Vol.2009-SE-166 No.4
2009/11/5

SOA にもとづくホームネットワーク サービス競合検出・解消基盤の提案

吉村悠平[†1]　稲田卓也[†1]
井垣宏[†1]　中村匡秀[†1]

ホームネットワークシステム (HNS) のアプリケーションの一つとして，複数の家電を連携制御する家電連携サービスの研究が進んでいる．連携サービスでは，単体では正常に動作しても，複数を同時に実行すると機能間の干渉・衝突によってユーザの意図しない振る舞いを引き起こす「サービス競合」と呼ばれる障害が存在する．HNS のサービス品質を損なわないためにも，このようなサービス競合を検出し解消することが求められている．我々は先行研究において，このようなサービス競合を動的に検出・解消する手法を提案した．本稿では，先行研究で提案したオンライン競合検出・解消システムを SOA 化し，競合検出・解消に関するより多様なアプリケーションを容易に開発できる競合検出・解消基盤を実現した．SOA 化に当たっては，競合検出・解消基盤の構成要素を競合検出部，解消部，実行管理部の 3 要素に分割し，それぞれを個別にサービス化した．ケーススタディでは，これらの 3 つの基盤サービスを利用して，オンライン競合検出解消機能付きの連携サービス実行・管理システムやオフライン競合検出システムを構築し，多様なアプリケーション開発が容易化されることを確認した．

A Service-Oriented Platform for Feature Interaction Detection and Resolution in Home Network System

Yuhei Yoshimura,[†1] Takuya Inada Hiroshi Igaki[†1,†1] and Masahide Nakamura[†1]

As one of the Home Network System application, smart appliance integration services are widely developed. However, combined use of variable appliance integration services may cause some conflicts which decrease quality of services. The conflicts are generally known as the feature interaction problem. In our precedence research, we proposed an online feature interaction detection and resolution method. In this paper, we applied SOA principles into our feature interaction detection and resolution mechanism. In our SOA-based FI detection and resolution system consists of a FI detection service and a FI resolution services, and a service activation manager. As a case study, we indicated that complicated hns applications can be developed easily with using these three fundamental services.

1. はじめに

ネットワーク技術の発展と共に，一般家庭にある家電機器を家庭内のネットワークに接続して，宅外からの制御や複数機器の連携を実現する**ホームネットワークシステム (HNS)** の研究開発が進んでおり，近年いくつかの製品が商品化されている[1)2)3)]．

HNS では，テレビや DVD レコーダー，エアコン，照明，扇風機といった複数の家電を連携制御することで，家電単体で利用する場合に比べてより付加価値の高い**家電連携サービス**（以下，連携サービス）を実現することができる．連携サービスは，ユーザの日常生活における快適性・利便性を高める主要な HNS アプリケーションの一つとして研究されている．

HNS に多くの連携サービスが提供されると，利便性が向上する反面，**サービス競合**という新たな問題が発生する[4)5)6)]．単独で正常に動作する連携サービスを，複数を同時に実行すると互いに干渉・衝突を起こし，ユーザの意図した通りに動作しなくなる現象である．サービス競合はユーザの快適性・利便性を損ない，HNS の品質を低下させる要因となるため，それらを**検出**し**解消**することが求められる．

下記に連携サービスの例を紹介する．

DVD シアターサービス： DVD レコーダー，テレビ，スピーカー，カーテン，照明を連携し，映画館の雰囲気でユーザが DVD を視聴できるサービス．ユーザがサービスを要求すると，自動的に DVD レコーダーとテレビが再生モードで ON になり，カーテンが閉まり，照明が暗くなり，スピーカーの 5.1ch が選択され，DVD が再生される．

おかえりサービス： ユーザが帰宅した際に，照明を明るくするサービス．

これらを使ってサービス競合の例を説明する．ユーザ A が DVD シアターサービスを実行しているときに別のユーザ B が帰宅し，おかえりサービスを実行したとする．このとき，ユーザ A の DVD シアターサービスにより照明が暗くなっているにもかかわらず，ユーザ B のおかえりサービスにより照明が明るくなってしまう．これは照明機器において 2 つの

†1 神戸大学
Kobe University

© 2009 Information Processing Society of Japan

連携サービスの要求が衝突してしまったことによるサービス競合である．この例においては，ユーザ B が帰宅サービスを実行したときに，システムは照明機器に対する競合を検出し，解消する必要がある．

我々は HNS におけるサービス競合の**オンライン競合検出・解消法**を提案した[7)8)]．先行研究ではまず競合検出を行うために，HNS に含まれる機器をメソッドと内部状態 (プロパティ）を持つ機器オブジェクトとして定義した．さらに，連携サービスを機器メソッドの系列 (連携サービスシナリオ) として定義することで，複数の連携サービス内のメソッド間に発生するサービス競合の定式化を行った．オンライン競合検出・解消法ではこの定式化にもとづいて，競合を動的に検出するための**競合監視マネージャ**を開発した．競合監視マネージャは実行中の連携サービス情報を管理しており，新しく連携サービスが起動される毎に，実行中機器メソッドとの間のサービス競合の有無をチェックする．競合が検出された後は，メソッドに予め定義された優先度を利用し，優先度の高い機器メソッドを実行する形式で競合解消が行われる．

我々の先行研究[7)8)] では，競合の検出や解消，機器メソッドの実行管理を，密に結合した一連の処理として実装している．そのため，現状では，決められた通りのオンライン競合検出・解消しか実行できない．したがって，検出・解消ロジックを差し替えたり，システムの機能の一部を他のプログラムに組み込んだりすることは非常に困難である．

そこで，この問題を解決するために，**サービス指向アーキテクチャ(SOA)** にもとづくサービス競合検出・解消基盤を提案する．具体的には，競合検出，解消，実行管理の各機能のサービス化を行う．これにより，必要なサービスメソッドを組み合わせることで新たなアプリケーションを作成することが可能となり，開発が容易化される．本稿では，提案する基盤を実装し，その競合検出・解消基盤を利用して実際にオンライン・オフラインでの競合検出・解消システムが容易に開発可能であることを示す．

2. 準　　備

2.1 HNS におけるサービス競合

HNS におけるサービス競合について，**静的検出手法**が提案されている[4)]．この手法では，競合を形式的に検出するために，HNS の**モデル化手法**を提案している．具体的には，各家電を状態 (プロパティ) と操作 (メソッド) を持つオブジェクトとして捉える．まず，各家電は自身の状態を保持する**機器プロパティ**を持つ．表 1 に機器プロパティ例を示す．例えば TV の場合，機器プロパティとして power, channel, volume を持ち，power は true(電源 ON) か false(電源 OFF) である boolean 型，channel と volume は整数の int 型で与えられる．

表 1　機器プロパティ例

ApplianceName	AppliancePropertyName	PropertyType
TV	power	boolean
	channel	int
	volume	int
Light	power	boolean
	brightness	int
Curtain	openLevel	int

表 2　メソッドモデル例

ApplianceName	ApplianceMethod	PreCondition	PostCondition
TV	on()		power=true
	off()		power=false
	ch(int c)		power=true
	vol(int vol)		power=true
Light	on()		power=true
	off()		power=false
	setBrightness(int level)		power=true
Curtain	open()		openLevel=100
	close()		openLevel=0

Priority 4
SS_1:DVD-Theater(DVD-T)

```
begin() {
 1.1. DVD_Recorder.on();
 1.2. DVD_Recorder.changeInput(0);
 1.3. TV.on();
 1.4. TV.changeInput(1);
 1.5. Curtain.close();
 1.6. Light.setBrightness(1);
 1.7. Speaker.changeMode(5.1ch);
 1.8. DVD_Recorder.play();
}
end () {
 1.11. DVD_Recorder.stop();
 1.12. DVD_Recorder.off();
 1.13. TV.off();
 1.14. Light.setBrightness(10);
 1.15. Curtain.open();
}
```

Priority 6
SS_2:Coming Home(CH)

```
begin() {
2.1. Light.setBrightness(10);
}
```

図 1　連携サービスシナリオ例

各家電の操作，すなわち，**機器メソッド**は，このプロパティを参照・更新するものとしてモデル化される．具体的には，各メソッドは，

- メソッドの実行に必要な機器に関する事前条件
- メソッド実行後に成立する機器に関する事後条件

でモデル化される．表 2 に機器メソッドのモデルの例を示す．例えば TV のメソッド vol(int vol) は，実行後に power が true，volume が引数 vol になっている必要がある．

連携サービスは任意の機器メソッドを組み合わせたものであり，機器メソッドの系列を**連携サービスシナリオ**と呼ぶ．図 1 に連携サービスシナリオの例を示す．begin() 内に記された機器メソッドが，連携サービス開始時に順番に実行される．また，end() 内の機器メソッドは，連携サービス終了時に実行されるもので，必要なサービスにのみ記してある．各サービスには優先度が記されており，優先度が大きいサービスが優先的に実行される．文献[4)] で

© 2009 Information Processing Society of Japan

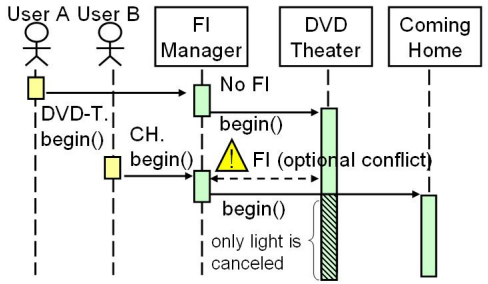

図 2 シーケンス図 (DVD-T,CH)

は，このモデル化手法を用いてサービス競合を「あるサービスに含まれるメソッド m と別のサービスに含まれるメソッド m' について，m の事後条件と m' の事後条件 (または，m' の事後条件) が，同じ機器プロパティにおいて矛盾するときに発生する」と定義している．与えられた連携サービスが含む全ての機器メソッドのペアに対して，上記の競合条件を評価することで，潜在的な全てのサービス競合を静的に検出することができる．

2.2 HNS におけるオンラインサービス競合検出・解消法

オンラインサービス競合検出・解消手法とは，静的競合検出において利用した機器モデルや連携サービスシナリオ定義を利用して，連携サービス実行時に動的に競合を検出・解消する手法である．先行研究[7)8)] ではサービス競合を実行時に検出するために，現在どの連携サービスのメソッドが実行中なのかを絶えず競合監視マネージャが把握している．新規に連携サービスが実行された際には，現在実行中の連携サービス内のメソッドと新規に実行予定のメソッドの間の競合を検出する．競合が検出された際には，優先度の高いメソッドを実行し，低いメソッドを停止することで競合解消を行う．

図 2 に，DVD シアターサービス実行後におかえりサービスを実行した場合のシーケンスを示す．まず，ユーザ A が DVD シアターサービスの実行要求を競合監視マネージャに送る．すると，競合監視マネージャは競合が発生しないことを確認した後，DVD シアターサービスを実行する．次にユーザ B がおかえりサービスの実行要求を競合監視マネージャに送る．競合監視マネージャは，新規に実行するおかえりサービスと，現在実行中である DVD シアターサービス間で，照明に関する競合を検出する．この場合は，おかえりサービスの優先度の方が高いので，おかえりサービスが優先的に実行されることで競合は解消される．

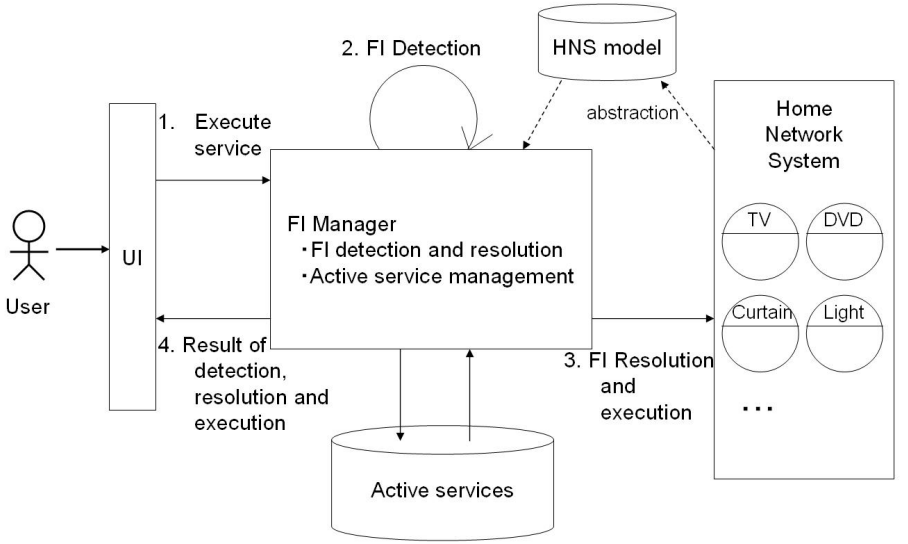

図 3 既存オンラインサービス競合検出・解消アーキテクチャ

既存の検出・解消システムの構造を図 3 に示す．円形で表されているものはサービスとして提供されていることを示している．各項目の詳細は以下のようになっている．

FI Manager： 競合検出・解消を行う競合監視マネージャ．

UI： ユーザが連携サービスの実行や終了を要求するインターフェース．

Home Network System： 各機器を制御するサービスを持つホームネットワークシステム．

HNS model： HNS の利用や競合検出・解消に必要な情報を持つ HNS モデル．機器モデル，連携サービスシナリオ定義を含む．

Active services： 現在実行中であるサービスの情報．

最初に，ユーザがユーザインタフェースを通して，連携サービスの実行を要求する．

競合監視マネージャは HNS モデルから，連携サービスシナリオ定義，機器モデルを取得する．その後，現在実行中のサービスと，実行要求を受けたサービスとの間の競合検出を行い，競合するメソッド情報を取得する．

© 2009 Information Processing Society of Japan

次に，競合の解消と，それに伴う機器の制御を行う．競合検出結果より，競合関係にある 2 つのメソッドのうち，優先度の高いメソッドを実行し，低いメソッドを停止する．実際の機器メソッドに関する処理は HNS に実行要求を送ることで実現する．さらに，競合監視マネージャは機器メソッドの実行内容にもとづいて各連携サービスの実行状態を更新する．

ユーザが連携サービスの終了を要求した場合は，終了時に実行するメソッドについて同様に競合検出・解消を行い，機器の制御を行う．この際，終了されたサービスは，実行中サービスから除外される．

2.3 既存の連携サービス実行管理システムにおける問題点

我々の先行研究における連携サービス実行・管理システムでは，前節で述べたプロセスに従って，一連の競合検出・解消プロセスを実行することができる．しかし，現在のシステムは事前に想定したプロセスを決められたとおりに実行することしかできない．

HNS において，連携サービスの実行・管理システムは多様な要求に対応するための柔軟性や拡張性が求められるシステムである．例えば，携帯電話や PC 画面，専用の操作パネル等の様々な UI の追加や競合検出・解消ロジックの変更などがシステムに対する変更要求として考えられる．また，競合検出部などの競合検出・解消システムの一部機能の再利用が可能であれば，オフライン競合検出のためのアプリケーション開発など，より多様な HNS アプリケーションの開発も可能となる．これらの変更や再利用への対応は，競合検出・解消から連携サービス実行までの一連の処理が密に結合した従来システムでは非常に困難である．

次節以降では，このような問題点に対応すべく，我々が行った既存システムへの SOA 適用について説明する．

3. SOA にもとづくホームネットワークサービス競合検出・解消基盤

3.1 キーアイデア

競合検出・解消が可能で，柔軟性が高く変更に強い連携サービス実行管理システムを実現するために，我々は SOA に基づく競合検出・解消基盤の構築を提案する．すなわち，これまで競合監視マネージャ内で一連のプロセスとして行われていた機能群から，サービスとしての再利用が有効であると思われる機能を抽出し，サービス化を行う．本稿では，競合検出・解消基盤におけるサービス化の対象を以下の 3 つとした．

競合検出サービス :新規に実行される連携サービス (以降では S_{new} と呼ぶ) と現在実行中の連携サービス (**実行管理サービス**より取得する) の間の競合を検出し，結果を返すサービス．

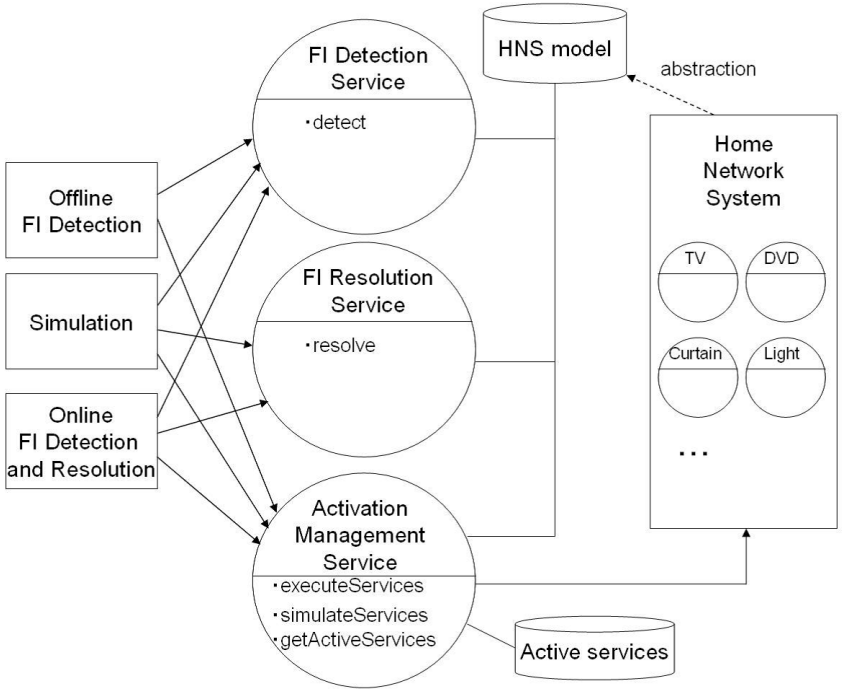

図 4 SOA にもとづく競合検出・解消基盤のアーキテクチャ

競合解消サービス :競合検出サービスの結果を利用し，検出された競合を解消し，連携サービスの実行内容を決定するサービス．

実行管理サービス : 競合解消サービスなどで生成された連携サービスの実行内容にもとづき，具体的な機器操作を行うサービス．また，実行中のサービス情報を把握することが出来る．

提案する基盤の構造を図 4 に示す．図 3 における競合監視マネージャの機能が，**競合検出サービス**，**競合解消サービス**，**実行管理サービス**としてサービス化される．

サービス化されたこれらの競合検出・解消基盤を利用することで，既存の連携サービス実行管理システムや新規に開発するオフライン競合検出システム，多様な UI 等を容易に構築することができるようになる．

© 2009 Information Processing Society of Japan

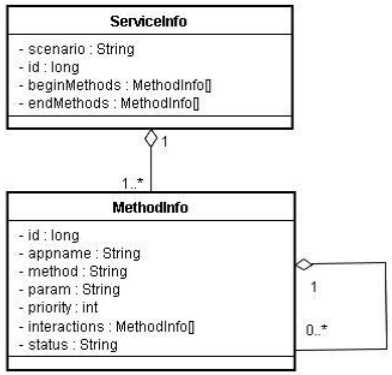

図 5 ServiceInfo オブジェクト

次節以降では，各サービスの詳細について説明する．

3.2 競合検出サービス

競合検出サービスは，新規に実行する連携サービス S_{new} と，現在実行中の全ての連携サービス間の競合を検出する．

このサービスが公開するサービスメソッドは以下の通り．

detect(ServiceInfo S_{new}**)** : **ServiceInfo**

引数と戻り値はともに図 5 に示す ServiceInfo である．ServiceInfo は連携サービスに関する情報を保持しており，我々の提案する競合検出・解消基盤における全てのサービスが共通して利用する．ServiceInfo の属性は以下の通り．

ServiceInfo

- scenario(String) : 連携サービスシナリオ名．
- id(long): 連携サービスを個別に識別するための ID.
- beginMethods(MethodInfo[]): 連携サービス開始時に実行される機器メソッド群.
- endMethods(MethodInfo[]): 連携サービス終了時に実行される機器メソッド群.

MethodInfo

- id(long): 機器メソッドが属する連携サービスの ID.
- appname(String): 制御される機器名.
- method(String): 実行される機器メソッド名.
- param(String): 機器メソッドに渡される引数.
- priority(int): 優先度．競合発生時に優先される機器メソッドを決定する際に用いられる．
- interactions(MethodInfo[]): 競合する機器メソッドの情報．競合検出の際に追加され，この情報を持つ機器メソッドについて競合解消が行われる．
- status(String): 機器メソッドの状態．競合解消と実行管理の際に変更，更新される．status は {"Running"(実行中), "WaitingRunning"(実行待ち), "Terminated"(終了), "WaitingTerminated"(終了待ち)} のどれかの値を取る．

detect メソッドは S_{new} を受け取ると，以下の Step で競合検出処理を行う．

Step1: 実行管理サービスの *getActiveServices* メソッドを呼び，現在実行中の連携サービス内の全機器メソッドを取得する．

Step2: HNS の持つ機器モデルを利用し，S_{new} と実行中連携サービスに含まれる全ての機器メソッド間に発生する競合を検出する．

Step3: 検出された競合結果にもとづいて，S_{new} の全ての機器メソッドに対して，検出された競合対象の機器メソッド情報 (競合メソッド情報) を MethodInfo の interactions 属性に追加する．

Step4: 競合メソッド情報が付与された ServiceInfo オブジェクト ($conflictedS_{new}$) を戻り値として返す．

3.3 競合解消サービス

競合解消サービスは，競合検出サービスによって検出された競合情報にもとづいて，競合解消を行うサービスである．

このサービスが公開するサービスメソッドは以下の通り．

resolve(ServiceInfo $conflictedS_{new}$**)** : **ServiceInfo**

引数と戻り値はともに ServiceInfo 型である．競合解消サービスは，$conflictedS_{new}$ 内の全ての機器メソッド (MethodInfo) が持つ interactions 属性を利用して競合解消を行う．resolve メソッドは，$conflictedS_{new}$ に基づいて以下の Step で競合解消処理を行う．

Step1: $conflictedS_{new}$ が持つ全ての機器メソッドについて，競合メソッド情報を持つか調べる．

Step2: 競合メソッドを持つ全ての MethodInfo を対象として，$conflictedS_{new}$ に含まれる機器メソッドと競合メソッドのどちらを優先するべきかを MethodInfo の

© 2009 Information Processing Society of Japan

priority 属性を利用して決定する．

Step3: Step2 で決定した情報に基づき，$conflictedS_{new}$ に含まれる機器メソッドの MethodInfo および競合メソッドの MethodInfo の status 属性を更新する．ここで，status 属性は，”WaitingRunning”(実行待ち) もしくは ”WaitingTerminated”(終了待ち) のどちらかの値が設定される．

Step4: status 属性が付与された ServiceInfo オブジェクト ($resolvedS_{new}$) を戻り値として返す．

3.4 実行管理サービス

実行管理サービスは，競合解消結果に従って，競合が解消された機器操作要求を HNS に伝達する．また，実行中サービス情報の管理を行う．このサービスが公開するサービスメソッドは以下の通り．

executeService(ServiceInfo $resolvedS_{new}$**)** : **ServiceInfo**

引数と戻り値はともに ServiceInfo 型である．引数の ServiceInfo は， $resolvedS_{new}$ が持つ機器メソッドに関する競合解消結果を保持している．

executeService メソッドは，$resolvedS_{new}$ を受け取ると，以下の Step で実行管理を行う．

Step1: $resolvedS_{new}$ に含まれる機器メソッドの MethodInfo の内，status 属性が $WaitingRunning$ であるものに対する実行要求を HNS に送る．

Step2: $resolvedS_{new}$ に含まれる機器メソッドと競合メソッドのの MethodInfo の内，status が”WaitingRunning”(実行待ち) のものを”Running”(実行中) に，”WaitingTerminated”(終了待ち) のものを”Terminated”(終了) に変更する．

Step3: 新しく開始した連携サービスを実行管理サービスが保持する実行中サービス群 (Active Services) に追加する．

Step4: 実行中サービス群の中で，全ての機器メソッドの status が”Terminated”(終了) の連携サービスを削除する．

Step5: status 属性が変更された ServiceInfo オブジェクト ($executedS_{new}$) を戻り値として返す．

simulateService(ServiceInfo $resolvedS_{new}$**)** : **ServiceInfo**

引数，戻り値は共に executeService メソッドと同じである．

simulateService メソッドは，$resolvedS_{new}$ を受け取ると，executeService メソッドと同じ Step で実行管理を行うが，機器の操作要求は行わない．すなわち，executedService メソッドの実行 Step において，Step1 のみを行わないのが simulateService メソッドである．戻り値となる ServiceInfo オブジェクトは，executeService メソッドを利用した場合と同じである．

getActiveServices() : ServiceInfo [] services

引数はなし．戻り値は現在実行中のサービスの ServiceInfo の集合である．

3.5 実　　装

競合検出・解消基盤に含まれる全てのサービスは以下の環境で開発・公開した．また，連携サービスに含まれる家電機器を制御する HNS は，我々の研究グループで以前より構築している実際の HNS(CS27-HNS と呼ぶ)[10] を利用している．

- サーバー:950MB RAM 2.00GHz WinXP Pro
- Tomcat 5.5
- Apache Axis2
- Java JDK5

次節では，ケーススタディとして，上記環境で実装した競合検出・解消基盤の各サービスを利用して開発した各システムについて説明する．

4. ケーススタディ

4.1 オンライン競合検出解消可能な連携サービス実行管理システム

図 6 は本稿で提案した競合検出・解消基盤を利用して開発したオンライン競合検出・解消機能付き連携サービス実行管理システムである．このシステムは以下の機能を備えている．

F1: 連携サービス実行およびオンライン競合検出・解消機能

F1 は競合検出サービスの detect メソッド，競合解消サービスの resolve メソッド，実行管理サービスの executeService メソッドを利用する．ユーザは UI 左上の連携サービスボタンから実行する連携サービスを選択し，実行ボタンをクリックするだけで，その連携サービスを実行することが出来る．このとき，競合検出・解消サービスの機能を利用してオンライン競合検出・解消が自動的に行われ，その結果が画面下部の実行ログに表示される．

F2: 現在実行中の連携サービスおよびサービスメソッド表示機能

現在実行中の連携サービス情報を実行管理サービスの getActiveServices メソッドを利用して表示する．

F3: 実行すると競合が起きるサービスメソッドの表示機能

© 2009 Information Processing Society of Japan

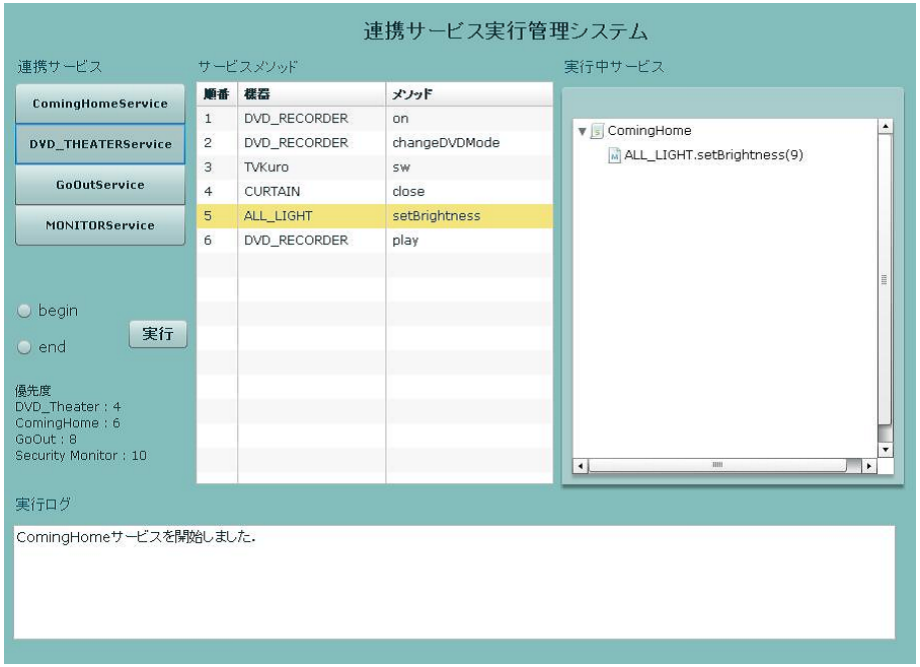

図 6 オンライン競合検出・解消機能付き連携サービス実行管理システム

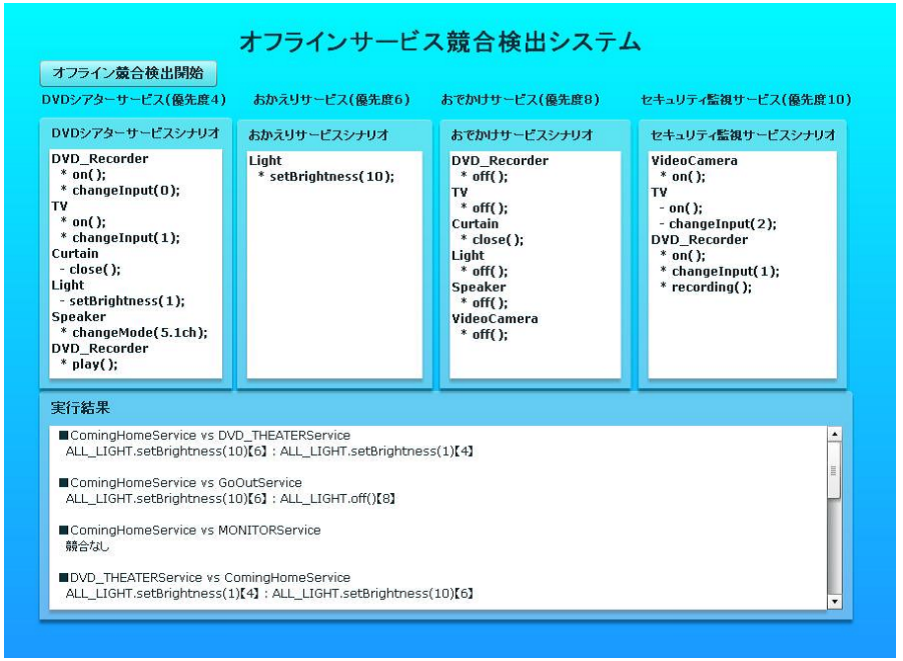

図 7 オフライン競合検出システム

ユーザが連携サービスボタンを選択すると，画面中央のサービスメソッドの項目に個々のサービスメソッドが表示される．このとき，競合検出サービスの detect メソッド，競合解消サービスの resolve メソッドを利用し，競合が発生する可能性があるサービスメソッドをその競合解消内容に従って色付けして表示する．図 6 の例では，おかえりサービス実行中に，DVD シアターサービスを選択している．この 2 サービス間では，照明に関して競合が発生する．この場合，優先度によりおかえりサービスが優先され，DVD シアターサービスの setBrightness メソッドが実行できないため，その内容が色付けして表示されている．

4.2 オフライン競合検出システム

サービスシナリオの確認や修正を目的として，オフライン競合検出を行うシステム (図 7) を作成した．このシステムでは，全ての連携サービス間に発生する可能性のある全ての競合を行う．利用するサービスメソッドは競合検出サービスの detect メソッドおよび実行管理サービスの simulateService メソッドの 2 つである．

特定の一つの連携サービスの情報を引数として，simulateService メソッドを実行する．simulateService メソッドによってシミュレート実行された連携サービスを対象として，detect メソッドに全ての連携サービスの情報を順番に与え，戻り値の情報を検出結果として表示する (図 7 下部)．このプロセスを全ての連携サービス間において繰り返し行うことで，発生する可能性のある全てのサービス競合を検出することが可能となる．

4.3 考　　察

競合検出・解消基盤を SOA 化することで，高度な UI を備えたオンライン競合検出・解消機能付き連携サービス実行管理システムを非常に容易に開発することができた．既存の競合検出・解消システムでは，密に結合した一連の処理としてシステムが実現されていた．

© 2009 Information Processing Society of Japan

そのため，オフライン競合検出のような競合検出と実行管理のみの利用や，連携サービス実行管理システムのような実行中サービスの取得は非常に困難であった．

本稿では，競合検出・解消基盤として，競合検出・競合解消・実行管理という 3 つの構成要素をそれぞれサービス化することで，ケーススタディで述べたような多様な機能を持つアプリケーションを非常に容易に開発することが可能となった．

今後の課題として，基盤サービスが引数もしくは戻り値として利用する ServiceInfo の扱いの見直しが挙げられる．ServiceInfo は現在，最もシンプルな競合検出・解消を行う際に必要な情報しか保持していない．例えば，現在の ServiceInfo ではどのユーザが連携サービスを実行したかという情報を含むことができないため，現状ではユーザ間の優先順位を利用して競合解消を行う競合解消サービスを提供できない．そのような，より複雑な競合検出・解消ロジックを基盤サービスとして実現する際には，新しいロジックで必要な情報を ServiceInfo の拡張もしくは他の場所から獲得する必要がある．

5. おわりに

本稿では，SOA にもとづき，HNS におけるサービス競合検出・解消システムの検出・解消・実行管理の各機能をそれぞれサービスとして公開し，それらの組み合わせによるシステム構築を実現した．また，ケーススタディとしてオンライン競合検出・解消機能付き連携サービス実行管理システムやオフライン競合検出が容易に構築可能であることを示した．

今後は，他に提案されているサービス競合検出・解消法[11)12)] の基盤サービス化と個々の競合検出サービスや解消サービスのアルゴリズムに依存する情報の管理方法等についての検討を進めていきたいと考えている．

参考文献

1) 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/
2) パナソニック電工株式会社, “ライフィニティ”, http://denko.panasonic.biz/Ebox/kahs/
3) 東芝, “東芝ネットワーク家電 Feminity”, http://www3.toshiba.co.jp/feminity/about/index.html
4) Masahide Nakamura, Hiroshi Igaki, and Ken-ichi Matsumoto, ”Feature Interactions in Integrated Services of Networked Home Appliances -An Object-Oriented Approach-,” In Proc. of Int’l. Conf. on Feature Interactions in Telecommunication Networks and Distributed Systems (ICFI’05), pp.236-251,2005.
5) M.Nakamura, H.Igaki, and K.Matsumoto. Feature interactions in integrated services of networked home appliances -an object-oriented approach-. In *Proc. Int’l. Conf. on Feature Interactions in Telecommunication Networks and Distributed Systems (ICFI’05)*, pages 236–251, 2005.
6) M.Wilson, M.Kolberg, and E.H. Magill. Considering side effects in service interactions in home automation - an online approach. In *Proc. Int’l. Conf. on Feature Interactions in Software and Communication Systems (ICFI’07)*, pages 172–187, 2007.
7) 吉村悠平，井垣宏，中村匡秀，”ホームネットワークシステムにおけるサービス競合の動的検出・解消システムの設計と実装”，電子情報通信学会技術研究報告，Vol.108, No.136, pp.35-40，July 2008.
8) 吉村 悠平, 池上 弘祐, 井垣 宏, 中村 匡秀, ”ホームネットワークシステムにおける家電連携サービスのための競合解消方式の考察”, 情報ネットワーク研究会, vol.IN2008-206, pp.439-444, March 2009.
9) 田中章弘, 中村匡秀, 井垣宏, 松本健一,“Web サービスを用いた従来家電のホームネットワークへの適応”, 電子情報通信学会技術研究報告, Vol.105, No.628, pp.067-072, March 2006.
10) M.Nakamura, A.Tanaka, H.Igaki, H.Tamada, and K.ichi Matsumoto. Constructing home network systems and integrated services using legacy home appliances and web services. *International Journal of Web Services Research*, 5(1):82–98, January 2008.
11) M. Wilson, M. Kolberg, and E. H. Magill, “Considering side effects in service interactions in home automation - an online approach,” in Feature Interactions in Software and Communication Systems IX (L. du Bousquet and J.-L. Richier, eds.), pp. 172-187, IOS Press, Amsterdam, 2007.
12) M. Kolberg, E. H. Magill, and M. Wilson. Compatibility issues between services supporting networked appliances. IEEE Communications Magazine, 41(11):136-147, November 2003.

© 2009 Information Processing Society of Japan