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

2006－UBI－12（6）
2006／11／9

センサイベント指向のサービス連携ミドルウェア：SENSORD

幸島明男 †,†† 池田剛 †,††
井上豊 †,†† 車谷浩一 †,††

さまざまなセンサデバイスからのセンサ情報をどのようにして管理し，人間向けの具体的なサービスへと結びつけるかという問題は，依然として，状況に基づくサービスを実現する上で重要な課題となっている．本研究では，このような問題の解決に向けて，複数の異なるセンサ情報を統一的に管理し，高次のサービスと連携させる機能を提供するミドルウェア SENSORD (Sensor-Event-Driven Service Coordination Middleware) を提案する．SENSORD は，低次元のセンサデータを高速に解析するために，センサデータをその時空間情報とともに共有メモリ上に保持する機能を持つ．SENSORD は，ユーザが登録したルールに基づいて，非同期同期的に蓄積されたセンサデータの評価を行い，ルールが適合する場合には対応するサービスを起動する．これらの機能を利用することで，ユーザは状況に基づくサービスを容易に実現することができる．本論文では，まず，状況に基づくサービスのためのミドルウェアに求められる機能について整理し，時空間情報管理とルールに基づくサービスに着目する．そして，そのような機能を提供するミドルウェアとして SENSORD を提案する．最後に，SENSORD のアプリケーションとして，現在開発中の緊急時における屋内避難誘導システムについて述べる．

SENSORD: Sensor-Event Driven Service Coordination Middleware for Ubiquitous Computing

Akio Sashima ,†,†† Takeshi Ikeda ,†,†† Yutaka Inoue †,††
and Koichi Kurumatani†,††

How various kinds of sensor devices are handled, and how numerous lower-level sensor data are managed and integrated into higher-level context services are important issues to realize context-aware information systems. We have been developing Sensor-Event-Driven Service Coordination Middleware (SENSORD) to fill coordination gaps betweepn higher-level services and lower-level sensors. The SENSORD system obtains and stores sensor data into an in-memory data container to achieve fast, complex analysis of the sensed data. The facility of real-time analysis, which includes periodical evaluations of spatio-temporal conditions, enables application developers to outsource context-aware mechanisms. As described in this paper, we first examine middleware of context-aware services and our approaches. We then show the outline of SENSORD and an exemplary application: an indoor emergency response system in our laboratories. Preparing for emergency situations, such as fire emergencies, it manages sensor devices (thermometers, hygrometers, vision systems, microphone arrays, etc.) to detect emergent situations.

1. はじめに

ユビキタスコンピューティング研究における中心的研究課題の一つに，ユーザの置かれた状況 (Context) の把握とその状況に応じたサービスの提供 (Context-Aware Services) がある．従来，こうした状況把握の問題に関しては，ユーザの位置情報を利用するものが多く，単一の位置測位デバイスを前提とした研究[1)2)11)]が多かった．しかしながら，近年では，無線センサネットワーク技術への関心の高まりと相まって，単一のセンサ情報だけからユーザの状況を把握するのではなく，環境中に存在する複数のセンサから得られる情報を組み合わせて，ユーザの状況を把握しようとする研究が増加の傾向を見せている．たとえば，Gellersen らは，ユーザの利用する日用品（artefact）にセンサを取り付け，そこから得られる複数のセンサ情報に基づく状況依存サービス[5)]を構築している．また，Wilson らは，ON/OFF だけを出力するような単純なセンサを複数用いて，ユーザの位置と行動推定を行う手法[15)]

† 独立行政法人 産業技術総合研究所 情報技術研究部門
ITRI, National Institute of Advanced Industrial Science and Technology
†† 独立行政法人 科学技術振興機構 CREST
CREST, Japan Science and Technology Agency

を提案し，Hiramatsu らは複数のセンサの変動とその共起関係から状況を定義する手法[6] を提案している．環境中に埋め込まれたセンサ以外では，異なるモダリティの複数のセンサを搭載した携帯型センサボードを用いて，ユーザの行動を推定する手法[12] なども提案されている．

しかしながら，複数のセンサを用いたこれまでの研究は，センサデバイスの開発やセンサデータの解析に重きをおいたものであり，異なる複数のセンサデバイスからのセンサデータをどのようにして管理するか，また，複数のセンサデータを同時・並列的に解析しながら，現在の状況を決定し，適切なサービスを起動するしくみをどのようにして実現するかといった，システムソフトウェア的観点からの検討は十分になされていない．

そこで，本研究では，多数のセンサ情報に基づく状況把握とサービス連携の問題に対して，ミドルウェア技術の観点からアプローチする．そして，低レベルのセンサイベント情報の解析と高次のサービス連携を実現するためのミドルウェア SENSORD (Sensor-Event-Driven Service Coordination Middleware) を提案する．SENSORD は，複数のセンサデータを高速に解析し，処理するために，取得したセンサデータを共有メモリ上に保持する機能を持つ．また，SENSORD に蓄積されたセンサデータに基づいて，アプリケーション開発者が記述したサービスルールを評価し，ルールが適合する場合には対応するサービスを起動する機能を提供する．アプリケーション開発者は，このサービスルールを，SENSORDScript と呼ぶルール記述と解釈のためのフレームワークに基づいて記述することで，状況に応じたサービスを提供するアプリケーションを容易に構築することができる．

以下では，まず，複数のセンサ情報を用いた状況に応じたサービスのためのミドルウェアに求められる機能について整理する．そして，そのような機能を提供するミドルウェアである SENSORD の概要について述べる．最後に，SENSORD のアプリケーションとして，現在開発中の緊急時における屋内避難誘導システムについて述べる．

2. 状況に応じたサービスのためのセンサミドルウェアに向けて

一般にミドルウェアとは，ハードウェアに密着した基盤ソフトウェアとユーザ向けのアプリケーションソフトウェアとの間を仲介するソフトウェアを指す．プラットフォームやフレームワークと呼ばれるソフトウェアが，異なる OS やデバイス上において共通の API を実現することを重視するのに対して，ミドルウェアでは，そのアプリケーションドメインにおいて共通化可能な機能を抽出して提供する．たとえば，データベースマネージメントシステムでは，データの管理という共通化可能な機能を抽出したミドルウェアである．

本研究では，複数のセンサ情報を前提とした状況に応じたサービスを実現する上で，共有化可能な機能を提供するミドルウェアの実現をめざす．具体的には，以下の３つの機能を提供するミドルウェアを目指した．

2.1 センサ情報の多様性の吸収

複数のセンサからの情報を統合的に扱うアプリケーションサービスの開発を考慮した場合，センサデータの多様性の吸収は重要な問題となる．ビデオ監視システムのように大量のビデオデータを常時送信し続けるものから，火災報知用の温度センサのように火災発生時にだけに警報を送信するものまで，さまざまな通信形態が存在する．また，そのデータ表現も単純なテキスト表現によるセンサデータからストリーミングによる大量のバイナリデータまで，さまざまなケースが考えられる．したがって，このような多様性，特にデータの通信方法とデータ表現の多様性の吸収は，状況に応じたサービスのためのミドルウェアの備えるべき重要な機能であると考えられる．

2.2 センサデータの時空間情報管理機構

複数のセンサデータを解析することで状況を把握する場合，複数のセンサデバイスからのデータの変化の時間的・空間的相関は，非常に重要な情報となる．すなわち，そのセンサデバイスの置かれた位置とそのセンサデータが得られた時刻の記録が不可欠となる．状況に基づくサービスアプリケーションのためのミドルウェアは，単なるセンサデータの管理機能だけではなく，任意のセンサデータに対して，その得られた時刻と空間的位置をキーとしてアクセスできるような管理機能が必要となる．本研究では，このようなセンサデータの管理機構をセンサデータの時空間情報管理機構と呼ぶ．

2.3 実時間センサデータ解析とサービス提供

状況に基づくサービスのためのミドルウェアは，単にセンサデータを蓄積するだけではなく，複数のセンサデータの変化の共起関係を解析して，現在の状況を把握し，実時間でなんらかのサービスを起動する機能が必要となる．すなわち，新しいセンサイベントを検知した際には，その時空間的関係を高速に計算し，即座に反応する仕組みが必要となる．例えば，災害に対する緊急避難システムでは，環境のセンサ情報に基づいて高速に情報を処理し，被災者に対して実時間で情報を提示することが被害の減少につながる．

こうした実時間解析とサービス起動のためのフレームワークをミドルウェアの機能として提供する点が，本研究で目指すミドルウェアと，従来のセンサデータベースとの最大の相違点である．

3. Sensor-Event-Driven Service Coordination Middleware

前節で述べた機能の実現を目指して現在開発中のセンサデータ管理・解析・サービス統合ミドルウェアが SENSORD (Sensor-Event-Driven Service Coordination Middleware) である．SENSORD は，さまざまなセンサデバイスからのセンサデータとその解析結果を統合的に管理し，ユーザ向けのサービスへと結びつける．SENSORD は，アプリケーションサービスが，必要とするセンサデータにアクセスするための時空間情報検索・格納・解析 API を提供する．加えて，API へのアクセスやサービスの起動のロジックを記述するためのフレームワークとして SENSORDScript と呼ぶフレームワークを提供する．SENSORDScript を用いることで，センサデータに応じたサービスの起動，停止，終了を制御することができる．SENSORD の実装は， Java (JDK 5.0) を用いて行っている．

3.1 SENSORD の概要

SENSORD は，アプリケーションシステムに機能を提供するミドルウェアである．したがって，センサモジュールや他のプログラムモジュールやサービスなどと連携して動作する．SENSORD の典型的アプリケーションは，図 1 に示すように，センサデバイス，センササーバ，SENSORD 本体，SENSORDScript からなる．アプリケーション開発者は，これらのモジュールを必要に応じて組み合わせて，動作させることで，異なるアプリケーションを実現することができる．

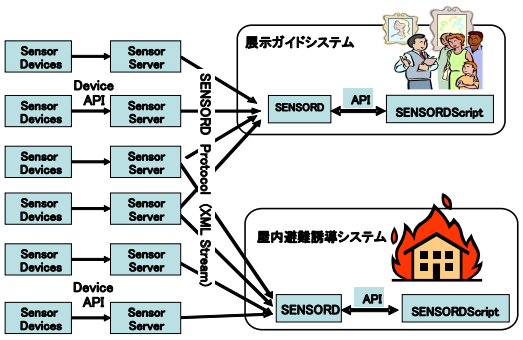

図 1 SENSORD: Sensor-Event-Driven Service Coordination Middleware の概要

3.1.1 センサデバイス

まず，SENSORD のアプリケーションの開発において必要不可欠なものとして，ユーザの挙動や状況，環境情報を把握するためのセンサデバイスを挙げることができる．ここでセンサデバイスは他のシステムと通信するための通信インタフェイスを持つものとする．現在，SENSORD が対応を想定しているセンサデバイスとしては，温度，湿度，加速度，地磁気，気圧，ビジョン，マイクロフォンアレイなどがある．

表 1 USV カメラ (第 4 章参照) のデータ通信モードのストリームセンサデータ

```
<?xml version="1.0" encoding="UTF-8"?>
<usv-tracking-data>
<header>
 …
</header>
<body>
<frame no= ”frame-no ”>
  <line id= ”line-id ”> x y z </line>
     …  % エリア内に存在する人の数だけ繰り返し
</frame>
   …   % 人を検知するごとに<frame …/>の送信を繰り返す
</body>
</usv-tracking-data>   % 通信切断
```

3.1.2 センササーバ

センササーバはセンサデバイスへアクセスし，センサデータを SENSORD 本体へと送信する機能を実装したサーバを指す．具体的には，SENSORD と接続するための独自通信プロトコルである SENSORD Protocol を実装したサーバシステムを意味する．センササーバの実装はアプリケーション開発者もしくはデバイス開発者が必要に応じて行うものとする．

SENSORD Protocol は XML を用いた TCP/IP 上の非同期通信プロトコルであり，センサデータの表現と通信インタフェイスの多様性を吸収するために開発されている．SENSORD Protocol は，センササーバの仕様などを通信する属性通信モードとセンサデータを XML ストリームで送信し続けるデータ通信モードに分かれている．データ通信モードで送信 XML データはストリーム全体で一つの妥当 (Valid) な XML 文書 (表 1 参照) となる．センササーバは，SENSORD からの送信されるコマンドに応じて，センサデータを送信開始・停止．中断・再開などを行う．一つのセンササーバは複数の TCP/IP 通信チャネルをオープンすることが可能であり，複数の SENSORD に対してデータを配信することができる．

3.1.3 SENSORD

SENSORD は，センササーバから受信したセンサデータを SENSORD のデータコンテナに蓄積する．データコンテナは，メモリ上に確保されたデータバッファと通常のデータベースシステムの組み合わせによって実現されており，取得したセンサデータは，まずメモリ上のデータバッファに書き込む．速度の遅いハードディスクやデータベースシステムへのアクセスをできるだけ避けることで，高速なデータ解析処理を実現する．一方で，データの永続性を考慮してデータバッファの内容は，定期的にデータベースシステムへコピーされる．

表 2 SENSORD の時空間 API の一部

```
/* 利用可能なセンサのリスト */
Collection getSensorListInGeometry(Geometry geom);
/* 時間検索 */
Collection getSensorData(String uri, long startTime,
                                    long endTime);
 /* 時間検索＋ソート */
Collection getTimeOrderdSensorData(String uri,
                      long startTime, long endTime);
 /* 空間的範囲に含まれるオブジェクト */
Collection getContainObjects(Geometry geom) ;
/* 空間的距離範囲に含まれるオブジェクト */
Collection getCentroidDistanceObjects(Geometry geom,
                                    long distance);
/* センサデータを記録 */
void putSensorData(ArrayList<SensorData> data,
             String sensorURI, long sensedTime);
```

SENSORD は，状況に応じたサービスを実現する上で必要となるさまざまな API を提供している．時空間 API はセンサデータを時間，空間をキーとして格納・検索するための API である (表 2 参照)．例えば,「私の周囲 5 メートルに存在するすべてのセンサデバイスに関して，過去 10 分間にわたるセンサデータを検索せよ」というような検索は，時空間 API を用いて実現できる． SENSORD は，センサデータに限らず，html ドキュメントのようなコンテンツの URL も格納することができるので，過去のある時点のある場所で撮影した写真の検索というようなことも可能である．

3.1.4 SENSORDScript

SENSORDScript は，アプリケーション開発者が SENSORD の API にアクセスするためのロジック (定型的処理や繰り返し，条件実行など) を容易に記述できるようにするためのフレームワークである．具体的には，次のような機能を提供する．

サービスのテンプレート定義 RMI, Web Service, や別の SENSORDScript の呼び出しインタフェイスのテンプレート

ルールに基づくサービスの実行制御 ルール記述のフォーマットを定義し，実行条件とサービスとのマッピングを定義する．SENSORD はサービスの実行と終了をルールに基づいて制御する．条件は最新のセンサデータおよび検索された過去のセンサデータの解釈に基づいて真偽が決定されるものであり，論理的関係を記述できる．

実装上は，SENSORDScript は，Java の抽象クラスとインタフェイスの集まりであり，開発者はその実装クラスを実現することで，ロジックを記述することになる．現状では，以下のような抽象クラスを提供している．

AsyncResponder SENSORD の API に対する Query の実行と実行結果の受信を，非同期通信で実現する機能 (Request/Reply) を提供する抽象クラス．

CyclicResponder SENSORD の API に対して，Query を繰り返し実行しその結果，非同期通信で繰り返し送信する機能を提供する抽象クラス．実行 Query の依頼は最初に一度だけ行う．

状況に応じたサービスを実現する場合，多くの開発者はセンサデータがある条件を満たしているか解析するルーチンを記述し，その状況が成立している場合に適切なサービスを起動する機能をアプリケーションプログラムの一部として実装する．これは，センサデータとサービスに関してルール解釈と実行系がアプリケーションプログラムに埋め込まれていることを意味する．しかしながら，このような機能は前章で述べたようにミドルウェアで提供すべき共通の機能であると考えられる．アプリケーション開発者は，このルール解釈と実行の部分に関しても，SENSORDScript の実装クラスを実現するだけで，これまで必要だったルールの解釈や実行等に関する詳細な実装を省略することができる．

たとえば,「ある人がある絵画の 3m 以内に来たらある情報を配信せよ」というようなルールを開発者が記述した場合， SENSORD は「ある人」と「ある絵画」の時空間的関係について 3m 以内という条件が満たされているか定期的に解釈する．そして，条件が満たされた場合には，配信サービスを起動して情報を配信する．ユーザは配信する情報と配信の条件を SENSORDScript のフレームワークで記述（実装クラスの定義）し，SENSORD に登録するだけで良い．具体的には，以下のような抽象クラスを提供している．

Condition ルールの条件部を記述するための抽象クラス．真偽値を判定する抽象メソッド `isTrue()` 持つ．サブ抽象クラス `And, Or , Not` などを用いて論理的関係を記述する．

Action ルールのアクション部を記述するための抽象クラス．行為を起動するための抽象メソッド `doAction()` を持つ．

Rule ルールを記述するための抽象クラス．`Condition` クラスと `Action` クラスをプロパティとして持つ．ルールの評価のための抽象メソッドとして `evaluate()` を持つ．

WatchDogResponder SENSORD のセンサデータ受信プロセスにフックし，センサデータをルールに基づいて監視する処理を行う抽象クラス．クライアントプロセスから渡されたルール `Rule` クラスに基づいて，センサデータを繰り返し評価 (`evaluate()` の実行) し，ルールにマッチする場合には，その `Action` クラスで定義された処理（SENSORD の API の実行や別プログラムの起動など）を実行する．実行結果は，クライアントプロセスに対して，非同期通信で送信される．ルー

ルの登録は依頼は最初に一度だけ行う．

加えて，Condition のサブクラスには以下のような抽象クラスも用意している．

StaticCondition　真偽値が一度決定するとそれに反するセンサデータが得られるまで真偽値を維持し続けるような条件のための抽象クラス

TemporalCondition　は，真偽値をあらかじめ定められた期間だけ維持し，その後はデフォルト値に戻るような条件のための抽象クラス．これは二つのセンサイベントがある時間内で発生したときだけ真になるようなルールを記述するために用いる．

PersistentCondition　は，真偽値が一度決定するとそれ以降は変わらないような条件のための抽象クラス．

状況に応じたサービスを実現するアプリケーションを開発・実行する場合には，開発者は以下のような手順で行う．

(1)　状況に応じたサービスを実現するためのルールを SENSORDScript のフレームワークを用いて実装する．
(2)　実装したルールクラスを登録する
(3)　必要に応じてセンササーバを実装し，ルールの評価のために不可欠なセンサデータが SENSORD に蓄積されるように設定する．
(4)　SENSORD を起動し，センサデータの収録とルールの評価を開始する．

3.2　アーキテクチャ

図 2 は SENSORD のアーキテクチャを示したものである．SENSORD は，主として**センサデータレコーダ**，**データコンテナ**，**サービスマネージャ**，**ルールマネージャ**という 4 つモジュールから構成される．

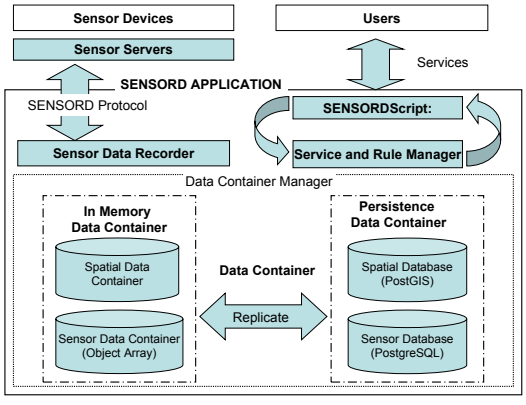

図 2　SENSORD アーキテクチャ

3.2.1　センサデータレコーダ

センサデータレコーダは，センササーバと通信し，センサデータを収録する処理を行う．センサーデータレコーダはマルチスレッドで動作し，複数のセンサーサーバから同時にデータの収集を行う．センサデータレコーダは，センササーバとの間で TCP/IP 通信チャネルを開き，送信されてくる XML データのストリームをオンラインで Java オブジェクトへと変換する処理（XML/Object バインディング）を行う．ここでは，XML Pull Parser ☆と呼ばれるストリーミングプルパーザーを用いて，バインディング処理を行う．XML Pull Parser を用いることで，XML の記述の柔軟性と再設定の容易さを保ちながら，効率的なセンサデータの受信が実現できている．

3.2.2　データコンテナ

データコンテナはセンサデータとその時空間情報を格納するデータベースであり，格納されたセンサデータへアクセスするための時空間 API を提供する．SENSORD のデータコンテナは性質の異なる 4 つのデータコンテナから構成されている．すなわち，インメモリセンサデータコンテナ, インメモリ空間データコンテナ, 永続センサデータコンテナ, 永続空間データコンテナの 4 種類である．

インメモリセンサデータコンテナはセンサデータレコーダから送信されるセンサデータのオブジェクトを格納する．インメモリセンサデータコンテナは各センサデバイスごとに用意され，担当するセンサデバイスからのデータを受信すると，取得した時間順にデータを格納する．

コンテナの構造は配列の先頭と末尾が環状につながったリングバッファとなっており，一定の時間周期ごとに同じ配列上の位置にセンサデータが書き込まれるような構造となっている．リングバッファは複数のブロックに分かれており，現在時刻より一周期前のブロックのデータを削除することで，周期の異なるデータが混在しないように構成されている．リングバッファは，可変長配列 m_i を要素として持つ配列によって実装されており，センサデータは受信時間 $sensedTime$ から計算されるインデックス $index$ が指す位置の可変長配列 m_{index} に格納される．インデックス $index$ は以下の式で計算される．

$$index = (t \bmod capacity)/thick$$

ここで，t はセンサデータの取得時刻（1970 年 1 月 1 日からの経過時間 (msec)），$capacity$ はリングバッファにデータが保持される期間 (msec) であり，デフォルト値は 30 分である．$thick$ はバッファの時間粒度であり，デフォルト値は 100 msec である．100msec 以内に複数のセンサデータが到着した場合は，同じインデックス k を持つ可変長配列 m_k の最後尾に格納される．時刻に対応するインデックスの位置が一意に定まるため，高速な時間検索が実現できる．

☆ http://www.extreme.indiana.edu/xgws/xsoap/xpp/mxp1/index.html

インメモリ空間データコンテナは空間的データ，すなわち人，センサデバイス，地物（部屋，建物や区画）などの空間的オブジェクトをその形状 (Polygon, Line, Point) および空間的位置をキーとしてメモリ上に展開して保持する．空間的オブジェクトの位置座標は，屋内外にかかわらず，緯度／経度で表現されており，GPS などで使われる標準の緯度／経度座標系である World Geodetic System 1984 (WGS 84) を用いて記述している．インメモリ空間データコンテナは，レイヤー構造を持っており，センサデバイスレイヤー，建物レイヤーなどの複数のレイヤーに分割して異なる種類の空間的オブジェクトを保持する．これにより，特定の空間的オブジェクトへ高速なアクセスが実現できる．アプリケーションが処理中（GUI 上に表示中など）の空間的領域に含まれる空間的オブジェクトを，下記の永続空間データコンテナから検索し，一時的に保持するメモリ上のバッファとして動作する．包含関係や距離などのオブジェクトの空間的関係は，レイヤー構造を考慮した独自の空間推論エンジンを用いて行っている．本空間推論エンジンは国土地理院発行の数値地図 2500（空間データ基盤）☆1の読み込み機能をもっており，数値地図上の地理データと屋内の空間オブジェクトのデータとをシームレスに扱うことができる．

永続センサデータコンテナと**永続空間データコンテナ**は，大量のデータを長期的に保持するために，インメモリデータコンテナに含まれる新規／更新データを定期的にコピーしたものである．永続データコンテナの実装は，オープンソースのリレーショナルデータベース PostgreSQL ☆2を用いて実現されており， 空間的オブジェクトの検索は，PostgreSQL に付属する地理オブジェクト処理用のプラグインである PostGIS ☆3を用いて実現している．

3.2.3 サービスマネージャとルールマネージャ

サービスマネージャは，SENSORDScript のライフサイクル管理（登録，起動，停止，登録解除）を行うモジュールである．サービスマネージャは，XML で記述された設定ファイルに基づいて SENSORDScript の初期化を行う．図 3 は，サービスマネージャの SENSORDScript 管理の概要を示している．SENSORDScript は，アプリケーション開発者によってプログラムされた時空間検索・格納クエリーを実行し，結果をアプリケーション側に送信する過程が記述されており，サービスマネージャによって起動・停止される．

SENSORDScript を用いてて何らかの条件が満たされた時に動作する機能を実現する場合は，前章で述べた `Rule` クラスを実装し，`WatchDogResponder` の仕組みを用いてルールマネージャーにルールを登録することによって行う．ルールマネージャは，非同期（センサデータが更新された時）もしくは一定の時間間隔で，繰り返しルールの評価を行い，条件が満たされた場合にはルールのアクション部を実行する．

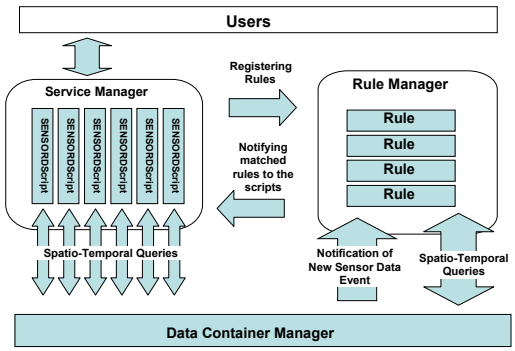

図 3 SENSORDScript の管理

4. 緊急時における屋内避難誘導システム

現在，我々は，SENSORD のアプリケーションとして，緊急時における屋内避難誘導システムの開発[8)20)]を行っている．このシステムは，屋内や地下街などの閉空間において，火災などの緊急時における情報配信や避難誘導を行うことを目的としたものであり，我々の研究室（秋葉原ダイビル 10F）のフロアに，さまざまなセンサデバイス（温度湿度センサ，ビデオ監視システム， マイクロフォンアレイなど）を実際に設置し，火災などの異常の発生を検知するためのシステムを開発中である．

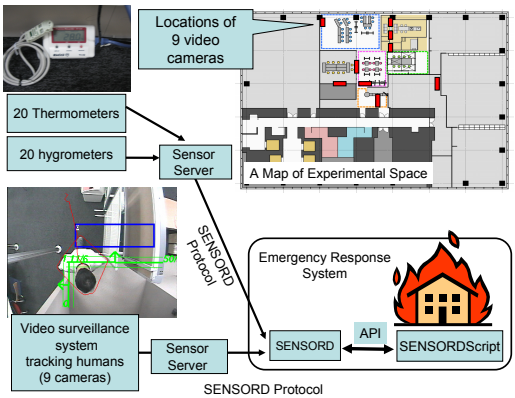

図 4 緊急時における屋内避難誘導システムの概要

図 4 は我々が開発中の屋内避難誘導システムの概要を示している． $700m^2$ の実験スペースに 20 台の温度・湿度センサ☆4，9 台のビデオ監視システムを設

☆1 http://www.gsi.go.jp/MAP/CD-ROM/2500/t2500.htm
☆2 http://www.postgresql.org/
☆3 http://www.postgis.org/
☆4 TR-72W; T&D Corp. http://english.tandd.com/

置している．監視用のビデオカメラは部屋の出入り口付近に設置されており，監視エリア内に設定された仮想的なボーダラインを通過した人の数をカウントし，SENSORD に通知するシステム☆1となっている．本システムでは，3 種類の異なるビデオ監視システム: Vitracom SiteView EP☆2, Ubiquitous Stereo Vision System (USV)☆3, IBS Counter ☆4を用いている．

SENSORD システムは，上記のようなセンサと通信し，センサデータ（温度・湿度，監視エリア内の通行人の検知結果など）を受信する．受信したセンサデータを解析することにより，ある部屋に現在在室中の人の数のような環境のモニタリングをリアルタイムで行う．図 5 はシステムのモニタリング画面のスナップショットである．図 5 の 2 次元地図は実験スペースの概観を示しており，各部屋の中央の数値はその部屋に在室中の人の数を示している．たとえば，右端の部屋には 3 名，右上の部屋には 2 名在室中であることが示している．在室者数のカウントと表示は，部屋の出入り口のセンサデータの変化に応じて人数を増減する処理を記述した `Rule` クラスを実装し，`WatchDogResponder` の仕組みを用いて GUI プログラムに実行結果を通知することで実現した．下部のグラフはある温度湿度センサから得られた温度・湿度データの履歴を示している．温度・湿度センサデータは定期的に更新されるだけなので `CyclicResponder` を用いて実現している．

図 5 屋内避難誘導システムの管理画面のスナップショット

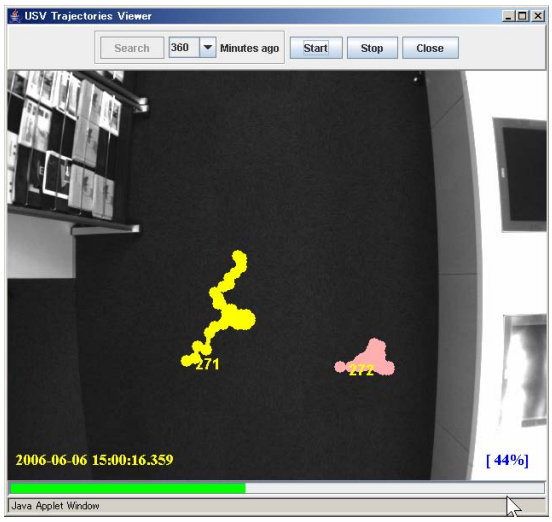

図 6 移動軌跡データ検索システムのスナップショット

3 種類のビデオ監視システムのうち，Ubiquitous Stereo Vision (USV)[17)18)] は，人の移動を検知し，その移動軌跡を記録することができる．そこで，SENSORD の時空間 API を用いて，フロアの受付付近における訪問者の移動軌跡データの記録，格納，検索機能も実現した．図 6 は，この軌跡データの検索システムのスナップショットである．この検索システムでは，監視エリア内を通過した人の軌跡を時間をキーとして検索し，時間順序でプレイバックする機能を持っている．この図は，右側に人が一名立っており，同時に中央を別の人が一名通過した場合の軌跡を示している．

5. 関 連 研 究

データベース研究の分野では，本研究に関連する，さまざまなタイプのデータベース技術が提案されている．時空間的に変化するデータを管理するためデータベース技術に関しては数多くの研究[3)10)16)] が存在している．センサデータの扱いをターゲットとした研究では，大量のセンサデータストリームを扱う研究[19)] や，センサネットワークを一種の分散データベースとみなしてデータ管理を行う研究[4)14)] などが存在する．データベース研究とは異なり，センサイベントの処理に注目したミドルウェアの研究としては GEM[9)], MASTAQ[7)], DSWare[13)] などがある．これらの研究と，SENSORD の相違は，SENSORD が時空間情報管理とサービス連携のフレームワークを統合したミドルウェアとして提案されている点にある．

6. ま と め

複数の異なるセンサ情報を統一的に管理し，高次のサービスへと結びつける機能を提供するミドルウェア SENSORD (Sensor-Event-Driven Service Coor-

product/tr_7w/tr_7w_01feature.html

☆1 プライバシーを考慮し，移動軌跡データおよび通過カウントデータのみを利用し，ビデオイメージ自体は利用していない．

☆2 Vitracom SiteView EP: `http://www.vitracom.de/downloads/siteview-st-ep-en.pdf`

☆3 Ubiquitous Stereo Vision: `http://staff.aist.go.jp/i-yoda/usv/index.html`

☆4 CED System, Inc. IBS Counter: `http://www.ced.cp.jp/IBS_CT_CAT.pdf`

dination Middleware) について述べた．SENSORD は，センサデータの解析と登録されたルールの適合性に基づいて，適切なサービスを選択・起動する機能を持つ．SENSORD の機能を利用することで，ユーザは状況に応じたサービスを提供するアプリケーションシステムを容易に実現できる．今後は，緊急時における避難誘導システムの実装と，実験環境で得られたセンサデータの解析を進めることで，SENSORD の API および SENSORDScript の完成度を高めていきたい．

参 考 文 献

1) G.D. Abowd, C.G. Atkeson, J.Hong, S.Long, R. Kooper, and M. Pinkerton. Cyberguide: a mobile context-aware tour guide. *Wirel. Netw.*, 3(5):421–433, 1997.
2) M.Addlesee, R.Curwen, S.Hodges, J.Newman, P. Steggles, A. Ward, and A. Hopper. Implementing a sentient computing system. *IEEE Computer Magazine*, (8):50–56, 2001.
3) K.K. Al-Taha, R.T. Snodgrass, and M.D. Soo. Bibliography on spatiotemporal databases. *SIGMOD Rec.*, 22(1):59–67, 1993.
4) A. Demers, J. Gehrke, R. Rajaraman, N. Trigoni, and Y. Yao. The cougar project: a work-in-progress report. *SIGMOD Rec.*, 32(4):53–59, 2003.
5) H. W. Gellersen, A. Schmidt, and M. Beigl. Multi-sensor context-awareness in mobile devices and smart artifacts. *Mob. Netw. Appl.*, 7(5):341–351, 2002.
6) K. Hiramatsu, T. Hattori, T. Yamada, and T.Okadome. A simple probabilistic analysis of sensor data fluctuations in the real world. In *In proceedings of 20th International Conference on Advanced Information Networking and Applications (AINA 2006)*, pages 399–406, 2006.
7) I.Hwang, Q.Han, and A.Misra. MASTAQ: A middleware architecture for sensor applications with statistical quality constraints. In *PERCOMW '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 390–395, Washington, DC, USA, 2005. IEEE Computer Society.
8) Y.Inoue, A.Sashima, and K.Kurumatani. Indoor navigation system for emergency evacuation in ubiquitous environment. In *Eighth International Conference on Ubiquitous Computing*, CD–ROM. ACM Press, 2006.
9) B. Jiao, S.H. Son, , and J.Stankovic. GEM: Generic event service middleware for wireless sensor networks. In *Proceedings of the 2nd International Workshop on Networked Sensing Systems (INSS)*, June 2005.
10) D.H. Kim, K.H. Ryu, and C.H. Park. Design and implementation of spatiotemporal database query processing system. *Journal of Systems and Software*, 60(1):37–49, 2002.
11) J.Krumm and E.Horvitz. Predestination: Inferring destinations from partial trajectories. In *Eighth International Conference on Ubiquitous Computing*, pages 243–260. ACM Press, 2006.
12) J. Lester, T. Choudhury, and G. Borriello. A practical approach to recognizing physical activities. In *Proceedings of The Fourth International Conference on Pervasive Computing (PERVASIVE 2006)*, pages 1–16, 2006.
13) S.Li, S.H. Son, and J.A. Stankovic. Event detection services using data service middleware in distributed sensor networks. In *IPSN*, pages 502–517, 2003.
14) S. Madden, M. J. Franklin, J. M. Hellerstein, and W.Hong. TinyDB: An acqusitional query processing system for sensor networks. *ACM Transactions on Database Systems*, 30(1):122–173, 2005.
15) D. Wilson and C. Atkeson. Simultaneous tracking and activity recognition (STAR) using many anonymous, binary sensors. In *Proceedings of The Third International Conference on Pervasive Computing (PERVASIVE 2005)*, pages 62–83, 2005.
16) O. Wolfson, B. Xu, S. Chamberlain, and L.Jiang. Moving objects databases: Issues and solutions. In *Proceedings of the 10th International Conference on Scientific and Statistical Database Management*, pages 111–122, 1998.
17) I.Yoda, D.Hosotani, and K.Sakaue. Ubiquitous stereo vision for controlling safety on platforms in railroad stations. In *Proceedings of the Sixth Asian Conference on Computer Vision (ACCV 2004)*, volume 2, pages 770–775, 2004.
18) I.Yoda and K.Sakaue. Concept of ubiquitous stereo vision and applications for human sensing. In *Proceedings 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA2003)*, pages 1251–1257, 2003.
19) 川島英之, 今井倫太, 遠山元道, and 安西祐一郎. センサデータベースシステム KRAFT の設計と実装. **情報処理学会論文誌：データベース**, 45(14):39–53, 2004.
20) 太田正幸, 山下倫央, and 車谷浩一. 屋内空間避難誘導シナリオのシミュレーションによる検討. In **第 *22* 回 ファジィ システム シンポジウム 講演論文集**, pages 819–822, 2006.