ビル管理システム連携におけるシステムとサービス競合の定式化

西澤　茂隆[†]　　中村　匡秀[††]　　井垣　　宏[††]　　松本　健一[†]　　三浦健次郎[†††]

† 奈良先端科学技術大学院大学 情報科学研究科
†† 神戸大学 大学院工学研究科
††† 三菱電機株式会社 情報技術総合研究所
E-mail: †{shigetaka-n,matumoto}@is.naist.jp, ††{masa-n,igaki}@cs.kobe-u.ac.jp,
†††Miura.Kenjiro@eb.MitsubishiElectric.co.jp

あらまし　建物の付加価値を高めるための様々なビル管理システムが商品化されており，今後は複数のシステムを組み合わせて，より付加価値の高いサービスを実現することが期待されている．しかし，複数のシステムを連携することで，システムの複雑化，異なるシステム間のサービス競合などの新たな問題が発生する．これらの問題に対処するため，本稿ではビル管理システムのモデル化手法を提案する．具体的には，各管理システムを，設備機器の集合とそれらによって実現されるサービスシナリオの集合とで性質付け，システムが単体で矛盾なく一貫して動作するための条件を定式化する．さらに，提案モデルに基づいて複数システム間のサービス競合を検出する手法も提案する．また本稿では，実用的なビル管理システムに対するケーススタディを行い，10 個の管理システム，33 本のサービスシナリオのモデル化を行った．また，466 通りのシナリオ組み合わせに対して，34 個のサービス競合を検出することが出来た．
キーワード　ビル管理システム，連携サービス，サービス競合，一貫性

Formalizing Heterogeneous Building Automation Systems and Feature Interaction Problem

Shigetaka NISHIZAWA[†], Masahide NAKAMURA[††], Hiroshi IGAKI[††],

Ken-ichi MATSUMOTO[†], and Kenjiro MIURA[†††]

† Graduate School of Information Science, Nara Institute of Science and Technology
†† Graduate School of Engineering, Kobe University
††† Mitsubishi Electric Corporation Information Technology R&D Center
E-mail: †{shigetaka-n,matumoto}@is.naist.jp, ††{masa-n,igaki}@cs.kobe-u.ac.jp,
†††Miura.Kenjiro@eb.MitsubishiElectric.co.jp

Abstract　The building automation system (BAS, for short) provides efficient management features of building equipments as well as various value-added services. According to the progress of network infrastructure, it is expected, in the near future, to achieve more value-added and sophisticated services by integrating multiple BASs over the network. However, the integration of heterogeneous systems yields complexity of the systems as well as the service interaction problem. To cope with these problems, we propose a modeling method for the BAS. Specifically, we characterize a BAS by sets of equipments and service scenarios achieved by the equipments. We then formulate conditions that the system itself operates consistently and compatibly. We also present a method for detecting service interactions between multiple BASs. Using the proposed model, we have conducted a case study, where 10 practical systems and 33 service scenarios were analyzed. Moreover, for 466 pairs of service scenarios, we could successfully detect 34 service interactions.
Key words　building automation system, integrated services, service interactions, consistency

1. はじめに

近年のオフィスビルや高級マンション等では，建物としての付加価値を上げるために，様々な**ビル管理システム**が商品化されてきている [8]．代表的なものに，空調やブラインドを制御する省エネシステムや，エレベータ・エスカレータの遠隔運用・保守システム，入退室管理やカメラ・ビデオ監視によるセキュリティシステムなどが挙げられる．

昨今のインターネット技術の普及により，高速で常時接続可能なネットワークを安価で利用できるようになってきた．また，インターネットのオープン性から，ネットワーク上のあらゆる種類の資源へアクセスが可能となってきている．このことを背景に，次世代のビル管理システムでは，ビル管理のみの枠にとらわれず，ネットワーク上の情報サービスを連携したり，異なるシステムをオープンな手段で統合して，より付加価値を高めることが期待されている．

しかし，システムの高付加価値化に伴い，管理システムの複雑化，一貫性保証，異なるシステム・サービス間の機能的な衝突・競合（サービス競合問題と呼ばれる）といった新たな問題が発生する．こうした問題に対処するため，本稿ではビル管理システムを形式的に記述する，**ビル管理システムモデル化手法**を提案する．ビル管理システムを，設備機器の集合とそれらによって実現されるサービスシナリオの集合とで性質付け，システムが単体で矛盾なく一貫して動作するための条件を定式化する．より具体的には，(a) 各サービスシナリオが実行可能であること，(b) 矛盾していないこと，および，(c) 任意のシナリオのペアが機能衝突しないこと，という 3 つの条件を定める．さらに，異種システムの連携の際に生じる**サービス競合問題**を検出する手法も提案する．具体的には，矛盾のない単体システムを組み合わせた際に，前述のシナリオの一貫性条件が満たされなくなる場合を，サービス競合と定義する．

また本稿では，我々が先行研究で提案した実用的なビル管理システム [11] に対するケーススタディを行い，10 個の管理システム，33 本のサービスシナリオのモデル化を行った．また，提案した枠組みに基づいて，これらのシステムの一貫性検査，および，サービス競合検出を行うツールを作成した．サービス競合検出においては，466 のシナリオ組み合わせに対して，34 個のサービス競合を自動検出でき，提案手法の有効性が示された．

2. ビル管理システムのモデル化

2.1 ビル管理システムの特徴

ビル管理システムは，ビル内に存在する様々な設備機器を遠隔・自動制御し，ビルの効率的な運用・管理を図るシステムである [8]．その一般的な特徴として，以下の性質を持っている．

(O1) ビル管理システムは，複数の設備機器を使う．

(O2) ビル管理システムは，設備機器を制御するための複数のサービスシナリオを持つ．

次に，ビル内の各設備機器は，管理システムから操作され，機器の内部状態に応じた振る舞いを実行する．従って，各設備・機器は，内部状態を性質づける「属性」と振る舞いをあらわす「操作」とを持つ「オブジェクト」とみなすことができる．

(O3) 設備機器は，属性 (attributes) とメソッド (methods) を持つオブジェクトである．

ビル管理システムは，システムに課せられた運用・管理業務の要求に従って，設備機器オブジェクトのメソッドを実行する．一般に，複数のオブジェクトのメソッドが，あるロジックに従って実行される．このとき実行されるメソッドは実行条件によって異なる．モデル化においては，こうしたシステムの振る舞いを，実行可能なメソッドの実行系列（＝シナリオ）の集合としてとらえる．各シナリオは，そのシナリオで使用される機器（群），シナリオで実行される設備機器メソッドの実行系列，シナリオが実行可能となるための事前条件，シナリオが実行された後の事後条件で性質づけることができる．

(O4) システムの実行シナリオは，使用する機器の集合，メソッドの実行系列，事前条件，事後条件で定義される．

多くの管理システムの場合,「あるシナリオを実行中は，それが完了するまで別のシナリオ実行しないこと」というように，シナリオ間の排他制御を運用でまかなう場合が存在する．

(O5) システムの各実行シナリオは，運用上同時には実行しないシナリオ（群）を持つ．

2.2 システムのモデル化

始めに観測 (O3) に基づき設備機器の定義を行う．

定義 1(設備機器): 設備機器 e は，$e =< A, M >$ で定義される．ここで，

- A は e の属性の集合，
- M は e のメソッドの集合

である．e の任意の属性 $a \in A$ を，$e.a$，e の任意のメソッド $m() \in M$ を，$e.m()$ と表すことがある．

次に観測 (O4),(O5) に基づき，サービスシナリオの定義を行う．

定義 2(サービスシナリオ): サービスシナリオ sce は，$sce =< E, seq, Pre, Post, X >$ で定義される．

- E は，sce で使用される機器の集合，
- $seq = e_1.m_1(); e_2.m_2(); ...; e_j.m_j(); (e_i \in E)$ は，機器メソッドの実行系列，
- Pre は，sce の事前条件で，E の機器の属性を用いて定義される論理式，
- $Post$ は，sce の事前条件で，E の機器の属性を用いて定義される論理式，
- $X = \{sce_1, sce_2, ..., sce_q\}$ $(sce \neq sce_i \ (1 \leqq i \leqq q))$ はシナリオの集合で，sce と同時に実行しないシナリオ (**排他シナリオ**) を現す．

便宜上，シナリオ sce の事前条件，事後条件をそれぞれ，$Pre(sce)$，$Post(sce)$ と書く．定義 1，定義 2 および観測 (O1), (O2) により，ビル管理システムを以下のように定義する．

定義 3(ビル管理システム): ビル管理システム Sys は，$Sys =< E, SCE >$ で定義される．ここで，

- E はシステムが使用するすべての機器の集合，
- $SCE = \{sce_1, sce_2, ..., sce_r\}$ はシステムが持つすべてのサービスシナリオの集合．ただし，各 $sce_i =<$

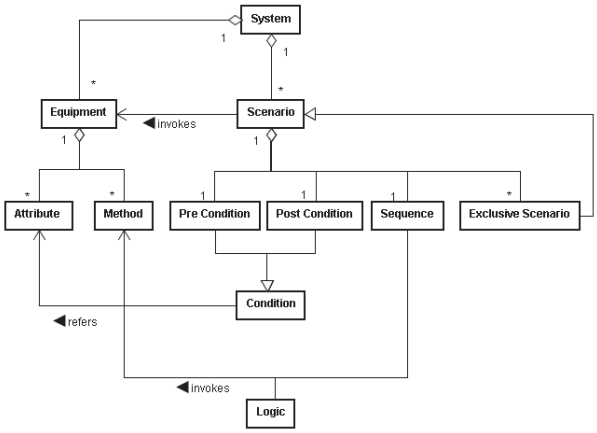

図 1　システムのメタモデル

$E_i, seq_i, Pre_i, Post_i, X_i >$ について，$E_i \subseteqq E$，$X_i \subseteqq SCE$ である．

ビル管理システムモデルの構成要素の構造を，UML クラス図で表すと，図 1 のように表される．

2.3　システムの一貫性

ビル管理システムモデル $Sys =< E, SCE >$ が与えられた時，そのモデルが矛盾なく動作するシステムを表現しているかどうかを考慮することは重要である．ここでは，システムモデルの一貫性 ($= consistency$) の定義を与える．

定義 4(システムの一貫性): ビル管理システム $Sys =< E, SCE >$ が与えられたとする．いま，あるシナリオ $sce =< E, seq, Pre, Post, X > \in SCE$ が以下の条件をすべて満たす時，**シナリオ** sce **は一貫している** ($consistent$) と呼ぶ．

条件 C1(シナリオ実行可能性): Pre が必ず真 (TRUE) になる場合が存在すること．

条件 C2(シナリオ無矛盾性): Pre が真のとき seq を実行すると，seq の実行後に，必ず $Post$ が真 (TRUE) になること．

条件 C3(シナリオ同時実行一貫性): $NX = SCE - X$ を sce の非排他的シナリオの集合とする．任意の $sce^* \in NX$ について，$Pre(sce)$ と $Pre(sce^*)$ が共に真になりうるならば，

- $Post(sce) \wedge Post(sce^*)$ が矛盾しない，かつ，
- $Pre(sce) \wedge Post(sce^*)$ が矛盾しない，かつ，
- $Post(sce) \wedge Pre(sce^*)$ が矛盾しない．

全てのシナリオ $\forall sce \in SCE$ が一貫している時，**システム** Sys **は一貫している**と呼ぶ．

条件 C1 は，そのシナリオが必ず実行されうることを保証している．条件 C2 は，実行された場合，実行後に想定した事後条件が必ず満たされることを保証している．さらに，条件 C3 の 1 番目の条件は，排他的でないシナリオを同時実行しても，実行結果が矛盾しないことを保証している．また，条件 C3 の 2 番目と 3 番目の条件は，一方のシナリオを実行した結果 (Post)，他方のシナリオの実行が妨げられない (Pre) ことを保証している．条件 C3 が満たされないシナリオの組 (sce, sce^*) は，モデル内で排他的シナリオとしてあらかじめ指定しておくべきである．一方，事前条件が同時に成り立たないシナリオは，そもそも排他的に実行されるため，条件 C3 による同時実行一貫性の評価は必要はない．

2.4　システムのモデル化例

提案モデルを用いて，実用的なビル管理システムをモデル化してみる．

2.4.1　入退室管理システム

入退室管理システムは，住人の持つ ID カードや指紋等を用いて，指紋認証などを行うことによって，各部屋への入室を制限するシステムである．また，認証の際にカメラによる撮影や，ドアセンサによる通過の確認を行うことで，入室記録に留まらず，室内の人数情報の把握を行うことも可能である．この情報は管理者側からいつでも参照することが可能である．この他にも，内部からの操作で，外部からの入室を完全に禁止する “警備モード設定” や，認証後に長時間住人が通行しないでいると警報を鳴らす “長期開放警告” などの機能も有する．以上のような機能を有する入退室管理システムは，カードリーダ，電気錠制御盤，ドア，カメラ，警報装置などの設備機器から構成される．カードリーダとカメラはそれぞれ入口側と出口側に 1 つずつあるとする．本節でも同様に，入口側と出口側に 1 つずつのカードリーダとカメラ，1 つの端末，ドア，警報装置システム管理装置を持ち，計 8 個のシナリオを有する入退室管理システムを想定し，モデル化を行い，図 2 に記す．

入退室管理システム：
　ELS(Enter or Leave contorol System)
Equipment(使用機器):
　$E\_ELS = \{CR_1, CR2, door_1, camera_1, camera_2$
　　$, IDC_1, alarm_1\}$
カードリーダ (CR) × 2
　attributes(属性)：
　CR.mode　　//通常 or 警備モード
　CR.lastID　　//最後に通過した住人の情報
　method(操作)：
　CR.getID()　　//カード読み取り
　CR.setSecurity()　　//警備モード設定
　CR.cancelSecurity()　//警備モード解除
ドア (door) × 1
　attributes(属性)：
　door.lock //ドアロック状態，LOCKED，UNLOCKED の 2 つの値をとる
　method(操作)：
　door.lock()　　//施錠
　door.unlock()　　//開錠
　door.checkPass()　　//通過確認
　door.detectPick()　　//こじ開け検知
カメラ (camera) × 1
　attributes(属性)：
　camera.power //カメラ電源
　method(操作)：
　camera.shoot() //撮影する

camera.getCameraData() //camera 内のデータを参照する
電気錠制御盤 (IDC) × 1
attributes(属性)：
IDC.mode //通常 or 警備モード
IDC.personNum //室内の人数
method(操作)：
IDC.updatePersonNum() //人数カウント
IDC.setSecurity() //警備モード設定
IDC.cancelSecurity() //警備モード解除
IDC.refRecord() //IDC 内に記録された ID，通過時間
警報機 (alarm) × 1
attributes(属性)：
alarm.power //警報の作動状態
method(操作)：
alarm.on() //警報作動
alarm.off() //警報解除
を参照する

シナリオ：
(1) 入室シナリオ (ELS_enter)

```
E={CR_1,door_1,camera_1,IDC_1,card_1}
X={ELS_leave,ELS_invalidity,ELS_setSecurity,
  ELS_LTOpen,ELS_pick}
Pre:IDC_1.mode==NORMAL  //警備モードでない
&& camera_1.power==ON //カメラが使用可
&& card_1.ID==MyID1  //住人が通行権限のある ID カードを所持
Post:door_1.lock==LOCKED  //ドアがロックされる
&& CR_1.lastID==MyID1 //通行者の ID 情報が記録される
seq:
CR_1.getID();  //ID 情報読み取り
camera_1.shoot();  //カメラが通行者を撮影
IDC_1.checkID(ID);  //ID の通行権限チェック
door_1.unlock();  //ドアロック解除
door_1.passCheck();  //通過確認
IDC_1.updatePersonNum();  //室内人数カウント
door_1.lock();  //ドア再ロック
```

(2) 退室シナリオ (ELS_leave)

```
E={CR_2,door_1,camera_2,IDC_1,card_2}
X={ELS_enter,ELS_invalidity,ELS_setSecurity,
   ELS_LTOpen,ELS_pick}
IDC_1.mode==NORMAL  //警備モードでない
&& camera_2.power==ON //カメラが使用可
&& card_2.ID==MyID2 //住人が通行権限のある ID カードを所持
Post:door_1.lock==LOCKED    //ドアがロックされる
&& CR_2.lastID==MyID2   //通行者の ID 情報が記録される
seq:
CR_2.getID();  //ID 情報読み取り
Camera_2.shoot();  //カメラが通行者を撮影
IDC_1.checkID(MyID2);  //ID の通行権限チェック
door_1.unlock();  //ドアロック解除
door_1.passCheck();  //通過確認
IDC_1.updatePersonNum();  //室内人数カウント
door_1.lock();  //ドア再ロック
```

ページの都合上，以下略.
(3) 認証不可シナリオ (ELS_invalidity)
(4) 警備モード設定シナリオ (ELS_setSecurity)
(5) 警備モード解除シナリオ (ELS_cancelSecurity)
(6) データ参照シナリオ (ELS_refData)
(7) 長期扉開放シナリオ (ELS_LTOpen)
(8) こじ開けシナリオ (ELS_pick)

図 2　入退室管理システムモデル

2.4.2 防災システム

防災システムは，煙センサ，火災報知機，ドアを連携し，火災発生時に，普段はセキュリティロックのかかっているドアを一斉に開放するシステムである．煙センサ，火災報知機，ドアをそれぞれ 1 つずつ持ち，2 つのシナリオを有する防災システム想定し，モデル化を行う．その結果を図 3 に示す．火災発生シナリオ (DPS_fire) は，火災発生時にドアのロックを一括で解除し，火災報知機を鳴らすシナリオ，鎮火シナリオ (DPS_extinction) は，火災が収まり，再びドアがロックされ，火災報知機が止まるシナリオである．

防災システム：
DPS(Disaster Prevention System)
Equipment(使用機器)：
$E\_DPS = \{smokeSensor_1, fireAlarm_1, door_1\}$
煙センサ× 1 attributes(属性)：
smokeSensor.status //煙検知状態. DETECT,NORMAL の値をとる
method(操作)：
smokeSensor.on() //火災検知
smokeSensor.off() //火災消火後，通常状態に移行
火災報知機 (fireAlarm) × 1
attributes(属性)：
fireAlarm.power //警報の作動状態
method(操作)：
fireAlarm.on() //火災報知機作動
fireAlarm.off() //火災報知機解除
ドア (door) × 1
ELS に同じ.

シナリオ：
(1) 火災発生シナリオ (DPS_fire)

```
E={smokeSensor_1,fireAlarm_1,door_1}
X=none
Pre:smokeSensor_1.status==DETECT //火災感知
Post:door_1.lock==UNLOCKED  //ドアロックが解除される
&& fireAlarm_1.power==ON  //火災報知機が作動
seq:
fireAlarm_1.on();  //火災報知器作動
door_1.unlock();  //ドアロック解除
```

(2) 鎮火シナリオ (DPS_extinction)

```
E={smokeSensor_1,fireAlarm_1,door_1}
X=none
Pre:smokeSensor_1.status==UNDETECT //火災が収まる
&& door_1.lockunlock==UNLOCKED  //ドアロックが解除されている
&& fireAlarm_1.power==ON  //火災報知機が作動中
Post:door_1.lockunlock==LOCKED  //ドアがロックされる
&& fireAlarm_1.power==OFF  //火災報知機が解除
seq:
fireAlarm_1.off();  //火災報知機が解除
door_1.lock();  //ドア再ロック
```

図 3　防災システムモデル

2.5　異種システム連携サービスのモデル化

複数の異なるビル管理システムを連携・統合して，より付加価値の高い**連携サービス** [11] を実現する場合を考える．連携サービスは複数のシステム間にまたがるサービスシナリオ群によって実現される．よって，まず連携するシステム群をひとつの大きなビル管理システムと考え，そのシステム上でのシナリオを考えるとよい．これにより，連携システムも単体システムと同様，提案モデルによって扱うことができる．

定義 5 (連携システム): $Sys_1 =< E_1, SCE_1 >$, $Sys_2 =< E_2, SCE_2 >$ をそれぞれビル管理システムとする．ここで $Sys =< E, SCE >$ が，$E_1 \subseteqq E$，$E_2 \subseteqq E$ を満たすならば，Sys **は** Sys_1 **と** Sys_2 **の連携システムである**という．

定義 4, 5 により，連携システムは，各単体システムの機器を両方使用することができる．さらに，連携システムにおけるサービスシナリオでは，それぞれの単体システムの機器メソッドを組み合わせて実行することができる．これにより「連携サービスシナリオ」が実現される．

定義 6 (連携サービスシナリオ): $Sys_1 =< E_1, SCE_1 >$, $Sys_2 =< E_2, SCE_2 >$ を，それぞれビル管理システムとする．また，$Sys =< E, SCE >$ を Sys_1 と Sys_2 の連携システムであるとする．このとき，Sys のサービスシナリオ $sce =< E, seq, Pre, Post, X >\in SCE$ が以下の条件 I を満たす時，sce **は連携サービスシナリオである**という．

条件 I: sce の実行系列 $seq = e_1.m_1(); e_2.m_2(); ...e_j.m_j()$ について，$e_p \in E_1, e_q \in E_2$ となるような p，q が存在する

3.　サービス競合問題

複数のビル管理システムを同時に運用する場合，または，それらを組み合わせて連携サービスを実現しようとする場合，もともと別々のシステムに存在していたサービスシナリオが機能衝突を起こし，望み通りの振る舞いを実行しない場合がある．この問題は一般に，**サービス競合問題** [3] [6] [7] と呼ばれ，システムの機能不全を引き起こしたり，安全性を阻害する問題として知られている．

3.1　サービス競合の例

2.4 節で述べた 2 つのビル管理システム，入退室管理システム (ELS，図 2)，防災システム (DPS, 図 3) を考える．両システムのモデルは，モデルの定義上一貫しており，それぞれ単体では矛盾なく動作する．

いま，両システムを同じビル内で使用することを考え，火災が発生したと仮定する．このとき，以下のようなサービスシナリオの競合が発生する．

■競合 1 – 火災発生シナリオ (DPS_fire) と入室シナリオ (ELS_enter): 火災が発生したため，DPS がドアのロックを解除する．一方，ユーザが入室しようと ELS の入室シナリオを実行すると，シナリオ実行後ドアがロックされてしまう．

■競合 2 – 火災発生シナリオ (DPS_fire) と退室シナリオ (ELS_leave): 火災が発生したため，DPS がドアのロックを解除する．一方，ユーザが退出に伴って ELS の退室シナリオを実行すると，シナリオ実行後ドアがロックされてしまう．

上記どちらの競合も，火災時に非難経路を遮断してしまうという重大な事故につながる．上記の例ではどちらも単体システムのサービスシナリオにおいて発生した競合である．しかし，こうしたシナリオ間の競合は，連携サービスシナリオにおいても発生することがある．

3.2　サービス競合の定義

3.1 節の例で見たように，サービスシナリオ間の競合は，もともと別々のシステムに存在していたシナリオが同時に実行された場合に発生する．より具体的には，システム A のシナリオ sce_a とシステム B のシナリオ sce_b について，両者を矛盾なく実行することができない，すなわち，**シナリオ同時実行一貫性が満たされない時競合が発生する** (2.3 節，定義 4 条件 C3)．

たとえば，3.1 節の競合 2 を考える．これらのシナリオの事前条件は以下のようになる．

```
Pre(ELS_leave):
   IDC_1.mode==NORMAL      //警備モードでない，かつ，
   && camera_1.power==ON   //カメラの電源が入っており，
   && card_2.ID==MyID2     //権限有カードが通される．
Pre(DPS_fire):
   smokeSensor_1.status==DETECT  //火災感知
```

両シナリオの事前条件は独立であるため，両者は同時に true になることがある．一方，事後条件は以下のようになる．

```
Post(ELS_leave):
   CR_2.lastID==MyID2  //退室者の情報が記録に残り，
   && door_1.lock==LOCKED //ドアが施錠される．
Post(DPS_fire):
   door_1.lock==UNLOCKED      //ドアが開錠され，
   && fireAlarm_1.power==ON  //火災警報機がなる．
```

上記のシナリオの事前条件が共に true になり，両シナリオが同時に実行された場合，ELS はドアを施錠しようとし，DPS は開錠しようとする，すなわち事後条件が矛盾してしまう．これは定義 4 の条件 C3 が満たされないことになる．

一般的に，各サービスシナリオはシステムごとに閉じた範囲内で矛盾なく（すなわち一貫性を持つように）定義される．しかし，複数のシステムを同時に運用したり連携したりする場合，ローカルなシステムで閉じていた一貫性が，連携システム上で保証されるとはかぎらない．このことに着目すると，ビル管理システムにおけるサービス（シナリオ）競合問題は，以下のように定義できる．

定義 7 (サービスシナリオ競合): $Sys_1 =< E_1, SCE_1 >$, $Sys_2 =< E_2, SCE_2 >$ をそれぞれビル管理システムとする．$sce_1 \in SCE_1$ と $sce_2 \in SCE_2$ が，以下の条件 D1 を満たす時，**サービスシナリオ sce_1 と sce_2 は競合する**という：

条件 D1：$Pre(sce_1)$ と $Pre(sce_2)$ が共に真になりえるとき，

- $Post(sce_1) \wedge Post(sce_2)$ が矛盾する，または，
- $Pre(sce_1) \wedge Post(sce_2)$ が矛盾する，または，
- $Post(sce_1) \wedge Pre(sce_2)$ が矛盾する．

Sys_1 と Sys_2 が上記の sce_1, sce_2 のような競合するシナリオ対を少なくとも 1 組持つ場合，**システム Sys_1 と Sys_2 は競合する**という．

定義 7 の条件 D1 は，定義 4 の条件 C3 の否定であることに注意されたい．このことを用いると，システムの競合は 以下のようも定義できる．

定義 8 (システム競合): $Sys_1 =< E_1, SCE_1 >$, $Sys_2 =< E_2, SCE_2 >$ をそれぞれビル管理システムとする．また，ある連携システム $Sys =< E_1 \cup E_2, SCE_1 \cup SCE_2 >$ $(= Sys_1 \oplus Sys_2$と書く) を考える．以下の条件 P が成り立つ時，Sys_1 と Sys_2 は競合する：

条件 P：

- Sys_1 は一貫している，かつ，
- Sys_2 は一貫している，かつ，
- $Sys_1 \oplus Sys_2$ は一貫していない．

上記演算子 $\oplus$ は，二つの異なるシステムを集合和で組み合わせただけの，単純な連携システムを構成する演算子である．定義 7 は，同時に運用する（または連携する）複数のシステム群を，ひとつの大きな連携システムとみなした場合の競合の定義を与えている．すなわち，各単体システムが一貫していても，それらを単純に組み合わせた場合に一貫性が阻害されるという，競合の直感的な定義にも適合する．

3.3 サービス競合の検出

定義 7 にしたがえば，ビル管理システムにおけるサービス競合検出問題は以下のように定式化できる．

[サービス競合検出問題]

入力： ビル管理システム Sys_1, Sys_2

出力： 競合する全てのサービスシナリオ対

Step1： Sys_1 が一貫しているかチェックする．Sys_2 が一貫しているかチェックする．この時点で一貫していなければ，競合検出手続きを中止する．共に一貫していれば次へ．

Step2： $Sys_1 \oplus Sys_2$ が一貫しているかチェック．一貫していなければ，条件 D1 が成立するシナリオのペア $(sce_1, sce_2), (sce_1 \in SCE_1, sce_2 \in SCE_2)$ を出力する．

4. ケーススタディ

提案手法に基づいて，実用的なシステムのモデル化，および，サービス競合検出を行った．

4.1 対象システム

以下の 6 種類のビル管理システム，および，4 種類の連携システムを提案モデル化手法に基づいてモデル化した．また，これら 10 システムに含まれる 33 のサービスシナリオをモデル化した．なお，以下のシステム，およびシナリオは，すべてが実存するものとは限らず，将来的な実現可能性を考慮した上で, [11] において考案したものも含まれている．

照明管理システム (LCS): ビル内の複数の照明機器を一括管理・制御するシステム．照明の点灯，消灯，生存確認等のシナリオを備えている．

ブラインド管理システム (BCS): ビル内の複数のブラインド機器を一括管理・制御するシステム．ブラインドの開閉，生存確認等のシナリオを備えている．

空調管理システム (ACS): ビル内の複数の空調機器を一括管理・制御するシステム．空調の運転開始，停止，運転モード切り替え，温度設定，風量設定等のシナリオを備えている．

入退室管理システム (ELS): ID カードや指紋認証によって，部屋への入退室の権限を制御するシステム．入室，退室，認証失敗，警備モード設定／解除，データ参照，扉長期開放，こじ開け等のシナリオを備えている．

エレベータ管理システム (ECS): ビル内の複数のエレベータ設備を遠隔監視，制御するためのシステム．通常運転，電源切，保守モード設定，遠隔監視，遠隔操作等のシナリオを備えている．

防災システム (DPS): 煙センサ，火災報知機，ドアを連携し，火災発生時に避難経路を確保するシステム．火災発生，鎮火シナリオ等が存在する．

また，以下は連携サービス（システム）である．

天気・省エネ照明サービス (WLS): 照明管理システム，ブラインド管理システム，Web 上の天気情報を連携し，ビル内の照明電力を節電するサービス．天気が晴れの場合には，照明を落としてブラインドを開け自然光を利用することで省エネを図る．

業務開始・終了サービス (OBE): 入退室管理，照明，空調

の 3 システムを連携し，業務開始・終了に伴って機器の開始，終了操作を行う．入退室管理システムで管理している部屋内の人数を監視し，人数に応じて照明・空調を制御する．

エレベータ認証サービス (EAS): エレベータの入り口に設置された入退室管理システムとエレベータ管理システムを連携したサービス．エレベータの入り口でユーザ ID の認証を行い，そのユーザに許可された行先階ボタンのみを選択可能にする．

エレベータ急行サービス (ERS): 各部屋の入退室管理システムとエレベータ管理システムを連携したサービス．ユーザが部屋から退出した際に，空きエレベータをその階に急行させ，ユーザのエレベータ待ち時間を減らす．

4.2 一貫性検査・競合検出ツール

実験のために，システムの一貫性，および，サービス競合を検出するためのツールを，Microsoft Excel VBA を用いて作成した．このツールは，システムモデルを入力すると，それぞれの単一システムの一貫性を定義 4 に基づいて自動的に検査する．さらに一貫性が確認されたシステムの任意のペアを組み合わせ，異なるシステム同士のシナリオ間のサービス競合をチェックする．

図 4 にその出力画面の例を示す．A 列の各行は，比較対象元のシステム名を表し，C,D 列 2 列で一組の塊で，C 列に比較したシナリオ対の名前，D 列に比較した結果を出力している．以降右方向へ，2 列ずつこれが繰り返される．図中のシナリオの比較出力結果はそれぞれ，以下の意味である．

OK: 同時に実行することが可能，

Pre 同時に成立せず： 事前条件が同時に成り立たないため，同時に実行されることはない，

Post/Post 同時に成立せず： 事前条件は同時に成り立つが，事後条件に矛盾が生じる，

Pre/Post，Post/Pre 同時に成立せず： 一方のの事前条件と他方の事後条件の間に矛盾が生じる．

例えば，2 行目の C，D 列を見ると，シナリオ「LCS_allOn」と「BCS_allOpen 」が「OK」なので，同時に実行することが可能であるということが分かる．また，7 行目の C，D 列を見ると，シナリオ「LCS_allOn」と「WLS_brightOut 」が「Post/Post 同時に成立せず」なので，同時に実行すると事後条件に矛盾が生じるため，システム間にはサービス競合が発生することが分かる．

4.3 実 験 手 順

実験は以下のステップに従って行った．

Step1: システムの一貫性チェック

6 個のシステム，4 個の連携システムが持つ，全 33 本のシナリオを対象に，各システムの一貫性のチェックを行った．ツールの実行により，4 個のシナリオに問題があることがわかった．例として防災システムには以下の様な一貫性欠如の問題があった．

当初の，防災システムの火災発生シナリオ「DPS_fire」は，Pre(DPS_fire) が “TRUE” になっており，鎮火シナリオ「DPS_extinction」と同時に実行できる状態になっていた．Post がそれぞれ異なるため，このままではシステムに一貫性はない．現実的にも，火災が発生していない状況と発生している状況が同じ場所で起こることはありえないため，両シナリオが同時に実行されることはありえない．現在は Pre(DPS_fire) に修正が加えられ，防災システムの一貫性は保たれている．

Step2: サービス競合の検出

Step 1 で確認した 10 個のシステム (連携システムを含む) の任意のペアにおけるサービスシナリオ間のサービス競合を検査した．

4.4 競合検出結果

競合検出実験は，ミドルクラスのデスクトップ PC（Intel-Core2Solo 2.40GHz, 2GB RAM, Windows XP SP2）を用いて行った．競合検出に要した実行時間は約 40 秒であり，実用上支障ない時間で検出が行えることを確認した．

競合検出の対象となったサービスシナリオの全組み合わせは 466 通りであり，34 種類のサービス競合が検出された．表 1 に実験の結果を示す．表の各項目は，異なるシステムの組み合わせにおいて，競合が発生したシナリオ組み合わせ数を表している．結果を分析したところ，想定した全ての競合を検出できたことを確認した．

4.5 検出された競合の例

以下に検出された興味深い競合例を紹介する．

■競合 3 – ブラインド制御システムの全閉シナリオと天気・省エネ照明サービスの晴天シナリオ: 晴天時，天気・照明サービスの晴天シナリオは，全ての照明を消し，全てのブラインドを開け，陽光を取り入れようとする．一方でブラインド管理システムの全ブラインド閉シナリオが実行されると，全てのブラインドが閉じられてしまう．天気が良いときは，外の陽光を取り入れることで省エネを行えるが，窓側にいる住人にとっては陽光がきつくて眩しいためブラインドを閉める，という状況が発生した場合，本競合が起こる．

■競合 4 – エレベータ認証サービスの認証シナリオとエレベータ急行サービスの籠急行シナリオ: 待機状態だったエレベータに対し，住人が認証シナリオを行ったことで，エレベータが輸送状態になる．一方で，住人が退室したときに，エレベータの待ち時間を減らすため，待機状態のエレベータを呼ぼうとするが，エレベータはすでに輸送状態であるため，呼ぶことができないという競合が発生する．また，認証を行おうとしたエレベータが，籠急行シナリオによって呼ばれてしまう競合も同様に考えられる．

■競合 5 – 空調管理システムの温度設定シナリオと業務開始・終了サービスの業務終了シナリオ: 管理者側が，省エネのために，空調の温度を設定しようとする．一方で，最後の住人が室内から退出してしまい，照明，空調の電源がオフになる．このような，機能的には競合であるが，省エネという観点から考えると競合とは呼べない例があるということは非常に興味深く，今後はこのような例を分類する枠組みも必要であると考えられる．

5. ま と め

本稿では，ビル管理システムの開発の複雑化や一貫性の保証などの問題に対処するために，ビル管理システムのモデル化手

	A	B	C	D	E	
1						
2	照明管理システム	ブラインド管理システム	LCS_allOn-BCS_allOpen	OK	LCS_allOn-BCS_allClose	OK
3		空調管理システム	LCS_allOn-ACS_allOn	OK	LCS_allOn-ACS_allOff	OK
4		エレベータ管理システム	LCS_allOn-ECS_allOn	OK	LCS_allOn-ECS_allOff	OK
5		入退室管理システム	LCS_allOn-ELS_enter	OK	LCS_allOn-ELS_leave	OK
6		防災システム	LCS_allOn-DPS_fire	OK	LCS_allOn-DPS_extinction	OK
7		天気照明サービス	LCS_allOn-WLS_brightOut	Post/Post同時に成立せず	LCS_allOn-WLS_darkOut	OK
8		業務開始・終了サービス	LCS_allOn-OBE_begin	OK	LCS_allOn-OBE_end	Post/Po
9		エレベータ急行サービス	LCS_allOn-ERS_rapid	OK	LCS_allOff-ERS_rapid	OK

図 4　ツール実行結果画面

表 1　競合検出実験結果

	LCS	BCS	ACS	ELS	ECS	DPS	WLS	OBE	EAS	ERS
LCS		--	--	--	--	--	2	2	--	--
BCS			--	--	--	--	2	--	--	--
ACS				--	--	--	--	5	--	--
ELS					--	6	--	4	--	--
ECS						--	--	--	2	2
DPS							--	4	--	2
WLS								2	--	--
OBE									--	--
EAS										1
ERS										

法の提案を行った．この手法に基づき，10 個の管理システム，33 本のシナリオに対してモデル化を行った．また，ビル管理システムを連携する際に考慮すべき問題のひとつとして，サービス競合問題が挙げられる．本稿では，提案したシステムモデル化手法を利用して，サービス競合の定式化を図った．それに伴い，システムの一貫性の検証，サービス競合の検出を行うツールを作成した．6 個のシステム，4 個の連携システムが持つ 33 本のシナリオに対して，システムの一貫性の検査，サービス競合の検出を行った．サービス競合検出の結果，466 組のシナリオ組み合わせの中から，34 個のサービス競合ケースを検出できた．

謝　　辞

本研究は，科学技術研究費 (若手研究 B 18700062，基盤研究 B 17300007，若手研究スタートアップ 18800060)，日本学術振興会日仏交流促進事業（SAKURA プログラム），および，三菱電機株式会社の助成を受けて行われている．

文　　献

[1] BACnet, http://www.bacnet.org/
[2] Echelon Corp., “LonWorks ネットワークテクノロジとは,” http://www.echelon.co.jp/products/lonworks.html
[3] “Feature Interactions in Telecommunications and Software Systems,” Vol.I-VIII, IOS Press, 1992-2005.
[4] 井垣 宏, 中村 匡秀, 石井 健一, 串戸 洋平, 松本 健一, “家電連携サービスにおけるサービス競合の動的な検出・解消法の設計と評価,” 信学技報, 情報ネットワーク研究会, Vol.IN2004-320, pp.373–378, Mar. 2005.
[5] 木村 隆洋, 中村 匡秀, 井垣 宏, 松本 健一, “データ依存解析に基づくレガシーソフトウェアからのサービス抽出法,” 信学技報 ソフトウェアサイエンス研究会, Vol.SS2005-42, pp.13-18, Oct. 2005.
[6] M. Kolberg, E. H. Magill and M. Wilson, “Compatibility Issues between Services Supporting Networked Appliances,” IEEE Communications Magazine, Vol.41, No. 11, pp.136-147, Nov. 2003.
[7] A. Metzger and C. Webel, “Feature Interaction Detection in Building Control Systems by Means of a Formal Product Model,” Feature Interaction in Telecommunications and Software Systems VII, pp.105-121, 2003.
[8] 三菱電機ビルテクノサービス, http://www.meltec.co.jp/
[9] M. Nakamura, A. Tanaka, H. Igaki, H. Tamada and K. Matsumoto, “Adapting Legacy Home Appliances to Home Network Systems Using Web Services,” IEEE International Conference on Web Services (ICWS 2006), pp.849-858, Sept. 2006.
[10] M. P. Papazoglou and D. Georgakopoulos, “Service-Oriented Computing,” Communications of the ACM, Vol. 46, No.10, pp.25-28, Sept. 2003.
[11] 西澤 茂隆, 中村 匡秀, 井垣 宏, 松本 健一, 三浦健次郎, “ビル管理システムにおけるサービス指向アーキテクチャの適用〜異種サービスの連携と安全性に関する考察〜,” 電子情報通信学会技術研究報告, Vol.107, No.261, pp.3-8, October 2007.
[12] 豊田 武二, “BACS とビル安全安心とのかかわり,” REAJ 誌, Vol.28, No.6(通巻 154 号), 2006.
[13] Web Services Activity, http://www.w3.org/2002/ws/
[14] Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/wsdl, Mar. 2001.
[15] What Is Service-Oriented Architecture, http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html, Sept. 2003.