情報処理学会論文誌 Vol. 52 No. 1 220–232 (Jan. 2011)

物理的相互作用に着目したスマート空間の形式仕様記述と検証

石川 冬樹[†1] 山本 佳代子[†2] 本位田 真一[†1,†3]

開発プロセスの早期においては，不整合や抜け等の誤りが後の過程に引き継がれ手戻りを引き起こすことを避けるため，系統的なモデル化・分析を行うことが信頼性，効率性の観点から重要である．しかし従来のソフトウェア開発に比べて，スマート空間の開発については早期のモデル化・分析手法が確立されていない．本論文においては，スマート空間仕様の早期モデル化と，モデル検査によるその検証のためのフレームワークを提案する．提案フレームワークにおいては，ユーザとデバイス間の視覚，聴覚等の物理的な相互作用の抽象モデル化に基づいたスマート空間仕様記述言語を提供する．また，特に計算機により制御できないユーザの存在を考慮した検証項目について，既存ツール SPIN を用いたモデル検査を行う仕組みを提供する．また，抽象的な仕様の分析と修正・追加をインタラクティブに，また段階的に繰り返す過程における提案フレームワークの有用性について，ケーススタディを通して確認した．

Formal Specification and Verification of Smart Spaces: Focusing on Physical Interactions

Fuyuki Ishikawa,[†1] Kayoko Yamamoto[†2] and Shinichi Honiden[†1,†3]

Systematic modeling and analysis methods in early phases of the development process are essential for reliable and efficient system development, in order to prevent deficiencies from being inherited to later phases and causing costly rollback. However, such methods have not been intensively explored for smart spaces, compared with traditional software engineering. This paper proposes a framework for early modeling of smart spaces specifications as well as verification by model checking. The framework includes a modeling language for smart space specifications based on abstract modeling of physical interactions between users and devices, e.g., visual or audio interactions. The framework also includes a mechanism for model checking of the specifications, against specific properties that reflect uncontrollability of users, through translation to inputs for an existing tool, SPIN. Case studies are conducted to discuss usefulness of the proposed framework in interactive analysis and modification of abstract specifications.

1. はじめに

ユビキタス・パーベイシブコンピューティングは，周辺環境に埋め込まれたセンサ・アクチュエータを，しばしばユーザの意識なく活用しユーザを支援することを目指すものとして広く注目されてきた[1),2)]．これらのパラダイムにおいては，ユーザと環境の間の，視覚，聴覚，触覚，はては嗅覚等に基づく物理的な相互作用がシステムの中心的側面となり，スマートホームからヘルスケアまで様々な適用対象が議論されている．これらパラダイムに対するさかんな取り組みにより，それらスマート空間の設計，実装ためのミドルウェアやフレームワーク，また先進的な物理的機能を提供する仕組みやデバイスが広く提案されている．

しかし，スマート空間はいまだ「特製」であり，実際にはその名のとおり「ユビキタス（いつでもどこでも）」となるには至っていない．その 1 つの理由として，一般的なソフトウェアに対する工学的手法が，スマート空間の開発にどう適合，適用できるかが定かでないことがあげられる．具体的には，開発の早期過程における，実装詳細を捨象しての，系統的なモデル化，分析手法が広く議論されていない．従来のソフトウェア工学においては，そのような早期過程における手法は非常に重要だと見なされてきた．なぜならば，早期過程において埋め込まれたが発見されずに引き継がれた誤り（不整合，抜け，誤解等）は，開発プロセスにおける手戻りを引き起こし大きなコスト増加につながるためである．スマート空間開発においては，物理的な環境の配備を要するために手戻りのコストが非常に大きく，上記の点はより留意されるべきである．

この問題に対し本論文では，信頼性，効率性のため数理的技法に基づいたシステム開発を行う形式手法の適用，適合について議論する[3),4)]．形式手法においては，開発の早期における仕様等の成果物に対し，抽象的なモデルを構築し，科学的・系統的な分析を行うことを考える．本論文では，抽象モデル化と分析というこれら 2 つの側面について，スマート空間の

†1 国立情報学研究所 GRACE センター
GRACE Center, National Institute of Informatics
†2 日立製作所
Hitachi, Ltd.
†3 東京大学
The University of Tokyo

固有の要求をふまえ以下のように取り組む．

第 1 に，「システムをどう実装するか」ではなく「システムが何をするか」に注目，分析する早期過程のための抽象的なモデル化手法を検討する．ここでスマート空間においては，ユーザの状況においてディスプレイやスピーカ等のデバイスを起動しユーザの活動を支援することを考える．このため，仕様や初期設計においては，「見る」「聞こえる」といったユーザとデバイス間の物理的な相互作用に関する要求について注意深く分析する必要があり，そのような物理的相互作用を主とした抽象モデル化が必要となる．

第 2 に，特定の状態でのみ顕在化する誤りを検出するために状態遷移の網羅的な検索を行うモデル検査の適用を検討する[5),6)]．スマート空間，およびそれを構成するソフトウェアにおいては，多様な状況に対ししばしば自発的に対応することが求められるため，抜け，漏れの発見のため網羅的な検査の重要性が高い．ここで，ユーザの自由な行動やシステム提案に対する許否が存在するため，計算機内の動作を正しく設計しても「決して起きない」「必ず起きる」といった性質を保証することができないことも多い．このため，ユーザの行動を考慮した検証項目の設定が必須であり，その記述パターンを支援する必要もある．

上記の議論をふまえ本論文では，スマート空間仕様の早期モデル化と，モデル検査によるその検証のためのフレームワークを提案する．具体的には，ユーザとデバイス間の物理的な相互作用の抽象モデル化に基づいたスマート空間仕様記述言語を提供する．また，特に計算機により制御できないユーザの存在を考慮した検証項目について，既存ツール SPIN[7)] を用いたモデル検査を行う仕組みを提供する．提案フレームワークを用いることにより，スマート空間システムの仕様，特に状況に対応して物理的な相互作用を行う複雑な振舞い仕様を，センサ等の実装詳細を捨象した形で抽象的に，かつ直接モデル化・記述し，モデル検査を通して系統的な検証を行うことができる．これにより，仕様の分析とそれに応じた修正をインタラクティブに，また仕様項目の追加を段階的に行うことができ，開発の早い段階で仕様の品質を向上させることができる．提案フレームワークのこの有用性については，ケーススタディを通して議論する．

以下ではまず 2 章において，簡単な例題を通してスマート空間仕様のモデル化と検証に関する要件について議論する．その議論を受けて，3 章では，提案フレームワークについて説明する．4 章では，提案フレームワークの活用についてのケーススタディを示す．5 章では，提案フレームワークの利点と限界，また関連研究について議論し，6 章ではまとめと今後の方向性について述べる．

2. スマート空間仕様のモデル化と検証

2.1 簡易例題

スマート空間の例として本論文ではスマートオフィスシステムを考える．このシステムにおいては，ディスプレイやスピーカ等を用いてプレゼンテーション，デモンストレーション，議論，社員による一時的な利用等を支援する（**図 1**）．このようなスマート空間システムにおいては，しばしばユーザによる明示的な指示がなくてもプレゼンテーション起動等の支援を行う一方で，望ましくない状態を防ぐことが期待される．

本章では，本論文の位置づけを議論するため，スマートオフィスシステムの 1 つの利用事例に対応する仕様について議論する．この仕様部分は，共有ディスプレイの社員による利用に関するものである（**図 2**）．図 2 においては，システムに関する要求 **R1**，**R2** とそれに対応する振舞い **B1**，**B2** が記述されている．これは非常に単純な例題であるが，以下で議論するようにスマート空間のモデル化，分析について本質的な側面を含んでいる．

2.2 モデル化

2.2.1 抽象化

開発の早い段階においては，「システムをどう実装するか」よりも「システムが何をするか」に注目し，記述，分析を行うべきである．この点に関し 2.1 節の例においては，センサの種類や設定，コンポーネント分割や配備に関する記述を行っていない．このような抽象化は，人間の開発者にとっても，支援ツールにとっても，集中による効率化のために非常に重要である．しかし 2.3 節で議論するように，この抽象度の高い記述においても，後の過程に

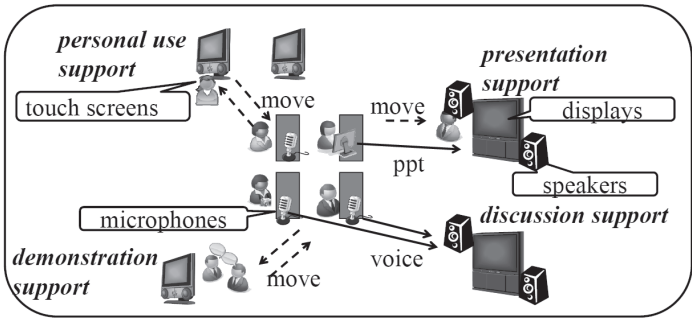

図 1　スマートオフィスシステム
Fig. 1　Smart office system.

R1 権限のあるユーザは，近くにあるディスプレイを用いて社内共有情報を見ることができる．

　B1 ディスプレイの近くにおり（それを見ることができ）かつ権限を持つユーザからの要求を受け，システムはそのディスプレイに要求された社内共有情報を表示する．

R2 社内共有情報は，権限を持たないユーザに見られることはない．

　B2 システムは，あるディスプレイに社内共有情報を表示しているとき，権限を持たないユーザが近くに来たならば（それを見ることができるようになったならば）その表示を停止する．

図 2　誤りを含むスマート空間仕様記述の例
Fig. 2　Example specification with deficiency.

引き継がれるべきではない誤り（曖昧さ，不整合，抜け）が含まれている．

2.2.2　物理的相互作用

スマート空間の機能は，反応的または自発的な，物理的な相互作用の有効化・無効化を通して説明することができる．上記の例においては，視覚に関する相互作用が含まれている．ほかにも聴覚や触覚による相互作用が一般的であり，将来的には嗅覚等も含まれるかもしれない．

2.1 節の例はさらに，「見る（聞く，触る）ために十分近くにいなければならない」という，ユーザとデバイス間の物理的な相互作用が有効に働くための必要条件の存在も示している．一方でこの条件は十分条件でもあり，ユーザが近くにいるならば望む望まないにかかわらずユーザは見る（聞く）ことができてしまう．これらの条件の存在により，ユーザが望ましい相互作用を持てない，または望ましくない相互作用を持ってしまう，という不整合が生じうる．このため 2.3 節で議論するように，そういった不整合の可能性を排除するような注意深い分析が求められる．

2.3　分析・検証

2.3.1　誤　　り

2.1 節の例において，この仕様記述は抜けのあるものである．なぜならば，振舞い **B1** と **B2** によって，要求 **R2** が満たされないためである．具体的には振舞い **B1** において，権限のないユーザが近くにいる場合であっても，社内共有情報の表示が始められてしまう．このため，たとえば，要求 **R2** を優先し，「近くに権限のないユーザがいなければ」という制約を振舞い **B1** に追加する修正を行う必要がある．

上記の抜けは，要求 **R1** と **R2** がそもそも両立しないことを示しており，**R1** も同様に修正を行う必要がある．このような漏れは，**R1** を厳密に，「つねに」と添えられた命題として明確化したり，数理的な検証を行ったりすれば顕在化するものである．しかし感覚的に読んでしまうと見逃してしまう，または問題ないかのように解釈し読み替えてしまう可能性がある．

上記のように，利用の可否等相反する要求の衝突，特定の状況に対応する振舞いの抜けといった誤りが，このような抽象的なレベルの仕様においても存在し，より具体的な設計段階に引き継ぐ前に注意深い分析を行うべきである．

2.3.2　モデル検査適用における課題

上記で議論した誤りは，「権限のないユーザが近くにいる場合に，権限のあるユーザが社内共有情報の表示を指示する」という特定の状況のみにおいて顕在化するものである．このため，シミュレーションやテストにおいては検出されない可能性があり，網羅的に状態探索を行うモデル検査の適用が有効である[5),6)]．

しかし，モデル検査ツールにおける記述言語は状態遷移を書くプリミティブなものであり，スマート空間の仕様記述を一から記述するのは負荷が大きい．このため，物理的な相互作用等，スマート空間固有の概念に基づいたモデル化，記述を行い，それに基づきモデル検査を行えることが望ましい．

加えて，モデル検査にあたっては検証項目の設定も必要となる．たとえば，直感的な意味を持つ検証項目パターンの一例として，「望ましくない状態に決して到達しない」という「安全性」があげられる．要求 **R2** はこの安全性の一例のように思える．しかし実際には，権限を持たないユーザが近づいてしまったときに，対応して表示が消える前に情報を見てしまう可能性がある．このようにスマート空間においては，計算機の振舞いにより制御できない，ユーザの自由な行動（移動や拒否等）が存在するため，望ましくないが確実に防ぐこともできない状況がしばしば存在する．すると，要求 **R2** は厳密には「望ましくない状況に対していつか対応する動作がとられ，その状況が解消する」という性質として記述されるべきである（検証項目パターンとしては「活性」となる）．ただし，「権限のないユーザはいつか離れるから（見られてしまっている状況は解消する）」ということでは検証の意味がないため，厳密には「もしユーザが何もしなくてもシステムが対応する」という性質を定めて検証する必要がある．この例のように，計算機の振舞いにより制御できない部分，すなわちユーザの存在を明示的に扱うような，検証項目パターンを検討する必要がある．

3. スマート空間仕様の記述・検証フレームワーク

本章では 2 章の議論をふまえ，本論文の提案であるスマート空間仕様の記述および検証のためのフレームワークについて説明する．

3.1 概　　要

図 3 は提案フレームワークの概要を示したものである．このフレームワークを用いる場合，まずスマート空間の仕様を，物理的な相互作用に基づいた抽象モデル化に基づき，提供された言語を用いて記述する．続いてモデル検査を用いた検証の設定として，空間内のユーザの数やそれぞれの持つモバイルデバイスについて設定を行うとともに，検証項目となる性質を時相論理を用いて，または提供されたパターンを用いて記述する．それらの記述はモデル検査器の入力である，状態遷移モデルと時相論理式に変換される．

3.2 抽象モデル化の方針

スマート空間の仕様においては，ユーザがデバイスと物理的な相互作用を行うことができる（見える，聞こえる等）範囲を，仮想的なスコープとして表現する[8]．いい換えると，

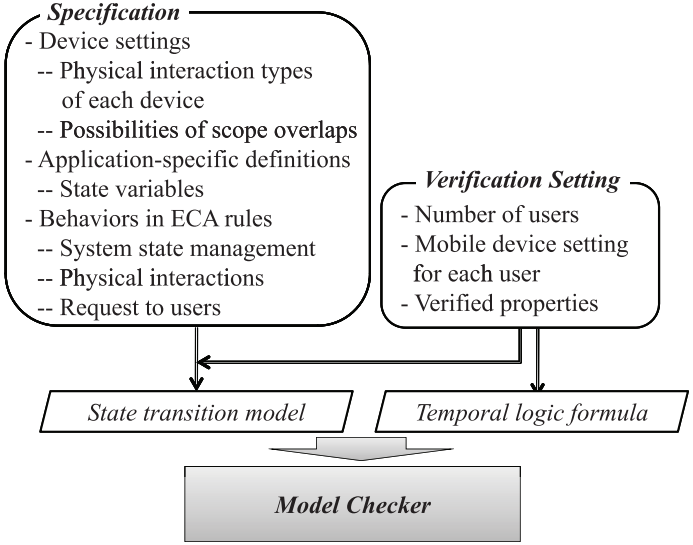

図 3　提案フレームワーク概要
Fig. 3　Overview of the proposed framework.

「あるユーザがあるデバイスと物理的相互作用を行う（見る，聞く等）ために十分近くにいる」ということを「スコープの中にいる」という形でモデル化する．スコープとユーザの関係，すなわち「あるユーザがあるデバイスを見ることができる・できない」といった条件は，ユーザごと，デバイスの持つ物理的な相互作用の種類ごとに識別される．このモデル化においては，そのスコープが実際にどう物理的に構成されるか，センサにより識別されるかについては捨象している．

図 4 は，このスコープに基づいたモデル化について 3 つの例を図示したものである．

- 左の図は 2 章において議論した例を表しており，あるユーザが情報を見ているときに他のユーザもそれを見えるようになる状況を示している．
- 真ん中の図においては，ユーザが 2 つのデバイスの音を同時に聞く可能性がある状況を表している．たとえば，スマートホームシステムにおいて，レシピの読み上げ機能の実行中に別の音楽再生機能が起動してしまう（互いのノイズになってしまう）という不整合の可能性を表現することができる．
- 右の図においては，ユーザが 2 つのデバイスを両方見ることができる状況を表している．たとえば，スマート博物館システムにおいて，ユーザが異なるガイドメッセージを同時に見てしまうという不整合の可能性を表現することができる．

以上では不整合の例に触れたが，「同時に見える」といった状況が望ましい状況である場合もある．いずれにしてもこのように，ユーザとスコープの関係，スコープどうしの関係（重なりの可能性）についてモデル化，分析することにより，望ましい状況や望ましくない状況について明示化，分析することができる．

また，提案フレームワークにおいてはシステムの振舞いを ECA ルールを用いて記述する（Event-Condition-Action）[9]．ECA ルールはイベント駆動型の動作を記述するのに適しており，ユビキタスコンピューティングにおける振舞い記述にもよく用いられてきてい

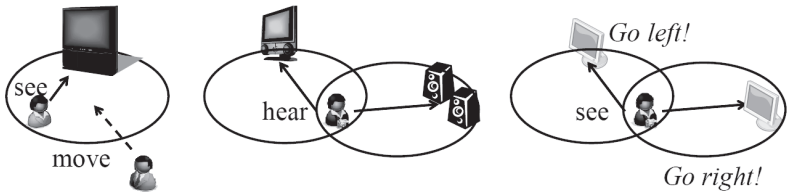

図 4　物理的相互作用のためのスコープのモデル化
Fig. 4　Modeling of scopes for physical interactions.

表 1　提案仕様記述言語の構成要素
Table 1　Constructs of the proposed specification language.

種別	要素	記述形式
環境設定	各デバイスの情報	各デバイスの識別子と提供する物理的相互作用の種類の集合（*visual*，*audio*，*tactile* 等）
	スコープ重なりの可能性	重なりえないスコープの組，包含関係にあるスコープの組
振舞い設定	システムの振舞い	下記要素により構成される ECA ルールの集合
	イベント	ユーザとスコープの関係の変化またはユーザによるシステムへの依頼発生
	条件	ユーザとスコープの関係，デバイスの稼働状態（on/off），ユーザの許否，アプリケーション固有の状態変数に関する論理式（その命題論理オペレータ $\wedge$，$\vee$，$\neg$ による組合せ）
	動作	デバイスの稼働状態変更（on/off），ユーザへの許否問合せ，アプリケーション固有の状態編に関する代入（それらの，原子的実行または非決定的実行オペレータによる組合せ）
アプリケーション固有要素定義	アプリケーション固有の列挙型定義	各列挙型に対して取り得る値の集合
	アプリケーション固有の状態変数定義	変数の識別子，型，初期値
アノテーション定義	あるユーザの許否によってその成功が定まるような動作への参照情報	そのような動作へのアノテーションとして用いるラベルの名前

る．一方で，整合性や完全性の観点から注意深い分析が必要であることもよく知られている[10),11)]．ECA ルールの記述においても，スコープに関連した記述を行うことができる．すなわち，E（Event）については「ユーザが該当スコープに入ったときに」といった記述，C（Condition）については「ユーザが該当スコープにいるならば」といった記述，A（Action）については「ユーザに該当スコープに入るように頼む」といった記述を行うことができる．

3.3　仕様記述言語

3.3.1　全　体　像

表 1 に，提案フレームワークにおける仕様記述言語の構成要素一覧を示す．仕様記述においては，デバイスやその物理的相互作用に関する定義を環境設定として，および，ユーザの行動に関連するアノテーションやアプリケーション固有の状態変数を含んだ ECA ルールを振舞い設定として記述する．以下では各項目について記述例を通して説明する．

3.3.2　環 境 設 定

仕様記述の一部として，デバイスやそれが提供する物理的相互作用に関する仮定を明示化するため，環境設定が与えられる．まず各デバイスの情報として，識別子と提供される物理的な相互作用の種類の集合が以下のように与えられる．

```
Devices = {
  dev0 : [visual],
  dev1 : [audio],
  ...
}
```

加えて，スコープの重なりの可能性が以下のように与えられる．

```
DisjointScopes = {
  ([dev0, audio], [dev1, audio]),
  ...
}
IncludingScopes = {
  ([dev0, audio], [dev0, visual]),
  ...
}
```

上記において 1 つ目の項目においては，デバイス *dev0* と *dev1* から同時に音が聞こえることはありえないという仮定を明記している（重なりえないスコープの明記）．また 2 つ目の項目においては，*dev0* の表示を見ることができるならば，必ずそれからの音を聞くこともできるということを明記している（スコープの包含関係の明記）．このように，デバイスの仕様やその配置に関する要求も明示的に与える．

3.3.3　ECA ルールによる振舞い設定

システムの振舞いは ECA ルール，すなわち「イベント E が起きたときに，条件 C が成り立つならば，動作 A を行う」という形式で与えられる．下記の例は，「ユーザ u が $dev0$ を見るために十分に近くに来たときに，$dev0$ が稼働していなければ，それを稼働させ $content0$ を表示させる」というルールを示している．

```
eca1{
  ON getInScope(u, dev0, visual)
```

```
  IF isOff(dev0)
  DO on(dev0, content0)
}
```

このルールにおいて，*getInScope*，*isOff*，*on* は提案フレームワーク（における仕様記述言語）に含まれる語彙である．スコープやデバイスの稼働状態に関するイベント，条件，動作記述については同様な語彙が提供されている．

ECA ルールの動作（Action）部分においては，複数の動作の原子的実行を指定することもできる．たとえば，あるイベントに対して複数の対応動作を行わなければならない場合を考える．このとき，「正しく複数の対応動作を起動したが，一部の対応動作が完了したものの残りは完了していない」ような中間状態も存在しうる．そういった中間状態を「不整合」と見なすことが意図に合わないのであれば，複数の動作が順次実行されるという側面を捨象し，原子的実行と見なすことができるということである．このように，「性質の検証となる状態はどれか」という観点での原子的実行は，形式的な抽象モデルに対する検証において一般的である．

また，複数の動作を非決定的に実行するような動作記述も指定することができる．たとえば，仕様の時点ではとりうる動作の可能性のみを列挙しておき，実装では知的機能を用いて適切な動作を選ぶ仕組みを導入することが考えられる．この場合でも仕様の段階において，単に非決定的な動作と見なし，どの動作を選んでも不整合が起きないことを確認することができる．

アプリケーション固有の列挙型や状態変数の定義も行うことができる．現在この記述は，変換先のツールである SPIN における記述言語 PROMELA の文法，記述能力に従っている（実装については 3.6 節で述べる）．

3.3.4　ユーザの振舞い

成否がユーザに依存するような動作，具体的にはユーザへの移動依頼や，許否の問合せを明示的に識別するために，アノテーションとしてのラベルを付加することができる．下記の例においては，ユーザがスコープから出たときに，そのスコープの中に戻るように依頼するルールを記述している．もしもユーザがそれを受け入れた場合結局状態は変化しないためシステムは何もせず，そうでない場合にはシステムはデバイスを切る．

```
eca1{
  ON getOutScope(u, dev0, visual)
  IF isOn(dev0)
  DO
    :: req1/accept: getInScope(u, dev0, visual)
    :: req1/deny: off(dev0)
}
```

起きうる状態遷移の表現という観点では，上記は単に 2 つの状態遷移の非決定的な実行を表現しているにすぎない．一方で検証を行う際には，「ユーザが拒否しなければ · · · が保証できる」といった性質を記述するためにこのようなラベルを利用することができる．

3.4　検証に関する設定

3.4.1　ユ　ー　ザ

検証設定としては，ユーザの数をまず与える．モデル検査は演繹的な検証方式ではないので，特定の数のユーザが様々な行動をとったときに起こりうる状態を網羅的に探索，検査することとなる．このユーザ数は，生成される状態遷移モデルに反映される．

加えて，各ユーザにモバイルデバイスを割り当てることもできる．提案フレームワークでは，モバイルデバイスは指定された種類の物理的相互作用を持ち，それらのスコープにつねに必ずそのユーザを含むようなものとして定義される．

3.4.2　検 証 項 目

検証設定としては加えて，検証項目の指定も行われる．ここで，一般的なモデル検査器同様に時相論理による式指定を行うことができる．本論文では，現在の実装で利用している SPIN ツールが用いている時相論理のクラスである LTL（Linear Temporal Logic）についてのみ議論する．もう 1 つの一般的なクラスであるる CTL（Computational Tree Logic）についての同様な議論は省略する．LTL においては，一般的な論理積等に加え，「いつか（$\mathbf{F}$）」「つねに（$\mathbf{G}$）」といったオペレータを用いて論理式を記述する．モデル検査器においては，どのような起こりうる実行パスにおいてもその性質が成り立つかどうかを検証する．

提案フレームワークにおいてはさらに，LTL による性質記述のパターンも利用することができる．2.3.2 項で議論したように，「決してある状態 P にならない」といった安全性（$\mathbf{G}\neg P$）は，スマート空間においてしばしば保証することができない．提案フレームワークにおいてはこれは「強い防止性」と呼び換え，「ある状態に到達してもその状態はいつか成り立たなくなる」という性質を「弱い防止性」パターンとして導入している（$\mathbf{G}(P \Rightarrow \mathbf{F}\neg P)$）．この「弱い防止性」に関しては加えて，「ユーザがいつか解決してくれる」のでは意味がないので，利用時にはあわせて，ユーザが無限にその場にとどまる可能性も状態遷移モデルに含めるようにしている．これにより，「ユーザが何もしなくてもシステムが反応的に対応する」ことを検証することができる．

また，3.3.4 項で述べたラベルを用い，ユーザの許否に依存するような性質を記述することができる．つまり，「ユーザがシステムの移動依頼や，機能起動に対する許否問合せを拒否しなければ」といった仮定の下での性質検証を行うことができる．ユーザによる拒否が発生した状態を A とすると，それが起きないという仮定の下での性質 P の検証は，$(\mathbf{G}\neg A) \Rightarrow P$ と表現される．実際には A においては，ラベルを指定することとなる．

加えて提案フレームワークにおいては，多くの場合アプリケーションによらず用いることができるであろう，「複数の音をユーザが同時に聞く（スコープ内におりかつデバイスが稼働中）ことがない」といった検証項目の生成機能も定義している．

3.5　状態遷移モデルとしての表現方針

以上で述べた仕様記述およびユーザに関する設定により，スマート空間において起きうる状態遷移モデルが定義される．この状態遷移は以下の方針で構築される．

- 状態遷移モデルは，EventProducer と EventHandler という 2 つの並行プロセスを含んでいる．実際には，これら 2 つのプロセスの表す状態遷移（オートマトン）の並行合成によりシステム全体の状態遷移が構成されることとなる．
- EventProducer プロセスは，ユーザの移動，すなわちスコープに対する出入り等のイベントを非決定的に起動していく．ユーザの移動は，スコープ間の関係を考慮して自由に起きうる．たとえば，「あるユーザがあるスコープの内部にいるときにさらに別のスコープの内部に入る」というイベントは，それらのスコープが重なりを持つときのみ発生しうる．また，3.4.2 項で述べたように，ユーザがある場所に無限にとどまる可能性を持たせるよう設定することもできる．
- EventHandler プロセスは，生成されたイベントに対応したシステムの振舞いとして ECA ルールを実行する．強い防止性と弱い防止性を明示的に区別するため，「状況が変化したがシステムがいまだそれに対応していない（対応する ECA ルールが実行されていない）状態」が存在するようなモデル化となっている．
- EventProducer，EventHandler の各プロセスは交互に起動される．つまり，システムの動作は十分速く，ユーザの移動等のイベントは，前のイベントに対応する動作をシステムが完了してから発生するとしている．これは，「ある状態変化に対し対応する動作を起動している間にすでに状態が変わっている」といった複雑な状況はこの仕様段階では考えない，とするものである．実際には後の設計・実装時において，「ユーザがある 2 つの位置空間の境界線（スコープの実装となるような境界）をごく短時間に何度もまたいだ場合にはそれを無視する」といった対策をとることとなるであろう．

3.6　実　　装

現在の実装においては，代表的な LTL モデル検査ツールである SPIN [7] を用いている．開発者はいずれにしても状態遷移や時相論理等，モデル検査の概念を理解する必要があるため，この実装においては SPIN の存在を隠蔽することは目標としていない．図 3 においては，SPIN への入力言語である PROMELA によるモデル記述や LTL による検証項目を生成するが，モデル検査の結果を元の仕様上で表現することはしていない．一方で，誤りがある最短の実行パス発見や，様々な種類のシミュレーション，最適化オプション等，多彩な SPIN のオプションを活用することができる．

図 5 に生成される PROMELA 記述の概要を示す．1 行目から 5 行目にあるように，on と off といった列挙型や，デバイス，ユーザ，スコープ，それらに関するイベントの識別子が定義される．こういった識別子は，SPIN 内部では無意味な整数値であるが，トレース表示時等にはマクロの文字列が表示される．このため，たとえば，「次に user0 が demo0display の視覚スコープに入るイベントが発生した」（**eid_user0_demo0display_visual_out2in**）といったことをトレースから読み取ることができる．7 行目から 21 行目は EventProducer プロセスを与えている．このプロセスでは，最初に **next** チャネルを通し，EventHandler プロセスからメッセージを受け取る．これにより，EventHandler による ECA ルールの実行が終わってから次のイベントが起きることとなる．ここで，EventHandler 側で，「ユーザに移動を依頼する（そしてユーザが受け入れる）」というように次のイベントが決まることがあるため，9 行目，18 行目にあるように異なるメッセージが EventHandler から与えられる．18 行目はそのようなメッセージの例で，特定のイベント（ユーザがスコープに入る）を起動している．逆に 9 行目では，非決定的にそのときに起きうるイベントを起こす．11 行目から 16 行目はそのようなイベントの発火記述の 1 つであり，ユーザの移動に相当するように状態変数やフラグを書き換えるとともに，EventHandler 側にメッセージを送り該当する ECA ルールを起動している．これに対し EventHandler 側では，そういったメッセージを受け取り該当する ECA ルールを起動する（23 行目から 35 行目）．

実際の実装においては，Domain-Specific Language（DSL）向けツールである Xtext を用いている [12),13)]．Xtext に対し文法定義と言語変換スクリプトを入力すれば，エディタ，文法チェッカ，言語変換器を出力することができる．本論文においては，スマート空間向けの仕様記述言語の文法と，その PROMELA 言語への変換スクリプトを与えてそれらのツール群を出力させている．

```
/* 変数・マクロ等の設定 */
mtype = {on, off}; /* デバイスの稼働状態 */
...
#define scid_user0_demo0_display_visual 0 /* スコープ ID の例 */
#define eid_user0_demo0display_visual_out2in 1 /* イベント ID の例 */

proctype event_producer(){ /* Event Producer */
  do
  :: next?0 -> /* イベントが非決定的に起こる場合 */
     :: atomic{ /* ユーザ移動イベントの例 */
          scstate[scid_user0_demo0_display_visual] == out ->
          scstate[scid_user0_demo0_display_visual] = in;
          waiting_user = false; /* フラグ管理*/
          ch_event!eid_user0_demo0_display_visual_out2in
          /* Event Handler 側の ECA ルールを起動する */
        }
     ...
  :: next?cid_user0_demo0_display_visual ->
     ... /* 次のイベントが Event Handler 側で決められた場合 */
  od
}

proctype event_handler(){ /* Event Handler */
  do
  :: ch_event?eid_user0_demo0_display_visual_out2in -> /* イベント対応 */
     atomic{ if
     :: scstate[scid_user0_demo0_speaker_audio] == in && ... -> /* 条件 */
        dvcstate[demo0_display] = on; ... /* アクション実行*/
        next!0 /* Event Producer 側で次のイベントを起こす */
     fi
     :: else -> next!0 /* 条件が満たされない場合 */
     fi }
  ...
}
```

図 5　生成される PROMELA 記述の概要
Fig. 5　Overview of generated PROMELA specification.

4. ケーススタディ

本章では提案フレームワークの利用についてケーススタディを通して議論する．

4.1 シナリオおよび初期仕様

以下では，2 章であげたスマートオフィスシステム（図 1）において，別の機能に関する部分を例題として提案フレームワークの利用に関する議論を行う．この機能は，学会やオープンハウスといったイベントにおいて，複数のプレゼンテーションやデモンストレーションをオープンスペースで行うことを支援するものである．プレゼンテーションを行う際には，空間の中央にある大きなメインディスプレイ，メインスピーカを起動してプレゼンテーションを支援する．一方，空間には小さなディスプレイ，スピーカも多数設置されており，個別にデモンストレーションを行うこともできる[⋆1]．

B-a1. ユーザがプレゼンテーション支援の開始を依頼した際には，指定されたコンテンツをメインディスプレイ，メインスピーカでそれぞれ表示，再生する．
B-a2. ユーザの依頼に応じて，メインディスプレイ，メインスピーカにおけるプレゼンテーション表示，再生は中止する．
B-b1. ユーザがデモンストレーション用のディスプレイ，スピーカに十分近くに来た際には，それぞれを起動しあらかじめ設定されたコンテンツの再生，表示を開始する．
B-b2. ユーザがデモンストレーション用のディスプレイ，スピーカから離れた際には，それぞれを停止しあらかじめ設定されたコンテンツの再生，表示を中止する．

図 6　ケーススタディにおける初期の振舞い仕様
Fig. 6　Initial behavior specification in case study.

この機能に関する初期の振舞い仕様を**図 6** に示す．プレゼンテーション支援はユーザの依頼によって起動されるが，個々のデモンストレーションはユーザが近くに来ることにより自動的に起動するような仕様となっている．実際にはこれらの振舞いは ECA ルールで記述されるが，ここでは直感的な説明を通してその意義，意味について議論する．本論文の説明では簡単のため，デモンストレーション用のディスプレイ，スピーカはそれぞれ 1 つずつとしており，**図 7** のようにスコープ間の包含関係が存在するとする．

4.2 音の衝突に関する分析

4.2.1 初期仕様の分析

図 6 に示した初期仕様に対し，3.4.2 項で述べた「複数の音をユーザが同時に聞く（スコープ内におりかつデバイスが稼働中）ことがない」という性質（強い防止性）の検証を考

⋆1 これは著者らの機関において，実際に導入された部屋の仕様である．

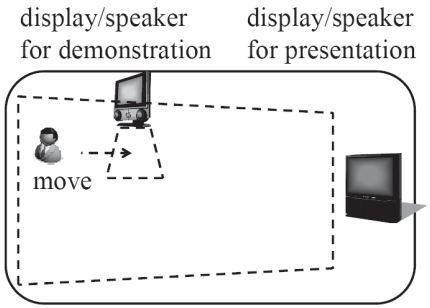

図 7　ケーススタディにおけるスコープの包含関係
Fig. 7　Scope inclusion relationship in case study.

える．しかしこの初期仕様では，デモンストレーションが動作しているときにプレゼンテーション開始の依頼が発生し，それぞれの音が衝突してしまう状況が発生してしまう．この対応漏れは，モデル検査器，特に SPIN の反例出力機能の利用により検出することができる（性質が成り立たないという検査結果が得られる）．

この初期仕様は，あまりにも単純で検討されていないものかもしれないが，提案フレームワークにおいては与えられた範囲で起きうることをすべてシミュレートまたは検査できるため，分析してみながら随時 ECA ルールを足していく，といった使い方も可能である．

4.2.2　修　正　1

上記の結果を受けて，開発者がプレゼンテーションを優先して下記の ECA ルールを足すことを考えてみる．

B-b3. プレゼンテーションが開始されたときに，デモンストレーション用のディスプレイとスピーカが稼働しているならば，それらを停止する．

検証を行うと，この修正は，デモンストレーションが動作しているときにプレゼンテーションが始まるという状況に対応しているが，逆の状況には対応していないことが分かる．

4.2.3　修　正　2

上記の結果を受けてさらに，振舞い **B-b1.** を下記のように修正する．

B-b1'. ユーザがデモンストレーション用のディスプレイ，スピーカに十分近くに来た際に，プレゼンテーションが行われていなければ，それぞれを起動しあらかじめ設定されたコンテンツの再生，表示を開始する．

検証を行うと，しかしこの修正後も，振舞い **B-b3** の起動には遅延があるため，「強い防止性」を実現することはできないことが分かる．

4.2.4　修　正　3

上記の結果を受けて可能な修正の 1 つは，振舞い **B-b3** を **B-a1** に組み込み，デモンストレーションを止める動作とプレゼンテーション動作を始める動作を連動させる（原子的に行う）ことである．

B-a1'. ユーザがプレゼンテーション支援の開始を依頼した際には，指定されたコンテンツをメインディスプレイ，メインスピーカでそれぞれ表示，再生すると同時に，稼働中のデモンストレーション用のディスプレイ，スピーカを停止する．
削除　**B-b3** を取り除く．

これにより最終的に，「音の衝突に関する強い防止性」という性質は満たされることとなる．

もしくは，代わりに「弱い防止性」を用いることも考えられる．その際には 3.4.2 項で述べたように，「ユーザが両方の音を聞いてしまう状況からいつか離れる」という結果ではシステム仕様に対する解決にならないため，無限にとどまるようなユーザを設定して検証を行うこととなる．

4.2.5　環境に関する仮定の明確化による解決

もしもスペースが十分広く，プレゼンテーションの音が聞こえる範囲とデモンストレーションの音が聞こえる範囲が重ならないように設置できるのであれば，それが最も簡単な解決である．その場合は，その制約をスコープ間の重なりの有無の設定として明示化することにより，検証に反映するとともに，実際の環境設営に関する仕様（たとえば設営マニュアルに記述する内容）として明示化することができる．

4.3　他のシナリオ

上記のシナリオのほかに，以下のようなシナリオを検討，記述，検証した．

複数ユーザ・デモンストレーションデバイス　上記の説明では単純化したが，実際には複数のユーザ，複数のデモンストレーション用デバイスが存在する．

モバイルデバイスの利用　デモンストレーションにおいてユーザが希望する場合，または混雑等によりユーザが十分に近づくことができない場合に，ユーザがモバイル端末も併用してデモンストレーションを視聴できることができるように仕様を変更した．3.4.1 項で述べたように，提案フレームワークにおいてはモバイルデバイスはそれとの相互作用のスコープにユーザをつねに含むデバイスとして定義されている．この際には，ユーザが視覚，聴覚による相互作用を組み込み，モバイルいずれかのデバイスを用いて同時に得ることができるという性質を検証した．厳密には，「視覚，聴覚のいずれかの相互作

用だけ得ている状態」を不整合として考えた．「ユーザが突然聞こえる範囲から離れる」といった状況変化がありうるため，この不整合に対して「弱い防止性」を検証した．

物理的な相互作用の種類変更　さらに，視覚，聴覚でそれぞれ伝わる情報（コンテンツ）について，視覚のみ，聴覚のみで伝達してもよい，という仕様変更も検討した．たとえば，聴覚による相互作用を通して伝わる情報について，字幕を用いて視覚による相互作用を通した実現としてもよい，という変更を考えた．3.3.3 項の例にあるように，デバイスを稼働させるときには表示・再生等行うコンテンツを指定することができるため，ある情報を聴覚デバイスではなく視覚デバイスで伝達する，という動作を表現することができている．この際には，「ユーザが片方のコンテンツのみ得ている状態」を不整合とし，上記同様に「弱い防止性」を検討した．これにより，プレゼンテーション中にデモンストレーションが静音モード（字幕つき）で動作する場合や，耳が悪いユーザがいたならば字幕をつけて動作する場合も記述，検証した．

5. 議　　論

5.1 利　　点

本論文では，スマート空間仕様の早期モデル化と，モデル検査によるその検証のためのフレームワークを提案した．このフレームワークにおいては，物理的な相互作用に関する非常に抽象的な，実装要素を捨象したモデル化を行っており，そのうえで網羅的な探索による検証（モデル検査）を行っている．しかし 4 章におけるケーススタディで示したように，この抽象度においても多くの不整合や漏れが発生しうる．このため，コンポーネントに分割しての動作設計や，センサの導入等の詳細を議論する前に，この抽象レベルであっても仕様を明確化，厳密化し分析することは非常に重要であると考えられる．

提案フレームワークが主に検証の対象とする性質は，スマート空間システムにおいて本質的である，ユーザが得るべき，または得るべきではない物理的な相互作用に関する要求である．こういった性質に対し，振舞い仕様によりその性質が実現できていないという不整合．特に，対応すべき動作が起動されていない，または誤っているような特定の状況を検出することができる．また修正対象は振舞い仕様に限らず，要求が強すぎて実現できない場合や，デバイスどうしを十分に離して置く等，要求の実現に必要となる環境設定に関する仮定が明示されていない場合についても分析の結果明らかになる．

上記のような検証にあたっては，要求および時相論理式を理解し妥当な検証式を記述することが必要となる．この点に関し本論文では，スマート空間システムにおいて本質的，共通である性質の記述パターンについて議論し，提供している．ケーススタディで示したように，それらの記述パターンはスマート空間システムにおいては本質的で，ほとんどの場合に意識して用いる必要があるものである．ただし，ソフトウェアシステムは個々異なる要求に基づいて開発されるため，開発者が自身の要件をふまえ注意深く取り組む必要性は残る．

提案フレームワークにおいては，上記のような記述および分析を行う際に，低レベルなモデル検査器に対して直接記述を行う必要がなく，スマート空間に関する語彙を直接用いて記述を行うことができる．また，ユーザがとりうる行動の網羅等は設定に応じて自動的に生成している．

5.2 限　　界

提案フレームワークの扱った範囲については，より様々な事例を通して評価，洗練を行う必要がある．特に，仕様記述言語の記述能力や記述形式，モデル検査器の利用に対するより強力な支援等の洗練が考えられる．

一方，本論文では開発初期におけるモデル検査による検証という作業のみを検討した．実際には下記のように，異なる作業も支援していく必要がある．

段階的詳細化　Event-B [14] 等の形式仕様記述において一般的であるように，詳細または広範囲な仕様・設計を得るために，段階的にモデルを詳細化していくことも支援する必要がある．

妥当性確認　検証はあくまで，開発者が与えた基準（検証項目）に対する正当性を評価するものである．一方で，それが利用ユーザやシステムの発注者にとって満足できるものかどうか，という妥当性確認もあわせて必要である．提案フレームワークを用いた結果，SPIN の機能を用いたシミュレーションは行うことができるが，可視化等の支援が必要だと考えられる．

5.3 関 連 研 究

5.3.1 ユビキタスコンピューティングにおける振舞い検証

ECA ルールにおける振舞いの記述は，ユビキタスコンピューティングを含めた分散システムにおいて広く用いられており，その記述の整合性や抜けに関する取り組みもなされてきた[10),11)]．しかしこれらの取り組みにおいては，ECA ルールの組における整合性や抜けに注目しており，スマート空間のようなシステム全体の仕様記述や分析は行っていない．一方本論文においては，環境に関する仮定を考慮しながら，ユーザの振舞いの可能性を列挙し，それに対し時相論理により与えられる高レベルな性質の検証を考えた．

5.3.2　ユビキタスコンピューティングにおける物理的側面のモデル化

モバイル・ユビキタスコンピューティングにおける移動のモデル化は，π-calculus[15] や Ambient Calculus[16] といったプロセス代数においても扱われてきた．π-calculus においては通信チャネルの移動の表現により，ローミング等をモデル化することができる．特に Ambient Calculus においては，スコープの表現により，建物やその中の部屋等，階層構造をなす位置関係やそこへの出入りをモデル化することができる．このようにスコープ自体を形式モデル化すること自体は新しい試みではない．本論文では物理的相互作用が可能である範囲をスコープとしてモデル化した際に，どのようにスマート空間システムにおける仕様の記述や分析を行うべきか，またどのようにそれを支援すべきかについて議論している．また要件の議論を通し，Ambient Calculus のようなスコープの階層構造だけではなく，スコープどうしの重なりについて直接的に記述，分析できるモデルを構築している．

ユビキタスコンピューティングにおいては，位置・場所に代表される物理的な側面は広く扱われてきた[17]．多くの取り組みが，文献 18) に代表されるような設計や実装，実行基盤に対し行われてきたが，形式的なモデル化，その系統的な分析については取り組みが限られている．Ambient Calculus[16] のような，場所を木構造として表現しそれに対する出入りを考える形式モデルが文献 19) 等においてしばしば議論されている．しかし，そのような木構造では，本論文で考えたスコープ間における重なりの可能性について直接表現，分析することができない．加えて，それらの取り組みにおいてはモデル検査のような分析について，理論的な可能性は示しているが具体的なツールが提供されていない．

物理的な相互作用を行うための前提条件を仮想的なスコープとして表すモデル化は，文献 8) において提案された．しかしこの取り組みは，実装・実行時の監視，イベント対応処理の実現を扱っており，本論文のように開発時の問題については扱っていない．またこの取り組みも木構造による表現を用いているため，スコープの重なりについては表現，分析できない．また，文献 20) においては本論文に似たモデル化や分析を検討しているが，プリミティブな形式言語である Event Calculus を用いており，またモデル検査の適用に関する具体的な課題やツールも議論されていない．

位置を表現するスコープの重なりを明示的にモデルに含め扱う取り組みとしては，文献 21) がある．しかしこの取り組みはコンテキストアウェアなアルゴリズムの検証を対象としており，スマート空間システム全体の検証は考えていない．また，RDF（Resource Description Framework）[22] 等を用いたオントロジベースのコンテキストモデル化においては，一般的な知識を表現でき，結果スコープの重なりも表現できる．しかし，それを分析するような試みや，スマート空間仕様の記述，検証を行う試みはこれまでない．

5.3.3　形式手法・モデル検査の適用

近年の形式手法に関する取り組みにおいては，設計の正しさ検証だけではなく早期のシステム分析も注目されている．EU においてさかんに取り組まれている Event-B[14] においては，開発の早期過程において，これから開発仕様とする部分だけでなくシステム全体をモデル化し，段階的に複雑化・詳細化していくことが考えられている．本論文では同様の思想に基づき，スマート空間仕様のモデル化と分析を考え，固有の要件に対する支援を行った．一方で本論文では，段階的詳細化については扱っていないため，どのように設計過程につないでいくかを検討する必要がある．

モデル化や分析，設計，実装等を一から行うことはコストが高いため，ドメイン依存のより密な支援を行う Domain-Specific Modeling/Language（DSM/DSL）の考え方が近年注目されている[13]．形式手法においてもドメイン固有の支援が必要であることは，形式手法を学んだ企業の開発者に関する統計[23] においても示されており，たとえば Web サービスに関してはそのようなツールも開発，提供されている[24]．本論文はスマート空間というドメインに対し，固有の要件を検討しそれを支援するフレームワークを提案した．

モデル検査の検証項目に関する時相論理記述パターンは，安全性等直感的な意味に基づくもの[6]，「前に」「後に」等時系列の順序に基づくもの[25),26] が知られている．本論文で議論したパターンは，スマート空間における検証の意図を考慮した，直感的な意味を持つもの（前者）であり，時系列という観点（後者）では一般的なパターンで表現できるものである．

6. ま と め

開発の早い段階における系統的なモデル化，分析は，誤りが検出，修正されずに引き継がれ手戻りによるコスト増を引き起こすことを防ぎ，システム開発の効率化・安定化のために非常に重要である．本論文では，スマート空間システムの特徴に言及しながら，その早期仕様に対するモデル検査の適用について議論した．その議論をふまえ，スマート空間仕様の早期モデル化と，モデル検査によるその検証のためのフレームワークを提案した．このフレームワークにおいては，ユーザとデバイス間の物理的な相互作用の抽象モデル化に基づいたスマート空間仕様記述言語を提供している．また，特に計算機により制御できないユーザの存在を考慮した検証項目について，既存ツール SPIN[7] を用いたモデル検査を行う仕組みを提供している．これにより，スマート空間システムの仕様，特に状況に対応して物理的な相互作用を行う複雑な振舞い仕様を，センサ等の実装詳細を捨象した形で抽象的に，かつ直接

モデル化・記述し，モデル検査を通して系統的な検証を行うことができる．ケーススタディを通して議論したように，提案フレームワークに基づき，仕様の分析とそれに応じた修正をインタラクティブに，また仕様項目の追加を段階的に行うことができ，開発の早い段階で仕様の品質を向上させることができる．本論文は，スマート空間に代表されるユビキタスコンピューティングシステムにおいて，ソフトウェア工学の観点から特に上流の開発過程を支援するための第 1 歩である．本論文の内容をふまえ，今後さらに 5.2 節で議論したような，本論文における提案が扱っていない可視化等ツールの洗練，また妥当性確認や設計・実装への詳細化の支援等，様々な事例適用を通した洗練を行っていく．

参 考 文 献

1) Weiser, M.: The Computer for the 21st Century, *SIGMOBILE Mobile Computing and Communications Review*, Vol.3, No.3, pp.3–11 (1999).
2) Satyanarayanan, M.: Pervasive Computing: Vision and Challenges, *IEEE Personal Communications*, pp.10–17 (2001).
3) 荒木啓二郎：フォーマルメソッドの新潮流：Part I：歴史と概要：1. フォーマルメソッドの過去・現在・未来—適用の実践に向けて，情報処理学会誌，Vol.49, No.5, pp.3–8 (2008).
4) Woodcock, J., Larsen, P.G., Bicarregui, J. and Fitzgerald, J.: Formal Methods: Practice and Experience, *ACM Computing Surveys* (2009).
5) Kim G. Larsen, Baier, C. and Katoen, J.-P.: *Principles of Model Checking*, The MIT Press (2008).
6) Berard, B., Bidoit, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, L., Schnoebelen, P. and McKenzie, P.: *Systems and Software Verification: Model-Checking Techniques and Tools*, Springer (2001).
7) SPIN – Formal Verification. http://spinroot.com/
8) Satoh, I.: A Location Model for Smart Environments, *Pervasive and Mobile Computing*, Vol.3, pp.158–179 (2007).
9) Dittrich, K.R., Gatziu, S. and Geppert, A.: The Active Database Management System Manifesto: A Rulebase of ADBMS Features, *The 2nd Workshop on Rules in Database Systems* (*RIDS'95*), pp.1–17 (1995).
10) Shankar, C.S., Ranganathan, A. and Campbell, R.: An ECA-P Policy-based Framework for Managing Ubiquitous Computing Environments, *The 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services* (*MobiQuitous'05*), pp.33–44 (2005).
11) Shankar, C. and Campbell, R.: Ordering Management Actions in Pervasive Systems using Specification-enhanced Policies, *The 4th IEEE International Conference on Pervasive Compputing and Communications*, pp.234–238 (2006).
12) Xtext. http://www.eclipse.org/Xtext/
13) Kelly, S. and Tolvanen, J.-P.: *Domain-Specific Modeling: Enabling Full Code Generation*, Wiley-IEEE Computer Society Pr (2008).
14) RODIN – Rigorous Open Development Environment for Complex Systems. http://rodin.cs.ncl.ac.uk/
15) Milner, R.: *Communicating and Mobile Systems: the Pi-Calculus*, Cambridge Univ. Press (1999).
16) Cardelli, L. and Gordon, A.D.: Mobile Ambients, *Foundations of Software Science and Computation Structures: First International Conference*, pp.140–155 (1998).
17) Strang, T. and Linnhoff-Popien, C.: A Context Modeling Survey, *Workshop on Advanced Context Modelling, Reasoning and Management* (*at UbiComp 2004*) (2004).
18) Roman, M., Hess, C., Cerqueira, R., Ranganathan, A., Campbell, R.H. and Nahrstedt, K.: A Middleware Infrastructure for Active Spaces, *IEEE Pervasive Computing*, Vol.01, No.4, pp.74–83 (2002).
19) Ranganathan, A. and Campbell, R.H.: Provably Correct Pervasive Computing Environments, *The 5th IEEE International Conference on Pervasive Computing and Communications* (*PerCom'08*), pp.160–169 (2008).
20) Ishikawa, F., Suleiman, B., Yamamoto, K. and Honiden, S.: Physical Interaction in Pervasive Computing: Formal Modeling, Analysis and Verification, *The 2009 International Conference on Pervasive Services* (*ICPS'09*), pp.133–140 (2009).
21) Schmidtke, H.R. and Woo, W.: Towards Ontology-Based Formal Verification Methods for Context Aware Systems, *The 7th International Conference on Pervasive Computing* (*Pervasive 2009*), pp.309–326 (2009).
22) Manola, F. and Miller, E.: RDF Primer (2004). http://www.w3.org/TR/2004/REC-rdf-primer-20040210/
23) Ishikawa, F., Taguchi, K., Yoshioka, N. and Honiden, S.: What Top-Level Software Engineers Tackles after Learning Formal Methods - Experiences from the Top SE Project, *The 2nd International FME Conference on Teaching Formal Methods* (*TFM 2009*), pp.57–71 (2009).
24) Foster, H., Uchitel, S., Magee, J. and Kramer, J.: Model-based Verification of Web Service Compositions, *18th IEEE International Conference on Automated Software Engineering* (2003).
25) Dwyer, M.B., Avrunin, G.S. and Corbett, J.C.: Patterns in Property Specifications for Finite-state Verification, *The 21st International Conference on Software Engineering* (*ICSE'99*), p.411 (1999).
26) SPEC PATTERNS. http://patterns.projects.cis.ksu.edu/

(平成 22 年 4 月 9 日受付)
(平成 22 年 10 月 4 日採録)

石川　冬樹（正会員）

2007 年東京大学大学院情報理工学研究科コンピュータ科学専攻博士課程修了．博士（情報理工学）．2007 年より国立情報学研究所助教，総合研究大学院大学複合科学研究科助教兼任，現在に至る．サービスコンピューティング，ソフトウェア工学の研究に従事．2009 年より電子情報通信学会情報・システムソサイエティサービスコンピューティング時限専門研究委員会副委員長．

山本佳代子

2010 年東京大学大学院情報理工学研究科創造情報学専攻修士課程修了．2010 年より日立製作所に所属．

本位田真一（正会員）

1978 年早稲田大学大学院理工学研究科修士課程修了．（株）東芝を経て 2000 年より国立情報学研究所教授，2004 年より同研究所アーキテクチャ科学研究系研究主幹を併任，現在に至る．2001 年より東京大学大学院情報理工学系研究科教授を兼任，現在に至る．2002 年 5 月〜2003 年 1 月英国 UCL ならびに Imperial College 客員研究員．2005 年度パリ第 6 大学招聘教授．早稲田大学客員教授．工学博士（早稲田大学）．1986 年度情報処理学会論文賞受賞．ソフトウェア工学，エージェント技術，ユビキタスコンピューティングの研究に従事．日本ソフトウェア科学会理事，情報処理学会理事を歴任．日本学術会議連携会員．