社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2006－MBL－36（41）
2006－UBI－10（41）
2006／2／17

系統的なテストを可能にする ユビキタスアプリケーションシミュレータの提案

西川 博志　山本 眞也　玉井 森彦　西垣 弘二
木谷 友哉　柴田 直樹† 安本 慶一　伊藤 実
奈良先端科学技術大学院大学情報科学研究科 † 滋賀大学情報管理学科

本稿では，多数の情報通信機器をコンテキストおよび利用者の好みに従い適切に制御するユビキタスアプリケーションを対象に，それらのアプリケーションを高信頼かつ低コストで開発できるようにするためのシミュレータを提案する．提案するシミュレータは，3D 仮想空間上でアプリケーションソフトウェアの動作を現実世界そっくりにシミュレートする．そのため，3D 空間への仮想デバイス設置支援機能，デバイス間の通信を MAC 層レベルでシミュレートする機能，エアコンなどの機器の動作による空間の物理量（室内温度など）の時間的変化を再現する機能などを実現するとともに，各デバイス制御用ソフトウェアはソースレベルで現実のものと互換性を持たせる．また，与えた環境設定およびアプリケーションソフトウェアが意図した通りに動作することを系統的かつ視覚的に調べるテスト機能を実現する．

A Simulator for Systematic Testing of Ubiquitous Applications in 3D Virtual Space

Hiroshi Nishikawa　Shinya Yamamoto　Morihiko Tamai　Kouji Nishigaki
Tomoya Kitani　Naoki Shibata† Keiichi Yasumoto　Minoru Ito
Nara Institute of Science and Technology † Shiga University

In this paper, we propose a simulator for facilitating reliable and inexpensive development of ubiquitous applications where each application software controls a lot of information appliances depending on the external environment and the user's contexts and preferences. The proposed simulator reproduces the realistic behavior of application software working with the virtual devices in a virtual 3D space. For this purpose, it provides the functions which facilitate deployment of virtual devices in a 3D space, simulate communication among the devices in MAC level, and reproduce the change of physical quantities (e.g., temperature) caused by devices (e.g., air conditioners). Also, we keep software portability between virtual devices and real devices. As the most prominent function, we provide a systematic and visual testing method which tests whether a given application software works satisfying specified requirements.

1　はじめに

ユビキタス社会の実現は，今日の情報通信分野における最も重要な課題の一つである．そのため，ネットワーク基盤の整備に加え，多数の情報通信デバイスおよびセンサを外部環境の状況，利用者のコンテキストや好みに従って適切に制御する**ユビキタスアプリケーション** を効率よく開発するための技術が求められている．そこで，様々なユビキタスアプリケーションに共通の機能をミドルウェアとして整備する研究や，ユビキタスアプリケーションの動作テストを目的としたテストベッドの構築に関する研究が盛んに進められている [6, 7, 8, 9].

ユビキタスアプリケーションは我々の生活に密接に関わってくるため，それらの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配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）に様々な行動を取ってもらう必要があり，実施には莫大な労力とコストがかかる．また，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み合わせは多数存在し，それら全てについてユビキタス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．

本稿では，多数のデバイスが仮想的に設置された 3D 仮想空間（スマートスペースと呼ぶ）上でユビキタスアプリケーションの実行を行うシミュレータを構築し，かつ，本シミュレータを用いてユビキタスアプリケーションの系統的なテストを行う機能を実現する．本シミュレータを以後 UbiREAL (Ubiquitous Application Simulator with REAListic Environments) と呼ぶ.

UbiREAL では，シミュレータ上で動作するデバイスの制御用ソフトウェアを現実のものと互換にすることで，シミュレーションにより正しく動作することが確認されたアプリケーションを現実空間でほぼそのまま動作させることを目指す．以上の目的のため，UbiREAL は，3D 空間への仮想デバイス設置支援機能，デバイス間の通信を MAC 層レベルでシミュレートする機能，エアコンなどの機器の動作による空間の物理量（室内温度など）の時間的変化を再現する機能，アプリケーションの動作状況を 3D グラフィックスにより可視化する機能などを提供する．さらに，UbiREAL の最大の特徴として，与えた環境設定およびアプリケーションソフトウェアが意図した通りに動作することを系統的に調べるテスト機能を実現する．

以降，2 章で関連研究，3 章で UbiREAL の概要を述べる．4 章でスマートスペース設計支援機能，5 章でネッ

トワークシミュレーション機能，6 章で物理環境シミュレーション機能，7 章で系統的なテスト機能について述べ，最後に 8 章でまとめを述べる．

2 関連研究

仮想空間を用いたユビキタスコンピューティング環境のシミュレータがいくつか提案されている．Hewlett-Packard Laboratories において UBIWISE [1] というシミュレータが提案されている．UBIWISE 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，または，デバイスの組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 という二つのシミュレータから構成されており，UbiSim でシミュレートされた 3D 仮想空間を生成し，WISE によって，シミュレートされたデバイスの 2D 表示を行う．

また，TATUS [2] というシミュレータが Trinity College の研究チームによって提案されている．TATUS とネットワークシミュレータを統合することで，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ション，テストを可能としている．ただし，TATUS は 3D ゲームエンジンを用いて開発されており，仮想空間上の利用者キャラクタをテスト実行者が操作して移動させる必要がある．また，仮想空間上で複数の人を動かす場合は，利用者キャラクタの動きは簡単な AI によって制御されるようになっている．

これら既存のシミュレータは，利用者キャラクタの行動を手動もしくは単純な方法でしか再現できないため，特定の限られたケースについてしかシミュレーションが行えない．そのため，与えられたユビキタスアプリケーションのプログラムが，実際に起こりうるあらゆる行動パターンに対し，予期したとおりに動作することを系統的に確かめたい，という目的には十分でない．また，これら既存シミュレータは，デバイスの動作による空間物理量の変化を正確にシミュレートしておらず，アプリケーションの実現の正しさを保証することは難しい．

3 シミュレータの概要

本章では，提案するシミュレータ 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ションの目的

UbiREAL がシミュレーションの対象とするアプリケーションの例を以下に示す．

「私」,「父」,「母」からなる家庭において，リビングルームにおける家電（デバイス）の制御について考える．リビングルームには，ネット対応のステレオ，テレビ，ビデオレコーダ，照明，エアコンが設置されており，外部環境および各ユーザのコンテキストに従ってそれらネット家電の制御を行うアプリケーションがホームサーバで実行されていると仮定する．本アプリケーションによる家電制御の一例を以下に示す．

> ある夏の日の夕方,「私」がリビングルームに行くと,「私」の好みに合わせ，部屋の明るさが薄暗く調整され（例えば，スポットライトによる間接照明），ステレオが好みの音楽（例えば Jazz）を大音量で演奏し，エアコンが快適な温度，湿度（気温 25 度，湿度 60%）に設定された．その後，父，母が帰宅しリビングに入ってくると，全員がリビングにいるときの設定（照明は明るく，気温 27 度，湿度 60%）に，また，ステレオが小音量になるよう，関連するデバイスが自動的に制御された．

上記は，アプリケーションの動作の一例であるが，アプリケーションが予期した通りに動作しているかどうかを，あらゆる状況を想定して確認できることは非常に重要である．しかし，上記のアプリケーションは，私と母が同時にリビングにいる時にも Jazz が大音量で演奏されたり，3 人がリビングにいる時に照明を消灯するなど，望ましくない動作，設定されていない動作を起こすかも知れない．しかし，上記の例のように限られたパターンでの動作を観測するだけでは，このような不具合を漏らさず検出することは難しい．

コンテキストを利用するユビキタスアプリケーションの開発を行う場合，様々なコンテキストに対して，アプリケーションが正常に動作する事を保証する必要がある．そのためには，あらゆる状況を想定してテストを行う必要がある．しかし，実際にテスト環境を構築するには莫大な労力と費用が生じる．例えば，情報家電の自動制御を行うアプリケーションのテストを考えた場合，実際の家，部屋を構築し，その環境下で起こりうる様々な条件において，アプリケーションが正常に動くかをテストする必要がある．このような労力，費用を軽減するために，我々はユビキタスアプリケーションを系統的にテストできるシミュレータ UbiREAL を提案する．UbiREAL では，テストで正しく動作することが確認できたアプリケーションソフトウェアを，実空間でも大きな変更無くそのまま動作させることができるようにすることを目標とする．

3.2 UbiREAL の構成

UbiREAL シミュレータは，(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の 4 つの部分から構成される (図 1)．UbiREAL への入力となる，各種デバイスで動作するソフトウェア，および，ホームサーバなどで実行されるアプリケーションプログラムは，実環境で動作する任意のものを対象とする．ただし，これらのソフトウェアは UPnP などの一般的な情報家電向け技術仕様で定義されている通信 API（SOAP や UDDI などのプロトコル）を用いて作成されると仮定する．ただし，デバイスドライバについては，デバイスの動作を可視化し，またテスト結果の判定を自動で行えるようにするため，シミュレータがいつでもデバイスの動作状態を取得できるよう，これらのソフトウェアのコードが若干修正されているものとする．また，デバイスドライバについてはセンサ等からの入力を受け付けるソフトウェアドライバとは別に，人などからの手動入力（スイッチを押す，リモコンを操作する等）を受け付ける機能も必要に応じて用意する．以下，シミュレータを構成する 4 つの機能の概要を述べる．

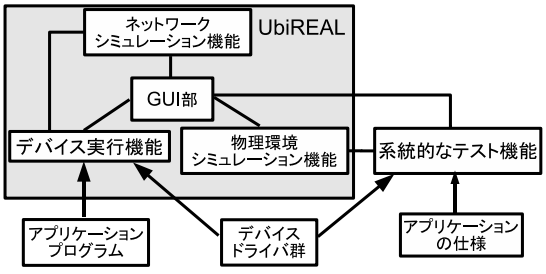


図 1: UbiREAL のアーキテクチャ

スマートスペース設計支援・可視化機能 仮想 3D 空間の作成支援，仮想のユビキタスアプリケーションデバイス（センサー，アクチュエーター，テレビ等）の配置支

援，人などのオブジェクトの移動軌跡の設定，結果の表示など，UbiREAL の GUI 部分に相当する．詳細は 4 章に記述する．

ネットワークシミュレーション機能　仮想空間上でのデバイス間の距離，遮蔽物などの影響を考慮して，デバイス間の通信を現実空間に近い精度でシミュレートする．詳細は 5 章に記述する．

物理環境シミュレーション機能　物理環境シミュレータでは，仮想空間上での物理環境（様々な物理量の変化）をシミュレートする．詳細は 6 章に記述する．

系統的なテスト機能　与えられたユビキタスアプリケーションがその仕様（要求）に対して正しく動作しているかを系統的にテストする機能を提供する．詳細は 7 章に記述する．

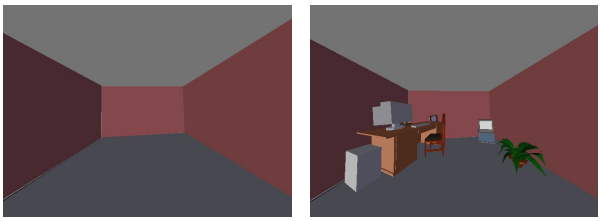
4　スマートスペース設計支援・可視化機能

UbiREAL は，スマートスペースを設計するための機能とシミュレーション結果を視覚的に表示する GUI を提供する．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．

4.1　スマートスペースの設計支援

本機能モジュールは，市販の 3D モデリングソフトなどで作成した VRML データをもとに，GUI によりデバイスの配置を行う機能を提供する．なお，ユビキタスアプリケーションが実行される 3D 空間は，既存建物の平面図の CAD データを市販の 3D モデリングソフトなどで 3D 化することによって作成する．作成した 3D 空間に，様々なオブジェクト（VRML 記述）を好きな位置に設置して行く（図 2）．オブジェクトは，机や椅子などの家具や，各種ユビキタスデバイス（センサ，テレビ，エアコン等）に相当し，その形状は 3D モデリングソフトで VRML データとして設計する．各デバイス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．

空間上への静止オブジェクトを配置は，登録されているオブジェクトのリストからの選択し，ドラッグ&ドロップで任意の位置に設置する．



(a) オブジェクト配置前　　(b) オブジェクト配置後

図 2: 仮想空間の様子

人などの稼動オブジェクトの 3D 空間上への配置は，移動軌跡を指定することで行う（図 3）．移動軌跡の指定は，手動で行うか，あるいは，系統的テスト機能により自動で行う．軌跡は複数の座標およびその訪問順序によって定義される．また，軌跡上を動くオブジェクトの速さや，軌跡上の任意の座標において，アクション（ボタンを押す，リモコンを操作する等）の実行を指定することもできる．人を表す稼動オブジェクトのグラフィックを以後，**アバタ**と呼ぶ．

なお，配置した全てのオブジェクトの位置情報は GUI 部により管理され，ネットワークシミュレーション部，物理環境シミュレーション部で利用される．

(a) 軌跡の設定前　　(b) 軌跡の設定後

図 3: アバタの移動軌跡の設定

4.2　スマートスペースの可視化

3D 空間上のあるデバイスでアクションが実行されることによりデバイスの状態が変化したり，外部環境が変化する際には（ライトが点く，エアコンが作動する），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．これは，GUI 部が，各デバイスのデバイスドライバと通信し，デバイスの状態を取得することにより行う（図 4）．

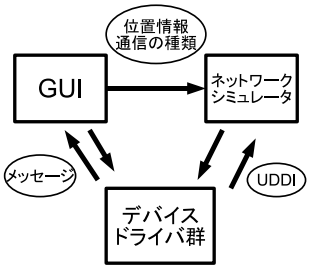


図 4: 各機能の相互通信

具体的な動作手順を，人（RFID タグ）がタグリーダに近づくことにより，エアコンでアクション（設定気温２５度でオン）が実行されるケースをもとに説明する．タグを持った人がタグリーダに近づいた時，図 4 に示すように，GUI 部が持つオブジェクト群（タグとタグリーダ）の位置情報をネットワークシミュレータ部が取得し，これらのデバイス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．次にこの情報をアプリケーションを実行するホームサーバ（デバイスドライバ）が取得し，エアコンに対し，アクション（設定気温２５度でオン）を実行する．ホームサーバ，エアコン間の通信の後，エアコンのデバイスドライバが指定されたアクションを実行し，その情報が GUI 部，物理環境シミュレータ部に送られる．最後に，GUI 部はデバイスの形状を「動作中」のものに変化させ，物理環境シミュレータ部は物理環境の状況を変化させる．

以上のことを可能にするためには，デバイスドライバが UbiREAL の GUI 部と通信できるよう実装されている必要がある．この詳細については，5 章で述べる．

UbiREAL では，シミュレーションの結果をログに残し，ログを入力として，スマートスペース上でのアプリ

ケーションの動作の様子を可視化する機能を持つ．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，指定したアバタの一人称の視点からの 3D ビュー，など任意のビューを設定できる．アバタが複数ある場合は，それらを切り替えて表示する機能も提供する．

5 ネットワークシミュレーション機能

本章では，UbiREAL のネットワークシミュレーション機能について述べる．はじめに，既存のネットワークシミュレータとの相違点について述べた後，提案シミュレータのアーキテクチャについて述べる．

5.1 実環境用ソフトとのソースコードの共用

ns2 等既存のネットワークシミュレータは，シミュレーション用の実装を実環境で動作させることは基本的に考えられていない．そのため，これらのシミュレータの利用者は，実環境用のコードとは別に，シミュレータ用のコードを新たに作成する必要がある．開発効率の観点からは，このような手間を省き，実環境用のコードとシミュレータ用のコードが共用できることが望ましい．しかし既存のシミュレータの多くは，シミュレータ用のコードをイベントドリブンで記述する必要があり，実環境用のフロードリブンで記述されたコードとの共用を行うことは困難である．

UbiREAL のネットワークシミュレータでは，シミュレータ用のコードとして，フロードリブンで記述されたコードを利用可能とする．これにより，シミュレータ用のコードと実環境用のコードの共有を可能とする．実環境で動作するバイナリとシミュレータ用バイナリは，コンパイル時のオプションを変更することで，それぞれ同一のソースから作成できるようにする．ただし，4 章で述べたように，UbiREAL でシミュレーションを行う際には，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付加する必要がある．

5.2 アーキテクチャ

UbiREAL のネットワークシミュレータは，シミュレータのユーザに対し，通常の TCP/IP プロトコルスタックのサービスプリミティブを提供する．この上に，UPnP 等のプロトコルを動作させるための既存のコードを追加し，実行する．シミュレートする通信プロトコルとして，MAC 層に関しては，通常のイーサネット (10Mbps, 100Mbps, 1000Mbps) に加え，IEEE802.11a/b/g，ZigBee，Bluetooth などをサポートする．また，ネットワーク層では，AODV や DSR などのアドホック通信プロトコルをサポートする．プロトコルの各層毎にシミュレーションの粒度の異なったモデルを用意し，シミュレーションの精度と実行時間に応じてユーザが選択できるようにする．

シミュレータの動作としては，既存のシミュレータと同様に，各層の動作をソフトウェアでシミュレートする．以下では，人の移動に伴って移動する無線 LAN デバイスが，アクセスポイントの電波範囲外から電波範囲内に移動する際の例を用いて，無線 LAN デバイス同士の通信がシミュレートされる様子を説明する．

可動デバイスの位置は刻々と変化する．これに伴ってデバイス間の通信状況も変化するが，これは，物理層のシミュレーションによって再現される．物理層では，各フレームがデバイスドライバから送信される毎に，デバイスの位置を管理する GUI 部にデバイスの位置を問い合わせ，また遮蔽物の位置なども考慮してそのフレームを受信することのできるデバイスを決定する．また，電波状況等に応じてフレームにエラーを混入させる．ここで，位置の問い合わせ等の GUI 部との通信は通常のメソッド呼び出しを用いる．電波範囲外から電波範囲内に可動デバイスが移動すると，無線範囲内の他のデバイスからのビーコンを受信できるようになり，その結果それらのデバイスとの通信が可能になる．フレームが受信できると判断されたデバイスは，フレームを受信し，それを上位層プロトコルに受け渡す．

5.3 仮想空間と現実空間を跨ったデバイス間通信

UbiREAL のネットワークシミュレータでは，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同士の IP プロトコルによる通信を可能とする．これにより，テストの際に特定のデバイスを現実空間に存在するものと置き換えることが可能となる．例えば，テストの対象となる空間にリビングと寝室が存在し，寝室に存在するデバイスには非常に高精度な動作が求められる場合，リビング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，といったことが可能となる．また，スマートスペース上の一部の動作を，実際に人間の感覚を通してテストしたいといった場合にも有用な機能である．

Linux 等の OS では，IP パケットのフレームを自由に作りだし，送受信することが可能なため，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へのルータに見せかけ，仮想デバイスと実デバイスの間で通信させることが技術的に可能である．また，一部のネットワークカードでは送信するパケット毎に MAC アドレスを変更することも可能であるため，このようなネットワークカードを使用すれば，仮想デバイスと実デバイスを同じネットワークの上で動作しているように見せかける事も可能である．

6 物理環境シミュレーション機能

物理環境シミュレータでは，物理環境（様々な物理量の変化）をシミュレートする．過去，様々なリアルタイム 3D 環境のためのシミュレータの研究や開発がなされており，文献 [3] においてサーベイされている．また，卑近な例でいうと，Unreal Game Engine[4] の中で高度な物理エンジンが使用されており，光や音の効果が計算されている．

我々の物理環境シミュレータでは，各物理量は，仮想デバイスのアクションや他の物理量の変化によって変化し，各物理量は，自身の変化を，仮想デバイスや他の物理量などの物理量情報を必要とするオブジェクトに通知する（以下，通知先のオブジェクトをサブスクライバと呼ぶ）．

また，我々の物理環境シミュレータでは，物理量として，気温，湿度，音，光，空気の動き，電力などをシミュレートする．各物理量のシミュレーションは，計算時間を少なくする目的で，できるかぎり単純化する．例えば，気温では，仮想デバイスなどから得られる熱量と空間の熱容量のみで計算する．例えば，次の式で記述する．

$\Delta T(t) = \frac{Q(t)}{C}$

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（部屋毎に定数を仮定）である．また，電波の干渉では，空間における遮蔽物（壁やドアを開けたときと開けないときなど）と減衰と透過だけをシミュレートする．

各物理量のシミュレーションは Plug-In として実装され，新たな物理量の追加およびより高度なシミュレーションの実装が可能である．物理量の変化をサブスクライバに通知するには，Publish-Subscribe モデルを用いる．あ

る物理量の変化の通知が必要な仮想デバイス，物理量は，その物理量に対してデータの通知を予約する．

7 系統的なテスト機能

UbiREAL シミュレータでは，与えられたアプリケーションの正しさを系統的にテストする機能を提供する．そのため，任意のアプリケーションの仕様を表すための形式モデルを定義する．アプリケーションの仕様をもとに，与えられたアプリケーションの実装の正しさを定義する．その後，正しさを機械的にテストする方法を提案する．

7.1 アプリケーション仕様の記述モデル

対象とするスマートスペース U を 2 項組 $U = (R, D)$ で定義する．ここで，$R = \{r_1, ..., r_n\}$ は U 内の部屋の集合，$D = \{d_1, ..., d_m\}$ は U 内に設置されているデバイスの集合である．

各部屋 $r_i \in R$ は，属性として，U における r_i の位置 pos，底面の形状・大きさを表す $bottom$，容積 cap，r_i の持つ各物理量（気温 $temp$，湿度 $humid$，熱容量 $heatcap$，明るさ $bright$，音量 $sound$ など）を持つ．これらの属性を $r_i.temp$ のように表記する．なお，部屋どうしを接続するドアが開いている場合には，接続された複数の部屋を一つの部屋とみなす．また，各部屋の物理量の初期値は予め与えられるものとする．

各仮想デバイス $d_j \in D$ は，属性として，どの部屋に設置されているかを表す r，部屋内の位置 pos，状態 $state$ を持つ．これらの属性を $d_j.state$ のように表記する．各デバイス d_j にはセンサを持つものがあり，$d_j.pos$ の位置で外部環境から物理量の現在値を取り込み，センサの値として保持するとする．また，d_j は，指定されたコマンド（冷房，除湿，ライトの点灯・消灯，手動でのスイッチオンなど）を実行することにより，状態 $d_j.state$ を変化させたり，部屋 $d_j.r$ の物理量を変化させることができる．

スマートスペース U において，各部屋の属性値，各デバイスの属性値は，それぞれ，ある範囲内で任意の値をとりうる．全ての部屋，全てのデバイスの属性のそれぞれに，対応する範囲内の値を代入してできる値と，日付，時間の組を U の状態と定義する．

提案するテスト手法では，アプリケーションの仕様 $Spec$ は有限個のルールの集合 $AP = \{l_1, ..., l_k\}$ および成り立って欲しい性質の集合 $P = \{p_1, ..., p_l\}$ により与えられるとする．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．発火条件 c_i は，各センサが外部環境から取り込んだ値を表す変数および定数からなる一次不等式により表されるものに限定する．また，アクション a_i には，あるデバイスに対するコマンドが指定される．

AP 中に同時に発火可能なルールが複数ある場合，それらが同一のデバイスに対し，相反するアクションを実行することはないと仮定する[1]．

アプリケーションにおいて成り立って欲しい各性質 p_k は，時相論理 CTL[5] を用いて次のような命題として指定する．

$$AG(\phi_1 \Rightarrow AF(\phi_2))$$

ここで，ϕ_1, ϕ_2 は命題を表し，スマートスペース U の状態ごとに真偽（$true$, $false$）が定義されているものとする．$AG(\phi)$ は，命題 ϕ が常に成り立つことを表し，$AF(\phi)$ は，ϕ がいつかは成立することを表す．すなわち，上記の命題は「『命題 ϕ_1 が真である状態から，いつかは命題 ϕ_2 が真となる状態に遷移する』が常に成り立つ」ということを表す．

例えば，「ユーザ 1 が部屋 A にいる時は，いつかは気温が 23 度，湿度が 60%近辺に調整される．」という性質を記述したいときは，ϕ_1 として，「ユーザ 1 が部屋 A にいる」，ϕ_2 として，「部屋 A の気温が 23 度± 1 度，湿度が 60% ± 1%である」のように指定すればよい．

7.2 アプリケーションの実装の正しさとテストの方法

与えられたユビキタス空間 U とその初期状態，アプリケーションの仕様 $Spec$（ルールの集合 AP，成り立って欲しい性質の集合 P），アプリケーションの実装 I に対し，以下の全ての条件が満たされる時，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) 全てのルール $l = (c, a) \in AP$ 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) 全ての性質 $p \in P$ 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）を全て生成し，I 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良い．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，全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，それら全ての値に対し，(1) を確認できれば，I は正しく実装されていると考える．C の値を大きくすることで，テストの精度を高めることが可能であるが，その分，テストのコストが増大する．そのため，C の値はトレードオフを考慮のうえ，テスト実行者が決定するものとする．例えば，$C = 5$ の場合，ルールの発火条件「気温が 28 度以上，かつ，湿度が 70%以上」に対し，(気温，湿度）= (28,70), (28, 80), (32, 70), (32,80), (36,90) のようにサンプル値を生成することが可能である．

一方，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，提案手法では，6 章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方法を採用する．

物理量のシミュレーションは部屋単位で行うこととする．また，部屋にドアや窓がある場合には，前述のように，ドアの開放により接続される複数の部屋を一つの部屋と想定する，あるいは，初期物理量の値をドアや窓の開閉状態に合わせて設定するなどの方法をとる．各ドア・窓について，閉じている場合，開けている場合について，U の初期状態を計算し，それぞれの場合に，提案するテストを実施することで，ドア・窓の開閉を考慮した系統的テストが可能である．

また，同じ部屋で複数のデバイスが同時に動作している場合，物理量の変化の仕方が変わるかも知れない．例えば，エアコンを動作させる場合，気温や湿度を制御しないデバイス（TV や PC など）が動作している場合とそうでない場合で物理量の時間変化は異なってくる可能性がある．そこで，同じ部屋に設置されている同時に起動される可能性のある n 個のデバイスについて，そのオン・オフの組み合わせ 2^n 通りについて，各性質 p がい

[1] そのような競合が生じる場合，優先順位をつけるなどの方法で事前に対処されているものとみなす．

つかは成立するかどうかをテストする．ただし，n が大きい場合には，影響の少ない組み合わせについては，手動で除外するなどの方法で，テストの実行時間を抑える必要がある．

7.3 テスト系列の生成

本節では，具体例を使って提案するテスト手法を説明する．アプリケーションの仕様 $Spec$ が以下のルールの集合 AP と性質の集合 P として与えられるとする．

$AP = \{(Exist(u_1, r_1) \wedge 28 \leq r_1.temp \wedge 70 \leq r_1.humid, Aircon_1.on(24, 60)),$
$(Exist(u_2, r_1) \wedge \neg Exist(u_1, r_1) \wedge 30 \leq r_1.temp \wedge 60 \leq r_1.humid, Aircon_1.on(28, 50)),$
$(Exist(u_1, r_2) \wedge Exist(u_2, r_2) \wedge Day = "Sun" \wedge Time = "6 : 30pm", TV_2.on("フジ")),$
$("11 : 00pm" < Time, Lamp_1.off), ...\}$

$P = \{AG(Exit(u_1, r_1) \Rightarrow AF(23 \leq r_1.temp \leq 25)),$
$AG("11 : 00pm" < Time \Rightarrow AF(r_1.bright \leq 10)\}$

AP 中の 1 番目のルールは，ユーザ u_1 が部屋 r_1 におり，かつ，気温，湿度がそれぞれ 28 度，70%以上の時に，エアコンを設定気温 24 度，設定湿度 60% で動作させることを指定している．2 番目のルールは，ユーザ u_2 が部屋 r_1 にいる時のエアコンの制御を指定しているが，ユーザ u_1 が同時に部屋 r_1 にいる時は u_1 の設定を優先させることを指定している．3 番目のルールは，u_1 と u_2 が日曜日の午後 6 時 30 分に同時に部屋 r_2 にいる時は，テレビ TV_2 のスイッチをオンにし，チャネルをフジテレビに合わせることを指定している．4 番目のルールは，時間が午後 11 時を超えたら，照明 $Lamp_1$ を消灯することを指定している．

P 中の 1 番目の性質は，ユーザ u_1 が部屋 r_1 にいるなら，いつかは r_1 の気温が 23 度から 25 度の範囲内に調整されることを表している．2 番目の性質は，時間が午後 11 時以降なら，いつかは部屋 r_1 の底面の照度が 10 ルクス以下になることを表している．

上記のルールに対し，提案手法では，以下のテスト系列を生成する．

(1) $u_1.pos \leftarrow r_1.pos + \%v11; r_1.temp \leftarrow \%v12; r_1.humid \leftarrow \%v13; Check(Aircon_1.on(24, 60))$
(2) $u_1.pos \leftarrow r_1.pos + \%v21; u_2.pos \leftarrow r_1.pos + \%v22; r_1.temp \leftarrow \%v23; r_1.humid \leftarrow \%v24; Check(Aircon_1.on(24, 60))$
(3) $u_2.pos \leftarrow r_1.pos + \%v31; u_2.pos \leftarrow r_3.pos; r_1.temp \leftarrow \%v32; r_1.humid \leftarrow \%v33; Check(Aircon_1.on(28, 50))$
(4) $u_1.pos \leftarrow r_1.pos + \%v41; u_2.pos \leftarrow r_1.pos + \%v42; Day \leftarrow "Sun"; Time \leftarrow "6 : 30pm"; Check(TV_2.on("フジ"))$
(5) $Time \leftarrow \%v51; Check(Lamp_1.off)$

上記において，$\leftarrow$ は値の代入を表し，$Check(a)$ は，アクション a が実行された時に $True$ を返す関数である．また，$\%v11$ などはパラメタであり，様々なパラメタ値を生成し，繰り返しテスト系列を実行する目的で使用される．上記全てのテスト系列を各パラメタ値に対して実行し，$Check$ の結果が全て $True$ の場合にテストは成功，そうでない場合テストは失敗と判定する．

また，上記の性質に対し，提案手法では，以下のようなテスト系列を生成する．

(1) $u_1.pos \leftarrow r_1.pos + \%v11; r_1.temp \leftarrow \%v12; r_1.humid \leftarrow \%v13; CheckProp(23 \leq r_1.temp \leq 25)$
(2) $Time \leftarrow \%v51; CheckProp(r_1.bright \leq 10)$

上記において，$CheckProp(\phi)$ は，一定時刻ごとに，スマートスペース U の現在の状態において命題 ϕ が成立しているかどうかを調べ，成立すると $True$ を返し，一定時間経過後までに成立しなければ $False$ を返す関数である．各ルールのテスト方法と同様に上記テスト系列を実行することでテストに対する成否を判定する．なお，各ルールのテスト系列と各性質のテスト系列で共通する部分については同時にテストを実行することで，テスト時間の短縮を図ることも可能である．また，テストしたい複数のデバイスの位置を最短距離で結ぶ経路を算出し，上記のテスト系列におけるユーザの位置 $u_1.pos, u_2.pos$ に代入することで，結果として最短時間で多くのテスト系列を連続的にテストすることが可能になると考えている．

8 おわりに

本稿では，ユビキタスアプリケーションシミュレータ UbiREAL を提案した．UbiREAL では，環境の変化，人などの可動オブジェクトの行動パターンを系統的に生成することで，アプリケーションの実装の正しさを系統的にテストすることを可能にする．今後，本稿で提案した各機能モジュールを実装し，シミュレータの性能や，系統的テストにかかる時間についての評価を行っていく予定である．

参考文献

[1] J. J. Barton, V. Vijayaraghavan：UBIWISE, A Simulator for Ubiquitous Computing Systems Design, Technical Report HPL-2003-93, HP Laboratories, Palo Alto, 2003.
[2] E. O'Neill, M. Klepal, D. Lewis, T. O'Donnell, D. O'Sullivan, D. Pesch：A Testbed for Evaluating Human Interaction with Ubiquitous Computing Environments, 1st International Conference on Testbeds and Research Infrastructures for the Development of NeTworks and Communities, pp 60-69, 2005.
[3] J. L. Asbahr: Beyond: A Portable Virtual World Simulation Framework, Asbahr.com.
[4] Unreal Engine: http://www.unrealtechnology.com/
[5] E. M. Clarke, E. A. Emerson, A. P. Sistla：Automatic verification of finite state concurrent systems using temporal logic specifications, ACM Trans. on Program Languages and Semantics, 8, 2, pp. 244-263, 1986.
[6] K. Nishigaki, K. Yasmoto, N. Shibata, M. Ito, T. Higashino：Framework and Rule-based Language for Facilitating Context-aware Computing using Information Appliances, Workshop on Services and Infrastructure for the Ubiquitous and Mobile Internet (SIUMI'05) (ICDCS'05 Workshop), pp. 345-351, 2005.
[7] T. Yamazaki, H. Ueda, A. Sawada, Y. Tajika, Michihiko Minoh：Networked Appliances Collaboration on the Ubiquitous Home, 3rd International Conference On Smart homes and health Telematic (ICOST 2005), Vol.15, pp.135-142, 2005.
[8] N. Kawaguchi：Cogma：A Middleware for Cooperative Smart Appliances for Ad hoc Environment, International Conference on Mobile Computing and Ubiquitous Networking(ICMU), 2004.
[9] G. Biegel, V. Cahill：A Framework for Developing Mobile, Context-aware Applications, 2nd IEEE International Conference on Pervasive Computing and Communications (PerCom2004), pp. 361-365, 2004.