Vol.2011-SLDM-149 No.31
Vol.2011-EMB-20 No.31
2011/3/19

無線によるユビキタス情報配信システムの設計・構築手法の検討

吉　田　信　明[†1]

多数の組み込み機器などを広範なエリアに展開し，無線を主としたネットワークを活用してサービスが提供されるユビキタス情報配信システムでは，ネットワークにサービスが強く依存しているため，使用されるソフトウェアの設計が困難になりがちである．また，開発時においても，多数の機器を取り扱うことによる特有の複雑さがある．本稿では，著者らが構築した実験システムに基づき，このような課題を捉えた形式手法による設計・構築手法について検討する．

A study of the designing method for ubiquitous information systems with wireless networks

Nobuaki Yoshida[†1]

In ubiquitous information systems, services are offered with numerous embedded devices and wireless networks which are deployed to wide service area. This feature makes their software development highly depends on their network architectures, therefore it tends to be difficult. In addition, setup and deployment process of devices also has particular complexity. In this paper, we consider designing method of such systems with formal methods which captures these problems.

1. はじめに

著者らは，Wi-Fi と Bluetooth を活用した観光情報ナビゲーションシステムのプロトタイプ[1)] を構築し，2008 年 12 月と 2009 年 1 月，京都市の三条通，烏丸〜寺町間およびその周辺店舗・施設において，実験を行った[2)]．このシステムでは，実験エリア内の各ポイントに組み込み Linux ボードで構築された Wi-Fi と Bluetooth のアクセスポイントを配置し，実験用のアプリケーションがインストールされた利用者端末（ノート PC） から無線による接続を行い，アクセスポイントに格納しているテキストおよび画像のコンテンツをダウンロードすることで，被験者（利用者）にその周辺の観光情報や店舗情報を配信した．

この実験システムの構築と実験運用を通じて，次のような課題が発生した:

- アプリケーションの動作が不安定なため，ネットワーク構成の見直しを迫られた
- 多数の組み込み機器を地理的に分散して配置するため，それらに格納されるソフトウェアやコンテンツに修正が必要となった場合に非常に手間取った

これらの課題は，より一般的な，ユビキタス環境における情報配信システムにおいても同様に発生すると考えられる．この種のシステムでは，しばしば，システムの目的に合わせてネットワーク構成が選択され，それにあわせたアプリケーションが開発されるため，ネットワーク構成の調整・変更を行うと，アプリケーションもこれに合わせて調整・変更が必要になる場合がある．また，このような開発では多数の組み込み機器をセットアップする必要があるが，システム展開後の修正はコストがかかるため，正しい構築が強く求められる．

よって，上記の課題の解決により，システム開発やその維持における負荷・コストを低減できると考えられる．本稿では，実験を踏まえ，この 2 つの課題を解決するための形式手法を用いたユビキタス情報配信システムの設計・開発手法について検討を行う． 2 節では，上記の実験で使用したシステムとその構築方法の概要を述べる．これを踏まえ，3 節では，このような形態をとる情報システムの設計手法について検討する．3.2 節においてネットワークとアプリケーションの設計段階での相互の影響をなるべく排除する手法を，検討する．3.3 節では，動作不良につながる構築時の誤りを排除する手法について，検討する．

2. 三条通における実験システム概要

実験システムは，コンテンツを受信し画面に表示する利用者端末（WindowsXP のノート PC）と，これらの端末にコンテンツを提供する，実験エリア（京都三条通烏丸〜寺町間，および周辺施設）に設置された Wi-Fi（5 台）および Bluetooth（10 台）のアクセスポイントから構成される（図 1）．利用者端末は，近傍のアクセスポイントに自動接続し，HTTP でコンテンツを取得する．遠ざかると，自動的に切断して他のアクセスポイントを探索する．この自動探索は，Wi-Fi は Windows の機能で特定の ESSID を探索するように設定し，Bluetooth については，本実験用のソフトウェアを使用し，特定のノード名を持つアクセス

†1 京都高度技術研究所, ASTEM RI / Kyoto

Vol.2011-SLDM-149 No.31
Vol.2011-EMB-20 No.31
2011/3/19

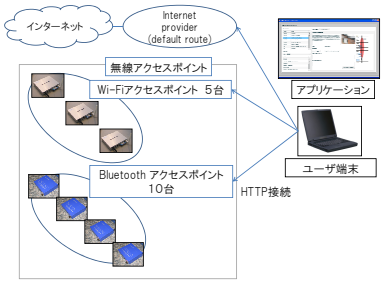

図 1　当初のシステム構成
Fig. 1　Original architecture.

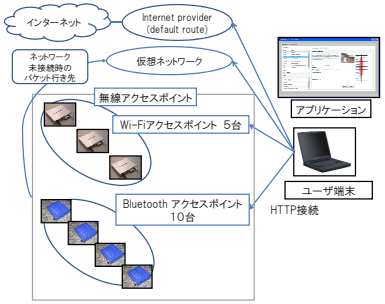

図 2　変更後のシステム構成
Fig. 2　Improved architecture.

ポイントに自動接続し，電波状態が悪くなると自動切断するようにした．ネットワーク構成は，1 回目の実験で使用した構成（図 1）ではアプリケーションや探索ソフトウェアが十分な性能を得られなかったため，経路テーブルやアドレス割り当てなどを見直した構成（図 2）で，再度実験を行っている．

アクセスポイントは，いずれも組み込み Linux を搭載し，以下の設定がなされている．

- 無線アクセスポイント機能（Wi-Fi あるいは Bluetooth PAN プロファイル）
- HTTP サーバ
- コンテンツ（XML および画像（PNG））
- 組み込みデータベースと登録・読み出し用 PHP プログラム（利用者コメント用）

これらの設定は，以下のような手順により組み込まれている．この手順の中では，シェルスクリプトや Makefile を使用し，必要に応じて，機器ごとに異なる設定がされている．

- アクセスポイントごとに固有の ID を決める
- アクセスポイントごとにマスタからコンテンツ一式や設定ファイル類を生成する
- 設定ファイルなどをアクセスポイントのフラッシュメモリに書き込む
- コンテンツなどを CF カードに書き込み，ボードに装着する

一方，利用者用のノート PC には以下の機能が設定されている．アプリケーションはインストーラを使用して組み込み，その他は手作業によっている．

- Wi-Fi アクセスポイントへの自動接続（Windows の機能による）
- Bluetooth アクセスポイントへの自動接続（専用ソフトウェア,PAN プロファイル）
- コンテンツ表示アプリケーション（Adobe AIR, HTTP 使用）

3. 設計手法の検討

3.1 課題と方針

前節で述べた実験では，求められる性能を得るためにネットワーク構成の変更の必要があり，それに伴ってアプリケーションの修正も必要となった．このように，ユビキタス環境でのシステムでは，ネットワーク構成の選択はシステムの性能・安定性に大きく影響する．特に，その上で動作するアプリケーションは，下位レイヤに必要以上に依存してしまうと，ネットワークの構成変更により新しい不具合が導入される可能性が出てくる．

一方，システム構築においては，アクセスポイント，実験用 PC をあわせて 20 台程の機器に様々な設定をしており，調整の過程で何度も設定作業を繰り返すこととなった．ネットワーク構成の変更にあたっても，アクセスポイントや PC に対して，設定変更を行っている．こういったシステムの構築手順は，煩雑になりがちであり，修正があった場合や間違いがあった場合などの手戻りのコストがきわめて大きい．

このような課題に対し，求められるサービスを実現するための開発や展開・調整などを効率化するための設計手法として，以下の 2 点を検討する（図 3）．

- ネットワークとアプリケーションの設計分離
  ネットワーク構成の調整・変更が，アプリケーションの挙動に必要以上に影響しないように，ネットワークとアプリケーションの設計を分離する手法を検討する．例えば，通常，TCP/IP アプリケーションは，ソケット API のみに依存して記述され，ネットワーク形態は問わないように記述される．これと同様に，ネットワーク構成が本質的な役割を果たすシステムにおけるこのような枠組みを検討し，設計の柔軟性を高める
- 構築プロセスも含めたシステム設計
  システムを構成する機器のセットアップ作業の過程も含め，正しさを検証できる設計手法を検討する．システムでは，コンテンツや設定などのファイルを機器ごとに正しく生成・格納することで，アクセスポイントは正しい設定の下必要な機能が起動され，利用者端末はこのアクセスポイントから正しくコンテンツを読み出すことが可能となる．この一連の手順を検証出来るようにし，構築時の誤りを未然に防止する

3.2 ネットワークとアプリケーションの設計分離

アプリケーションがネットワークと対話するのはプロトコルを通じてであり，プロトコルが正しい動作を提供できるかどうかが，システムの安定性にとって重要である．そこで，このプロトコルの挙動とともに，システムを構成する機器のプロトコルスタックなどネット

Vol.2011-SLDM-149 No.31
Vol.2011-EMB-20 No.31
2011/3/19

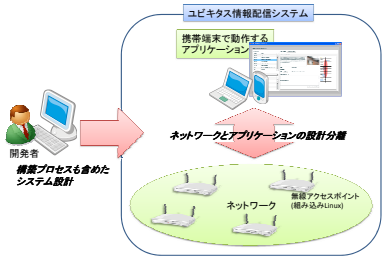

図 3　設計手法の検討方針
Fig. 3　Concept of the designing method.

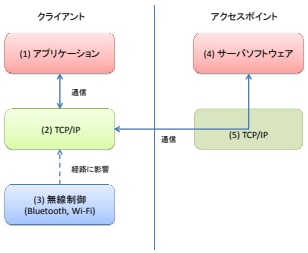

図 4　ネットワークのモデル
Fig. 4　Network model in ubiquitous systems.

ワークレベルの挙動も明示的に記述する手法を検討した．これにより，想定するネットワーク構成の下でプロトコルの挙動が仕様を満たしていることが確認されれば，アプリケーション自身の仕様については，別途このプロトコルを用いて記述すればよいこととなり，ネットワークとアプリケーションの設計分離が期待できる．なお，この記述には，分散システムや通信の記述・検証に用いられる，Promela[3] を使用した．

記述内容の構成図を，図 4 に示す．通常，サーバとアプリケーションの間の一般的な通信をモデルする場合，最上位の（1）アプリケーションと（4）サーバの間の通信を記述すればよい．しかしながら，ここでは，ネットワークの挙動を明示的に表現するため，クライアント側に（2）TCP/IP のレイヤを導入し，アプリケーションは，直接サーバと対話する代わりに TCP/IP のレイヤにデータを渡し，その時点で接続しているアクセスポイントに転送させるように記述する．未接続であれば，TCP/IP レイヤはそのデータは破棄し，アプリケーションにエラーを返す．通常の処理を行う（5）アクセスポイント側の TCP/IP スタックについては記述する必要はないが，必要に応じて明示的に記述することもできる．クライアントのアクセスポイントの接続状態の管理は，実験における Bluetooth 探索アプリケーションに相当する（3）が行う．ここでの状態遷移に従って，(2) TCP/IP は送信先を決定するか，通信を失敗させる．これにより，この種のシステムに特徴的な無線ネットワークレベルでの挙動を加味したプロトコルレベルでの整合性の検証が可能となる．

具体的な記述の抜粋を，図 5 に示す．ここでは， 2 つのアクセスポイント NODE_ID_0, NODE_ID_1 があるとする．クライアントからアクセスポイントへのリクエストは，client_application（左上）で sock を通じて送信されるが，このチャンネルは client_tcpip_stack（右上）との通信である．TCP/IP のスタックは，接続先のネットワークレベルの

```
proctype client_application() {
  do  /* 省略 */
    :: client_state_http == INITIAL
       -> atomic { dst_node = NODE_TYPE_BT;
                   client_state_http = MAIN_REQ;
                   contents_text = NODE_ID_NONE;
                   contents_image = NODE_ID_NONE;}
    :: client_state_http == MAIN_REQ ->
       sock!dst_node,GET_MAIN_CONTENT,NODE_ID_NONE;
       if
         :: sock?resp_node,resp_msg,resp_id ->
            if  /* 一部ケースは省略. */
              :: resp_msg == RESP_MAIN ->
                 atomic {
                   client_state_http = CHECKING_STATUS;
                   contents_text = resp_id;
                   contents_image = NODE_ID_NONE; }
              :: resp_msg == RESP_MAIN_WITH_IMAGES ->
                 atomic { client_state_http = IMAGE_REQ;
                   contents_text = resp_id; }
            fi
       fi
    :: client_state_http == IMAGE_REQ ->
       sock!dst_node,GET_IMAGE_CONTENT,NODE_ID_NONE;
       if
         :: sock?resp_node,resp_msg,resp_id ->
            if /* 一部ケースは省略. */
              :: resp_msg == RESP_IMAGE &&
                 resp_id == contents_text ->
                 atomic { contents_image = resp_id;
                   client_state_http = CHECKING_STATUS; }
            fi
       fi
    :: client_state_http == CHECKING_STATUS ->
      assert(((contents_image!=NODE_ID_NONE) &&
              (contents_text==contents_image))
             || (contents_image==NODE_ID_NONE));
       sock!dst_node,CHECK_STATUS,NODE_ID_NONE ->
       if
         :: sock?resp_node,resp_msg,resp_id ->
            if  /* 一部ケースは省略. */
              :: resp_msg == RESP_CHECK_STATUS &&
                 resp_id != contents_text ->
                 client_state_http = INITIAL
            fi
       fi
    :: else -> assert(false)
  od;
}
```

```
mtype = { NOT_CONNECTED, CONNECTED, UNSTABLE_1}

proctype client_tcpip_stack
  (chan chn; chan svchn1; chan svchn2; mtype nodeId) {
  /* 省略 */
  do
    :: chn?dst,data,remoteNodeId ->
       if
         /* つながっている先にデータを送信する */
         :: atomic { client_state == CONNECTED
                     && connected_node == node1 ->
                        svchn1!data,NODE_ID_NONE; }
         :: atomic { client_state == CONNECTED
                     && connected_node == node2 ->
                     svchn2!data,NODE_ID_NONE; }
         /* 以下，不安定な時は時々失敗する場合など省略 */
         :: else -> chn!dst,SOCK_NO_PEER,NODE_ID_NONE;
       fi
    :: svchn?data,remoteNodeId ->
       chn!nodeId,data,remoteNodeId;
  od
}

proctype client_bluetooth_state () {
  client_state = NOT_CONNECTED;
  connected_node = NODE_ID_NONE;
  do
    :: client_state == NOT_CONNECTED ->
       if
         :: atomic { client_state = CONNECTED;
                     connected_node = NODE_ID_0 }
         :: atomic { client_state = CONNECTED;
                     connected_node = NODE_ID_1 }
         :: skip
       fi
    :: client_state == CONNECTED ->
       if
         :: skip
         :: client_state = UNSTABLE_1
         :: client_state = NOT_CONNECTED
       fi
    :: client_state == UNSTABLE_1 ->
       if
         :: client_state = CONNECTED
         :: skip
         :: client_state = NOT_CONNECTED
       fi
  od
}
```

図 5　Promela によるモデルの記述（抜粋）
Fig. 5　Model description in Promela (excerpt).

IDconnected_node に従って，svchn1 あるいは svchn2 を通じて HTTP サーバへ中継されるように記述される．このリクエストに対するレスポンスも逆の経路で送り返される．送信先のアクセスポイントは，client_bluetooth_state（右下）が切り替えており，電波状態により不安定で必ずしも送信できない場合 UNSTABLE_1 は， client_tcpip_stack では失敗する場合がある．

これにより，使用するプロトコルとネットワークの組み合わせが求められる仕様を実現するかどうか，検証可能となる．例えば，アプリケーションは，テキストコンテンツとそれに

Vol.2011-SLDM-149 No.31
Vol.2011-EMB-20 No.31
2011/3/19

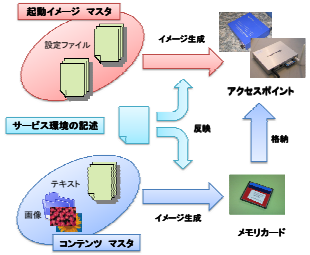

図 6 実験システムの構築手順
Fig. 6 Setup proccess of the system.

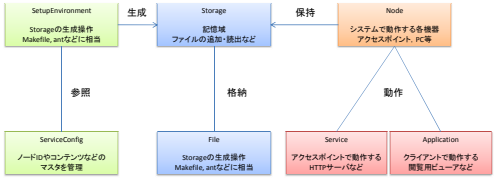

図 7 ストレージを明示的に表現したモデル
Fig. 7 A system model with explicit storages.

```
class SanjoSetupEnvironment
 /* 省略 */
  public setupNode :
   SanjoService`serviceNodeId ==>
                SanjoWirelessServerNode
   setupNode (id) ==
      ( let s = new SanjoHttpServerStorage(id)
        in ( s.setMainFile(new SanjoSpotFile(id));
          return new SanjoWirelessServerNode(id, s)))
      pre id in set SanjoService`serviceNodes;
  public setupNodes :
   () ==> map SanjoService`serviceNodeId
           to SanjoWirelessServerNode
   setupNodes () ==
     return { id |-> setupNode(id) |
              id in set SanjoService`serviceNodes};
  /* 省略 */
end SanjoSetupEnvironment
```

```
class SanjoWirelessServerNode
      is subclass of WirelessTcpIpServerNode
 /* 省略 */
  public SanjoWirelessServerNode :
     SanjoService`serviceNodeId *
          SanjoHttpServerStorage
                ==> SanjoWirelessServerNode
       SanjoWirelessServerNode(id, storage) ==
        ( nodeId := id; httpStorage := storage;
          setupNetwork();  /* ネットワークを設定 */
          httpService :=
          new SanjoHttpNodeService(self, storage);
          return self;)
       pre id in set SanjoService`serviceNodes;
        /* 省略 */
end SanjoWirelessServerNode
```

図 8 VDM++によるストレージを明示的に表現したシステム構築モデル（抜粋）
Fig. 8 Model description of system setup with explicit storages (excerpt).

付随する写真を同じアクセスポイントから続けて取得しなければならない．上記のプロトコルでは，resp_id として，ネットワークレベルの ID とは別に，データを送信してきたサーバ固有の ID を含んでいる．client_application で画像を受け取った際に，この ID をテキストの送信元 ID と比較して，同じであれば接続を維持して画面表示する CHECKING_STATUS に遷移し，異なればコンテンツを何も表示していない初期状態 INITIAL に遷移するようにすればよい．これで上記の条件を満たせることは，アサーションにより，実際に SPIN を用いて検証できる．また，client_application がプロトコルに含まれる ID にのみによって条件分岐等の動作をしている．これは，アプリケーションがネットワークレベルの ID の設計に依存していないということであり，上記のアサーションの条件が満たされている範囲で，下位レイヤのアドレス設計などネットワーク構成を変更しても良く，プロトコルをインターフェースとして，設計分離を図る事が可能となる．

3.3 構築プロセスも含めたシステム設計

前節では，アプリケーション層でのプロトコルをインターフェースとして，アプリケーションとネットワークの設計を分離する手法を検討した．このプロトコルを使用する前提で，本節では，アプリケーションを含めたシステム設計について検討する．記述言語として VDM++[4),5)] を使用する．

前述のように，ユビキタス環境でのシステム構築の特徴として，地理的に分散して配備される機器それぞれについて，サービスの提供に必要なソフトウェアのインストールや設定が個別になされる点がある（図 6）．このような構築作業は煩雑なため，make ユーティリティやシェルスクリプトを用い，設定ファイルやコンテンツなどの生成から，フラッシュメモリや CF カードへの書込まで自動化することが望まれる．そこで，我々は，この手順も含めた検証を行うため，システムが使用するファイルなどを格納するストレージを明示的に扱ったシステムのモデリング・検証手法を検討する（図 7）．ここでは，システムの各機能は，使用するソフトウェアやコンテンツをそれぞれの機器が持っているストレージから読み出すとして記述する（図右）．一方，構築時（図左）は，ファイルのストレージへのコピーや，マスタやテンプレートからのファイルの生成まで含んだ手順を記述する．これにより，構築時に生成・格納されたファイルを使用してシステムが動作する一連の流れが表現され，誤って生成・コピーされたファイルを使用すると，検証をパスしなくなってしまう．以下，実際に VDM++で記述した実験システムのモデルの抜粋（図 8，9）で，この方針を検討する．

クラス SanjoSetupEnvironment（図 8 左）は， Makefile などに相当する構築手順の全体を記述するクラスであり，setupNodes では，全てのアクセスポイントを生成している．この中で，すべてのアクセスポイントのアプリケーションレベルの ID について，setupNode を呼び出し，アクセスポイントを生成する．setupNode は，まず，HTTP サーバが使用するストレージ SanjoHttpServerStorage を生成し，必要なコンテンツを追加する．そして，SanjoWirelessServerNode のコンストラクタを呼び出し，この中では，ホストのネットワーク関係の設定をした上で，引数として渡されたストレージを使って HTTP サーバ（SanjoHttpNodeService）を構築する．

その上で，システムの動作は，アプリケーションからのリクエストに応答して，サーバがストレージからコンテンツを読み出して送信するまでを記述する（図 9）．この際使用されるプロトコル（sock_request や sock_status） は，Promela で使用したものと同等のものを使用し，アプリケーションとネットワークの間の関係づけを行う．

Vol.2011-SLDM-149 No.31
Vol.2011-EMB-20 No.31
2011/3/19

```
class SanjoHttpNodeService
      is subclass of NodeService
     /* 省略 */
instance variables
private storage : Storage;
operations
 public sendTextContents : URL ==> Worker`sock_resp
  sendTextContents(url) ==
    ( let fname = urlToFileName(url)
      in if storage.exist(fname)
       then return mk_Worker`sock_resp
                   (<RESP_OK>,sanjoStr.mainFile)
       else return mk_Worker`sock_resp
                   (<RESP_ERROR_NOT_FOUND>,nil));
     /* 省略 */
end HttpNodeService

class Test
/* 省略 */
/* 環境を生成して，実際にアプリケーションを動作させる */
 public Test : () ==> Test
  Test () ==
    ( env := new SanjoSetupEnvironment();
      sua := new SanjoUserApplication
               (new SanjoWirelessClientNode(env)););
 public getTextContents : () ==>
   map SanjoService`serviceNodeId to SanjoSpotFile
  getTextContents () ==
   return { a.nodeId |-> sua.readContentsByUrl
                     (a,SanjoService`rootFileName)
          | a in set nodesSet() };
end Test
```

```
class Worker
types
 public sock_request = <GET_MAIN_CONTENT>
                     | <GET_IMAGE_CONTENT> | <CHECK_STATUS>;
 public sock_status  = <RESP_OK> | <RESP_NOT_FOUND>
                     | <SOCK_NO_PEER> | <SOCK_TIMEOUT>;
 public sock_resp :: status : sock_status
                     data : [SanjoSpotFile|ImageFile];
     /* 省略 */
operations
 private getRemoteContents :
   sock_request * SanjoService`serviceNodeId
    ==> sock_resp
   getRemoteContents(req, n) ==
    if client.isConnected(n) then   /* 本来はスタックの役割 */
    ( let rn = client.getConnectedNode(),
          url = storage.remoteNodeInfo.getUrlForNode(n)
      in return rn.getService().sendTextContents(url))
     else return mk_sock_resp(<SOCK_NO_PEER>, nil)
 thread
  ( while true do
    for all n in
        set storage.remoteNodeInfo.nodeIds(remoteType) do
    ( let v = getRemoteContents(<GET_MAIN_CONTENT>, n)
      in if v.status = <RESP_OK>
         then owner.setContent(remoteType, v.data)))
end Worker
```

図 9　ストレージを明示的に利用するシステムのモデル（抜粋）
Fig. 9　Model description with explicit storages (excerpt).

ユーザアプリケーション中で動作する Worker（右）は，アクセスポイントにポーリングを行い，接続できたところからコンテンツをダウンロードするスレッドである．呼び出される getRemoteContents は，接続できたアクセスポイントの HTTP サーバ SanjoHttpNodeService（左上）に対し，HTTP リクエストに相当する sendTextContents の呼び出しを行う．sendTextContents は，ストレージにアクセスして，ファイルが存在すれば HTTP レスポンスとして，コンテンツを返す処理を行う．

このような記述の下で，セットアップしたストレージを使って，コンテンツがアプリケーションから読み出せることを確認できる．図左下の Test では，SanjoSetupEnvironment によりストレージを含むシステム全体を構築した上で，全てのアクセスポイントからコンテンツを読み出すテスト getTextContents を記述している．この操作を実行すると，アプリケーションからサーバの sendTextContents が呼び出され，返ってきたテキストコンテンツを写像型に格納する．また，実際に Worker を使ってコンテンツを読み出すことで，クライアントに格納されているアクセス先 URL（storage.remoteNodeInfo）を使用して動作の確認も可能である．これにより，セットアップすれば，そのストレージから正しくコンテンツを読み出せることが確認できる．画像の読み出しの確認もできるが，スペースの都合で割愛する．

ここまでが確認できると，より詳細なアプリケーション側の仕様の記述も可能となる．例えば，Worker では，owner.setContent を呼び出してアプリケーションの内部状態を変更している． Bluetooth，Wi-Fi それぞれに Worker は起動されているため，同じ変数を同時に更新しないように setContent に排他処理を設定することなどが出来る．なお，VDM++ において invariant などを記述する際には，通常のシステム実行時における invariant は，構築時には成り立たない場合がある．このような状態については，実行時にのみ成り立つことを明示した条件式としなければならない．

4. 関 連 研 究

ユビキタスコンピューティング[6] におけるソフトウェア開発・システム開発に対しては，様々なデバイスへの開発ツールの対応，ユーザのコンテクストの多様性・変化への対応，また，システムを構成する要素の組み合わせなど，通常のソフトウェア開発とは異なる課題が指摘されている[7),8]．このような課題に対し，位置や利用者のコンテクストを表現可能なラピッドプロトタイピング[9] やシミュレーション[10] により，シナリオの検証を行うようなツールの研究など，システムが提供するサービスを検証するための研究がなされている．

これらの研究では，ユビキタスコンピューティングにおけるサービスそのものの開発手法に焦点をあてているが，組み込みシステム開発においても，多様な外部環境との対話やハードウェアの取り扱いについては課題であり，それに対し，プロダクトライン開発の観点から形式手法を適用する研究[11] などがなされている．本研究では，より実装に近い側面の課題を扱っているが，ユビキタスシステムが持つ上記のような特徴は，デバイスの展開まで含めた実装に反映されており，多様なデバイスの取り扱いや，ネットワーク環境の変化への対応といった，類似する課題を解決しようとするものである．

情報システムの設計という観点からは，形式的手法は重要なツールとなりつつあり，ビジネスプロセスのモデルの正当性検証など，多数の研究がなされている[12),13]．本研究では，上述のように実装時の課題解決を主眼としているためこのような上流設計を対象とはしていない．

5. ま と め

ユビキタス環境におけるシステムでは，しばしば，システムの目的に合わせてネットワーク構成が選択され，その上で動作するアプリケーションが開発される．システム開発や調整

Vol.2011-SLDM-149 No.31
Vol.2011-EMB-20 No.31
2011/3/19

の過程では，ネットワーク構成の調整・変更もなされることがあるが，アプリケーションの動作もこれに影響されてしまう可能性がある．また，このような開発・調整作業では，煩雑な作業の中で多数の組み込み機器を正しくセットアップする必要に迫られる．本稿では，このような，ユビキタス環境における情報配信システムに固有の課題に着目した設計・開発への形式手法の適用について，著者らが構築した実験システムに基づき，検討した．

ネットワーク構成とアプリケーション設計に関しては，アプリケーションのプロトコルをインターフェースとし，プロトコルがサービスに必要な情報を提供できる設計とした上で，分離設計を行う手法を検討した．ネットワークレベルの挙動については，アプリケーションのプロトコルに加えて，TCP/IP や Bluetooth のプロトコルスタックの挙動も含めたモデリングを行っい，求められる仕様の検証方法を検討した．アプリケーションについては，ネットワークレベルの詳細な設計・挙動を前提とせず，プロトコルに依ったモデリングを行うことで，ネットワークとアプリケーションの設計分離を図った．また，機器が持つストレージをモデルの中に組み込み，ストレージに書き込まれたファイルが正しく実行時に読み出せる事を検証する事で，上記のような煩雑な作業の検証を行えるようにした．

現時点では，ブートプロセスのようなセットアップとシステム稼働の間に行われる作業や，システムのアップグレードやノードの追加などといったサービスの障害や陳腐化を防ぐ作業はモデルに含まれていない．このような作業も，ユビキタス環境でのシステムにおいては必須の要素であり，モデリングの仕組みの中に取り込んでいくべきであろう．

また，VDM++で記述する際，どのメソッド呼び出しが Makefile のどの操作に対応しているかなどといった規約化はなされていない．システム構築は単一の言語を使用してなされるのではなく，様々なスクリプトなどを組み合わせて行うのが一般的であり，モデルに基づいた実装を確実に行うためには，何らかの規約化が必要と考えられる．これにより，さらには，モデルから構築用スクリプトなどを自動生成するツールなども考えられるだろう．

謝辞　本研究を進めるにあたり多くの貴重な指導・助言を頂いた南山大学情報理工学部ソフトウェア工学科 沢田篤史教授，張漢明准教授，京都高度技術研究所 神原弘之氏に感謝する．総務省近畿総合通信局「観光ナビゲーションのためのユビキタスネットワークの構築に関する調査検討会」（座長 中村行宏京都大学名誉教授，2008 年）においては，本研究の端緒となる貴重な実験の機会を頂いた．この実験にあたり，多くの助言を頂いた座長をはじめとする委員および関係の各位，システム構築にあたって協力いただいた京都高度技術研究所 槌田義之氏，池上周作氏に感謝する．

参 考 文 献

1) 吉田信明，池上周作，槌田義之：無線を活用した街歩き観光ナビゲーションシステムの開発，組み込みシステムシンポジウム 2009(ESS2009)，情報処理学会組み込みシステム研究会 (2009).
2) 総務省近畿総合通信局：観光ナビゲーションのためのユビキタスネットワークの構築に関する調査検討報告書 (2009 年).
3) Holzmann, G. J.: *The SPIN Model Checker: Primer and Reference Manual*, Addison-Wesley Professional (2004).
4) The VDM Tool Group: The IFAD VDM++ Language, Technical report, CSK Systems (2008).
5) CSK: VDMTools homepage, *http://www.vdmtools.jp/* (2007).
6) Weiser, M.: The Computer for the Twenty-First Century, *Scientific American*, Vol.265, No.3, pp.94–104 (1991).
7) Abowd, G.D.: Software engineering issues for ubiquitous computing, *Proceedings of the 21st international conference on Software engineering*, ICSE '99, New York, NY, USA, ACM, pp.75–84 (1999).
8) Abowd, G. D. and Mynatt, E. D.: Charting past, present, and future research in ubiquitous computing, *ACM Trans. Comput.-Hum. Interact.*, Vol. 7, pp. 29–58 (2000).
9) Li, Y., Hong, J. I. and Landay, J. A.: Topiary: a tool for prototyping location-enhanced applications, *Proceedings of the 17th annual ACM symposium on User interface software and technology*, UIST '04, New York, NY, USA, ACM, pp.217–226 (2004).
10) O'Neill, E., Lewis, D. and Conlan, O.: A simulation-based approach to highly iterative prototyping of ubiquitous computing systems, *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, Simutools '09, ICST, Brussels, Belgium, Belgium, ICST, pp.56:1–56:10 (2009).
11) 鵜林尚靖，金川太俊，瀬戸敏喜，中島　震，平山雅之：コンテキストベース・プロダクトライン開発と VDM++の適用 (ソフトウェアプロダクトライン開発, 特集:ソフトウェア工学の理論と実践)，情報処理学会論文誌，Vol.48, No.8, pp.2492–2507 (2007-08-15).
12) Mili, H., Tremblay, G., Jaoude, G. B., Lefebvre, E., Elabed, L. and Boussaidi, G. E.: Business process modeling languages: Sorting through the alphabet soup, *ACM Comput. Surv.*, Vol.43, pp.4:1–4:56 (2010).
13) 飯島淳一：情報システム開発における形式的手法 (特集:情報システム設計論)，日本情報経営学会誌，Vol.28, No.2, pp.16–24 (2007-12-20).