仮想空間を用いたスマートスペースアプリケーション向けシミュレータ

西川博志[†1] 山本眞也[†1] 玉井森彦[†1]
西垣弘二[†1] 木谷友哉[†1] 柴田直樹[†2]
安本慶一[†1] 伊藤実[†1]

本論文では，スマートスペースを実現するアプリケーションソフトウェアを高信頼かつ低コストで開発するための仮想テストベッドを提供するシミュレータを提案する．提案シミュレータは，3D 仮想空間上に仮想のデバイスを設置し，デバイスの動作，デバイス間の通信をシミュレートする機能に加え，エアコン等のデバイスの動作による空間の物理量（室内温度等）の時間的変化を再現する機能，仮想空間の様子を任意の視点で描画する機能，仮想ユーザを空間内で自由に移動させる機能等を提供する．デバイスの動作・通信を再現する既存シミュレータはいくつか存在するが，仮想空間におけるユーザの位置，各種物理量等の多彩なコンテキストの変化を再現する機能をあわせ持つシミュレータはこれまで開発されていない．提案シミュレータが持つ多彩なコンテキストの再現機能により，スマートスペースアプリケーションの開発が容易になることが期待できる．有効性を評価するために，仮想空間上に 8 部屋からなるスマートホームとコンテキストに従って情報家電を制御するアプリケーションソフトウェアを構築し実験を行った．結果，本シミュレータが，実用規模のスマートスペースアプリケーションを実行するのに十分な性能を有すること，アプリケーションソフトウェアおよびデバイス制御シナリオ開発時の不具合発見に有用なこと等を確認した．

A Simulator for Smartspace Application with 3D Virtual Space

Hiroshi Nishikawa,[†1] Shinya Yamamoto,[†1] Morihiko Tamai,[†1]
Kouji Nishigaki,[†1] Tomoya Kitani,[†1] Naoki Shibata,[†2]
Keiichi Yasumoto[†1] and Minoru Ito[†1]

In this paper, we propose a simulator which provides a virtual testbed for reliable and inexpensive development of application software for Smartspace. The proposed simulator allows application programmers to set up virtual devices in a 3D virtual space, simulate communication between the devices, and observe their behavior. Moreover, the simulator can reproduce transitions of physical quantities such as room temperature as a consequence of device actions, display the situation of the virtual space in real-time from an arbitrary viewpoint, and move virtual inhabitants in the space. Some existing simulators can simulate behavior and communication of devices, but no existing simulator can reproduce richer context such as transitions of physical quantities and user movements. Reproducing richer context by our simulator will make it easier to develop smartspace applications. We constructed virtual Smarthome consisting of 8 rooms and application software which controls information appliances according to context change, and conducted experimental validation of usefulness of the simulator. As a result, we confirmed that our simulator has enough performance to simulate Smarthome with practical size and is useful to detect bugs of application software.

1. はじめに

近年，多数のセンサや情報通信デバイスが設置された**スマートスペース**と呼ばれる空間において，各デバイスを，コンテキストや利用者の好みに従い適切に制御するアプリケーションに関する研究・開発がさかんに行われている[1)–4)]．スマートスペースは我々の生活に密接に関わってくるため，その上で動作するアプリケーションは起こりうる様々な状況において，予期したとおりに，かつ，安全に動作するように設計・開発されなければならない．しかし，動作の正しさを現実空

†1 奈良先端科学技術大学院大学情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology
†2 滋賀大学経済学部情報管理学科
Faculty of Economics, Shiga University

間においてテストするためには，多数のセンサやデバイスを設置したテストベッドを構築し，その上で被験者（サービス利用者）に様々な行動をとってもらう必要があり，その実施には莫大な労力とコストがかかる．さらに，センサやデバイスの設置箇所，環境の変化，利用者の行動パターンの起こりうる組合せ数は膨大であり，現実空間において，それらすべての可能性に対してアプリケーションの動作を確認することは，時間や労力の点から困難である．また，アプリケーションのユーザ（住人もしくはスマートホームの設計者）が，アプリケーションの動作を自分もしくは顧客の好みに合わせて設定する際には，(i) センサやデバイスの設置箇所の変更，(ii) 制御シナリオの設定，(iii) 制御シナリオに応じたコンテキストの生成による動作確認，等の操作を何度も繰り返し，試行錯誤により最適な設定（設置箇所およびシナリオ）を発見することが求められる．このような試行錯誤によるアプリケーションの設定を現実空間において実施することは，センサ等の設置箇所の変更に要する労力や温度等の物理量に基づいたコンテキストの生成が容易でないことを考慮すると，ユーザにとって大きな負担である．

以上の問題に対する解決策の 1 つは，アプリケーションのテストをシミュレータ上で実施することである．そのためには，シミュレータに以下の 3 つの機能が必要である．(1) スマートスペースの設計支援と可視化：自由に 3D の仮想空間を設計し，その空間上に簡単に情報通信デバイスを配置できる．また，コンテキストによってデバイスがどのように動作するのかをアニメーションを通して直観的に把握できる．(2) デバイス間の通信とソフトウェア互換性：仮想空間に設置したデバイス間で行われる通信をシミュレートする．その際，実デバイス上で動作するソフトウェアおよびプロトコルがシミュレータ上でもそのまま動作する．さらに，実デバイスと仮想空間に設置された仮想デバイスが通信し連動できる．(3) 現実的なコンテキストの生成と系統的なテスト：3D 空間上でのユーザの行動やデバイスの動作，通信，周辺の物理量（温度，明るさ等）を基にしたコンテキストを生成できる．また，発生しうるコンテキストを系統的に生成し，各コンテキストについてアプリケーションが正常に動作するかをチェックする．

スマートスペースの開発支援を目的とした既存シミュレータとして UBIWISE[12]，TATUS[13] が提案されている．これらのシミュレータでは，3D 仮想空間に設置したデバイスの動作テストを行うことができ，上記の機能 (1)，(2) は部分的に実現されているが，現実的なコンテキストを生成したうえでスマートスペース全体の動作を再現しテストする機能 (3) は実現されていない．

本論文では，スマートスペースアプリケーション設計開発支援のための仮想テストベッドの提供を目的に，上記 (1)〜(3) の機能すべてを備えたシミュレータ UbiREAL（Ubiquitous Application Simulator with REAListic Environments）[5] を提案する．UbiREAL の実現の際には，デバイスの通信，空間の物理量，ユーザの位置等多くのシミュレーション項目を同時に扱いかつそれらを時間的に同期させること，空間の可視化も含めた複雑なシミュレーションを実時間で行えること，が必要である．そのため，UbiREAL を空間設計支援・可視化機能，ネットワークシミュレーション機能，物理シミュレーション機能，テスト機能の 4 つの独立したモジュールで構成し，複数の計算機での並列実行にも対応できるようにするとともに，各モジュールでのシミュレーションの粒度を調整できるよう，設計上の工夫を行った．

UbiREAL の有効性を評価するために，仮想空間上に，8 部屋からなるスマートホームを構築し，コンテキストに従って情報家電を制御するアプリケーションソフトウェアとルールベースの制御シナリオを開発して実行した．結果，アプリケーションのテストにかかる時間が十分に短いこと，可視化をともなったシミュレーションが実時間で実行可能なこと，制御シナリオやアプリケーションに含まれる不具合の発見に UbiREAL が有用であることを確認した．

2. 関 連 研 究

ユビキタスアプリケーションの開発の容易化や信頼性の確保に関して多くの研究が行われている．本章では，関連研究をミドルウェア，テストベッド，シミュレータの 3 つに分類して説明する．

最初にユビキタスアプリケーションの開発を効率的に行うためのミドルウェアやフレームワークに関する研究について述べる．Biege らは，コンテキストアウェアアプリケーションの開発を容易化するため，センサとアクチュエータ，アプリケーションコンポーネントをイベントベースで制御するミドルウェアを提案している[3]．Niemel らは，コンポーネント間の相互接続性，適応性，拡張性を実現することが重要との認識のもと，これらを実現するミドルウェアのアーキテクチャについて述べている[6]．Roman らは，Gaia と呼ばれるスマートスペース上で様々なサービスを提供するためのミドルウェアを提案している[7]．加えて，Gaia の

サービスをシンクライアントで利用できるようにするため，マイクロサーバのプロキシを使った J2ME ベースのミドルウェアを提供している[8]．Messer らは，異なる CE（Customer Edge）デバイスを統合し，インタフェースを通して，ユーザが自分のやりたいことを選択するだけで済むためのミドルウェアを開発している[9]．これらのミドルウェアは，UbiREAL 上のアプリケーションとして実行することが可能であり，アプリケーションの開発の容易化やアプリケーションのユーザビリティの向上という観点において，UbiREAL と互いに補完し合う関係にあるといえる．

次に，テストベッドに関する研究について述べる．Consolvo らは，Labscape と呼ばれるスマートスペースを設計する際に，様々な既存手法を使った場合の長所，短所を，実験により評価している[10]．また，Nakata らは，多数のノードが存在するセンサネットワークやホームネットワークをテストベッド上のみでシミュレートするのは困難と考え，実際のノードと仮想ノードを組み合わせてシミュレーションする方法を提案している[11]．これらテストベッドに関する研究は，スマートスペース向けアプリケーションの効率良い開発や信頼性向上のために非常に重要であるが，すべての対象環境を現実空間に設営するのはコスト等の面で困難である．Nakata らの研究は，UbiREAL と同様に現実空間での動作と仮想空間でのシミュレーションを連携して行うことを目標としているが，詳細な実装方法や性能評価は公開されていない．

UbiREAL と同様，スマートスペースの設計・開発支援を目的とした，3D 可視化機能を持つシミュレータがいくつか提案されている．UBIWISE[12] は，ユビキタスコンピューティング環境で用いられる新しいハードウェアやデバイスの組み込みソフトウェアのプロトタイプの開発とテストを主な目的としている．UBIWISE は UbiSim と WISE という 2 つのシミュレータから構成され，UbiSim で 3D 仮想空間を生成し，WISE によってデバイスをシミュレートし 2D 表示を行う．TATUS[13] は無線ネットワークのシミュレーション機能を持つシミュレータであり，複数のデバイスにまたがるユビキタスアプリケーションを 3D 仮想空間上でシミュレートし，テストすることを可能にしている．TATUS は 3D ゲームエンジンを用いて開発されており，仮想空間内の仮想ユーザをテスト実行者が操作して移動させたり，簡単な AI に基づいて移動させたりすることができる．また，デバイス間の無線通信におけるライン・オブ・サイトを仮想ユーザが横切る頻度を基にした無線通信のパケットのロス率のシミュレーションを行うことができる．

これら既存のシミュレータは，デバイスの動作確認等を主な目的としているが，スマートスペースにおけるコンテキストの変化の再現をほとんど扱っていない．スマートスペースアプリケーションは，そのときどきのコンテキストによって動作が変わるため，実テストベッドと同様にアプリケーションのテストを行うには，ユーザ位置，空間物理量等からなる様々なコンテキストを設定できる機能，デバイスの動作による物理量の自然な移り変わりをシミュレートする機能が必要である．UbiREAL は，これらコンテキストの変化をきめ細かく設定，シミュレートできるよう設計されており，既存のシミュレータでは実現できなかった，スマートスペースの仮想テストベッドを提供する．

3. UbiREAL の機能

本章では，スマートスペース向けシミュレータに求められる機能について述べ，次に，UbiREAL の概要について述べる．

3.1 対象アプリケーションとシミュレータに求められる機能

提案シミュレータは，スマートスペース向けアプリケーションを対象とし，それらを高信頼かつ低コストで開発するための仮想テストベッドの構築を目的としている．提案シミュレータが対象とするアプリケーションの一例として，以下のような家電制御アプリケーションが考えられる．リビングルームに各種センサ，家電が配置されており，家電をネットワーク経由で制御する．たとえば，以下のルールに従って制御がなされる．

- アリス（娘）がリビングルームに入ると，部屋の照明が薄暗く調整され，ステレオがジャズを演奏し，エアコンは室温 25 度，湿度 50%に設定される．
- デイブ（父），または，キャロル（母）がリビングに入ると，照明が明るく調整され，ステレオがクラシックを演奏し，エアコンは室温 27 度，湿度 60%に設定される．

このとき，照明が点灯と消灯を繰り返したり，同じ部屋でクーラとヒータが同時に動作したりするのは望ましくない．このようなアプリケーションの不具合を仮想空間上のシミュレーションを通して検出できることが求められる．そのためには，以下の機能がシミュレータに求められる．

(1) シミュレータ内の仮想デバイスの動作が直観的に確認できること

アプリケーションが適切に動作しているかを確認す

る方法として，アプリケーションに対するテストケースを記述し，テストにパスするかどうかを機械的にチェックすることが考えられる．しかしテストケースを作成するためには，アプリケーションが満たすべき性質に関し，それを機械的にチェック可能な形式として表現する必要があり，この作業には相応の労力を必要とする．一方，print 文によるデバッグのように，テストケースを記述する以前に，アプリケーションの動作を気軽にチェックしたいという要求がある．また，特にアプリケーションのユーザにとっては，自分の設定した制御シナリオが適切に動作するかを確認するために，自らテストケースを記述することは大きな負担である．したがってルール設定の結果，どのように家電が動作するのか，あるいはどのような場合に不具合が発生するのかを，シミュレータ内の仮想デバイスの動作の様子を視覚化することで，直感的に確認できる仕組みが求められる．

(2) シミュレータ内の仮想デバイスと実空間のデバイス間の連携動作が可能なこと

たとえば，冷暖房機器の設定温度や音を発する機器の音量等について，シミュレータ上の数値のみで動作を確認するだけでなく，実際に人間の感覚を通して確認したい場合がある．このような場合，一部の仮想デバイスを実デバイスに置き換えてシミュレーションできると便利である．

(3) シミュレータ内で，実空間のデバイス用ソフトウェアが大きな変更なく動作可能なこと

実デバイスの動作をシミュレータ上に再現するうえで，実デバイス用ソフトウェアを一から作り直すのは多大な労力を必要とする．したがって，実デバイス用ソフトウェアを，大きな変更なしにシミュレータ上でも動作させられることが望ましい．

(4) 発生しうるコンテキストを系統的に生成し，アプリケーションの正しさをテストできること

シミュレーションを行う際に，手動で多くのコンテキストを生成し，それらのコンテキストに対してデバイスが予期したとおりに動作するかをチェックすることは，労力がかかるうえにチェック漏れが生じる可能性が高いため，自動的に行えることが望ましい．

3.2 UbiREAL の構成

前節で述べた機能を実現するため，UbiREAL を**図 1** に示す 4 つの部分，(1) スマートスペース設計支援・可視化機能（GUI 部），(2) ネットワークシミュレーション機能，(3) 物理環境シミュレーション機能，(4) 系統的なテスト機能，から構成する．

UbiREAL では，各仮想デバイス上で動作するソフトウェアとして，実デバイス上で動作するものをほぼそのまま実行可能とする（詳細は 3.4.1 項で述べる）．ただし，デバイスの内部状態を取得する仕組みを追加するため若干の変更を要する．これは，デバイスの動作の可視化とテスト結果の自動判定のために使用される．また，デバイスに人が操作するためのスイッチがついている場合，これらの入力を仮想デバイス上のソフトウェアが受け取れるようにするための変更も必要になる．以下，シミュレータを構成する各機能の概要について述べる．

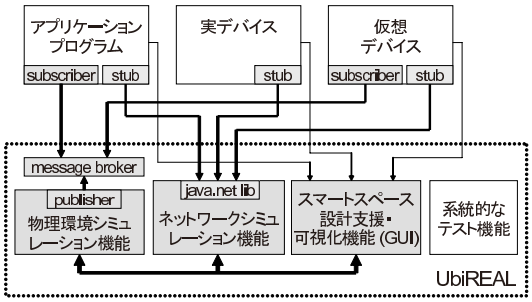


図 1 UbiREAL のアーキテクチャ
Fig. 1 UbiREAL architecture.

3.3 スマートスペース設計支援・可視化機能

スマートスペース設計支援・可視化機能では，仮想デバイスを動作させる部屋の作成，人等のオブジェクトの移動軌跡の設定を行う機能を提供する．また，仮想デバイスの動作を 3D アニメーションにより可視化する機能を提供する．これらの機能は，GUI を通じて直観的に利用可能である．これらを行うソフトウェアモジュールを，以後 GUI 部と呼ぶ．なお，GUI に関する，以下に述べる機能すべてが，現在の実装で利用可能である．

3.3.1 スマートスペースの設計支援機能

本機能モジュールは，市販の 3D モデリングソフト等で作成した 3 次元形状データをインポートし，GUI によりそれらを仮想空間上の任意の位置へ設置することを可能にする．仮想空間は，建物の平面図の CAD データを市販の 3D モデリングソフト等で 3D 化することによって作成される．設置されるオブジェクトは，机や椅子等の家具や各種デバイス（センサ，テレビ，エアコン等）に相当する．オブジェクトは静止オブジェクトと可動オブジェクトに分類され，それぞれ設置の仕方が異なる．静止オブジェクトは，登録されているオブジェクトのリストのうちから，プルダウンメニューを用いて選択され，ドラッグ&ドロップで任意の位置に設置される．新たにオブジェクトを登録する場合，そのオブジェクトの形状ファイルとデバイス用のソフ

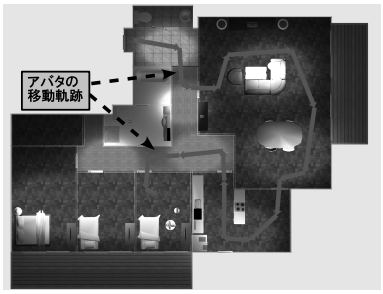


図 2 可動オブジェクトの配置
Fig. 2 Specifying route of movable object.

トウェアをインポートして登録する．ただし，デバイス用のソフトウェアにはオブジェクトの形状の変化を行うためのコードを付加する．人等の可動オブジェクトは，仮想空間を真上から見た視点で，各オブジェクトの移動軌跡を指定することで設置される（**図 2**）．移動軌跡の指定は手動または系統的テスト機能により自動で行う．オブジェクトの移動速度や軌跡上の特定の座標におけるアクション（ボタンを押す，リモコンを操作する等）を指定することもできる．人を表す可動オブジェクトを以後，**アバタ**と呼ぶ．

3.3.2 スマートスペースの可視化機能

本機能モジュールは，シミュレーション結果を視覚的に表示する機能を提供する．仮想デバイスの状態の変化（ライトの点灯，エアコンの作動等）や外部環境の変化（部屋が暗くなる等）の際，デバイスの形状や色，空間の見え方を変化させることでスマートスペースの動作状況を可視化する．可視化による空間の見え方は，仮想空間を真上から見た 2D ビュー，特定のアバタの 1 人称の視点からの 3D ビュー等が設定できる．アバタが複数存在する場合，各アバタごとにビューを切り替えて表示する機能も提供する．**図 3**，**図 4** にスマートスペースの可視化の様子を示す．

3.4 ネットワークシミュレーション機能

ネットワークシミュレーション機能では，仮想空間上のデバイス間の通信をシミュレートする．無線通信を行う場合には，仮想空間上のデバイスの位置やデバイス間に存在する遮蔽物の影響等も考慮してシミュレートできるようにする．提案するネットワークシミュレータは，Java 言語で通常用いられる java.net パッケージと互換のライブラリを提供し，その上で UPnP 等のプロトコルを実現することが可能である．図 1 に示すように，このシミュレータとアプリケーションプログラムや各デバイス間の通信は，そのライブラリを用いた API で実現し，現実のアプリケーションプログラムやデバイスドライバの変更を最小限にして利用

(a) ヒータがオフの状態 (b) ヒータがオンの状態

図 3 ヒータの状態変化の可視化
Fig. 3 Visualization of heater's state.

(a) テレビがオフの状態 (b) テレビがオンの状態

図 4 テレビの状態変化の可視化
Fig. 4 Visualization of TV's state.

することが可能である．

本節では，まず，既存のネットワークシミュレータとの相違点を示し，提案シミュレータのアーキテクチャを述べる．

3.4.1 実デバイス用ソフトウェアとのソースコードの共用

ns-2 等既存のネットワークシミュレータでは，実デバイス用の制御ソフトウェアをシミュレータ上で動作させることは考慮されていない．そのため，シミュレータの利用者は，実デバイス用のソフトウェアとは別にシミュレータ用のソフトウェアを新たに作成する必要がある．

UbiREAL のネットワークシミュレータは，ネットワーク関連のライブラリと互換の API を提供しており，実デバイス用のコードをネットワークシミュレータとリンクすることで，大幅な改変なく利用可能である．提案シミュレータでは，現在のところ，Java 言語で通常用いられる java.net パッケージと互換のライブラリを提供する．実デバイス用のコードを UbiREAL 上で用いるためには，3.3 節で述べたように，デバイスの位置や状態を GUI 部と交換するためのコードを新たに付加する必要がある．Java 言語以外の，たとえば C 言語で書かれた実デバイス用のコードを UbiREAL 上で利用するためには，これらのネットワーク関連 API と互換の機能を実装し，JNI（Java Native Interface）を用いて呼び出すことで実現可能である．

シミュレータ用の互換ライブラリが現時点で提供するのは，java.net パッケージのうち，DatagramSocket,

MulticastSocket，ServerSocket，および Socket クラスであり，これらのクラスのうち代表的なメソッドが現在実装済みである．これにより，Java 言語用の UPnP ライブラリである CyberLink for Java（http://sourceforge.net/projects/cgupnpjava/）とのリンクが可能である．

3.4.2 アーキテクチャ

UbiREAL のネットワークシミュレータ[14] は，Java 言語で通常用いられる java.net パッケージと互換のライブラリを提供する．この上に，UPnP 等のプロトコルを動作させるためのコードを追加し，実行することができる．将来的に，ネットワーク層における経路，リンク層・物理層の特性を考慮した通信のシミュレーションを可能にする予定である．ネットワーク層における経路制御プロトコル，リンク層，物理層の特性をプロトコルの種別ごとに正確に再現する機能は未実装であるが，AODV や DSR 等のルーティングプロトコル，イーサネット，IEEE802.11a/b/g，ZigBee，Bluetooth 等のリンク層プロトコルを今後サポートする予定である．

各デバイスの IP アドレス等の設定項目は，デバイスごとの設定ファイルにより指定する．物理的なネットワークの構成（イーサネットケーブルの接続関係やハブの位置等）は，ネットワークシミュレータ専用の GUI を用いて設定できるようにすることを予定している．無線通信のシミュレーションでは，スマートスペース上の家具やデバイスの位置を取得し，デバイス間の電波干渉や障害物による影響を反映できるようにする予定である．これらは，オブジェクトの位置の時間的変化を，3.3 節で述べた可視化機能から定期的に取得することで実現可能である．

3.4.3 仮想空間と現実空間をまたがったデバイス間通信

UbiREAL のネットワークシミュレータにより，シミュレータ上で動作するソフトウェアと，実環境で動作するソフトウェア間の IP プロトコルによる通信を可能にする．現在の実装では，実環境で動作する Java プログラムを，CLASSPATH 等の設定を変更して実行することで，この機能が利用可能である．この機能により，実デバイスをシミュレーション対象の仮想空間のデバイスとして用いるハイブリッドシミュレーションが可能になり，実デバイスの想定利用環境での動作確認，動作テストが行える．たとえば，リビングルームに存在するデバイスをシミュレータ上で実行し，寝室のデバイスには実機を使用する，といったシミュレーションも行える．この機能は，シミュレータを実行している PC を，仮想デバイスのつながったネットワーク上のスイッチングハブに見せかけ，その PC 上でネットワークアドレス変換を行うことにより実現する[14]．

3.5 物理環境シミュレーション機能

物理環境シミュレーション機能では，仮想空間上での様々な物理量（室温，音の大きさ，明るさ等）の変化を，デバイスの動作が空間に与える影響を考慮してシミュレートする．図 1 に示すように，提案する物理シミュレータでは，シミュレートした物理量の変化を Publish/Subscribe モデル[16] を用いてアプリケーションプログラムや各デバイスに通知する．

シミュレーションによってアプリケーションが正常に動作するかを確かめるためには，空間の形状的な特徴を再現することに加え，デバイスの動作および人の行動の影響により変化する仮想空間内の温度や照明等の物理量の変化を再現する必要がある．これまで，リアルな 3D 環境を再現するための様々なシミュレータの研究や開発がなされており，文献 15) においてサーベイが報告されている．最近では，Unreal Game Engine[17] 内の物理エンジンにおいて，光や音の効果がシミュレーションされている．これらのシミュレータは，視覚，音響に関係する物理量にのみ焦点をあてている．一方，UbiREAL の物理環境シミュレータは，室温，湿度，音，光，電波，振動等，コンテキストを形成する様々な物理量を取り扱う．個々の物理量シミュレータは，物理学の適切な定式に基づいて実装する．各物理量に関連するデバイス（センサ等）は，関連する物理シミュレータが管理する物理量が変化したとき通知を受けるよう，物理シミュレータに対しサブスクライブ要求を出す．物理シミュレータは，定期的に物理量の現在値を計算し，値が更新されたとき，サブスクライバに通知を行う．なお，UbiREAL の GUI 部は各物理シミュレータに対してサブスクライブを行っており，たとえば室温の変化であれば，図 3 のように可視化される．それぞれの物理シミュレータが必要とするパラメータは，前回の計算時の値，仮想空間の部屋のサイズや容量，デバイスの状態，人間の振舞い，他の物理量および前回の計算から経過した時間となる．

たとえば室温に関しては，仮想デバイス等から得られる熱量と，空間の熱容量に基づいて，次の式で計算を行う：$\Delta T(t) = Q(t)/C$．ここで，それぞれ，ΔT：温度変化，Q：得られる熱量，t：単位時間，C：熱容量（部屋ごとに定数を仮定）である．

物理量のシミュレーションは，各物理量ごとに独立して行い，各物理量の変化は Publish/Subscribe モデ

ルに基づいた API を用いて通知する．UbiREAL の現時点の実装では，各部屋の室温をシミュレートすることが可能である．室温以外の物理量のシミュレーションについては，Publish/Subscribe モデルに基づいた API で制御される個別モジュールとして用意することで，柔軟に追加することが可能である．物理量の演算において，精度と計算量にはトレードオフの関係がある．そのため，トレードオフを考慮して複数のモジュールを用意し，それらを変更することで，シミュレーションの目的に応じた精度と処理速度を容易に得られるようになるという利点がある．また，電波強度等，計算機上でのシミュレートが困難な物理量については，実際の外部デバイスから得られた物理量の測定値を計算機内部にフィードバックするモジュールを作成することで，さらに精度を向上させることができると考えている．

3.6 系統的テスト機能

系統的テスト機能では，与えられたアプリケーションがその要求仕様に対して正しく動作するかどうかを，コンテキスト（ユーザ位置や気温等の物理量）を系統的に生成しテストする機能を提供する．

本テスト機能は，アプリケーションの仕様 *Spec* は有限個のルールの集合 $AP = \{l_1, \ldots, l_k\}$ および成り立ってほしい性質の集合 $P = \{p_1, \ldots, p_l\}$ により与えられることを想定する．ここで，各ルール $l_i = (c_i, a_i)$ は発火条件 c_i およびアクション a_i からなり，条件 c_i が成り立つとき，アクション a_i が実行されるものとする．アプリケーションにおいて成り立ってほしい各性質 p_k は，スマートスペースの状態間の時相論理を用いて，『「ユーザ 1 が部屋 A にいる状態」なら「いつかは必ず部屋 A の気温が摂氏 23–25 度，湿度が 59–61%の状態に遷移する」がつねに成り立つ』のような命題として指定する．

与えられた空間 U とその初期状態（すべてのデバイスの状態，部屋の状態，住人の状態を含む），アプリケーションの仕様 *Spec*（ルールの集合 AP，成り立ってほしい性質の集合 P），アプリケーションの実装 I に対し，以下のすべての条件が満たされるとき，I はその仕様 *Spec* に対し正しく実装されていると定義する．

(1) すべてのルール $l = (c, a) \in AP$ に対し，発火条件 c が満たされれば，必ずアクション a が実行される．

(2) すべての性質 $p \in P$ が成立する．

実装 I が上記の正しさ (1) を満たすかどうかを確かめるには，仕様のルール集合 AP の各ルールの発火条件を満たすセンサの値（の組）をすべて生成し，I への入力として与えた際に，ルールに設定されたアクションが実行されることを確認すればよい．しかし，ルールの発火条件は，「気温が 28 度以上，かつ，湿度が 70%以上」のように連続物理量の範囲として与えられるため，すべての値に対し上記の確認を行うことは不可能である．そこで，提案手法では，条件を満たす値の範囲を，定数 C 個のサンプル値でカバーし，それらすべての値に対し，(1) を確認できれば，I は正しく実装されていると考える．

なお，3.3 節で述べた GUI モジュールが空間 U の状態を保持しているので，GUI モジュールの保持するデバイス d の状態を参照することで，デバイス d でアクションが正しく実行されたかどうかチェックできる．

実装 I が P の各性質を満たすかどうかは，ルールの発火により実行されるアクションが U の物理量に与える変化の度合いと経過時間に依存する．そこで，3.5 節で述べた物理環境シミュレータにより，アクション実行後の物理量の時間変化をシミュレートし，一定時間ごとに各性質 $p \in P$ が成立しているかどうかをチェックする．

以上で述べた各機能が連携動作することでシミュレーションを行う．シミュレータの具体的な動作を，RFID タグを身につけた人がタグリーダに近づくことにより，エアコンの電源が入る，というアプリケーションを例に説明する．タグを身につけた人がタグリーダに近づいたとき，タグとリーダ間の通信がネットワークシミュレータを介して行われる．このとき，タグとリーダの仮想空間上での位置を考慮することで，無線による通信が可能かどうかが判定される．リーダは，仮想空間上のホームサーバ（デバイス制御用のルールデータベース）と通信し，エアコンの電源を入れる．ホームサーバ，エアコン間の通信の後，エアコンの制御ソフトウェアが動作し，エアコンの状態を変化させる（ルーバの向きの変更等）．この状態変化を可視化機能モジュールが検出し，デバイスの形状をあらかじめ用意してある「動作中」のものに変化させる．また，物理環境シミュレータは，エアコンの設定温度に応じて部屋の室温を変化させる．

4. 評価実験

UbiREAL がスマートスペースアプリケーションの設計開発のための仮想テストベッドとして有用であるためには，（評価項目 1）デバイス間の通信のシミュレーションが十分高速に行われること，（評価項目 2）

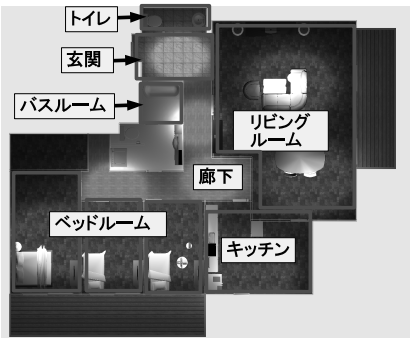


図 5 実験に用いた仮想空間
Fig. 5 Virtual space used for experiment.

可視化機能において仮想空間の変化の様子が滑らかに描画されること，（評価項目 3）開発中のソフトウェアやシナリオに潜む不具合を仮想テストベッドで動作させることで発見できること，が少なくとも必要である．そこで，仮想空間上に，**図 5** に示す 8 部屋からなるスマートホームを構築し，コンテキストに従って情報家電を制御するアプリケーションソフトウェアとルールベースの制御シナリオを開発し，実行した．以下，評価項目 1〜3 に対する実験を 4.1–4.3 節で述べる．実験に使用したマシンは次のとおりである：CPU: Intel Core 2 Duo E6600（2.4 GHz），メモリ：2 GB，OS: Linux 2.6.18.1，グラフィックカード：NVIDIA GeForce 7600 GS.

4.1 テストに要する時間

UbiREAL において，デバイス間の通信のシミュレーションが十分高速に行われることを確認するため，3.6 節で述べたアプリケーションのテスト（ただし，ルールのみ）に関する以下の実験を行った．なお，グラフィックス表示機能はオフに設定している．

仮想空間に温度センサとエアコンを設置する．ルール l として「室温が 26 度以上のとき，エアコンの電源をオンにする」を設定する．温度センサに対し，l が発火する温度を設定した時刻 t_1 と，エアコンの冷房機能をオンにするというアクションの実行が完了した（設計支援・可視化機能モジュールにおけるエアコンの状態がオンに更新された）時刻 t_2 を測定し，$t = t_2 - t_1$ を求める．

実験では，デバイス間は有線で 1 つのハブに接続されているものとした．各デバイスは UPnP プロトコルを介して他のデバイスと通信を行う．ネットワークシミュレータでは，UPnP プロトコルを含むセッション層以上のプロトコルのシミュレーションを行った．また，設定されたルール l に対し，l の発火条件が満たされたとき，対象となるデバイスにアクションの実行要求を送信するためのソフトウェアとして，CADEL フレームワーク[4] を用いた．CADEL フレームワークでは，デバイスの動作ルールとして，センサからの入力の不等式の論理結合からなるルールの発火条件と，条件が成り立ったときに実行されるコマンドの組を指定可能であり，このような動作ルールを GUI 等を用いて直観的に設定すること等が可能である．

上記のテストを 10 回繰り返し，1 回あたりのテストにかかる平均時間を算出したところ，17.5 ms であった．これにより，たとえば，ユーザが 4 人，デバイスが 100 台，各ユーザと各デバイスの組に対しルールが 1 つ存在するような典型的なスマートホームを想定した場合，各ルール，4 人のユーザの存否（$2^4 = 16$ 通り）に対し 100 種類のコンテキストで発火するかどうかのテストを行ったとしても，すべてのテストを $16 \times 100 \times 100 \times 17.5\,\mathrm{ms} \approx 47$ 分程度で行えることが分かる．この値は実用上十分短い．実際には，複数のデバイスに対し複数のルールが同時に発火するような場合が考えられるため，上記の見積りよりも若干遅くなることが予想される．

4.2 GUI 部を用いたシミュレーションの性能

シミュレータのユーザが，現実空間に設置するデバイスが期待どおりに動作するかどうかを仮想空間を通して視覚的に確認するためには，第 1 に現実空間でテストを行う際と同等の速度でシミュレートできることと，第 2 に十分スムーズに仮想デバイスの動作が表示されることが必要である．この 2 点を確認するため，UbiREAL 上で GUI 部を用いてシミュレーションを行う際の性能評価を行った．実験では，GUI 部，ネットワークシミュレータ，CADEL フレームワークを連携して動作させ，デバイスと，そのデバイスに対するルール数を増加させ，ルールすべてを発火させた際の GUI 部のフレームレートを測定した．GUI 部は UXGA（1,600 × 1,200）の解像度で動作させた．図 5 の各部屋には，あらかじめセンサやライト等のデバイスが設置されており，それぞれのデバイスを制御するルールが設定されている．設定されたルールは，可動オブジェクトであるアバタが部屋に入室，退室した際に発火する．アバタの移動軌跡は，玄関を始点とし，廊下，リビングルーム，キッチン，廊下を通り，ベッドルームに至るように設定した（図 2）．

あらかじめ設置済みの合計 21 個のデバイスのみでシミュレーションを行い，その際の平均フレームレートを計測したところ，約 54 fps であった．次に，あらかじめ設置済みのデバイスに加えて，2 つのルールが設定されたデバイスを 10 から 50 まで順次追加して

表 1 制御シナリオの一部
Table 1 Part of specified rules.

発火条件	動作
アリスがいない	ライト 1 の電源を切る
アリスがいない	ヒーター 1 の電源を切る
アリスがいない	扇風機 1 の電源を切る
キャロルがいない	テレビ 1 の電源を切る
アリスがいる	ライト 1 の電源を入れる
アリスがいる ∧ 室温が 28 度以上である	扇風機 1 の電源を入れる
アリスがいる ∧ 室温が 26 度以下である	ヒーター 1 の電源を入れる
キャロルがいる ∧ 月曜日の 21 時以降 22 時以前である	テレビ 1 の電源を入れる
キャロルがいる ∧ 月曜日の 21 時以降 22 時以前である	テレビ 1 のチャンネルを 1 にする

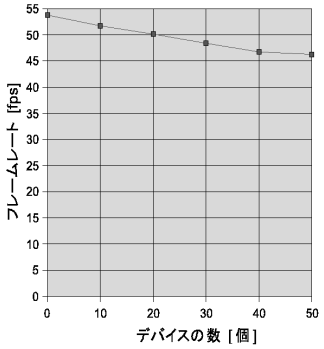


図 6 ノード数の増加に対する GUI 部のフレームレートの変化
Fig. 6 Achieved framerate vs. number of rules.

いき，ルールすべてが発火するように設定した際の平均フレームレートを計測した．いずれの条件においても，30 fps を下回ることはなく，十分スムーズに表示されていることが確認できた．計測結果を**図 6** に示す．図 6 より，デバイス数が 50 個程度まで増え，発火するルール数が増えたとしても，通信，アプリケーションが GUI 部のフレームレートに与える影響はそれほど大きくなく，スマートホーム規模のアプリケーションなら十分実時間でのシミュレーションが可能であることが分かった．

4.3 UbiREAL の有効性の調査

制御シナリオやアプリケーションソフトウェアの不具合の発見に UbiREAL が有用であることを検証するため，図 5 に示す仮想空間上で，デバイスを制御する 3 つのシナリオを用意し，それぞれのシナリオの不具合を被験者 10 人がそれぞれ見つけることができるか，実験を行った．評価実験では，一般家庭の住人のうち，比較的家電製品に対する知識の豊富な人を被験者として想定し，各シナリオに対し，目視でルールの不具合を発見できるかどうかを調べた．それぞれのシナリオには，あらかじめ不具合を含ませており，被験者がそれらの不具合を UbiREAL によりどの程度発見できるかを調べた．設定した制御シナリオのうち，寝室の 1 つに対して設定されたものを**表 1** に示す．

表 2 設置されたセンサとデバイス
Table 2 Deployed sensors and devices.

種類	個数
人感センサ	13
照明	9
テレビ	7
空調	5

また，スマートスペースに設置されたセンサとデバイスの種類と個数を**表 2** に示す．それぞれのシナリオには，あらかじめ不具合を含ませており，被験者がそれらの不具合を UbiREAL によりどの程度発見できるかを調べた．使用したシナリオは以下のとおりである．

シナリオ S1：リビングルーム

(1) アリスが部屋に入ると，照明を薄暗くし，TV を音楽チャネルに設定し，エアコンの温度を 28 度に設定する．(2) デイブが部屋に入ると，照明を明るくし，TV をスポーツチャネルに設定し，エアコンの温度を 24 度に設定する．(3) キャロルが部屋に入ると，照明を明るくし，TV をニュースチャネルに設定し，エアコンの温度を 26 度に設定する．ただし，デイブ，キャロル，アリスの順に優先順位が高いものとする．

シナリオ S2：ベッドルーム

(1) デイブが部屋にいて，室温が 25 度以上のとき，エアコンの温度を 22 度に設定する．(2) キャロルが部屋にいて，室温が 24 度以下のとき，エアコンの温度を 27 度に設定する．ただし，デイブ，キャロルの順に優先順位が高いものとする．

シナリオ S3：フォローミー TV

(1) 月曜の 21 時から 22 時の間，キャロルが TV のある部屋にいれば，そのテレビをドラマチャネルに設定する．(2) キャロルが部屋にいなければ，テレビの電源はオフにする．ただし，テレビはリビングルーム，ベッドルーム，バスルーム，トイレ，キッチンに設置してある．

アプリケーションが予期したとおりに動作しない原因として，ルールの誤設定，デバイス制御用のソフトウェアの不具合，ネットワークの障害のいずれかが考えられる．それぞれのシナリオに含まれる不具合は以下のとおりである．

シナリオ S1 に含まれる不具合

ユーザ間の優先順位設定のミスにより以下の不具合が起こる．(i) アリスとデイブが一緒にいるとき，ライトが明るい状態と薄暗い状態が交互に繰り返される．(ii) アリスとキャロルが一緒にいるとき，テレビのチャネルがスポーツチャネルと音楽チャネルを交互に繰り返す．(iii) デイブとキャロルが一緒にいるとき，エアコンの温度が，優先順位の高いデイブの値に設定されず，26 度に設定される．(iv) アリス，キャロル，デイブが一緒にいるとき，エアコンの温度が，優先順位の高いデイブの温度に設定されず，26 度に設定される．

シナリオ S2 に含まれる不具合

ルール実行系のバグにより，以下の不具合が起こる．(i) ルールの条件以外でも，エアコンの電源が入る．(ii) ルール間で優先順位を設定しても，部屋の温度が振動する．

シナリオ S3 に含まれる不具合

デバイス制御ソフトウェアのバグにより，以下の不具合が起こる．(i) バスルームにあるテレビがドラマチャネルに設定されない．(ii) トイレにあるテレビの電源がオフにならない．

被験者には，UbiREAL の可視化機能を用いて不具合の発見を依頼した．各シナリオにおけるアバタの軌跡は，各ルールが発火するようにあらかじめ設定済みのものを使用した．また，時刻，部屋の温度は，GUI 部を用いて，各被験者が自由に変更できる．

実験結果を**表 3** に示す．結果より，部屋のライトが点滅したり，テレビのチャネルが頻繁に変更を繰り返したりする視覚的に分かりやすい不具合が含まれているシナリオ S1 での発見率が 85%と高いことが分かる．しかし，シナリオ S2 のように温度変化が関係しているルールの不具合の発見率は 60%とシナリオ S1 に比較して低いことが分かった．これは，UbiREAL 上での温度の変化が遅く，被験者がそれを待ちきれずに正常だと判断してしまったことが原因だと考えられる．また，シナリオ S3 の不具合発見率も 65%とシナリオ S2 と同様程度であった．ここで，発見されにくかった不具合は，ルールが設定された住人キャロルが去ったあとに起こるものであり，キャロルを中心にした視点で評価を行う被験者が多かったためであると考えられる．

表 3 各被験者の不具合の発見数
Table 3 Number of bugs found by testees.

シナリオ	被験者										平均
(バグ数)	A	B	C	D	E	F	G	H	I	J	(発見率)
S1 (4)	4	4	3	4	4	4	4	2	3	2	3.4 (85%)
S2 (2)	1	2	0	1	2	1	2	1	1	1	1.2 (60%)
S3 (2)	1	2	1	2	2	1	1	0	1	2	1.3 (65%)

以上より，UbiREAL を用いたシミュレーションにより，ソフトウェア開発の知識がなくても，視覚情報を活用して直観的にルールの誤りを発見することが，ある程度可能であることが分かった．特に，各住人が 1 人で存在するときは正常に動作するが，複数の住人が同室に存在する場合に不具合が起きる場合や，ルールの設定に矛盾はないもののデバイスの設計に問題がある場合等に，実際にそれらのシステムが正常に動作するかを確認することに効果があることが分かった．評価実験では，住人の設定したルールに比較的よくあると思われる誤りに対し，10 人中 9 人の被験者が半分以上の誤りを発見することができ，2 人の被験者がすべての誤りを発見できた．視覚で分かりにくい物理量や，時間的な変化が重要な物理量に関しては，温度等を色をつけて表示する機能やシミュレーション速度を動的に変更する機能を追加することで改善できると考えられる．

本節で用いたシナリオ S1〜S3 は，一般的な邸宅，および，典型的に設定されうるルールベースシナリオであり，これらのシナリオを実際のテストベッドやシミュレータで実行することもできる．今後，テストベッドやシミュレータの評価を目的として，ベンチマークのためのシナリオ集を整備することを計画している．

5. おわりに

本論文では，スマートスペース向けシミュレータ UbiREAL を提案した．情報家電の自動制御を行うルールベースアプリケーションのシミュレーションを通じて UbiREAL の有効性を検証したところ，アプリケーションのテストにかかる時間が十分に短いこと，可視化をともなったシミュレーションが実時間で実行可能なこと，また，実ユーザによるスマートスペースの制御シナリオの記述および動作実験を通して，シナリオやアプリケーションソフトウェアの不具合の発見に UbiREAL が有用であることを確認した．なお，UbiREAL によるアプリケーションの実行の様子を UbiREAL ホームページ[18]で公開している．

今後の課題として，物理環境シミュレータが扱う物理量の数を増やすこと，各物理量のシミュレーション

の精度を向上させること，があげられる．また，ネットワークシミュレータが扱うプロトコル数を増やすことや，物理層等の低位層プロトコルにおいて，特に無線通信のシミュレーション精度を向上させることも今後の課題である．UbiREAL は，仮想デバイスと実デバイスを混在させたシミュレーションが可能である．今後，この機能の有効性を実験により検証したいと考えている．そのため，2 章で述べた既存のテストベッドおよびミドルウェアを UbiREAL 上で実行・連携させることを今後の課題として検討したい．

参 考 文 献

1) Yamazaki, T., Ueda, H., Sawada, A., Tajika, Y. and Minoh, M.: Networked Appliances Collaboration on the UbiquitousHome, *Proc. 3rd Int'l. Conf. on Smart homes and health Telematic* (*ICOST2005*), Vol.15, pp.135–142 (2005).
2) Kawaguchi, N.: Cogma: A Middleware for Cooperative Smart Appliances for Ad hoc Environment, *Proc. Int'l. Conf. on Mobile Computing and Ubiquitous Networking* (*ICMU2004*), pp.146–151 (2004).
3) Biegel, G. and Cahill, V.: A Framework for Developing Mobile, Context-aware Applications, *Proc. 2nd IEEE Int'l. Conf. on Pervasive Computing and Communications* (*PerCom2004*), pp.361–365 (2004).
4) Nishigaki, K., Yasumoto, K., Shibata, N., Ito, M. and Higashino, T.: Framework and Rule-based Language for Facilitating Context-aware Computing using Information Appliances, *Proc. 1st Int'l. Workshop on Services and Infrastructure for the Ubiquitous and Mobile Internet* (*SIUMI'05*) (*ICDCS'05 Workshop*), pp.345–351 (2005).
5) Nishikawa, H., Yamamoto, S., Tamai, M., Nishigaki, K., Kitani, T., Shibata, N., Yasumoto, K. and Ito, M.: UbiREAL: Realistic Smartspace Simulator for Systematic Testing, *Proc. 8th Int'l. Conf. on Ubiquitous Computing* (*UbiComp2006*), LNCS4206, pp.459–476 (2006).
6) Niemelä, E. and Vaskivuo, T.: Agile Middleware of Pervasive Computing Environments, *Proc. 2nd IEEE Annual Conf. on Pervasive Computing and Communications Workshops* (*PerCom 2004 Workshop*), pp.192–197 (2004).
7) Roman, M., Hess, C.K., Cerqueira, R., Campbell, R.H. and Narhstedt, K.: Gaia: A Middleware Infrastructure to Enable Active spaces, *IEEE Pervasive Computing Magazine*, Vol.1, pp.74–83 (2002).
8) Chan, E., Bresler, J., Al-Muhtadi, J. and Campbell, R.: Gaia Microserver: An Extendable Mobile Middleware Platform, *Proc. 3rd IEEE Int'l. Conf. on Pervasive Computing and Communications* (*PerCom2005*), pp.309–313 (2005).
9) Messer, A., Kunjithapatham, A., Sheshagiri, M., Song, H., Kumar, P., Nguyen, P. and Yi, K.H.: InterPlay: A Middleware for Seamless Device Integration and Task Orchestration in a Networked Home, *Proc. 4th IEEE Int'l. Conf. on Pervasive Computing and Communications* (*PerCom2006*), pp.296–307 (2006).
10) Consolvo, S., Arnstein, L. and Franza, B.R.: User Study Techniques in the Design and Evaluation of a Ubicomp Environment, *Proc. 4th Int'l. Conf. on Ubiquitous Computing* (*UbiComp2002*), LNCS2498, pp.73–90 (2002).
11) Nakata, J. and Tan, Y.: The Design and Implementation of Large Scale Ubiquitous Network Testbed, *Proc. Workshop on Smart Object Systems* (*SOBS'05*) (*UbiComp 2005 Workshop*) (2005).
12) Barton, J.J. and Vijayaraghavan, V.: UBIWISE, A Simulator for Ubiquitous Computing Systems Design, Technical Report HPL-2003-93, HP Laboratories, Palo Alto (2003).
13) O'Neill, E., Klepal, M., Lewis, D., O'Donnell, T., O'Sullivan, D. and Pesch, D.: A Testbed for Evaluating Human Interaction with Ubiquitous Computing Environments, *Proc. 1st IEEE Int'l. Conf. on Testbeds and Research Infrastructures for the Development of Networks and Communities* (*TRIDENTCOM'05*), pp.60–69 (2005).
14) Tamai, M., Shibata, N., Yasumoto, K. and Ito, M.: Network Simulation Architecture for Smartspace, *Proc. 2006 System Support for Ubiquitous Computing Workshop* (*UbiSys2006*) (2006).
15) Asbahr, J.L.: Beyond: A Portable Virtual World Simulation Framework, *Proc. 7th Int'l. Python Conf.* (1998).
16) Tanenbaum, A.S. and Steen, M.v.: *Distributed Systems, 2nd edition*, Prentice Hall (2007).
17) Unreal Engine. http://www.unrealtechnology.com/
18) UbiREAL Project. http://www.ubireal.org/

(平成 19 年 5 月 16 日受付)
(平成 19 年 11 月 6 日採録)

西川 博志

2005年神戸大学工学部電気電子工学科卒業．2007年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．現在，（株）ACCESSに勤務．

山本 眞也（学生会員）

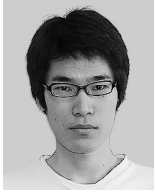
2006年奈良先端科学技術大学院大学情報科学研究科前期課程修了．現在，同研究科博士後期課程在学中．P2P，インタラクション，仮想空間アプリケーションの研究に従事．

玉井 森彦（正会員）

2002年岡山県立大学情報工学部情報システム工学科卒業．2007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修了．現在，日本学術振興会特別研究員．マルチメディア通信システム，分散処理方式に興味を持つ．

西垣 弘二

1994年神戸大学理学部物理学科卒業．同年メルコ・パワー・システムズ（株）に入社，2002年まで勤務．2004年奈良先端科学技術大学院大学情報科学研究科前期課程修了．2006年同研究科博士後期課程修了．現在，（株）ニコンシステムに勤務．



木谷 友哉（正会員）

2002年大阪大学基礎工学部情報科学科卒業．2006年同大学大学院博士後期課程修了．博士（情報科学）．現在，奈良先端科学技術大学院大学情報科学研究科助教．組合せ最適化問題，組み込みシステム，ネットワークシステムに関する研究に従事．IEEE，電子情報通信学会各会員．

柴田 直樹（正会員）

2001年大阪大学大学院基礎工学研究科情報数理系専攻博士後期課程修了．現在，滋賀大学経済学部情報管理学科助教授．分散システム，ITS，遺伝的アルゴリズム等の研究に従事．IEEE会員．

安本 慶一（正会員）

1991年大阪大学基礎工学部情報工学科卒業．1995年同大学大学院博士後期課程退学後，滋賀大学経済学部助手．現在，奈良先端科学技術大学院大学情報科学研究科准教授．博士（工学）．分散システム，マルチメディア通信システムに関する研究に従事．ACM，IEEE/CS各会員．

伊藤 実（正会員）

1977年大阪大学基礎工学部卒業．1979年同大学大学院基礎工学研究科博士前期課程修了．1983年同研究科博士後期課程修了．1979年より大阪大学基礎工学部助手．1986年より大阪大学基礎工学部講師．1989年より大阪大学基礎工学部助教授．現在，奈良先端科学技術大学院大学情報科学研究科教授．関係データベース理論，オブジェクト指向のデータベースのアプリケーション等の研究に従事．ACM，IEEE各会員．