「情報処理学会論文誌」 第46巻 第2号別刷 平成17年2月発行

サービス指向アーキテクチャを用いたネットワーク家電連携サービスの開発

井垣 宏 中村匡秀
玉田春昭 松本健一

社団法人 情報処理学会

サービス指向アーキテクチャを用いたネットワーク家電連携サービスの開発

井垣 宏[†] 中村匡秀[†]
玉田春昭[†] 松本健一[†]

ホームネットワークに接続された家電機器を連携制御しユーザの快適性・利便性を高める，家電機器連携サービスの一実現手法を提案する．既存の機器連携システムでは，ホームサーバが家電機器を中央集権的に制御する方式が一般的である．しかしながら，家電機器の多様化・高性能化により，ホームサーバへの負荷集中や信頼性・相互接続性の低下が問題となる．そこで我々は，サービス指向アーキテクチャ（SOA）に基づいた新たな連携サービス実現方式を提案する．提案手法では，各機器が自己の機能をサービスとしてネットワークに公開し，他の機器が公開するサービスを互いに実行することで連携を行う．これにより，各機器はサービスを介して疎結合され，ホームサーバも不要となる．したがって，より柔軟で障害や負荷に強い連携サービスの実現が可能となる．本稿では，SOA に基づいて連携サービスを設計・実装するための枠組みを示し，Web サービスを用いたプロトタイプ開発を行う．また，連携サービスの評価尺度として，信頼性，負荷，結合度を定義し，従来システムと提案システムの定量的な比較評価を行う．

Implementing Integrated Services of Networked Home Appliances Using Service-oriented Architecture

Hiroshi Igaki,[†] Masahide Nakamura,[†] Haruaki Tamada[†] and Ken-ichi Matsumoto[†]

This paper presents a method to implement the integrated services of networked home electric appliances, which provide more convenient and comfortable living for home users. The conventional methods generally employ a home server to achieve the integrated services. The server controls all the networked appliances in a centralized manner. However, as the number of sophisticated appliances increases, the centralized server suffers from the concentration of load, as well as a decline in the reliability and interoperability. To cope with this problem, we adopt the service-oriented architecture (SOA) for the implementation of the integrated services. In the proposed framework, each appliance exports own features as a service. The appliances autonomously execute the exported services one another to achieve the integrated services. Thus, the appliances are loosely coupled via the exported services, without the centralized home server. This enables more flexible, balanced and reliable integrated services. In this paper, we present a framework to design and implement the integrated services based on the SOA, and illustrate a prototype system developed with Web services. We also define three kinds of metrics (i.e., reliability, workload, and coupling) and conduct a comparative evaluation between the proposed and the previous systems.

1. はじめに

プロセッサの小型化・高機能化，ネットワークの性能・信頼性の向上にともない，様々な家電機器をネットワークに接続するための基盤技術が注目を集めている[4),9),16)]．テレビ，エアコン，照明，AV 機器などの家電機器を家庭内のネットワークに接続することで，機器の連携や宅外からの操作が可能となる．このようなシステムは一般にホームネットワークシステム（HNS）と呼ばれ，近年，研究開発が進められている．また，いくつかの製品がすでに商品化されている[10),12),14),18)]．

HNS アプリケーションの 1 つとして，複数の家電機器を連携動作させ，ユーザの日常生活の利便性・快適性を高める家電機器連携サービス（以降，連携サービスと呼ぶ）がある[8),15)]．以下に連携サービスの例をあげる：

帰宅サービス： ユーザが帰宅すると，照明および空

† 奈良先端科学技術大学院大学
Nara Institute of Science and Technology

調が起動し，適切な明るさ・室温に設定される．

DVD シアターサービス： DVD プレーヤを起動すると，照明が暗くなり，スピーカが 5.1ch に設定され音量が調節される．

連携サービスを実現するために，従来の HNS では，中央に高性能な制御サーバ（**ホームサーバ**と呼ぶ）を置き，ホームサーバが中央集権的にすべての家電機器を制御する方式が一般的である．この方式を**サーバ中心型アーキテクチャ**（Server Centralized Architecture, SCA）と呼ぶ．機器の連携制御は，サーバから各機器へ制御コマンドを送信することで実現できる[12]．連携サービスにおけるインテリジェントな処理をすべてサーバで行うため，アーキテクチャが非常にシンプルとなる．

しかしながら，今後家電機器の多様化や高性能化が進むにつれて，SCA に基づく連携サービスでは次のような問題が顕在化すると考えられる．

信頼性・負荷集中： すべての機器がサーバによって制御されるため，サーバに障害が発生するとすべての連携サービスが利用不能になってしまう．また，接続される機器が増えるたびに，サーバへの負荷が増大する．

機能拡張性： 接続される家電機器が高性能化しても，ホームサーバが対応していない限り，サービス機能として利用できない．この制限により，連携サービスを柔軟に拡張しにくい．

相互接続性： ホームサーバは連携する家電機器すべての下位プロトコルを意識する必要があり，ミドルウェアの実装が複雑になる．また，サーバと家電機器とが密に結合しやすく，機器やプロトコルのバージョンアップにともなって相互接続に関する問題を生みやすい．

これらの問題点に対処するため，本稿では，**サービス指向アーキテクチャ**[26]（Service oriented Architecture, SOA と呼ぶ）に基づいた新たな連携サービス実現方式を提案する．SOA とは，厳密に定義されたインタフェースを持つ自律分散コンポーネントを，標準的な通信手段によって疎結合し，統合サービスとして提供するためのシステムアーキテクチャである．

提案手法では，家電機器の構成をサービス層とデバイス層の 2 つに分ける．各機器は，サービス層において，機器制御に必要なインタフェースを（基本）サービスとしてネットワークに公開し，サービスが実行されると，対応するデバイスへ機器固有の通信手段で制御コマンドを送る．また，各機器は，サービス層において標準的な通信手段により他の機器のサービスを実行する．すなわち，各機器はサービス層において疎結合され，中央集権的なホームサーバが不要となる．したがって，より柔軟で障害や負荷に強い連携サービスの実現が可能となる．

本稿ではまず，具体的な連携サービスのシナリオをあげ，SOA に基づいた連携サービスの設計を行う．次に，この設計に基づいた連携サービスの具体的な実装の枠組みについて考察し，**Web サービス**[5] を用いた実装例を示す．さらに，グラフに基づいた連携サービスの評価手法を提案し，信頼性，負荷，結合度の 3 つの観点から，新アーキテクチャ（SOA）と既存アーキテクチャ（SCA）の違いを定量的に分析する．

2. 準　　備

2.1 サービス指向アーキテクチャと連携サービス

サービス指向アーキテクチャ（**SOA**）は，ネットワーク上に分散したシステムを，標準的な通信手段によって疎結合するためのソフトウェアアーキテクチャである[26]．各システムは自己が提供する機能をサービス（一連のタスクの集合．オブジェクトよりも粒度が高い）という単位でネットワークに公開する．サービスの内部ロジックや実装は各システム内で自己完結しており，システムはサービスへのインタフェースのみを，厳密に型定義された**公開メソッド**としてネットワークに公開する．

サービスの利用者は，公開メソッドを遠隔実行することで，希望するサービス結果を得る．この遠隔実行は標準的な通信手段を用いて行われる．また，いったん公開されたメソッドの型定義は基本的に変更が許されない．したがって，サービス利用者は，サービスの内部ロジックの変更や実装プラットフォームの影響を受けない．このように，サービスの利用者と提供者間の疎な結合が実現される．なお，SOA の代表的な実現手段として，Web サービス[5),23)] が標準化されている．

図 1 は SOA の例である．サービス利用者はクライアントを通じてサービス A の公開メソッドを呼び出している．サービス A の内部は，密結合されたオブジェクトにより実装されており，その内部で別のサービス B の公開メソッドを呼び出して利用している．この例では，サービス利用者は，サービス A とサービス B を統合したサービス（太線の楕円で示す）を利用していることになる．

2.2 想定する家電機器

本稿では HNS に収容する家電機器として，以下の条件を満たすものを想定する．

条件 C1： 各家電機器は，ソフトウェアによって制

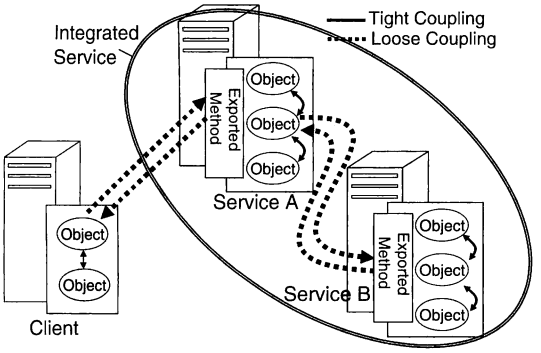


図 1 サービス指向アーキテクチャ
Fig. 1 Service-oriented architecture.

御可能なインタフェース（API など）を備える．

条件 C2： 各家電機器は，アプリケーションソフトウェア（サーバおよび機器制御アプリケーション）を格納するための記憶領域，アプリケーションを処理するプロセッサ，ネットワークインタフェースを備える．

これらの条件は，非現実的な仮定ではなく，次世代の家電機器が標準的に備えることが望ましい機能と考えられる．条件 C1 については，すでにいくつかの標準化が進んでおり，機器ごとのオブジェクトの詳細なテンプレートが規定されているものもある[3),4)]．条件 C2 については，プロセッサやメモリの小型化・低価格化によって，これらを家電機器に組み込むことが容易になってきていることに裏付けられる．実際に，Web サーバを内包し PC から Web ブラウザを用いて設定や制御が可能な製品も存在する[17),20)]．

2.3 連携サービスのシナリオ

以降の議論における理解を深めるため，連携サービスのシナリオ例を導入する．まず，HNS を構成する要素として，前節の条件 C1，C2 を満たした 9 種類の家電機器（DVD プレーヤ，TV，スピーカ，照明，照度計，ドア，電話，エアコン，温度計）を想定する．さらに，これら 9 種類の家電機器がそれぞれ 1 つずつ，すべて同じ部屋に収容されているものと仮定して，以降の議論を進める☆．

以下に，連携サービスとして実現する 8 つのサービスシナリオ例（以降では SS_i $(1 \leq i \leq 8)$ と書くことにする）を示す．これらのシナリオは実際に商品化されている連携サービスを参考に決定した[8),18)]．

SS_1： ユーザが照明の電源を入れると，照度計が示す照度をもとに明るさを調節する．

SS_2： ユーザがドアを開けて中に入ると，照明が自動的に点く（照度計も利用する）．

SS_3： ユーザが DVD プレーヤの電源を入れると，照明が暗くなりテレビとスピーカが DVD モードで起動する．

SS_4： ユーザがテレビの電源を入れるとスピーカも連動して電源が入る．

SS_5： ユーザに電話がかかってくると，自動的にスピーカのボリュームが下がる．

SS_6： ユーザがエアコンの電源を入れると，温度計の示す温度をもとに室温が調節される．

SS_7： ユーザがドアを開けて中に入ると，エアコンが起動する（温度計も利用する）．

SS_8： ユーザの外出時や就寝時にすべての機器の電源を落とし，ドアの鍵をかける．

3. SOA に基づく連携サービスの設計

3.1 キーアイデア

本研究のキーアイデアは，HNS における連携サービスに SOA を適用し，従来の HNS では困難であった以下の 2 つを達成することである．

(A) 家電機器間の標準的な通信・疎結合

各機器の機能を公開メソッドとしてネットワークに公開し，SOA における標準的な方法でアクセス可能にする．これにより，機器間が疎結合され，相互接続性や機器拡張性の向上がより期待できる．SOA を用いてシステム間の疎結合を実現するアプローチ自体は一般的であるが，主にビジネスプロセスを対象とするものである[2)]．これに対し，本研究では新たに HNS に対して SOA を適用し，それにともなう具体的な機器の構成法・実装方法を提案する．

(B) 中央サーバを利用しない自律連携

SOA の適用により，各機器は特別なサーバを介することなく，他の任意の機器の公開メソッドに直接アクセス可能になる．このことに着目し，従来ホームサーバがすべて担ってきた家電機器の連携部分を，機器側で自律分散的に行うための枠組みを新たに提案する．具体的には，機器の公開メソッドが実行されると，その機器自身が次に実行すべき他の機器の公開メソッドを決定して，遠隔実行する方式を提案する．

3.2 機器の構成

前節で述べたキーアイデア (A) および (B) を実現するためには，各機器にそれぞれ以下の仕組みが必要

☆ 同種の家電機器が複数存在する場合には，1 つずつ別個の機器と見なして，同様の議論を進める．たとえば，宅内に 4 つの照明機器がある場合は，照明 1，照明 2，照明 3，照明 4 という 4 つの独立した機器と考える．

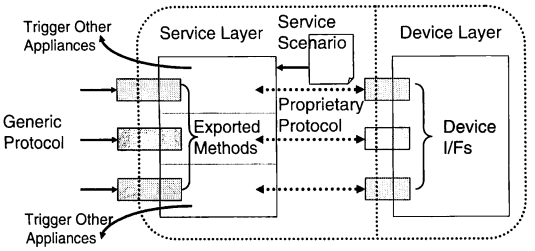


図 2 単一家電機器の構成
Fig. 2 Architecture of each home appliance.

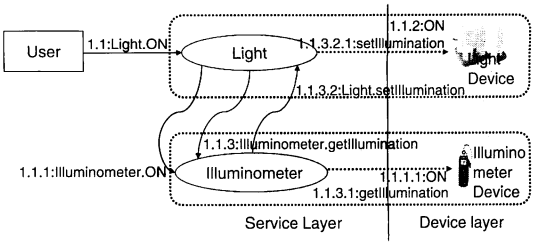


図 3 サービスシナリオ (SS_1) の設計例
Fig. 3 An example of the service scenario design (SS_1).

となる.

(1) 機器依存部を吸収し，ネットワークに対して標準的な通信手段で機能を公開・提供するための仕組み（自己機能提供）

(2) サービスシナリオに応じて，適切に他の機器の機能を制御する仕組み（他者機能利用）

これらの具体化のために，本研究では，各家電機器をデバイス層とサービス層の 2 つの部分で構成する．図 2 に詳細を示す．

まずデバイス層は，機器のハードウェア（含ミドルウェア）部分を指す．条件 C1 によって，機器は制御インタフェースを通して，ソフトウェアから制御可能である．ただし，この制御方法はその機器が準拠する規格により異なる（例：センサ・照明類は ECHONET，デジタル家電は IEEE1394，UPnP など）[3),22)]．

一方，サービス層は，機器の制御インタフェースをラップし，サービスとしてネットワークに公開する．これは，SOA を用いた連携サービスの実現のため，本研究で新たに提案・実装する部分であり，条件 C2 に基づいた家電機器上に，ソフトウェアアプリケーションとして構築する．

具体的には，機器のインタフェース（照明であれば ON，OFF，照度調節など）のそれぞれを，1 つのメソッドでラップし，機器の規格に依存しない形でネットワークに公開する．これには，Web サービス (SOAP/XML，WSDL) など SOA における標準的なサービス公開手段[1),23)] を用いる．こうして，家電機器のすべてのインタフェースは，公開メソッドの集合 (=サービス) として公開される（上記 (1) の実現）．さらに，公開メソッドには，サービスシナリオに応じて，自律的に他の機器の公開メソッドをトリガする仕組みを持たせ，サービス層における複数機器の連携を実現する（上記 (2) の実現）．具体的な実装法については，次章で述べる．

例として，提案する機器構成を有する照明 (Light) および照度計 (Illuminometer) を考え，2.3 節で述べた SS_1 を SOA に基づいて設計してみる．図 3 に設計例を示す．図中，各家電機器のサービス層を楕円，デバイス層をアイコンで示している．サービス A からサービス B へ描かれるラベル付き実線矢印は，サービス A がサービス B の公開メソッド（メソッド名をラベルに記述）を利用（実行）することを示す．点線矢印は，サービス層からデバイス層への制御コマンドを表す．また各メソッドは，UML のコラボレーション図の記法[6)] に従って，階層的な順序付けがなされている．

この例では，ユーザが Light サービスの公開メソッド Light.ON を実行すると，Light サービスはメソッド内部から Illuminometer サービスの ON を実行する．次に Light デバイスを ON にし，Illuminometer デバイスの公開メソッドである getIllumination を呼び出す．Illuminometer サービスは，Illuminometer デバイスに対して，電源 ON，照度取得の制御を順に行い，Light サービスに対して照度設定を行う．最後に，Light サービスは，Illuminometer から得た現在照度に応じて，Light デバイスの setIllumination を呼び出し，最適な照度に照度調節を行う．

このように，家電機器をサービス層において自律的に連携させることで，中央集権的なサーバを用いずに，連携サービスを実現できる．

3.3 サービス連携グラフ

図 3 に示したように，各サービスのシナリオは，ユーザ，サービス，デバイス，ホームサーバといった HNS 上の構成要素と，要素間の機能の利用関係によって性質づけられるため，ここで連携サービスのグラフ表現を導入する．

ラベル付き有向グラフ G は $G=(N, L, E)$ と定義される．ここで，N はノードの集合，L はラベルの集合，$E \subseteq N \times L \times N$ はラベル付き有向辺の集合を表す．ここで，あるサービスシナリオ s に対して，$G_s=(N, L, E)$ が以下の条件を満たすとき G_s を s のサービス連携グラフと呼び，$SIG(s)$ と記述する．

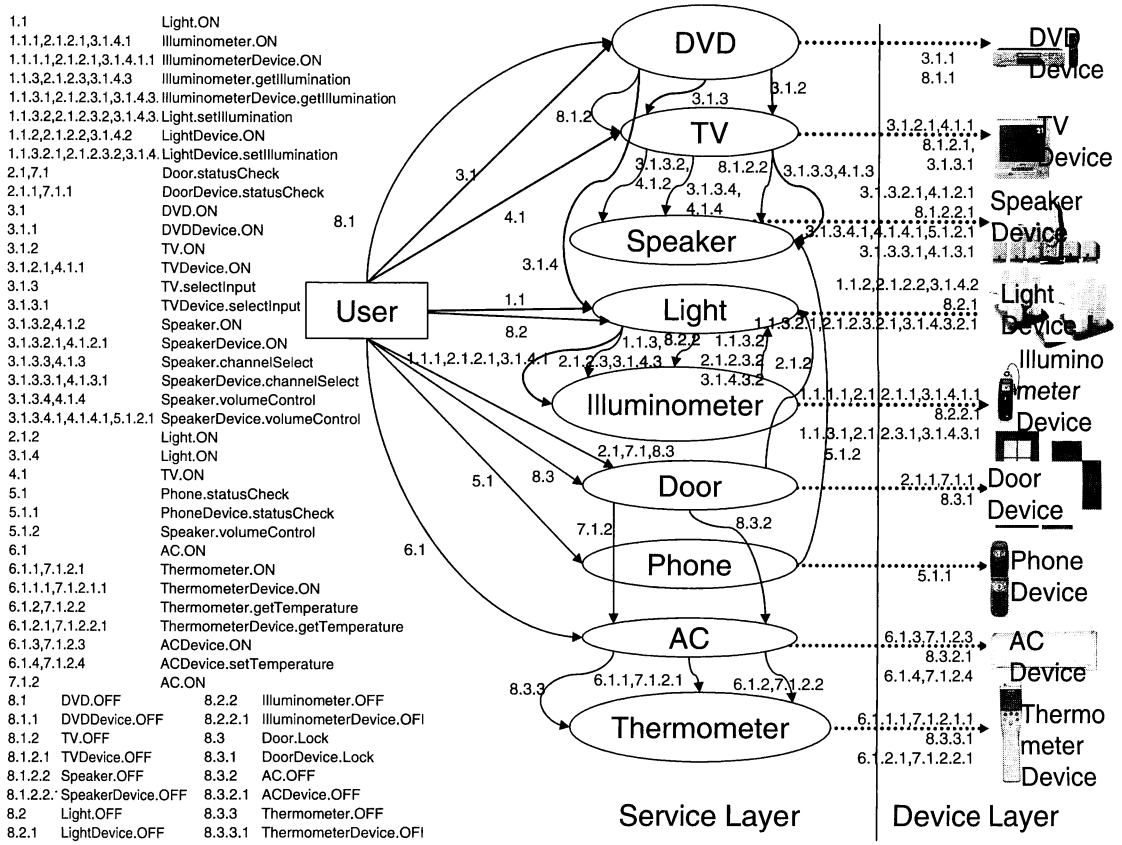


図 4 SOA に基づいて作成した連携サービス
Fig. 4 Integrated service based on SOA.

(1) N は HNS に現れるすべての構成要素の集合
(2) L は HNS において実行されるすべての公開メソッド（または制御インタフェース）の集合
(3) s において，p が q のメソッド m を実行するとき，$(p, m, q) \in E$

さらに，この定義を複数のサービスシナリオに対して拡張する．今，k 個のサービスシナリオ $s_1, s_2, \ldots, s_k$ のそれぞれについて，$SIG(s_i) = (N_i, L_i, E_i)$ が与えられたとき，$s_1, \ldots, s_k$ を同時に含むサービス連携グラフを $SIG(\{s_1, \ldots, s_k\}) = (\cup_i N_i, \cup_i L_i, \cup_i E_i)$ と定義する．さらに，$s_1, s_2, \ldots, s_n$ が与えられたすべてのサービスシナリオである場合，$SIG(\{s_1, s_2, \ldots, s_n\})$ を**全サービス連携グラフ**（$FSIG$）と呼ぶ．定義より，すべての SIG は $FSIG$ の部分グラフとなる．

たとえば，図 3 において，楕円，アイコンをノード，矢印を有向辺と見なしたグラフは $SIG(SS_1)$ である．

3.4 サービス指向アーキテクチャ（SOA）に基づく連携サービスの設計

2.3 節で例示した 8 つの連携サービスシナリオを SOA に基づいて設計する．

図 4 に，全シナリオの設計例を含む，全サービス連携グラフ $FSIG(= SIG(SS_1, SS_2, \ldots, SS_8))$ を示す．図中，ラベル番号に対応する実際のメソッド名を図中左に記述する．ラベル番号の先頭の数字 i は，実行される SS の番号を表す．下位の数字は，SSi における各メソッドの実行順序を階層的に表している．紙面の制限上，同じメソッド名を持つ有効辺を 1 つの矢印で表している．

例として，SS_4 を考える．SS_4 は "4" で始まるラベルを持つ有向辺に示される手順で実行される．ユーザが TV の電源を入れる（TV.on）と，TV サービスが自律的に Speaker サービスと連携し，音量設定やチャネル設定を行う．また，図 4 の $FSIG$ は，図 3 で説明した $SIG(SS_1)$ を部分グラフとして含んでいることも分かる．

SS_3 では，SS_1 と SS_4 のシナリオを再利用できる．ユーザが DVD.ON を呼び出し，DVD サービスが Light サービスと TV サービスの ON メソッドなどを呼び出す．次に，Light，TV サービスはそれぞれ SS_1 と SS_4 を実行することで，全体として SS_3 を実現する．

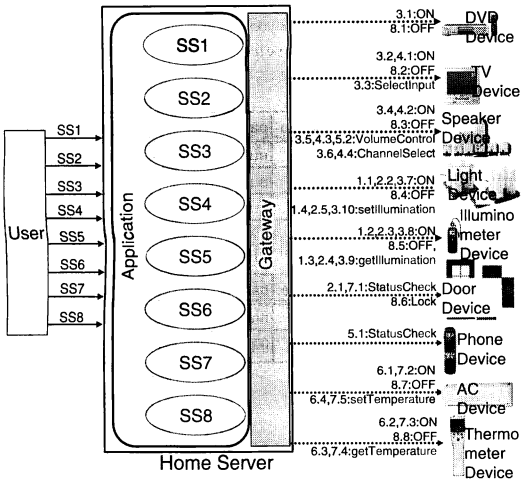


図 5 SCA に基づいて作成した HNS
Fig. 5 HNS based on SCA.

このように，SOA を用いて自律分散的に連携サービスを実現する新たな HNS を，以降では **HNS-SOA** と呼ぶことにする．

3.5 サーバ中心型アーキテクチャ（SCA）に基づく連携サービスの設計

比較のため，2.3 節のサービスシナリオを，従来のアーキテクチャである SCA に基づいて設計してみる．近年商品化されている HNS では，家庭内のホームサーバから，専用のアプリケーションおよびプロトコルを用いて，複数の家電機器に制御コマンドを送る方式を採用している[8),12)]．なお，SOA との定量的な比較評価は，5 章で述べる．

SCA に基づいて連携サービスを実現する場合，各機器にはサービス層は必要でなく，サーバは機器が提供する制御インタフェースに対して直接通信を行う．したがって，ネットワーク規格の異なる家電機器を連携制御する場合には，サーバ内でプロトコル変換を行うためのゲートウェイ処理が必要となる．そのため，サーバの実装が複雑となり，機器の相互接続性を完全に保証することが難しい．

図 5 に設計例を示す．この例では，SS_1～SS_8 の各サービスシナリオは，ホームサーバ内で密に結合したアプリケーションオブジェクトによって実現されている．各オブジェクトは，ゲートウェイ部を通して，連携サービスに必要な機器に対して制御コマンドを送受信する．たとえば，ユーザがホームサーバに対して，SS_3 の開始指示を与えると，サーバは SS_3 のオブジェクトを実行し，DVD プレーヤ，TV，スピーカ，照明の連携制御を行う．

このように，ホームサーバが中央集権的に連携サービスを実現する従来の HNS を，以降では **HNS-SCA** と呼ぶこととする．

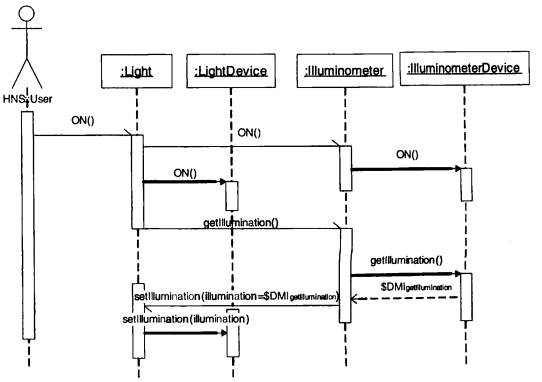


図 6 シーケンス図（SS_1）
Fig. 6 Sequence chart (SS_1).

4. 実　　装

4.1 サービス層実装の枠組み

前章で述べたように，HNS-SOA では，各家電機器がサービス層において互いの公開メソッドを実行し，サービスシナリオを実現する．このような機器連携を行うためのサービス層の実装方式について述べる．

サービスシナリオ s のサービス連携グラフ $SIG(s)$ は，その定義から UML のコラボレーション図[6)] と等価であり，シーケンス図で表現可能である．たとえば，図 3 の SS_1 は，図 **6** のシーケンス図で表される．この例から分かるように，サービス層における連携のためには，各公開メソッドの内部で，以下の 2 種類のメソッド呼び出し（Method Invocation）を実装する必要がある．

DMI（Device Method Invocation）： サービス層が対応するデバイス層へ制御コマンドを送る処理を指す．提案する機器構成（図 2 参照）により，1 公開メソッドあたり 1 つの DMI が存在し，実行するサービスシナリオにかかわらず，固定的に実行される．

SMI（Service Method Invocation）： 他のサービスの公開メソッドに対する呼び出しを指す．DMI の前後のタイミングで行われる．実行するサービスシナリオよって，異なるサービス・公開メソッドが実行される．

例として，図 6 を考える．図中，DMI を太線矢印，SMI を細線矢印で表している．たとえば，Light サービスの公開メソッドである `Light.ON` では，1 つの DMI - `LightDevice.ON` と，2 つの SMI - `Illuminometer.ON`，`Illuminometer.getIllumination` が実行され

表 1 Light サービスおよび Illuminometer サービスの SMI 定義ファイル (SS_1)
Table 1 SMI definition file of light service and Illuminometer service.

(a) Light service (http://light.myhome.net/service.jws)

Context	SSID	ServiceURI	Pre/Post	methodName	paramName	paramType	paramValue
ON()	1	http://illuminometer.home.net/service.jws	pre	ON	null	null	null
	1	http://illuminometer.home.net/service.jws	post	getIllumination	null	null	null
setIllumination()	1	null	null	null	null	null	null

(b) Illuminometer service (http://illuminometer.myhome.net/service.jws)

Context	SSID	ServiceURI	Pre/Post	methodName	paramName	paramType	paramValue
ON()	1	null	null	null	null	null	null
getIllumination()	1	http://light.home.net/service.jws	post	setIllumination	illumination	int	$DMI_{getIllumination()}$

る．前者は DMI の前，後者は DMI の後に実行されている．

サービス層の実装方針は以下のとおりである．まず家電機器が提供する各制御インタフェース d に対して，1 つの公開メソッド m_d を作成し，m_d から d を呼び出すことで DMI を実現する．ここで，d が返り値を持つ場合は，$\$DMI_d$ で返り値を表すことにする．

一方，SMI は実行するサービスシナリオに依存し，サービスシナリオはユーザによって後から変更・追加される可能性もあるため，公開メソッド内にハードコードすべきではない．代わりに，すべてのサービスシナリオと SMI を関連付けた定義ファイル（SMI 定義ファイルと呼ぶ）をサービスごとに用意し，実行するシナリオに応じて，適切な SMI を動的実行する方針を採る．これにより，シナリオの変更や追加は，実装を変更せず SMI 定義ファイルのみを差し替えることで対応可能となる．

図 6 における Light サービスおよび Illuminometer サービスの SMI 定義ファイルの例を，それぞれ**表 1** (a)，(b) に示す．表の各エントリは，*Context*（SMI の呼び出し元メソッド名），*SSID*（実行されるサービスシナリオの ID），*ServiceURI*（SMI を行うサービスの URI．機器のネットワークアドレスと，サービス層のインスタンス名から構成される．例：http://illuminometer.myhome.net/service.jws（jws は java web service の略）），*Pre/Post*（DMI の事前・事後呼び出し），*methodName*（SMI を行う公開メソッド名），*paramName*（公開メソッドのパラメータ名），*paramType*（公開メソッドの引数型），*paramValue*（公開メソッドに渡す引数値）から構成されている．Light サービスは，自身の公開メソッドが実行された際，この表を参照して，適切な SMI を動的に検索・実行する．この例では，両サービスの ServiceURI をそれぞれ http://light.myhome.net/service.jws，

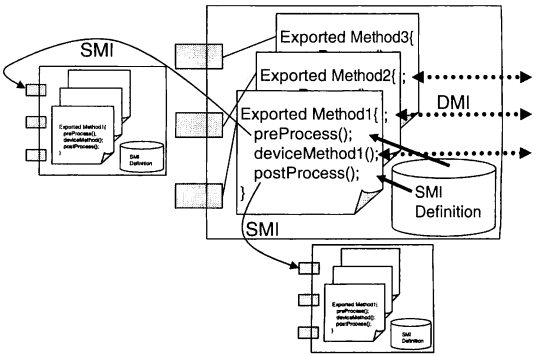


図 7 サービス層実装テンプレート
Fig. 7 Implementation template of service layer.

http://illuminometer.myhome.net/service.jws と仮定している．

以上に基づいて，各家電機器のサービス層は次のような実装テンプレートによって実装可能である（図 7 参照）．

- サービス層は，デバイス層の制御インタフェースと 1 対 1 に対応する公開メソッドを持つ．
- 各公開メソッドの内部では，DMI として 1 つの `deviceMethod()` と，その前後に，`preProcess()`，`postProcess()` が実装される．ここで，`preProcess`（`postProcess`）は，サービス層が SMI 定義ファイルを参照して，DMI の前（後）に，動的に SMI を行うルーチンであり，サービス内の全公開メソッドで共通である．

例として図 6 および表 1 を考える．ユーザが Light サービスの公開メソッド `Light.ON` を実行すると，`Light.ON` メソッドは `preProcess()` を実行するために，表 1 (a) から DMI の事前 SMI を検索する．その結果，Illuminometer サービスの `Illuminometer.ON()` を発見し，この SMI を定義ファイルに従って実行する．その後，DMI として ON を実行し，最後に

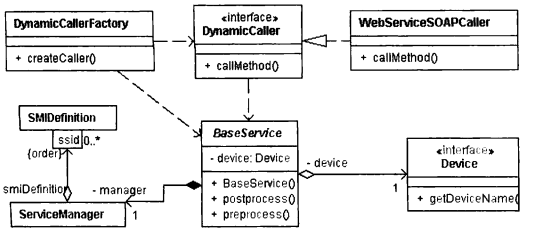


図 8 プロトタイプシステムのクラス図（全体）
Fig. 8 Class diagram of the prototype system (overview).

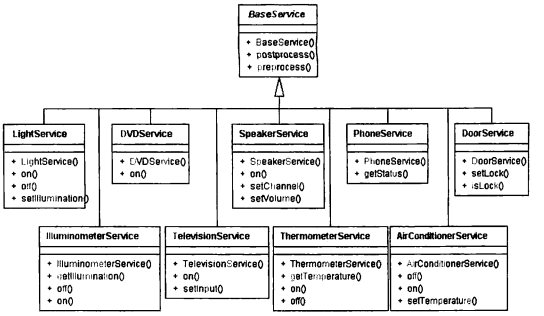


図 9 プロトタイプシステムのクラス図（サービス層）
Fig. 9 Class diagram of the prototype system (service layer).

postProcess() として SMI 定義ファイルに記述されている Illuminometer.getIllumination() の実行を行う．

Illuminometer サービスは，表 1(b) に従って SMI を行う．Illuminometer.ON() が実行された際は，SMI 処理は特になく DMI のみが行われる．一方，Illuminometer.getIllumination() では，DMI が行われた後の事後 SMI として，Light.setIllumination() が実行される．このとき，パラメータ illumination に，DMI で得られた返値 ($\$DMI_{getIllumination()}$) を渡して，SMI を行っている．これにより，照度計で取得した現在の照度を Light にセットするサービスシナリオ SS_1 (図 6) が完結する．

4.2 Web サービスを用いた実装例

前節で述べた実装の枠組みに従い，プロトタイプシステムを試作した．サービス層の公開・利用手段としては，Web サービス[1] を用いた．具体的なシステム環境は以下のとおりである．

- Web サーバ：Jakarta Tomcat 4.1.18
- SOAP ライブラリ：Apache-AXIS 1.1
- 言語環境：Java2 SDK SE 1.4.1_02

また，サービス層に対応するデバイス層を仮想デバイスとして実装した．

図 8 に作成したプロトタイプシステムのクラス図[6] を示す．各サービスは図 9 に示すように，BaseService クラスを継承している．この BaseService クラスは以下の処理を行っている．

(1) SMI 定義ファイルを ServiceManager クラスを用いて解釈し，各サービスの preprocess()，postprocess() で行う SMI 処理を動的に決定する．
(2) SMI を行う Web サービスを DynamicCallerFactory クラスを利用して呼び出す．

次に，図 4 のサービス連携グラフから，各サービスの SMI 定義ファイルを作成し，プロトタイプシステム上で $SS_1 - SS_8$ の連携サービスシナリオを実行したところ，各シナリオが正常に動作することを確認した．

なお，SMI 定義ファイルは，サービス連携グラフ上のサービス層（ノード）に接続する有向辺とその実行順を解析することで，自動生成できる．サービスシナリオの変更や追加の際には，新しい定義ファイルを各機器へ再アップロードすることでサービスシナリオを変更する．このとき，サービス層の実装・設定の変更や，Web サーバの再起動はいっさい不要となっている．

4.3 システムの開発・運用形態

提案する枠組みを用いてシステムを開発・運用する際，本研究で想定しているベンダおよびユーザの役割について述べる．

まず，各機器のサービス層におけるアプリケーション開発は，4.1 節で述べた実装テンプレートに基づいて機器のベンダが行う．その際，ベンダはそのアプリケーションが他の機器にどのように利用されるかは意識する必要はない．その代わり，公開メソッドの厳密な型定義を行い，Web サービスなど SOA における標準的なサービス公開法を用いることが重要である．これにより，機器間の疎結合が実現し，機器の柔軟な追加や変更が可能となる．

一方，連携サービスのシナリオ作成は，ユーザが行うものとする（もちろんベンダがサービス例を組み込んでおくことは可能である）．4.1 節で述べた実装テンプレートにより，サービスシナリオは各機器のサービス層の実装と独立している．したがって，ユーザはサービス連携グラフを作成し，グラフから得られる SMI 定義ファイルを各機器にアップロードすることで，任意の機器を組み合わせたサービスシナリオを容易に実現できる．

サービスシナリオ作成の際，ユーザは各機器の公開メソッドの詳細な定義を知る必要があるが，Web サービスの WSDL や UDDI など SOA のサービス発見技術を備えた連携サービス作成環境を用意することで，

効率的な支援が可能である．この考えに基づき，現在シナリオ作成支援ツールを開発中である．

これに対しHNS-SCAにおいて，家電機器やサービスシナリオの追加・変更が生じた場合，ホームサーバの制御アプリケーションを更新する必要がある．しかしながら，一般のユーザにとって，アプリケーションを開発することは困難であり，ベンダが開発したものを設定変更して利用するしかない．結果として，サービスシナリオの自由度や拡張性が限定される．

5. 評　価

提案するHNS-SOAを，アーキテクチャの側面から定量的に評価する．具体的には，3.3節で定義したサービス連携グラフを用いて，連携サービスの信頼性，負荷，結合度の3つの評価尺度を定義し，従来のHNS-SCAとの比較を行う．

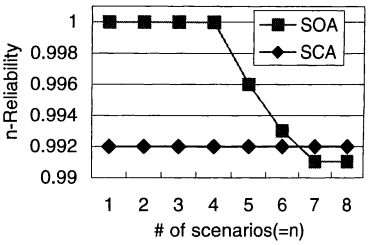
5.1 信 頼 性

HNS上の構成要素に障害が発生しうるとき，HNS上で少なくともn個の連携サービスシナリオが正常に動作する確率を考える．これを連携サービスの*n*-信頼性[7]と呼ぶ．*n*-信頼性は，HNSの各構成要素の信頼性と，ネットワークトポロジ（ここではアーキテクチャ）に依存する．

n-信頼性を計測するため，本稿ではサービス連携グラフに対して*Sum of Disjoint Products*（*SDP*）法を適用する[7),19),21)]．SDP法は，グラフの各ノードと各辺にそれぞれ信頼性を与えると，グラフ内の特定の部分グラフ（の集合）が稼動する確率を，ノードや辺の重なりを考慮して算出する．

3.3節で述べたように，HNSにおける各サービスシナリオはSIGで表現でき，SIGは$FSIG$の部分グラフである．したがって，SDPによって，$FSIG$におけるn個のSIGが正常動作する確率（すなわち，*n*-信頼性）を算出できる．たとえば2.3節のサービスシナリオと図4において，1-信頼性は，$SIG(SS_1),\ldots,SIG(SS_8)$のうち，少なくとも1つが正常に動作する確率である．同様に2-信頼性は，2つのサービスシナリオを含む，$SIG(\{SS_1,SS_2\}), SIG(\{SS_1,SS_3\}),\ldots, SIG(\{SS_7,SS_8\})$の内，少なくとも1つが正常動作する確率である．このように，シナリオのすべての組合せに対してSDPを適用し，*n*-信頼性を算出する．

本稿では，HNS-SOA（図4）とHNS-SCA（図5）のアーキテクチャ差異によってのみ生じる信頼性を評価するため，ネットワーク，および，家電機器のデバイス部の障害を考えない．障害が発生しうるHNSの構成要素として，HNS-SOAにおける各家電機器のサービス層と，HNS-SCAにおけるホームサーバを考える．



	n	1	2	3	4	5	6	7	8
Architecture Type	SOA	0.99999	0.99999	0.99999	0.99998	0.996	0.99302	0.99104	0.99104
	SCA	0.992	0.992	0.992	0.992	0.992	0.992	0.992	0.992

図10 *n*-信頼性
Fig. 10 *n*-reliability.

各構成要素の信頼性は数値化することが難しいが，ここでは，HNS-SOAの各サービス層が正常稼動する確率を一律0.999に設定した．また，HNS-SCAのホームサーバは，8つのシナリオを実現するオブジェクトから構成されると仮定し，その信頼性を0.992（$=0.999^8$）とした（厳密には，ゲートウェイ部の信頼性も考慮する必要があるため，もう少し低くなると予想される）．

図4と図5に示されるサービス連携グラフのそれぞれに対し，SDPを適用し*n*-信頼性を計測した．計測結果を図**10**に示す．図において，横軸はサービスシナリオの数（$=n$），縦軸は*n*-信頼性を示す．

結果において，HNS-SCAではホームサーバの信頼性がそのまま*n*-信頼性と等しくなっている．これは，すべてのサービスシナリオがサーバに依存しているためであり，サーバに障害が発生するとあらゆるシナリオの実行が不可能になることを示している．一方HNS-SOAでは，サービスシナリオが依存するサービス層は分散している．したがって，あるサービス層に障害が発生しても，それに依存しないシナリオは実行可能である．

$n=7,8$のときには，HNS-SOAの*n*-信頼性は，HNS-SCAよりも若干低くなっている．これは，HNS-SOAにおける構成要素の数がHNS-SCAよりも多くなるため，HNS-SOAのすべての構成要素が正常に動作する確率はホームサーバ1台と比べると低くなってしまうからである．このように，障害発生時の耐故障性とシステム全体が正常稼動する場合の信頼性との間に，トレードオフが存在することが分かる．

5.2 サービスにかかる負荷

連携サービスの実行時に，各構成要素（ここでは，HNS-SOAの各サービス層とHNS-SCAのホームサー

表 2 サービス負荷
Table 2 Workload.

Element	WL
DVD	10.7
TV	26.1
Speaker	29.8
Light	57.4
Illuminometer	57.4
Door	18.7
Phone	3.7
AC	18.1
Thermometer	18.1
StandardDev	17.925

Element	WL
Home Server	86.3
StandardDev	86.3

表 3 結合度
Table 3 Coupling.

	coup	
Element	use	used
DVD	3	1
TV	2	2
Speaker	1	2
Light	2	3
Illuminometer	1	1
Door	3	1
Phone	2	1
AC	2	2
Thermometer	1	1
HS	9	8

バ）にどの程度の負荷がかかるかを見積もる．負荷はユーザが利用するシナリオの種類と利用頻度によって変化する．

今，$FSIG = (N, L, E)$ と，サービスシナリオ $s_1, s_2, \ldots, s_k$ の $SIG(s_i)$ $(1 \leq i \leq n)$ が与えられ，さらに各 s_i $(1 \leq i \leq n)$ の利用頻度 f_i が与えられると仮定する．ここで，以下のような出現関数 $c_i : N \to \{0, 1\}$ を定義する：$FSIG$ の各ノード $v \in N$ に対して，v が $SIG(s_i)$ に含まれるときは $c_i(v) = 1$，それ以外は $c_i(v) = 0$．関数 c_i は，そのノードがシナリオにおいて利用されるかどうかを判別する．この c_i とシナリオの利用頻度 f_i を用いて，すべてのサービスシナリオを実行する際，各構成要素 v にかかる負荷 $WL(v)$ を以下のように定義できる．

$$WL(v) = \sum_{i=1}^{n} f_i \times c_i(v)$$

図 4 および図 5 の各構成要素に対して，負荷評価を行った．シナリオの利用頻度を求めるため，12 人の被験者に対してアンケートを実施した（うち独身 8 人，既婚者（子供なし）が 2 人，4 人家族が 2 人）．アンケートでは，まずシナリオ $SS_1 \sim SS_8$ の内容説明を被験者に対して行った．その後，各被験者の各週あたりのシナリオ予想利用頻度を調査した．このアンケート結果をもとに，HNS-SOA のサービス層，および，HNS-SCA のホームサーバにかかる負荷の算出を行った．

評価の結果を**表 2** に示す．WL の列は 9 個の各機器のサービス層，および，ホームサーバにかかる負荷（ここでは週あたりの利用頻度）と，負荷の標準偏差を示している．結果から，HNS-SOA では各サービスに負荷が分散しているのに対し，HNS-SCA ではホームサーバに集中している．

次に，負荷分散の観点から考察する．HNS-SCA では，アーキテクチャの性質上，ホームサーバに負荷が一極集中するので，ホームサーバそのものを負荷分散（たとえば多重化するなど）するしかない．結果的に，利用頻度の低い機器に対しても，冗長な負荷分散を強要してしまうことになり，非効率である．一方 HNS-SOA では，すべての機器がサービス層において対等，かつ，分散しているので，ユーザが選んだ任意の機器を負荷分散の対象にできる．たとえば，表 2 の評価結果から，仮にユーザが Light と Illuminometer を選択したと仮定すると，これらのサービス層のみを多重化すればよい．このように，HNS-SOA では，局所的かつ柔軟な負荷分散が実現できる．

5.3 結 合 度

結合度は，ある構成要素が他の構成要素にどの程度依存しているかを表す評価尺度である．依存度が非常に高い構成要素に障害や変更が生じた場合，それに依存する多くの構成要素が影響を受けるため，連携サービスの正常動作を妨げる要因となる．

今，与えられた $FSIG = (N, L, E)$ に対して，ノード $v \in N$ の結合度を，v に接続するノードの総和であると定義する．厳密には，$FSIG$ における $v \in N$ に対し，$use(v) = |\{v' | \exists m; (v, m, v') \in E\}|$（$v$ が利用する構成要素の数），および，$used(v) = |\{v' | \exists m; (v', m, v) \in E\}|$（$v$ を利用する構成要素の数）とするとき，v の結合度を $coup(v) = use(v) + used(v)$ と定義する．

たとえば，図 4 の TV サービスを例にあげると $use(TV) = |\{Speaker$ サービス，TV デバイス $\}| = 2$，$used(TV) = |\{User, DVD$ サービス $\}| = 2$ となり，TV サービスの結合度は $coup(TV) = 4$ となる．

以上の定義に基づいて，図 4 と図 5 のそれぞれにおける各構成要素の結合度を計算した（**表 3**）．この結果において，HNS-SOA の各サービスの結合度はバランスがとれている．一方で HNS-SCA ではホームサーバへの一極集中が見られる．ゆえに，5.1 節でも述べ

たように，サーバでの障害発生はすべてのサービスシナリオにおいて致命的であることが分かる．

さらに，構成要素間の結びつき（すなわち，サービス連携グラフにおける各有効辺）を考える．HNS-SOAでは，サービスどうしがWebサービスなど標準的な枠組みで疎結合されている．そのため，家電機器が準拠するネットワーク規格やデバイスの内部実装に変更が生じても，(公開メソッドの型定義が変更されない限り）結合している他の機器が影響を受けない．

一方，HNS-SCAではホームサーバと各家電機器は密に結合しており，上記のような変更は相互接続性を大きく低下させる．たとえば，既存のHNSに新たなネットワーク規格に準拠した機器を追加する場合，ホームサーバのゲートウェイ実装を更新する必要がある．ゲートウェイ実装の変更は，それまで接続されていたすべての既存機器に影響を与える．結果として，新規—既存機器間，および，既存—既存機器間の相互接続性を低下させる要因となる．このことを考慮すると，表3におけるHNS-SCAの結合度は，さらに大きくなると考えられる．

6. 考　　察

6.1 提案手法の特長と限界

提案手法の主な特長をまとめると以下のようになる．

(1) SOAを用いて家電機器をサービス層で疎結合することにより，機器間の相互接続性が向上し，HNSへの機器の追加・変更が容易である．

(2) サービスの連携は機器どうしの協調によって自律的に行われるため，中央のホームサーバが不要となる．したがって，耐故障性，負荷バランスに優れた連携サービスの実現が可能となる．

(3) 提案する実装テンプレートにより，機器のサービス層の実装とサービスシナリオとが独立しているため，シナリオの追加・変更が容易である．その結果，連携サービスのより柔軟な実現が可能となる．

これらの特長のそれぞれとSOAとの関連を考察する．まず，上記(1)は，HNSという適用ドメインにかかわらず，SOAによって得られる一般的なメリットであるといえる．(2)は，SOAの疎結合を利用して本稿で提案した独自の「サービス層」によるものである．機器をSOAに適応させるために，自己機能提供(3.2節参照）だけでなく，他者機能利用を各機器のサービス層に含めることで，元来ホームサーバが担ってきた連携作業を，機器側で分担できた．(3)の実装テンプレートは，任意の機器の機能を「公開メソッドの実行」として一元的に扱うSOAの特徴を利用している．この特徴により，サービス層における他者機能利用はSMI，自己機能提供はDMIとして，すべての機器の実装を共通化できた．また，提案した実装テンプレートによって，サービスシナリオに依存するSMIの内容を外部の定義ファイルから動的に参照する方式で実装した．これによって，サービス層の実装とサービスシナリオを独立させることが可能となった．

もちろん，提案手法がすべての面で従来手法より優れているわけではない．連携サービスを自律分散制御することのトレードオフとして，現状では以下のような課題が想定される．

家電機器コスト： サービス層処理のため，収容する家電機器が条件C2（2.2節参照）を満たさなければならない．結果として，従来手法と比べて家電機器のコストが高くなる．

通信オーバヘッド： 機器間の自律連携に必要な通信オーバヘッドが従来法と比べて大きくなると予想される．ハードリアルタイム性を要する連携サービスに適用するには，深慮が必要である．

機器の一括管理： 連携サービスの制御が分散するため，すべての機器を一括管理しにくい．そのため，障害発生機器の特定やセキュリティ管理においてより高度な仕組みが必要になると考えられる．

6.2 関連研究

サービス（特にWebサービス）連携を行うための関連技術として，BPEL4WS[2]が知られている．BPEL4WSは，ネットワーク上に分散するサービスを連携・結合した新たなサービスを記述するための言語である．これを本研究で提案したサービス連携グラフの代わりとして用いることは可能である．しかしながら，既存のBPEL4WSの実行環境（BPELエンジン[2),11)]）では，BPELエンジンが分散するサービス群を中央集権的に実行していくサーバ中心型アーキテクチャが採られている．したがって，本稿で提案するSOA-HNSを，既存のBPEL4WSの枠組みだけを用いて実装することは難しいと考えられる．

また，Webサービスの Peer-to-Peer な連携・相互作用をモデル化する言語として，WS-CDL（Web Services Choreography DescriptionLanguage）[24]が現在策定中である．WS-CDLは，企業レベルのオンライン商取引などサービス間で複雑な順序・依存関係が生じる場面において，複数のサービス間の観測可能なやりとり（メッセージ交換，企業間協定の確認など）を，大域的な視点から規定するための言語である．WS-CDLはπ計算[13]に基づいた数学モデルも採用

しており，チャネル入出力や並列・選択などのプロセス代数演算によって，サービス間の連携相互作用を大域的に規定できる．

このWS-CDLをHNS-SOAにおける機器連携のモデル化に適用することも考えられる．しかしながら，エンドユーザにとって，家電機器間の詳細かつ厳密な関係定義を行うことは，実用上容易なことではない．また，著者らの知る限りでは，現状のHNS連携サービスは，機器機能（公開メソッド）の逐次実行（およびその重ね合わせ）のみで実現でき，複雑な連携を必要としないことがほとんどである．このように，エンドユーザの利便性とHNS連携サービスの複雑さを考えると，今のところ，WS-CDLやπ計算を最大限に活用できる場面は少ないと考えられる．より高度なHNSサービスの調査およびWS-CDLの応用については将来の課題である．

7. まとめ

本稿では，サービス指向アーキテクチャ（SOA）を用いて，連携サービスを設計・実装する手法を提案した．また，信頼性，負荷，結合度の3つの観点から，従来のHNS-SCAとの比較評価を行った．

今後の研究では，より実用規模のシステムを開発し，上で述べた課題を実用面から評価することを計画している．これに基づき，SOAに適した新たな機器・サービス管理方式やセキュリティ実現方式を提案していきたい．また，ホームネットワークでは，複数のユーザが複数のサービスシナリオを同時に実行する可能性があるため，サービス競合問題[15),25)]が発生することが考えられる．したがって，SOAの特性を生かしたサービス競合の検出・解消法の開発も重要な将来の課題である．

謝辞 この研究は，日本学術振興会・科学技術研究費・若手研究（B）15700058，および，奈良先端科学技術大学院大学情報科学研究科21世紀COEプログラム「ユビキタス統合メディアコンピューティング」の支援を受けている．

参考文献

1) 青山幹雄：Webサービス技術とWebサービスネットワーク，信学技報，Vol.102, No.560, pp.47–52 (2003).
2) Business Process Execution Language for Web Services, Version 1.1. http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/
3) DLNA. http://www.dlna.org/
4) ECHONET Consortium. http://www.echonet.gr.jp/
5) Cerami, E.: *Web Services Essentials*, 1st Edition. O'Reilly & Associates Inc., United Stated of America (2002).
6) Fowler, M. and Scott, K.: *Uml Distilled: A Brief Guide to the Standard Object Modeling Language*, Addison-Wesley, Boston (1999).
7) Hariri, S. and Raghavendra, C.S.: SYREL: A Symbolic Reliability Algorithm Based on Path and Cutset Methods, *IEEE Trans. Comput.*, pp.1224–1232 (Oct. 1987).
8) 日立ホーム&ライフソリューション株式会社：ホラソネットワーク．http://www.horaso.com/
9) iReady. http://www.sharp.co.jp/corporate/news/031217-2.html
10) LG E. Home Network. http://www.lge.com/products/homenetwork/homenetwork.jsp.
11) Loke, S.W.: Service-Oriented Device Ecology Workflows, *Proc. 1st Int'l Conf. on Service-Oriented Computing* (*ICSOC2003*), LNCS2910, pp.559–574 (Dec. 2003).
12) 松下電器産業株式会社：くらしネット．http://national.jp/appliance/product/kurashi-net/
13) Milner, R.: *Communicating and Mobile Systems: the Pi-Calculus*, Cambridge University Press (1999).
14) 三菱レイヨン株式会社：ホームネットワーク．http://pofeska.com/tec/homenet1/homenet1.htm
15) 日本電信電話株式会社：ホームサービスハーモニー．http://www.ntt.co.jp/news/news04/0403/040308.html
16) OSGi Alliance. http://www.osgi.org/
17) PLANEX COMMUNICATIONS Inc.: BRC-14V. http://www.planex.co.jp/product/broadlanner/brc14v.shtml
18) Samsung: Home Network. http://www.samsung.com/HomeNetwork/index.htm
19) Soh, S. and Rai, S.: CAREL: Computer aided reliability evaluator for distributed computing networks, *IEEE Trans. Parallel and Distributed Systems*, pp.199–213 (July 1991).
20) Toshiba Corporation: ネットdeナビ．http://www.rd-style.com/index_j.htm
21) Tsuchiya, T., Kajikawa, T. and Kikuno, T.: Parallelizing SDP (Sum of Disjoint Products) Algorithms for Fast Reliability Analysis, *IEICE Trans. Inf. Syst.*, Vol.E83-D, No.5, pp.1183–1186 (2000).
22) UPnP Forum. http://www.upnp.org/
23) W3C Web Service Activity. http://www.w3.org/2002/ws/
24) Web Services Choreography Description Lan-

guage Version 1.0. http://www.w3.org/TR/ws-cdl-10/
25) Weiss, M.: Feature Interactions in Web Services, *Proc. 7th Int'l. Workshop on Feature Interactions in Telecommunication Networks and Distributed Systems* (*FIW'03*), pp.149–156 (2003).
26) What is Service-Oriented Architecture? http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html

(平成 16 年 5 月 19 日受付)
(平成 16 年 11 月 1 日採録)

井垣 宏（学生会員）

平成 12 年神戸大学工学部電気電子工学科卒業．平成 14 年奈良先端科学技術大学院大学情報科学研究科修士課程修了．現在，同大学博士後期課程．修士（工学）．秘書エージェントシステム，Web サービスを利用したサービス指向アーキテクチャに関する研究に従事．電子情報通信学会，IEEE 各学生会員．

中村 匡秀

平成 6 年大阪大学基礎工学部情報工学科卒業．平成 8 年同大学院博士前期課程修了．平成 11 年同大学院博士後期課程修了．同年カナダ・オタワ大学ポスドクフェロー．平成 12 年大阪大学サイバーメディアセンター助手．平成 14 年奈良先端科学技術大学院大学情報科学研究科助手となり，現在に至る．博士（工学）．通信ソフトウェアの検証，サービス競合，ソフトウェアプロテクション等の研究に従事．電子情報通信学会，IEEE 各会員．

玉田 春昭

平成 11 年京都産業大学工学部情報通信工学科卒業．平成 13 年同大学院博士前期課程修了．同年住商エレクトロニクス（株）入社．Web アプリケーション開発に従事．平成 15 年奈良先端科学技術大学院大学情報科学研究科博士後期課程進学，現在に至る．修士（工学）．ソフトウェアプロテクション，エンタープライズアプリケーションの研究に従事．電子情報通信学会，IEEE 各学生会員．

松本 健一（正会員）

昭和 60 年大阪大学基礎工学部情報工学科卒業．平成元年同大学院博士課程中退．同年大阪大学基礎工学部情報工学科助手．平成 5 年奈良先端科学技術大学院大学情報科学研究科助教授．平成 13 年同大学教授．工学博士．収集データに基づくソフトウェア開発/利用支援，ウェブユーザビリティ，ソフトウェアプロセス等の研究に従事．電子情報通信学会，IEEE，ACM 各会員．