジェットを再配置する必要がある．

(6) サービス合成機構への非依存性
ホームネットワークにおいて，サービスは多様なサービス合成機構上で動作する．そのため，多様なサービスを制御可能にするため，サービス合成機構非依存に動作する必要がある．

4 ユーザインタフェース記述言語の設計

本章では，RINCS 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，RINCS で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[7] や UPnP[10] 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組み合わせである．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．

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

図 3: CUIL の構成

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．

```
<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

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，RINCS では，実行コマンドの発行に遠隔メソッド呼び出しを用いる

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．

4.5 GUI ウィジェットの記述

CUIL では他の言語と同様に，ユーザからの入力値の型を用いて GUI ウィジェットを抽象的に記述する．図 4 において <mainWidget> および <subWidget> の属性として定義される type が入力値の型である．現状では，CUIL で利用可能な入力値の型は，boolean 型,int 型，double 型，string 型の 4 つである．これらの入力値と GUI ウィジェットの対応表は，個々の端末上に配置された RINCS が持ち，対応は個々の端末上で利用可能な GUI ツールキットによって異なる．たとえば，boolean 型は Button，int 型は List，double 型は Slider，string 型は TextField に対応することが想定される．

5 ユーザインタフェース合成機構の設計

本章では，RINCS の全体の設計と動作概要，およびユーザインタフェースの合成アルゴリズムについて述べる．

5.1 システム構成と動作概要

図 5 に RINCS のシステム構成を示し，各モジュール群の動作概要を述べる．

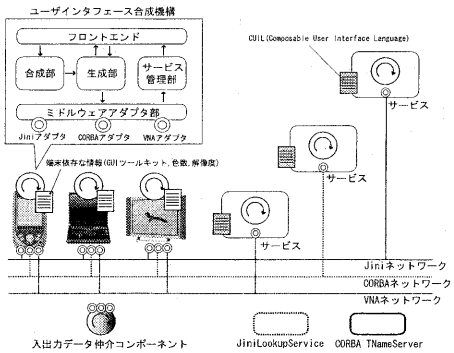

図 5: ユーザインタフェース合成機構の設計

RINCS は，ユーザが利用する端末上に配置され，サービスから動的にユーザインタフェースの記述をダウンロードする．本稿では，PC や PDA に接続されたプラズマディスプレイ，ラップトップ PC 等の画面表示能力を持った端末をサービス制御用端末として想定する．端末ごとに RINCS を配置し，端末の解像度や利用可能な GUI ツールキット等の端末依存な情報に合わせてユーザインタフェースを生成することにより，多様なサービスを制御可能なユーザインタフェースを構築する．ユーザインタフェースの合成は，自動的に定義される分散コンポーネント間の関係に従い，分散コンポーネント毎に定義されたユーザインタフェース記述文書を合成することによって実現する．以下に，RINCS を構成するモジュール群の動作概要を述べる．

5.2 モジュール構成

- **フロントエンド**
  フロントエンドは，ユーザに対してサービスの選択，およびユーザインタフェースの操作を行うためのインタフェースを提供するモジュールである．ユーザは，フロントエンド上に表示される利用可能なサービスの情報から，再構成するサービス群を選択する．サービスを選択する際には，入出力データの流れやユーザインタフェースのレイアウトの優先度をシステムが判別するため，サービスに優先度を設定する必要がある．
- **生成部**
  生成部は，サービスからダウンロードしたユーザインタフェース記述文書を，利用可能な GUI ツールキット，色数，解像度等のユーザが持つ端末依存の情報に合わせてユーザインタフェースを構築するモジュールである．ユーザインタフェースの生成に必要な処理として，ユーザインタフェース記述文書の構文解析，抽象的に記述された GUI ウィジェットと具体的な GUI ウィジェットへのマッピング，ユーザインタフェースの構築がある．
- **合成部**
  合成部は，第 3.1 節で述べた分散コンポーネント間の関係定義，およびユーザインタフェースの合成を行うモジュールである．ユーザインタフェースの合成では，GUI ウィジェットの合成および分散コンポーネントに対して発行される実行コマンドの合成を行う．
- **サービス管理部**
  サービス管理部は，利用可能なサービスへの参照を保持するモジュールである．RINCS はサービス合成機構非依存にサービスを制御するため，利用可能なサービスの情報は各サービス合成機構が提供する情報を個々の端末上で管理する必要がある．
- **ミドルウェアアダプタ部**
  サービス合成機構に対応したアダプタを持つことにより，RINCS はサービス合成機構非依存なサービスの制御を実現する．Jini が提供する様なイベント駆動型のディレクトリサービスでは，プッシュされたサービスの状態の変化イベントを用いて利用可能なサービスの一覧を更新する．VNA のようにディレクトリサービスを持たないアーキテクチャでは，アダプタが VNA ネットワークのサービスとして振る舞うことにより，サービスの一覧を更新する．

5.3 ユーザインタフェース合成アルゴリズム

本節では，前節で述べた合成部におけるユーザインタフェース合成アルゴリズムについて述べる．合成アルゴリズムとして，RINCS では UI 集約合成規則と UI 接続合成規則を用いる．これらのユーザインタフェース合成アルゴリズムは同時に適用される．

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 6(a) で

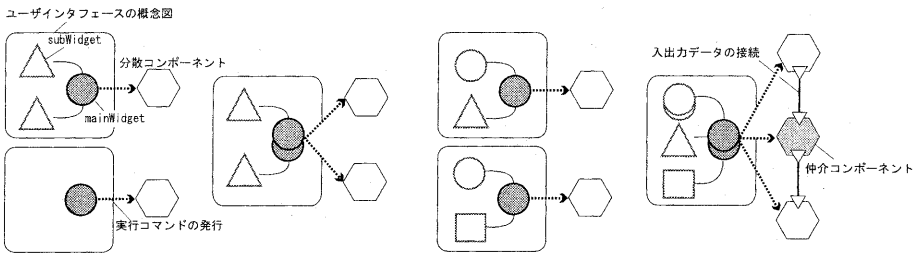

図 6: GUI ウィジェットの合成と実行コマンドの合成

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，集約合成規則と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，RINCS が GUI ウィジェットを変更し，GUI ウィジェット間のラベルを矢印で結ぶ．

5.4 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 RINCS 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，RINCS 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．図 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 ユーザインタフェース合成機構の実装と基本性能評価

本章では，RINCS のプロトタイプシステムの実装および基本性能の評価を行う．

6.1 実装環境

表 1 にプロトタイプシステムの実装環境を挙げる．プロトタイプシステムの実装には，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して，計算機 (1) を RINCS が動作する端末として用い，計算機 (2) を Jini のディレクトリサービスが動作するサーバとして用いた．これらの計算機群は 100Mbps のイーサネットで接続される．

表 1: 実装環境

	計算機 (1)	計算機 (2)
CPU	Mobile PentiumIII 750MHz	Mobile PentiumIV 1.6GHz
OS	VineLinux2.6	WindowsXP
メモリ	256MB	512MB

6.2 フロントエンドの実装

プロトタイプシステムでは，ユーザが制御対象となるサービスを選択するためのフロントエンドの一例として，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CUICs (Card-oriented User Interface Composition sytem) の実装を行った．CUICs 上に表示されるカードが個々のサービスを制御するユーザインタフェースである．カードのメタファを用いる利点は，RINCS ではサービスに優先度をつける必要があるため，カードを重ねるインタラクションによってサービス間の関係を明示的に示すことが可能な点にある．これにより，ユーザはカードを重ねることにより，簡易にユーザインタフェースを合成できる．図 7 に CUICs の実装画面を示す．図では，3 種類のライトの合成例と，デジタルビデオカメラ，エアコンのユーザインタフェース生成例を示している．

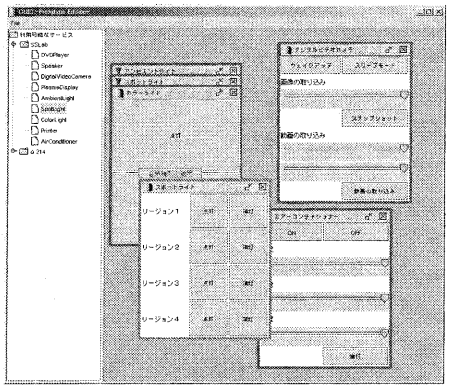

図 7: CUICs とユーザインタフェース合成例

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

6.3 アダプタの実装

下位サービス合成機構に対するアダプタとして，プロトタイプシステムでは Jini に対するアダプタの実装を行った．CORBA や VNA 等の他のサービス合成機構に対するアダプタの実装は現在開発中である．利用可能なサービスの一覧はユーザが持つ端末上に集約され，実行コマンドは端末を中心にして発行される．図 7 の左側のパネルが集約されたサービスの情報を示している．

6.4 基本性能の評価

本節では，RINCS 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．

ユーザインタフェース生成時間

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．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，今後の機能拡張や GUI 数の増加に比例して処理時間が増加することが予測される．

ユーザインタフェース合成時間

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．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と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

図 8 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．RINCS 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．

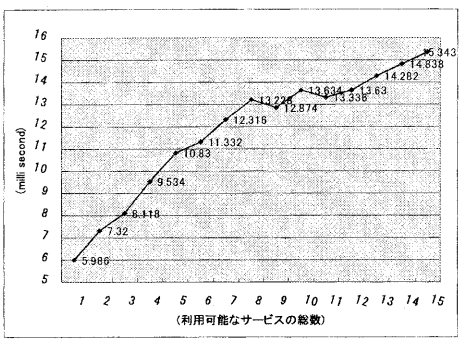

図 8: 利用可能なサービスの一覧取得時間

7 関連研究

本節では，RINCS に関連する機構について述べ，定性的な評価を行う．

7.1 Document-based Framework

Document-based Framework[2] は，ISL (Interface Specification Language) を用いてユーザインタフェースを記述し，制御用端末非依存にユーザインタフェースを生成する機構である．各制御用端末上に ISL のインタープリタが配置され，ISL 内で記述された抽象的な GUI ウィジェットの型と，特定の GUI ツールキット依存な GUI ウィジェット間の一致を行う．GUI ツールキットとしては，Tcl/Tk および独自の GUI ツールキットである MASH ツールキットを用いている．

表 3: ユーザインタフェース生成時間の測定結果 (単位:milli 秒)

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 秒)

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

Document-based Framework の特徴として，GUI ウィジェットの集合自体をあらかじめ定義するため，プログラミング言語で記述されたユーザインタフェースと動的に生成されるユーザインタフェース間の入れ替えが可能であることが挙げられる．これにより，プログラミング言語で記述された高品質なユーザインタフェースをプラグイン可能になる．しかし，特定の機能に対するユーザインタフェースを静的に記述するため，ユーザインタフェースの実装の多様性を考慮できない問題点がある．RINCS では，ユーザインタフェースの実装の多様性を考慮し，GUI ウィジェット単位での組み換えを実現している．

7.2 ICrafter

ICrafter[6] は，Document-based Framework と同様に機能単位でユーザインタフェースを記述し，制御用端末非依存にユーザインタフェースを生成する機構である．ICrafter では，SDL(Service Description Language) を用いてユーザインタフェースを記述する．SDL は Java 言語で記述されたサービスのバイトコードからリフレクション API[8] を用いて動的に生成される．

ICrafter の特徴の 1 つとして，独自のイベントモデルに基づくフレームワーク上での動作が挙げられる．単一のミドルウェア上で動作することにより，より信頼性のある制御機構やミドルウェア独自の機能を利用した高機能な制御機構を実現できる反面，多様なミドルウェア上で動作できないため，汎用性の点で問題がある．RINCS では，制御用端末でサービスの情報を集約するオーバーヘッドがあるが，多様なミドルウェア上で動作可能である．

また，ICrafter では，RINCS と同様にユーザインタフェースの合成機構を提供するが，分散コンポーネント間の関係定義にテンプレート主導型のモデルを利用するため，静的に定義されたユーザインタフェースしか生成できない問題がある．これに対して，RINCS ではルール生成モデルに基づいて，ユーザの要求を動的に反映したユーザインタフェースを構築可能である．

8 まとめ

本稿では，既存のサービス合成機構を用いてサービスを再構成した際に，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構築できない問題を解決し，再構成されたサービスを直接制御可能なユーザインタフェースを動的に合成する機構 RINCS の設計および実装について述べた．RINCS におけるユーザインタフェースの合成は，再構成されるサービスの構成要素となる分散コンポーネント間の関係をメタ情報を用いて自動的に求め，求められた関係に従ってユーザインタフェース記述文書を動的に合成することで実現される．本稿で提案するユーザインタフェース記述文書 CUIL において，ユーザインタフェースは分散コンポーネント単位で記述される．RINCS において，ユーザインタフェースは分散コンポーネント単位で記述される必要があるため，RINCS では独自のユーザインタフェース記述言語を設計した．

また，RINCS を利用するためのフロントエンドの一例として，プロトタイプシステムでは，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 CUICs を開発した．CUICs を用いることにより，ユーザは分散コンポーネントのメタ情報や，ユーザインタフェース合成機構を意識すること無く，カードとして表示されるユーザインタフェースを重ねることによってユーザインタフェースを合成できる．

RINCS を用いることにより，複数のサービスの電源を一括制御するユーザインタフェースや，デジタルビデオカメラでキャプチャした画像を複数のプリンタやディスプレイに表示させるユーザインタフェースを動的に構築できる．これにより，ユーザは複数のサービスを選択するのみで，複数のサービスを構成する分散コンポーネントを横断的に制御可能なユーザインタフェースを利用可能になる．

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] Todd D. Hodes and Randy H. Katz. A document-based framework for internet application control. In *2nd USENIX Symposium on Internet Technologies (USITS'99)*, Oct 1999.

[3] 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.

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

[5] 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.

[6] Shankar R. Ponnekanti, Brian Lee, Armando Fox, Pat Hanrahan, and Terry Winograd. Icrafter: A service framework for ubiquitous computing environments. In *Ubicomp 2001: Atlanta, GA, USA*, 56-75, 2001.

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

[8] Sun Microsystems Inc. Java core reflection api and specification, 1996.

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

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

[11] 岩井将行, 中澤仁, 西尾信彦, 徳田英幸. 分散コンポーネントによる即興的アプリケーション構成機構の実現. 情報処理学会論文誌, Vol. 43, No. 6, pp. 1664–1676, 6 2002.