卒業論文 2003 年度 (平成 15 年度)

異常状態の自動定義による状況監視アプリケーションの支援

指導教員

慶應義塾大学環境情報学部

徳田　英幸

村井　純

楠本　博之

中村　修

南　　政樹

慶應義塾大学 環境情報学部

出内 将夫

ide@sfc.wide.ad.jp

卒業論文要旨　　2003 年度 (平成 15 年度)

異常状態の自動定義による状況監視アプリケーションの支援

本論文では，実世界の異常を自動的に検知し，アプリケーションに通知するミドルウェア，SARADA を提案する．SARADA を用いることで，状況監視アプリケーションの開発，導入に必要なコストが削減できる．

近年，ネットワーク接続性を備えたセンサや機器が環境に遍在する，ユビキタスコンピューティング環境が実現しつつある．また，このような環境に存在する多種のセンサや機器を用いて環境の異常を検知し，異常への対処を行う状況監視アプリケーションが開発されている．

環境の異常検知には異常の判定基準が必要である．しかし，判定基準は環境ごとに異なるため，各々の環境に適応した判定基準の設定が必要である．また，多種のセンサや機器を多数用いる際には，設定する判定基準が複雑になる．そのため，判定基準の設定作業は，アプリケーション開発者やユーザにとって大きな負担となる．

SARADA は，決定木学習を用いてセンサや機器によって取得できる情報の履歴から環境に応じた異常の判定基準を動的に定義する．さらに，異常を検知し，アプリケーションへの通知を行う．本論文では，SARADA の実装，評価を行った．評価として他の定義手法と SARADA との機能比較，判定基準の設定に必要な時間の計測を行い，他の定義手法に対して導入時の負担が軽減することを示した．

慶應義塾大学　環境情報学部
出内 将夫

Abstract of Bachelor's Thesis

SARADA: Support for Application of Recognizing environment by Auto Detection of Anomalies

This paper introduces "SARADA", the middleware that automatically detects the real world anomalies and notify applications of it. SARADA reduces the cost of development and deployment of applications that observe the environment.

Recently, the applications that detect and handle the user concerning anomalies in the environment, by networked devices and sensors are being developed. The criterion of anomalies is needed to detect the anomalies of environment. But the criterion of an anomaly varies in each environment. Defining such criterion for each environment is burden for applications developers and users.

In ubiquitous computing environments, as there are various devices and sensors around that can recognize many kinds of contexts. So the anomalies that are targeted by applications also vary. But it is difficult to define criterion for such anomalies on each device.

This research shows the design and implementation of middleware SARADA, which dynamically defines the criterion for anomalies and detects it. SARADA create decision tree from history of environmental information to define the criterion. For evaluation, we compare SARADA with other definition methods and measure the time needed for creating a definition, and show advantages to other methods.

Masao Ideuchi
Faculty of Environmental Information Keio University

目 次

図目次

表目次

第1章　序論

1.1 本研究の背景

近年，情報技術の進歩により，計算能力を持った様々なデバイスの小型化や低価格化が進んでいる．さらに，これらのデバイスがネットワーク接続性を備えることにより，協調的な動作が可能となった [15]．このようなデバイスが遍在し，それらが協調動作することで，人々の日常生活を支援する環境を，ユビキタスコンピューティング環境 [22] と呼ぶ．現在，ユビキタスコンピューティング環境の実現に向け，様々な施設が実験的に構築されている [9][14]．また今後，このような情報環境が，実験施設から，オフィス，キャンパス，家庭や公共空間へと浸透していくと考えられる．

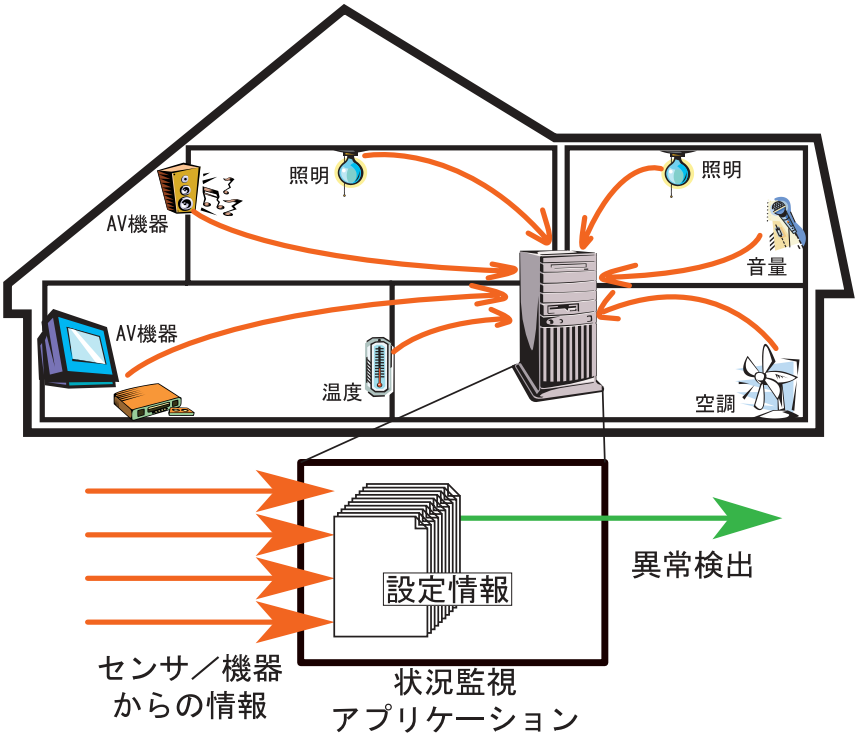

図 1.1: 状況監視アプリケーションの概念図

上述のようにネットワークに接続されたセンサや機器を利用するアプリケーションの一つとして，状況監視アプリケーションが開発されている [3][18]（図 1.1）．状況監視アプリケーションとは，センサや機器を用いて環境をモニタリングし，異常への対応やユーザへの通知を行うアプリケーションである．異常とは，ユーザに不利益をもたらす状況，あるいはその可能性のある状況など，ユーザが関心を持つ普段と異なった状態である．状況監視アプリケーションの例としては，カメラや人感センサを用いて侵入者を検知し，通報する防犯警備システムや，位置センサや機器の使用状態を用いて居住者の生活状態を割り出し，介護に役立てる遠隔介護システムなどが挙げられる．

ユビキタスコンピューティング環境では，ネットワークを介することで，様々なセンサや機器が，様々なアプリケーションから利用可能となる．また，このような環境では，機

能や性能の異なるセンサや機器が混在すると考えられる．そのため，アプリケーションから機能や性能の差異を意識することなく，センサや機器を透過的に利用するための研究が行われている．例として，センサから得られる値を抽象化する手法 [10] や，情報家電への Java プラットフォームの搭載 [1] などが挙げられる．これにより，アプリケーションは多様なセンサや機器に対して個々に対応する必要は無くなり，アプリケーションの開発コストが下がる．

このような環境下で，状況監視アプリケーションはネットワークを介して，様々なセンサや機器を容易に活用できる．これにより，多くのセンサや機器を連携させて様々な異常を検出できる．例えば，ガスの使用量の計器とガス使用機器の状態を調べることによる，ガス漏れの検知や，情報家電の動作パターンを監視することによる，住人の昏倒の検知が可能となる．

1.2 問題意識

状況監視アプリケーションが，センサや機器によって取得した情報から異常を判定するためには，異常の判定基準が必要である．異常の判定基準の例として，人が倒れていることを検知するための基準を挙げる．まず，人が同じ場所に 10 時間以上滞在したら異常，のように，１つの情報から判定する異常が挙げられる．また，部屋の照明が on の状態で人が部屋に 5 時間以上滞在したら異常，部屋の照明が off の状態では 12 時間以上滞在したら異常，など，複数の情報を連携させることで判定できる異常も存在する．多様なセンサや機器が利用可能な場合，複数の情報から判断することで，状況をより詳細に記述でき，検知の精度を高めることができる．

ある状態を異常とする定義は，センサが設置されている場所や，機器を使用している状況や使用頻度などによって異なるため，一意に決定することは難しい．従って，様々な異常の判定基準を設定する際には，センサが設置された環境や，機器の利用環境を考慮する必要がある．

環境に応じた設定をユーザや開発者が手作業で行うことは，増加し続けているセンサや機器の数や種類を考慮すると，大きな負担になる．さらに，上述のユビキタスコンピューティング環境では，複数の情報から異常を判定することが可能となる．このような環境では，異常の判定基準の設定の際に考慮しなければならないセンサや機器の組み合わせはさらに増加する．そのため，開発者やユーザが環境に応じた設定を手作業で行うことは，膨大な労力が必要となる．

1.3 本研究の目的

本研究の目的は，異常の検知に用いる判定基準を，環境に応じて自動生成することで，ユーザや開発者にかかる負担を軽減することである．これにより，状況監視アプリケーションの開発，導入を支援する．本研究では，環境に応じた異常状態の定義を動的に生成するミドルウェア，SARADA(Supporting system for Applications of Recognizing environment

by Auto Detection of Anomalies) を構築する．

SARADA は大きく分けて，学習と検知の２つの動作がある．１つは，センサや機器から得られる環境情報の履歴から判定基準を自動生成する学習の動作である．これにより，ユーザによる判定基準の入力を必要としない．もう１つの動作は，異常が検知された場合，アプリケーションに対して異常発生と異常情報を通知する動作である．本システムを用いることにより，アプリケーション開発者もしくはユーザにかかる負担を軽減できる．

1.4　本論文の構成

本論文では，第２章で状況監視アプリケーションと異常検知の手法について述べ，ユビキタスコンピューティング環境における状況監視アプリケーションについて考察する．第３章では，本研究で用いる異常状態の自動定義手法について述べる．そして第４章で，異常状態を自動定義する SARADA の設計について，第５章で実装について述べる．第６章で SARADA を評価し，第７章で本論文をまとめる．

第2章　異常状態の検知

本章では，異常状態の検知について述べる．まず，異常状態を判定する基準となる環境属性について説明し，異常状態と定常状態の関係を述べ，異常状態の定義とは何かについて述べる．次に，状況監視アプリケーションの役割と，検知対象である異常についての指標について考える．そして，従来の状況監視アプリケーションとユビキタスコンピューティング環境での状況監視アプリケーションを，環境の違いと，その環境で用いる異常状態の検知手法，対象とする異常状態について比較する．

2.1　異常状態

本節では，本論文で扱う実世界における環境の異常状態とその判定基準となるデバイスから取得する環境属性，異常状態と定常状態の関係，定常状態の環境による違いについて述べる．本節で述べる内容を図に表すと，図 2.1 のようになる．

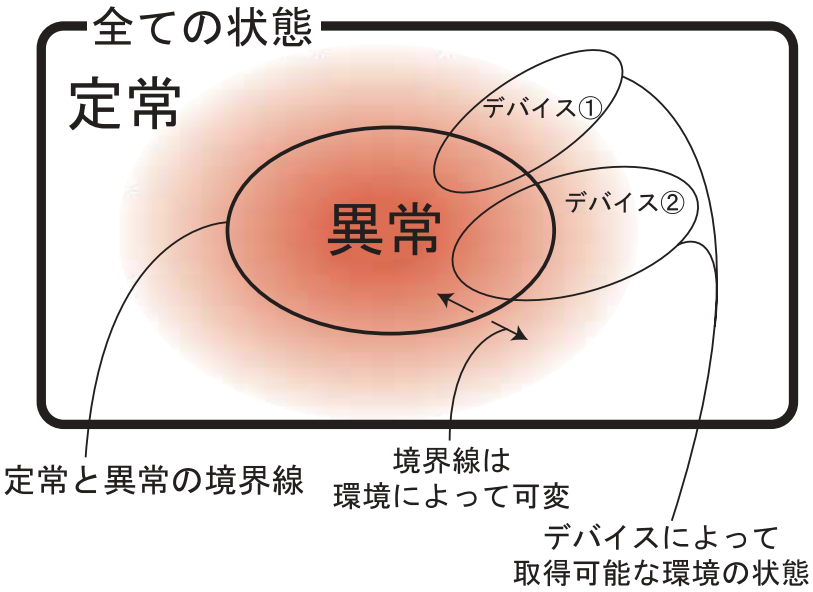

図 2.1: 定常状態と異常状態の関係

2.1.1　環境属性による環境状態の表現

本論文で扱う異常状態とは，人々の生活環境上で生じる普段と異なった状態を指す．例えば，家が誰かに侵入された，家にいる家族の様子がおかしい，などの人が不利益を及ぼされる，あるいはその可能性がある状態である．

異常状態は，環境に設置されたセンサや機器などのデバイスから取得する環境属性により判定する．**環境属性**とは，環境の状態を表現する情報である．例として，温度や湿度，音量，光量，機器の使用状態が挙げられる．温度が 20 度，湿度は 55 度，ライトは点灯しているなど，環境属性が示す値の集合によって環境の状態を記述できる．

2.1.2　異常状態と定常状態

異常状態は定常状態の対となる．環境が取りうる全ての状態を全体集合として考えたとき，定常状態と異常状態は互いに補集合の関係となる．したがって，異常状態の定義は定常状態の定義に等しい．

定常状態とは，環境属性が普段の値やその近似値を指している状態である．環境属性が

示す普段の値は，センサが設置されている位置や，機器が使用されている状況や使用頻度によって異なるため，全ての環境における定常状態は一意に決定できない．したがって，環境ごとに定常状態と異常状態の境界線を定める必要がある．

2.2 状況監視アプリケーション

本節では，まず本研究が対象とする状況監視アプリケーションの役割について述べ，次に状況監視アプリケーションが対象とする異常について説明する．

2.2.1 状況監視アプリケーションの役割

状況監視アプリケーションは，2.1 節で述べた異常状態を検知し，管理者やユーザへの通知や，警報を発することにより，異常状態への対処を行うアプリケーションである．これにより，身体的危険，金銭的損害などを未然に防いだり，緩和できる．

2.2.2 検知対象となる異常状態の指標

状況監視アプリケーションが対象とする異常状態について述べる．その際，ある異常状態を検知することへの需要と，異常状態の抽象度の 2 つの指標を用いる．

- 需要

  ある異常状態の検知をどの程度多くのユーザが望んでいるかを指す．対象とする異常検知の需要が高いほど，状況監視アプリケーションの価値も高くなる．そのため，状況監視アプリケーション作成の際には，より多くの需要がある異常が検知対象として選ばれる．

- 抽象度

  ある異常状態の検知が，どの程度困難かを示す．抽象度が低い異常とは，センサや機器が単体で検知できる異常を指し，抽象度が高い異常とは，センサや機器単体では検知することが難しい異常を指す．抽象度が低い異常の例としては，温度の異常，ガスの使用量の異常などがあり，抽象度が高い異常の例としては，老人が倒れたことの検知が挙げられる．抽象度が高い異常は，検知に必要なデバイスが多く，判定基準も複雑になるため，検知の難易度が高い．

以降，2.3 節や 2.4 節で，状況監視アプリケーションが対象とする異常について述べる際，この 2 つの指標を用いて評価する．

2.3　従来の状況監視アプリケーション

本節では，従来の状況監視アプリケーションが行う異常検知について述べる．まず動作環境について述べ，次に異常定義の手法について述べる．最後に従来の状況監視アプリケーションが対象とする異常について，2.2.2 項で述べた指標を用いて考察する．

2.3.1　動作環境

従来の環境では，状況監視アプリケーションがあらかじめ利用可能なセンサや機器は少ない．そこで，環境の異常状態を検知するために新たにセンサや機器とそれらを繋ぐネットワークを設置していた [2][24][28]．このようにして設置するセンサや機器は，状況監視アプリケーション導入にかかるコストを増加させるが，状況監視アプリケーションに特化できる．

従来の状況監視アプリケーションは，多くのユーザが共通して興味を持つ異常状態に候補を絞り，検知対象としている．そして，状況監視アプリケーションが利用するセンサや機器は，それらの異常を判定することを目的として環境に設置される．

2.3.2　異常状態の定義手法

上述の動作環境では，状況監視アプリケーションは利用するセンサや機器を統一できるため，異常の定義をアプリケーションごとに静的に行うことが多い [2][28]．静的な定義とは，状況監視アプリケーションが異常の判定に用いる異常の定義が，アプリケーション開始時から与えられおり，再設定を行わない限り，周囲の状況に応じて変化しないことを指す．静的な定義の方法として，アプリケーション開発者やユーザによる手動定義がある．

この定義手法は，適切な設定が行われていれば，アプリケーション開発者やユーザの意図した異常のみが検知されるため，誤検知を少なくできる．センサの設置場所が移動することや，ユーザの行動パターンが変化することで，デバイスの置かれる環境が変化することは想定していないため，設置条件に制約がある．

2.3.3　検知対象

従来の状況監視アプリケーションが対象としていたのは，図 2.2 に示すように，需要が高く，抽象度が低く検知が簡単な異常である．この図は，2.2.2 項で述べた需要を縦軸にとり，抽象度を横軸にとった図である．右に行くほど抽象度が高く，検知が難しい異常を示し，上に行くほど検知の需要が高い異常を示す．需要は異常検知の価値を表し，抽象度は異常検知の難易度と見ることができるため，図中の線より上側は，検知の難易度に見合う需要がある異常を指す．その領域の中に従来の状況監視アプリケーションが対象とする異常が含まれる．

従来の状況監視アプリケーションとしては，防犯や警備，老人介護を目的とした，セ

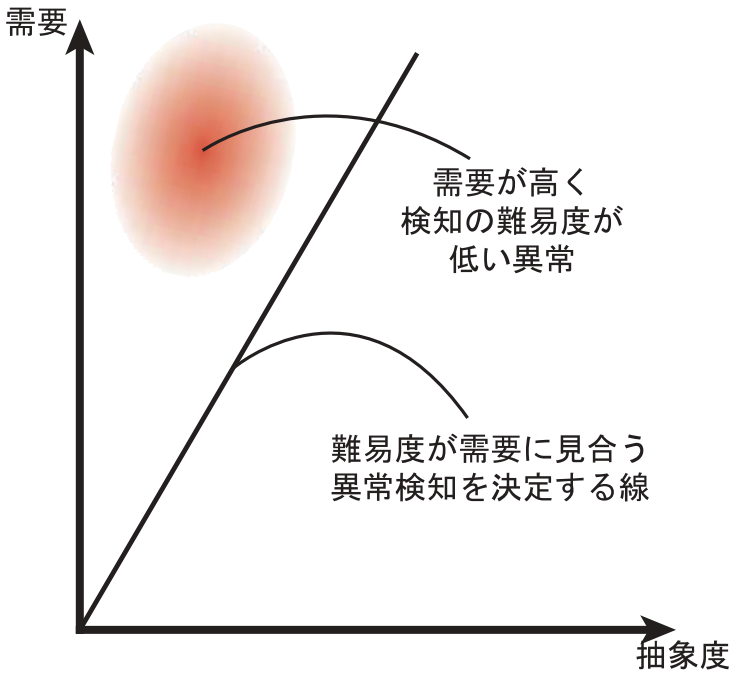

図 2.2: 従来の状況監視アプリケーションによる異常検知

キュリティ関連の異常検知が挙げられる．現在，家庭における盗難の増加 [25] や老人のみで暮らす家庭の増加 [27] によって，これらの異常検知に関心を持つ人々の数は増えている．そのため，防犯警備システムはオフィスや家庭で，老人介護システムは住宅や老人ホームで普及している．

2.3.4 既存のシステム

本項では，既存のシステムを列挙し，それぞれについて，問題点や本研究との差異を述べる．

ケアモニタ

松下電工 [13] による，ケアモニタ [28] は，高齢者福祉施設を対象としたシステムである．近赤外光を用いたカメラによる画像処理技術を用いて，高齢者の位置情報を取得することで，居室における居住者のふるまいの異常を検知できる．このシステムは，高齢者福祉施設を対象としたもので，居住者が異常な振る舞いをした場合に，介護スタッフが持つ携帯端末に自動通報することにより，居住者の安全の確保と，即応的なサービス提供を可能にする．また，環境に設置したセンサのみを利用しており，居住者が，ボタンを押すなどの必要がないため，実際に異常が起きた場合に通報できなくなることも無い．

しかし，このシステムは使用する際に，ある場所に何分滞在したら異常と判定する，あるいは，何分動きが無かったら異常と判定する，などの数値設定を行う必要がある．また，

高齢者福祉施設を対象としており，部屋の構造が画一的で，個室を想定しているため，家具の配置の変化や，部屋の中に複数人存在する状況には対応できない．

みまもりほっとライン i-PoT

象印マホービン株式会社 [4] によって開発された，みまもりほっとライン i-PoT[3] は，動作状態を取得可能な電気ポットを [2] を用いて，高齢者の行動をモニタリングするシステムである．このシステムは，一人暮らしの高齢者を対象としたもので，高齢者の電気ポット使用履歴を携帯やパソコンから確認できる．確認方法には，電子メールサービスを用いた文字情報や，Web や携帯アプリを用いたグラフ情報がある．また，指定時間使用されない場合に自動送信させる設定が可能である．

しかし，自動検知を行うためには，ユーザが普段の行動を参考にしながら，異常と判定するための適切な時間をユーザが設定しなければならない．また，この設定は静的であるため，想定外の事象には対応できない．この場合，電気ポットが使用されない時間で異常を判定するため，高齢者が外出しただけで異常と判定されてしまい，異常検知の精度が低くなる．

2.4 ユビキタスコンピューティング環境下の状況監視アプリケーション

本節では，ユビキタスコンピューティング環境下の状況監視アプリケーションにおける異常検知について述べる．まず動作環境について考え，次に異常検知の手法について考える．最後に，ユビキタスコンピューティング環境下の状況監視アプリケーションが対象とする異常について，2.2.2 項で述べた指標を用いて考察する．

2.4.1 動作環境

今後，環境にセンサや機器が増加し，状況監視アプリケーションを含む様々なアプリケーションが，それらのデバイスをネットワークを介して利用可能になると考えられる．このような環境では，状況監視アプリケーションが利用可能なセンサや機器のインフラを用いて環境属性を取得し，異常状態を検知できる．それらの既存のデバイスは状況監視アプリケーションに特化したものではないが，状況監視アプリケーションの導入コスト削減や，複数のセンサや機器を用いた多様な異常の判定が行える．例えば，ガスの総使用量とガス使用器具の状態を関連付けることで，ガス漏れ検知を行うことや，老人が部屋で倒れているのか寝ているのかを識別するために，位置センサや家電機器の使用状態，部屋の温度や明るさなどの情報を総合して判断できる．

2.4.2　異常状態の定義手法

上述の環境では，状況監視アプリケーションは既存のセンサや機器が利用できる．逆に，利用するセンサや機器を事前に定義することは困難である．そこで，異常の定義を動的に行う必要がある．動的な定義とは，アプリケーションが異常の判定に用いる異常の定義が，アプリケーション動作中に周囲の状況に応じて変化することを指す．動的な定義の方法には，ソフトウェアによる定義の自動生成がある．

この定義手法は，ネットワーク上の侵入検知システムにおける異常検知に用いられており，異常の定義を生成するのに，まず定常を定義し，その補集合，つまり定常の定義に当てはまらないものを異常と定義する手法が採られている [11][12]．この方法には，アプリケーション開発者やユーザによる設定を行う必要がないという利点がある．しかし，定常を定義するための計算負荷がかかることや，環境の変化やユーザの行動パターンの変化などによって誤検知が生じること，定常を定義するために必要な情報の収集に時間がかかること，などが問題として挙げられる．この問題への対処法として，自動定義するだけでなく，その結果をユーザが確認でき，定義の編集や追加を可能にする必要がある．

2.4.3　検知対象

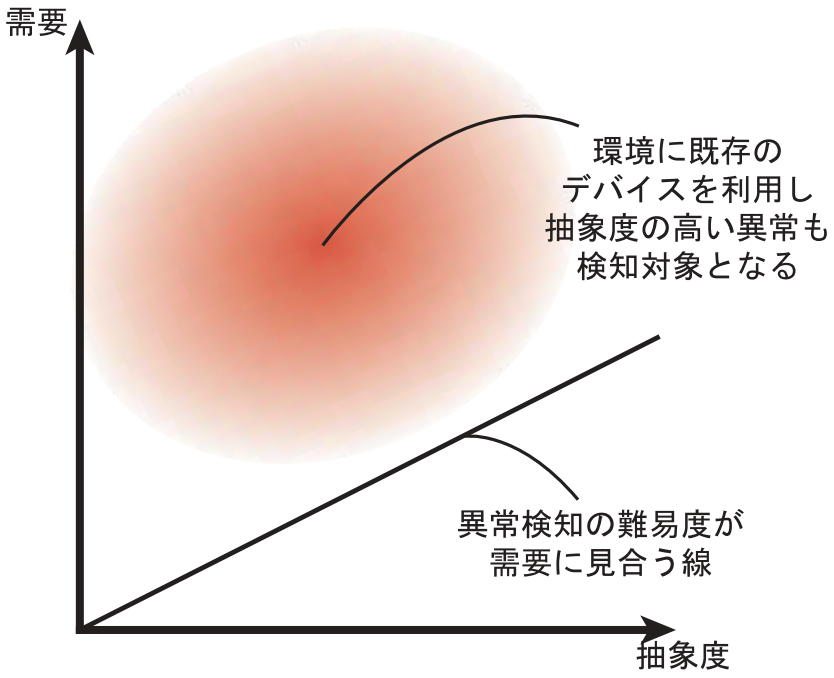

図 2.3: ユビキタスコンピューティング環境下の状況監視アプリケーションによる異常検知

今後の状況監視アプリケーションでは，先に述べた動作環境の変化や検知手法の発達により，図 2.3 に示すように，検知対象となる異常も多様化すると考えられる．図中の線の傾きが小さくなっているのは，従来の動作環境に比べて，種類や数が多いセンサや機器が

利用可能となるため，それらのデバイスを用いることで，抽象度が高い異常も検知可能となるからである．また，このような環境ではアプリケーションとデバイスの関係が動的であるため，状況監視アプリケーションが利用するデバイスを組み替えるだけで，個人の関心に合わせた異常検知も行える．例として，現状では全てのものにタグをつける必要がある自動忘れ物検知や，ペットの行動監視などが挙げられる．

2.5 本章のまとめ

本章ではまず，本論文で扱う異常状態について述べ，状況監視アプリケーションとの関係を示した．次に現状の状況監視アプリケーションについて述べ，今後の状況監視アプリケーションについて考えた．

次章では，異常状態の自動検知について述べる．

第3章　異常の自動検知

本章では，本研究で用いる異常を自動検知する手法について述べる．まず，自動検知手法の概要について述べ，次に，検知に用いる基準を自動定義する際に考慮する指標について述べる．そして，その指標を用いて自動定義アルゴリズムを選別し，決定木学習における定常状態の定義の記述方式と学習方法のそれぞれについて，決定木での手法と，本研究での利用方法を述べる．さらに，異常状態の検知手法について述べ，本章をまとめる．

3.1　定常状態の自動定義による異常検知

本研究で用いる，異常状態の自動的な検知手法について説明する．本研究では，まず定常状態を定義し，その定義に反する状態を判定することで異常状態を検知する．以降，定常状態の定義方法について説明する．

定常状態は対象となる環境ごとに異なるため，それぞれの環境に応じた設定を行う必要がある．本研究では，ある環境において，設置されたセンサや機器によって取得される環境属性の履歴情報の多くは，その環境の定常状態を反映していることに着目した．履歴情報の処理方法については，3.2 節で自動定義の要件に述べた後，3.4 節と 3.5 節で詳しく述べる．

3.2　自動定義の要件

定常状態を自動定義するシステムの要件として，環境適応性と定義記述の柔軟性を挙げる．それぞれについて以下で説明する．

3.2.1　環境適応性

環境適応性を持つとは，監視対象となる環境に応じた定常状態の定義が設定可能であることを指す．つまり，センサが設置されている場所や機器の使用している状況，使用頻度などを考慮した異常の判定が可能となることを指す．これにより，異なった環境で，それぞれに適した異常の判定が行える．

3.2.2　定義記述の柔軟性

開発者もしくはユーザが，自動生成された定常状態の定義を理解できる，あるいは理解可能な形に翻訳できることを指す．すなわち，自動生成された定常状態の定義を，開発者やユーザが確認することや編集可能であることを指す．これにより，ユーザが自動生成された定常状態の定義を手動で操作し，思い通りの設定が行える．

3.3　自動定義アルゴリズムの比較検討

前節で挙げた要件を考慮しながら，自動定義に用いるアルゴリズムについて検討する．検討したアルゴリズムは，数量化法アプローチ [7]，ベイジアンネットワーク [5]，ニューラルネットワーク [8]，決定木学習 [26] の 4 つである．検討した結果，決定木学習を用いることとした．結果のまとめを表 3.3 に示す．

表 3.1: 定常状態の定義に用いる手法とその比較

用いる手法	環境適応性	定義記述の柔軟性
数量化法アプローチ	△	×
ベイジアンネットワーク	○	△
ニューラルネットワーク	△	×
決定木	○	○

3.3.1 数量化法アプローチ

ある１つの環境属性が他の環境属性とどの程度相関があるかを，数式化することで表現する．この数式は常にデータ全体の規則性を表すため，局所的な規則性に対応できない．局所的な規則性とは，例えば人感センサが部屋内にあった場合，部屋内に人がいるときの規則性，またはさらに条件が加わって，部屋内に人がいて，かつ照明が消えているときの規則性などの，条件に制約を与えた場合の規則性のことである．

また，全ての属性を数値として扱うため，属性が数値で表せない離散値を取る場合，属性が取る値と同じ数の変数を用いることになり，変数の数は膨大になる．さらに定常状態の定義を，それぞれの変数に係数を持つ項の総和を表す数式で表現する．各項の係数が相関の正負を表し，２つの属性間の関連を捉えることはできる．しかし，全体との関連を把握するためには数式全体を理解する必要があり，人間が解釈するのは困難である．

3.3.2 ベイジアンネットワーク

複数の属性間の因果関係を条件付き確率で表したグラフ構造を構築する．この方法は，履歴データから各属性同士の因果関係を記述できるが，初期のグラフ状態を設定する必要がある．ベイジアンネットワークのアルゴリズム単独では，この初期設定については定義されていないため，初期のグラフ状態の設定が必要となる．初期のグラフ状態の設定に学習アルゴリズムを用いれば自動的な定常状態の定義は可能であり，様々なアルゴリズムが検討されている．

定義記述の柔軟性については，グラフ構造における連結が因果関係の有無を表すため，属性同士の因果関係は直感的に理解可能である．しかし，条件付き確率を表す場合，因果関係が数値の表を用いて表現され，定常状態か異常状態かの判定は複数の数値を考慮して判断しなければならず，人間が解釈，編集するのは困難である．

3.3.3 ニューラルネットワーク

入力と出力の属性間の写像関係を表現するグラフ構造を構築する．このグラフ構造は，初期設定で関連のない属性を連結しても，学習が進むにつれて，関連の深さによって連結の重みが調整されるため，自動的な構築が可能である．しかし，自動的に定常状態を定義

するためには，入力と出力のセットを与える必要があり，履歴データに加えて，それが異常か定常かを表すデータも必要である．履歴には定常状態のみが含まれるとは限らないので，自動的な定常状態の定義は難しい．

また，学習後のネットワークそのものが定常状態を表しているが，ネットワーク内部はブラックボックスともいえる複雑な構造となるため，可読性が無く，定義記述の柔軟性を満たしていない．

3.3.4 決定木学習

決定木学習は，与えられたデータの集合を基に分類の基準となるデータ項目を，最も効率良く分類できる規則を探し出す．これにより，複数のデータ項目間の関係を木構造で表現する学習アルゴリズムである．この方法は，条件付き確率を考慮した履歴データの分類を自動的に行い，履歴データ以外の情報や初期設定も必要としない．

この木構造は，トップダウンにたどると，フローチャートのように読めるため，容易に理解できる構造である．そのため，この木構造による記述方法は，定義記述の柔軟性を満たす．

3.4 木構造による定常状態の記述

本研究では決定木学習を用いるため，環境属性の定常状態を木構造で記述する．そこで，木構造による定常状態の記述方法について説明する．

まず，木構造について簡単に説明し，次に，木構造が決定木学習でどのように用いられているか説明する．最後に，本研究で用いる木構造による定常状態の表現方法を述べる．

3.4.1 木構造

本項で扱う木構造とは，閉路を持たない１つの根を持つ連結有向グラフを指す．グラフとは，頂点の集合と，頂点同士のつながりを表す辺の集合である．連結グラフとは，任意に選んだ２つの頂点が，辺や他の頂点を介して，直接あるいは間接に結ばれているグラフである．有向グラフとは，辺が方向性を持つグラフである．

”根”とは，そこに入り込む辺を持たない頂点である．また，根以外の頂点はそこに入り込む辺を１つ持ち，出て行く辺を持たない頂点は**”葉”**と呼び，それ以外の頂点を**”節”**と呼ぶ．節と節，あるいは節と葉を結ぶ辺を**”枝”**と呼ぶ．これらの性質を持つ点の集合と辺の集合を合わせて**”木”**と呼ぶ [23]．図 3.1 に木構造の例をを示す．

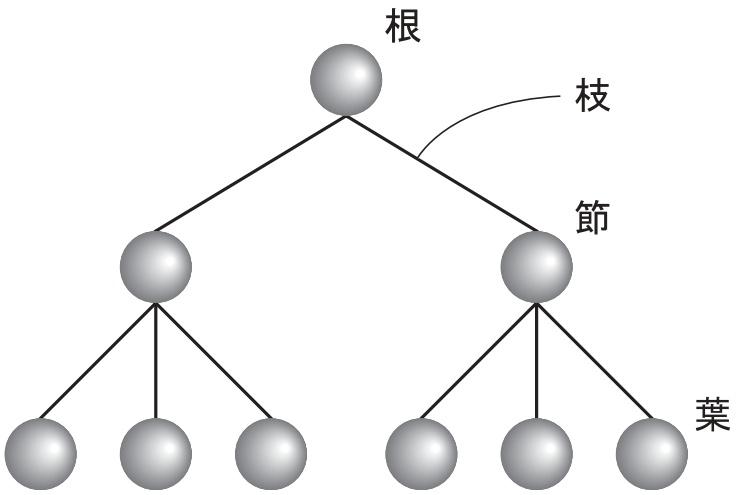

図 3.1: 木構造の例

3.4.2 決定木における木構造

決定木は，以上に述べた木の性質を持つ．それに加えて，決定木では根，節，葉や節などのそれぞれの部分にそれぞれ役割を持つ．根と節は，与えられたデータに対する質問文の役割を持ち，辺は，節における質問文の答えとなる選択肢の役割を持つ．また，根は一番最初の質問文となる．そして葉は，分類の終端であるクラスとなる．クラスとは，分類の基準となるデータ項目が取る値である．

決定木は通常，根を上にした形で描く．そして，上から順になぞりながら，データに質問文に答えていくことを繰り返すことで，分類の終端となるクラスへと到達する．決定木の例を図 3.2 に示す．

この決定木は，０～３の数当て問題の例である．０～３の整数のいずれかを示すデータがあり，根にある質問文から始めて，該当する選択肢，次の質問 … と繰り返すことにより，最終的に正解の数字へと到達する．この例で，もし x=2 が与えられた場合，まず，根に該当する x ＜ 2 の質問文に対して，答えが no であるため，右側の枝に進む．さらに x ＜ 3 の質問文に対して答えることにより，実際の正解である 2 へと到達する．

3.4.3 本研究での利用方法

以下に，本研究で用いる定常状態の記述方法を，決定木の構造と対応させながら説明する．

定常状態を記述する木は，環境に設置されたセンサや機器から取得される環境属性を用いて記述する．クラスには，環境属性の１つを割り当て，質問文には，クラスとなった環境属性以外の環境属性を用いる．質問文に対する答えには，環境属性が取る値を割り当てる．このようにして構築された木が定常状態を表す決定木となる．図 3.3 に，環境属性を用いて定常状態を定義した決定木の例を示し，表 3.2 に，木構造や決定木との対応表を

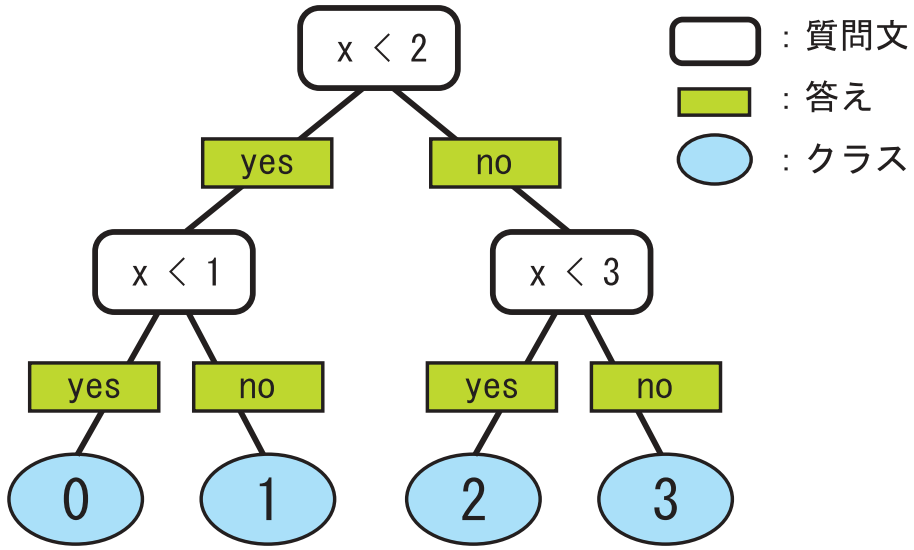

図 3.2: 決定木の例

表 3.2: 木構造，決定木，定常状態を記述した木の対応表

木構造	決定木	定常状態を記述した木
根，節	質問文	クラス属性以外の環境属性
枝	質問文の答え	上の節とした環境属性が取る値
葉	クラス	クラス属性とした環境属性

示す.

この例では，”扇風機の電源状態”という環境属性をクラス属性とし，その値が ON か OFF かをクラスとして分類することで，環境の定常状態を定義した木を表している．この木が表す定常状態は，部屋の温度が 24 度以下の場合に扇風機が OFF の状態，部屋の温度が 24 度以上だが，人が部屋にいない場合に扇風機が OFF の状態，部屋の温度が 24 度以上で，人が部屋にいて 10 分経った場合に扇風機が ON の状態，である．

3.5 決定木学習

本節では，本研究で用いる決定木の学習アルゴリズムの説明と，本研究での工夫点について述べる．まず，節を決定する方法と枝を生成する方法について説明し，次に，冗長な枝を排除することで決定木を簡潔にする，枝刈りについて述べる．最後に，本研究で用いる際に行った工夫について述べる．なお，決定木を構築するアルゴリズムは複数存在するが，本研究で参考にしたアルゴリズムは，ID3[16] と C4.5[17] である．

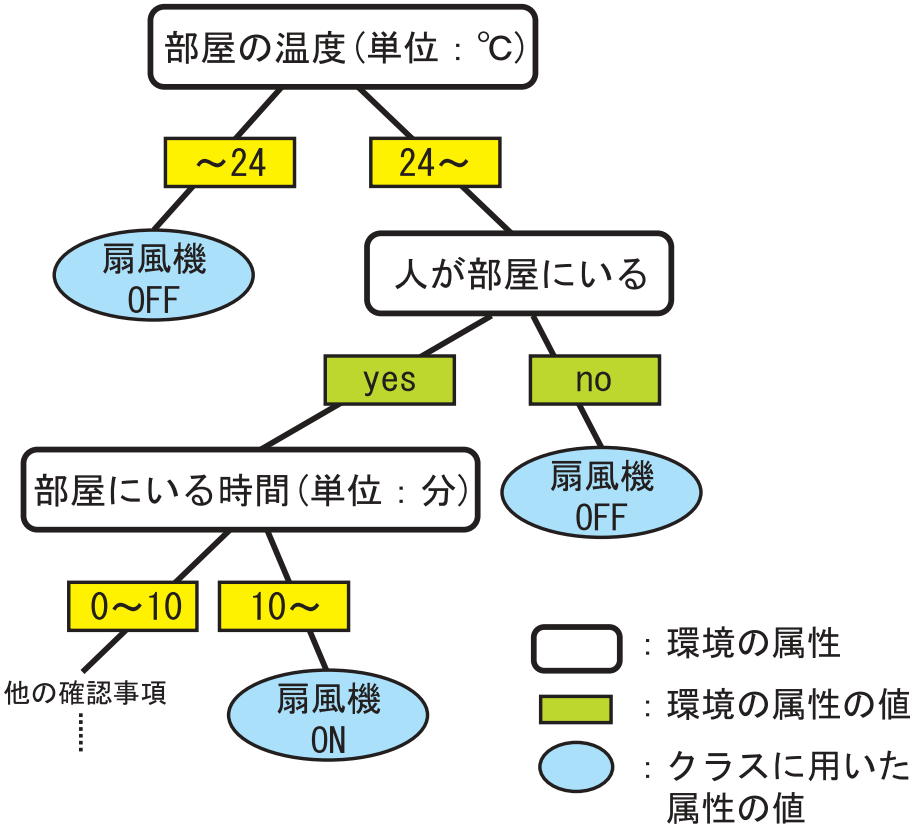

図 3.3: 扇風機の電源状態を異常判定に用いた木

3.5.1 木の構築方法

決定木を構築する際，節は分岐点の役割を果たすため，適切な節を選ぶことで，枝の本数や続く分岐を少なくできる．よって，簡潔な木の作成には適切な節の選択が必要である．まず，節の決定方法を述べ，次に，枝の生成方法と木全体の構築方法について説明する．

節には，クラス属性以外の属性の中から，学習対象となる集合全体をクラスを基準として最も効率良く分割できる属性を選ぶ．本研究では，学習対象となる集合として，システムに蓄積された履歴情報を用いる．その際，分割の効率を示す基準には情報利得比を用いる．情報利得比は，情報利得を分割情報量で正規化したものである．集合 X を属性 A によって分割したときの情報利得比 $GR_A(X)$ は，情報利得を $G_A(X)$ とし，分割情報量を $SI_A(X)$ とすると，以下の式で表せる．情報利得と分割情報量については後述する．

$$GR_A(X) = \frac{G_A(X)}{SI_A(X)}$$

情報利得は，シャノンの情報理論 [19] によるエントロピーが分割によってどれだけ減少したかを表す基準であり，この値が大きいほど，クラスによる分類が進んだことを表す．集合 X のエントロピー $E(X)$ は，X が n 個の互いに重ならない集合に分割でき，X にお

ける i 番目の集合が占める割合を p_i とすると，$E(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$(bit) で表すことができる．学習対象の集合が全てクラスに分類された後は，p_i が 0 または 1 となり，エントロピーが 0 となるので，分割前後のエントロピーの差分を表す情報利得も 0 となる．上述の式での情報利得 $G_A(X)$ は分割後の集合を $S_j(i = 1, 2, ..., k)$ として，以下の式によって表される．

$$G_A(X) = E(X) - \sum_{j=1}^{k} \frac{|S_j|}{|X|} \times E(S_j)$$

分割情報量は，選んだ属性が集合全体に対して，どれだけ分割を細かく行うかを表す基準であり，この値が大きいほど，細かい分割を行うことを意味する．上述の情報利得のみで分割を行った場合，分割が細かければ細かいほど，情報利得の値は高くなる．そのため，各要素を識別する ID による分割のように，集合に含まれる他の要素に応用できない分割に対する評価も高くなる．そこで，細かすぎる分割の評価を下げるために導入されたのが，この分割情報量である．分割情報量は以下の式で表すことができる．

$$SI_A(X) = -\sum_{i=1}^{n} \frac{|X_i|}{|X|} \log_2 \frac{|X_i|}{|X|}$$

この情報利得比を分類基準として用いることで，決定木を構築する際に，学習対象となる集合を効率良く分類し，かつ，学習対象となった集合以外にも応用が利く属性が節として選ばれる．

上述の方法で節に入る属性を選んだ後，その節から出て行く枝を生成する．節に決定した属性が取る値の中で，学習対象となる集合内に含まれる値を枝とする．そして，生成された各々の枝に対して，学習対象となる集合から，節に入れた属性が枝に入れた値を取っているデータの集合を割り振る．ある枝に割り振ったデータの集合の中に，クラスが 1 種類しか含まれないとき，その枝はクラスとなり，分類は終了する．

それぞれの節に割り振られたデータを元に，本項で述べたアルゴリズムを再帰的に行うことにより，木全体を構築する．

3.5.2 枝刈り

決定木を構築する際に，学習対象となる集合に微量に含まれるデータに対してまで枝を生成することは，木構造が複雑になる原因となる．そこで，簡潔な木構造にするために行う作業が**枝刈り**である．

枝刈りとは，学習対象となる集合の中で，誤差の範囲とみなせる微量な要素に対して作成された枝を取り除くことで，木全体の構造を簡潔にする手法である．枝刈りを行うことで，学習対象となる集合に過剰に適応させずに，学習対象以外の集合にも応用が利く，分類効率が良い決定木が構築できる．

3.5.3　本研究での利用方法

決定木学習を用いた定常状態の自動定義の際に行う工夫点について述べる．まず，決定木学習を利用する前に行う数値データの離散化について述べ，次に，枝の作成方法，葉の定義方法について述べる．

数値データの離散化

扱う属性が数値データであった場合，枝の生成方法を統一し，木の作成を効率良く行うため，あらかじめ閾値を決定して，離散的な値として扱う．閾値を一定間隔で区切ると，データの分割が過度に行われる可能性があるため，図 3.4 のように，データの平均と標準偏差を用いてデータを分割し，データの余剰な分割を防ぐ．

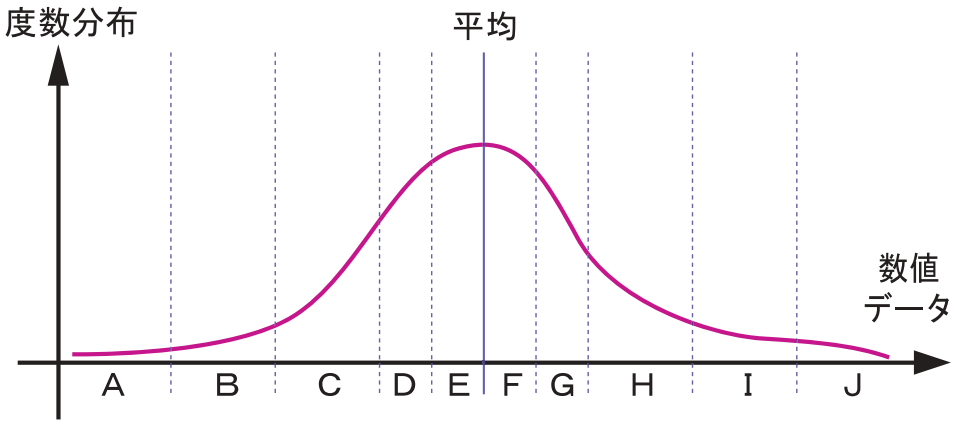

図 3.4: 数値データの離散化の例

未定義の枝

決定木では，分類結果として１つのクラスに到達するように木が構築されるため，分類後の集合の要素数に関係なく，分類後の集合には１つのクラスを定義する．もしくは，要素数が過度に少ない場合は，枝刈りによって，上の節に吸収される．

しかし，定常状態を定義する場合は，必ずしもクラスを定義する必要は無い．そこで，データ数が少ない枝は未定義とすることで，不十分なデータ数による定常状態の定義を避ける．これにより，過度な異常検知を防ぐことができる．

葉の定義手法

上述したように，決定木学習では分類結果として１つのクラスに到達する必要があるため，１つの枝に対して葉は１つしか定義されない．学習を行う過程で，分類に用いる基準を全て使い切った，分類後の集合に複数のクラスが含まれる場合でも，分類結果として１

つのクラスに導く．その手法には，分類後の集合からランダムに要素を選ぶ方法や，１番多く含まれるクラスを選ぶ方法がある．

しかし，定常状態を定義する場合は，必ずしも分類結果として１つのクラスに定義する必要はない．分類後の集合に複数のクラスが含まれるときには，クラス数に大きな偏りが見られる場合を除いて，複数のクラスが定常状態と定義されるべきである．

例として，クラスとしてA，B，C，Dの４種類の値があり，分類を行ったある枝では，A，C，Dの３種類が定常といえる状態を考える．決定木学習では，A，C，Dの３つのうち，１番数が多い値を選ぶ方法や，ランダムに１つを選ぶ方法により，枝として１つの値を決定している．本研究で用いる定常状態の学習では，A，C，Dの３種類の全てを定常状態と定義する．

3.6　異常の検知

本節では，3.5.3 項で述べた定常状態を表現する木の定義方法を用いた，異常の検知手法について述べる．まず，木の作成について述べ，次に，作成した木から異常を判定する方法について述べる．

3.6.1　木の作成

3.5 節で述べたアルゴリズムに従い，定常状態を表す木を構築する．学習対象とする集合は，環境属性の履歴情報であり，履歴情報の中に多く含まれる状態が定常状態として表現される．履歴情報の中には異常状態も含まれていると予想されるが，その情報は集合全体に対して少ないことが予想される．そのため，枝刈りによって削られ，木には影響しない．クラスとする属性が異なることで，定義される定常状態も異なるため，全ての環境属性に対して，それぞれの属性をクラスとした木を構築する．

3.6.2　異常の判定

本項では，前項で述べた手法を用いて定常状態を定義した木に対して，取得した環境属性の値がどのような場合に，定常状態，異常状態と判断するかについて述べる．まず，定常状態と判断する場合について述べ，次に，異常状態と判断する場合について述べる．

定常状態と判定する状態は以下の２種類である．木をたどった結果，クラスに到達し，そのクラスが現在の状態に反していない状態と，木をたどった結果，未定義の枝に到達した状態である．

異常状態と判定する状態は，大きく分けて２つ存在する．１つ目は，木を最後まで辿っていった結果，現在のクラス属性が取る値と定常状態を表す木で定義されているクラスが反した場合である．２つ目は，3.5.2 項で詳しく述べた枝刈りや，前例が無いなどの理由により，木をたどる途中で条件に合う枝が発見できない場合である．

3.7　本章のまとめ

本章では，本研究で行う異常の自動定義に必要な要件を考え，異常状態の自動定義の手法である決定木とその利用法について述べた．

次章では，環境における異常状態の自動定義を行うミドルウェア，SARADA の設計について述べる．

第4章　SARADAの設計

前章では，異常状態の自動検知手法について述べた．本章では，環境の異常を自動定義し，アプリケーションに提供するミドルウェア，SARADA の設計について述べる．まず，設計方針，想定環境，全体の概要について述べる．次に SARADA を構成するそれぞれの機能について述べ，本章をまとめる．

4.1 設計方針

本研究の目的を達成するために必要な，SARADA の設計方針として，プラットフォーム独立性，取得データの抽象化，デバイスの抽象化を挙げる．

- プラットフォーム独立性

  本システムは多様なハードウェア環境で動作する必要がある．そのため，本システムの各モジュールをプラットフォームから独立させる．

- 取得データの抽象化

  センサから得られるデータには様々な種類が存在する．それらのデータを扱う際に，それぞれに解釈を行っていては，対応できるデータの種類や数に限界がある．そのため，本システムではセンサから取得したデータを抽象化し，統一的に扱うことで，様々なデータに対応できる．

- デバイスの抽象化

  ユビキタスコンピューティング環境では，様々なセンサや機器が設置されている．それらのデバイスから環境属性を取得するためには，デバイスそれぞれで異なった処理が必要となるが，それらを抽象化することで，本システムは異なるデバイスを統一的に扱える．これにより，新たなデバイスを組み込む際，システムへの変更を必要としない．

4.2 想定環境

SARADA の動作する想定環境は，家やビルなどの建物内に以下の３つの要素が設置され，それらがネットワークを介して通信できる環境である．これらをまとめて図 4.1 に示す．

- サーバ

  PC やワークステーションなど，SARADA が動作する端末．部屋もしくは建物内に１つ以上存在する．

- センサ

  位置情報，温度，湿度や音量などの環境属性を取得するデバイスである．サーバ上で動作する SARADA からセンサで取得した情報を取得できる．

- 機器

  照明，AV 機器やエアコンなどのデバイスである．環境属性の値として電源の ON/OFF など動作状態を表す情報をサーバ上で動作する SARADA から取得できる．

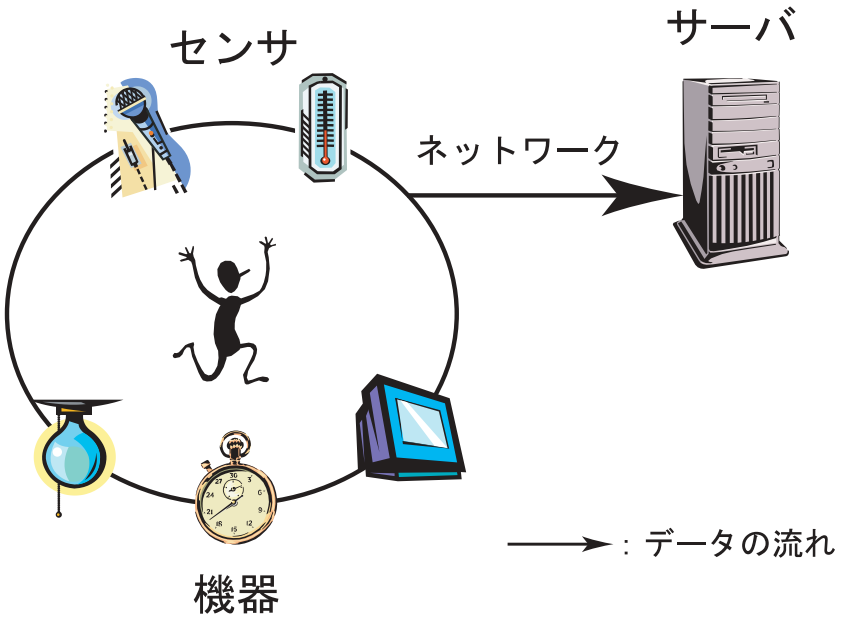

図 4.1: ハードウェア構成図

4.3　ソフトウェア構成

まず，ソフトウェア構成図を図 4.2 に示し，システム全体の概要について述べる．次に，SARADA を構成する５つのモジュールについて，各部の役割を簡単に説明した後，入力データ，出力データ，内部処理について詳しく述べる．

4.3.1　概要

SARADA のモジュールは，データ取得部，履歴管理部，木作成部，判定部，通知部の５つから構成される．これらのモジュールは大きく２つの役割に分かれる．環境属性の履歴情報から定常状態を表現する木を作成する学習フェーズと，定義された定常状態と検知対象となる環境属性から異常状態を判定する検知フェーズがある．以下に，それぞれのフェーズに分けて，各モジュールの概要を説明する．

学習フェーズ

学習フェーズでは，環境属性の履歴データから定常状態を表現する木を自動的に作成する．学習フェーズでは，データ取得部，履歴管理部，木作成部の３つのモジュールが動作する．データ取得部は，センサや機器からデータを取得し環境属性を抽出する．履歴管理部は，履歴として環境属性を表すデータを蓄積し離散化を行う．木作成部は，離散化されたデータを基に定常状態を表す木を作成する．

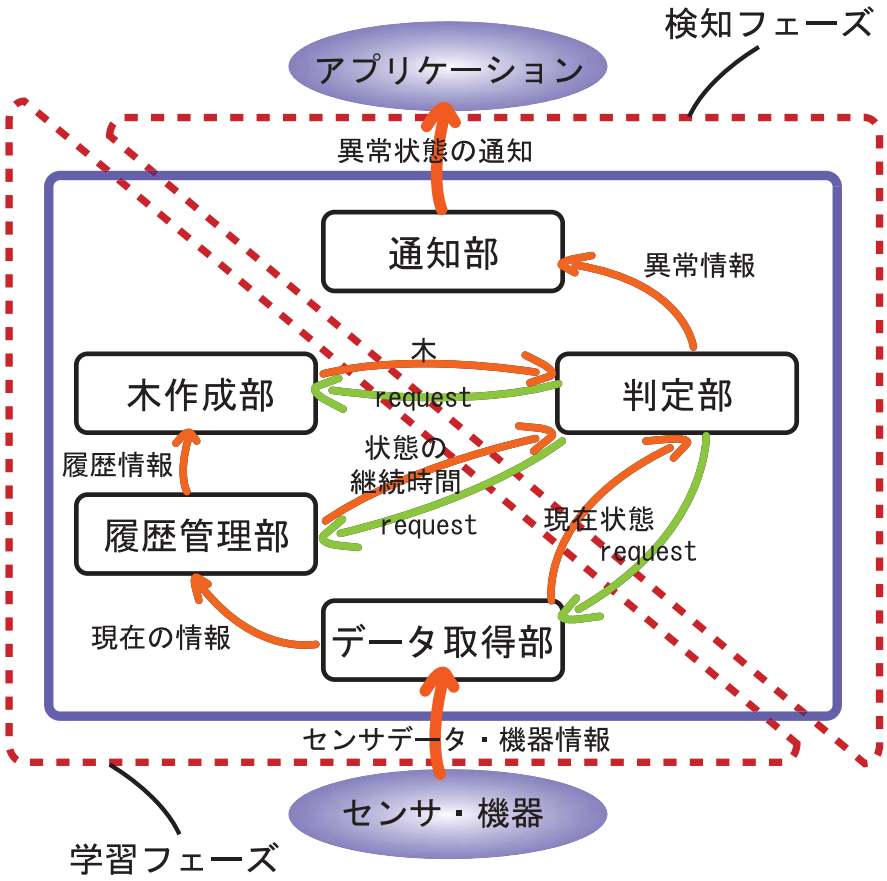

図 4.2: ソフトウェア構成図

検知フェーズ

検知フェーズでは，学習フェーズで定義された定常状態を基に，異常状態を検知する役割を持つ．検知フェーズでは，判定部と通知部の 2 つのモジュールが動作する．判定部は，定常状態を表す木を基に，検知対象となる環境属性から異常を検知する．通知部は，アプリケーションに対して異常情報を通知する．

4.3.2 データ取得部

センサや機器からデータを取得し，履歴管理部に渡す．取得した raw データから環境属性を抽出し，数値データと離散データの 2 種類に分類する．数値データとは，連続的に値が変化し，近い値は近い意味を持つデータである．離散データとは，非連続的に値が変化し，値が異なると意味は全く異なるデータである．

入力データ

入力データは，センサからの raw データである．raw データとは，センサから直接取得するデータであり，本システムが必要とする環境属性以外の情報を含む場合もあるため，必要に応じて，解釈する必要がある．

出力データ

出力データは，環境属性を表すデータである．SARADA では，環境属性を基に異常を判断するため，以降のモジュールでは環境属性を表すデータを扱う．出力データは数値データと離散データの 2 種類である．

内部処理

センサから得られる raw データを環境属性を表すデータへと変換する．その際，データの種類を数値データと離散データとに分類する．

4.3.3 履歴管理部

データ取得部から受け取ったデータを履歴として蓄積し，全てのデータを，木作成部が木を作成しやすい形である離散データに統一する．環境属性がある値を取り続けた時間の情報を抽出し，時間情報を考慮した定常状態の定義を可能にする．

入力データ

入力データとして，情報取得部から環境属性を表すデータが渡される．渡されたデータが数値データならば離散化を行う必要があるため，渡されたデータの種類によって区別して扱う．

出力データ

蓄積された環境属性の履歴データが全て出力データとなる．出力するデータ形式は全て離散データに統一する．環境属性が数値データを取る場合，閾値を設定し離散化した値とする．

内部処理

全てのデータを履歴として保存しておく．木作成部には，離散化済みのデータを渡す必要があるため，情報取得部から渡されたデータが数値データの場合は，3.5.3 項で説明した統計的な手法を用いて離散化する．また，全てのデータに対して，同じ値を取り続けて

いる時間を計算し，その時間の値を一つのデータとして扱う．これにより，一時的な値だけでなく，値の変化や継続時間など，時間を考慮した定常状態の定義が可能になる．

4.3.4　木作成部

履歴管理部から渡される離散化済みの履歴データを用いて，定常状態を表現する木を作成する．

入力データ

入力データとして，初期データの種類に応じた形で離散化された，環境属性の履歴データを受け取る．また同時に，そのデータが継続して取り続けている時間のデータも受け取る．

出力データ

出力データとして，3.4.3 項で述べた，環境属性を用いて定常状態を表現する木が出力される．

内部処理

3.5 節で述べたアルゴリズムを用いて，履歴データを基に定常状態を表現する木を作成する．

4.3.5　判定部

学習フェーズで動作する各モジュールからデータを取得し，異常状態の判定を行う．異常と判定された場合，通知部に異常情報を渡す．

入力データ

入力データとして，学習フェーズで動作する各モジュールのそれぞれからデータを受け取る．以下に，各モジュールから受け取るデータを説明する．データ取得部からは，判定を行う瞬間の環境属性データを受け取る．履歴管理部からは，判定を行う環境属性が取り続けている時間のデータが渡される．木作成部からは，定常状態を表現する木を受け取る．

出力データ

判定の対象となる環境に異常が存在した場合，異常が発生したことを通知部に知らせる．またその際，異常が発生したときの環境属性データと，異常と判定した理由を示す定常状態の木も同時に出力する．

内部処理

定常状態を表現する木を用いて，判定の対象となる環境の環境属性データが異常かどうかを調べる．判定方法となる木のたどり方は，3.6.2 項で述べた．

4.3.6　通知部

アプリケーションに対して，異常を通知する．通知する内容には，異常と判定された理由となる木の情報や統計情報を含める．

入力データ

判定部から，異常情報と判定基準の情報，判定対象となった状態などを入力データとして受け取る．

出力データ

どの環境属性がどのように異常と判定されたかを，アプリケーションが解釈できる形で伝える．判定基準となった規則や，異常と判定された状態を出力データとして渡す．

内部処理

判定部から受け取った，異常と判定されたときの木の状態と判定対象となった環境属性データを，アプリケーションに渡す形式に統一する．

4.4　システムの動作

SARADA の動作について説明する．まず，学習フェーズの動作について説明し，次に，検知フェーズの動作について説明する．

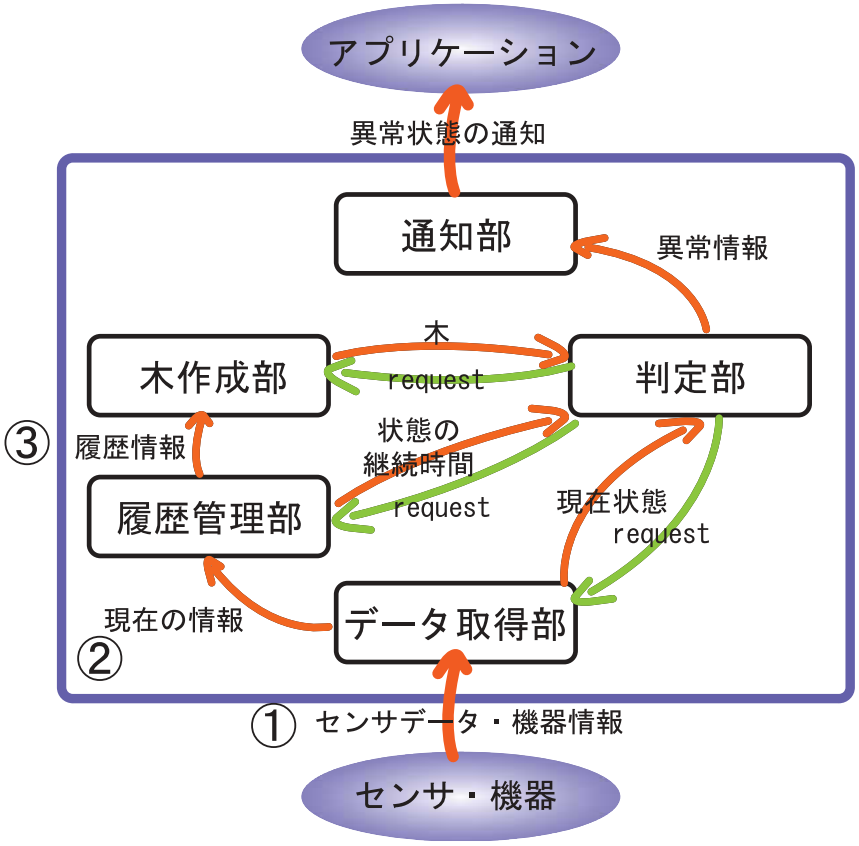

図 4.3: システム動作図：学習フェーズ

4.4.1 学習フェーズ

学習フェーズの動作を図 4.3 に示す．学習フェーズは，SARADA 起動時から常に動作し，センサや機器の状態が更新され，データ取得部がデータを取得するたびに処理を行う．センサや機器の状態更新がイベントドリブンで行える場合は，イベントドリブンでデータを取得し，行えない場合は，ポーリングを行いデータを取得する．取得したデータを環境属性データに加工し，履歴管理部に渡す．

4.4.2 検知フェーズ

検知フェーズの動作を図 4.4 に示す．検知フェーズは，学習フェーズが十分な量の環境属性の履歴データから定常状態を定義してから動作する．十分な量の明確な基準は実際に動作をさせた後，データの信頼性や動作時の誤検知率を考慮して決定する．判定部は，環境属性の値に変化があった場合や，一定時間変化が無い場合に再判定を行う．通知部は，判定部で異常が検知された場合に動作する．

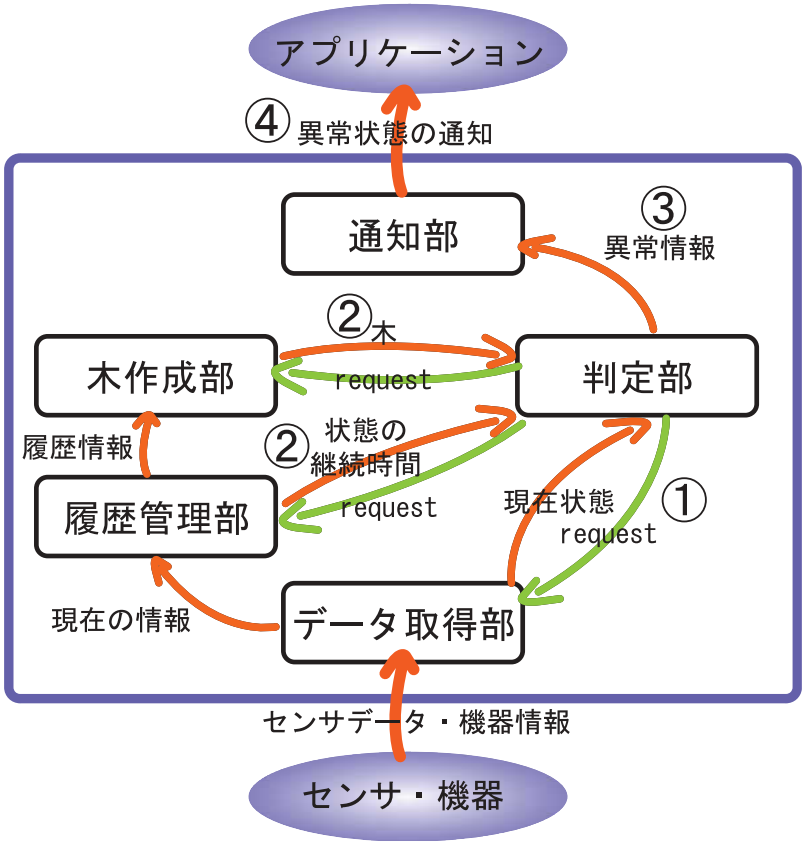

図 4.4: システム動作図：検知フェーズ

4.5　本章のまとめ

本章では，前章で示した自動検知手法を用いて，異常状態を自動定義し，アプリケーションに提供するミドルウェアである SARADA の設計を行った．

次章では，本章で行った設計に基づき，SARADA のプロトタイプ実装について述べる．

第5章　SARADAの実装

前章では，環境の異常を自動定義し，状況監視アプリケーションに提供するミドルウェア，SARADAの設計について述べた．本章では，前章の設計をもとに，SARADAのプロトタイプ実装について述べる．

表 5.1: 実装環境

項目	環境
CPU	Athron XP 2500+
Memory	1GB
OS	Windows2000 Professional
JDK	Java 2 SDK, Standard Edition Version 1.4.2

5.1 実装概要

SARADA のプロトタイプ実装の概要について述べる．学習フェーズのモジュールである，データ取得部，履歴管理部，木作成部を実装した．また，プロトタイプ実装の評価をするために，検知フェーズでは判定部の一部を実装した．実現した機能は，履歴データから異常の判定基準を自動生成する機能と，生成された判定基準から異常を検知する機能である．

5.1.1 実装環境

SARADA は表 5.1 に示す開発環境で実装した．本システムは，4.1 節で述べた，３つの設計方針を実現するために，実装言語として，Java 言語を用いた．その理由として，プラットフォーム非依存性と，抽象クラスの利用が挙げられる．以下に詳細を述べる．

プラットフォーム非依存性

Java 言語では，プラットフォームの差異を JavaVM が吸収し，JavaVM 上で Java プログラムの中間コードが実行される．そのため，JavaVM がインストールされたマシン上であれば，プラットフォームに関わり無く実行可能である．これにより，SARADA はサーバとして利用する端末のプラットフォームに依存せずに動作できる．

抽象クラスの利用

Java 言語では，抽象クラスを作成し，それを継承することで，クラスに予定されている機能と，その実際の処理を分離することが可能である．そのため，共通の性質を持つクラス全体に必要な機能であるが，個々のクラスで異なる処理を行わなければならない機能が容易に実現できる．

```
public abstract class Value {
    protected final Object data;
    protected final Date sensed_time;
    .....

    protected Value(Object d) {
        data = d;
        sensed_time = new Date();
        .....
    }

    //このデータが表現する環境属性の値を返す
    public Object getData() {
        return data;
    }

    //このデータが取得された時間を返す
    public Date getTime() {
        return sensed_time;
    }

    //離散化したデータを返す
    public abstract String getClassificatedData();
    .....
}
```

図 5.1: Value クラス

5.1.2 実装方針

前項で述べた Java の性質を利用して，4.1 節で挙げた，プラットフォーム独立性，取得データの抽象化，デバイスの抽象化を実現する手法について詳細に述べる．

プラットフォーム独立性

前項で述べたように，本システムの実装に Java 言語を用いることで，プラットフォーム独立性を実現した．

取得データの抽象化

前項で述べた，抽象クラスを利用し，データを抽象化した Value クラスを作成した．概要を図 5.1 に示す．システム内部では，センサや機器が取得したデータを Value クラスのオブジェクトとして扱う．これにより，センサや機器が取得したデータを統一的に扱え，システムはデータ非依存に実装できる．

システムが取得するデータに対して必要とするのは，データの値と取得された時間，離

```
public abstract class Device {
    public static final int CONTIGUOUS = 1;
    public static final int DISCRETE   = 2;
    protected final int value_type;

    protected Device(int type, String device_name) {
        value_type = type;
        .....
    }

    //初期設定で指定した型のデータを返す
    public Value getValue() {
        if(value_type == CONTIGUOUS) {
            //型チェックを行う
            if(makeValue() instanceof ContiguousValue)
                return makeValue();
        }
        .....
    }

    //デバイス独自の処理を行う
    protected abstract Value makeValue();
    .....
}
```

図 5.2: Device クラス

散化後の値である．データの値と取得された時間は，全てのデータで同じ処理が可能だが，離散化の処理はデータの種類により手法が異なる．抽象メソッドとして，離散化を行うメソッドを定義することで，離散化はデータの種類に応じて行える．

デバイスの抽象化

前項で述べた，抽象クラスを利用し，デバイスを抽象化した Device クラスを作成した．概要を図 5.2 に示す．システム内部では，環境属性を取得する際に，個々の異なったデバイスを Device クラスとして統一的に扱える．また，デバイスごとにデータ形式が異なるため，数値データ，離散データの２つのどちらを取得するかを初期設定で決定し，設定通りのデータを返すことを保証できる．

個々のデバイスがデータを取得する方法は異なるため，raw データを取得する部分や，それを環境属性を示すデータに変換する部分は，Device クラスを拡張した，個々のデバイスを表すクラスで実装する．

5.2　各部の実装

本節では，SARADA の各モジュールの主要クラスの役割について説明する．

5.2.1　データ取得部

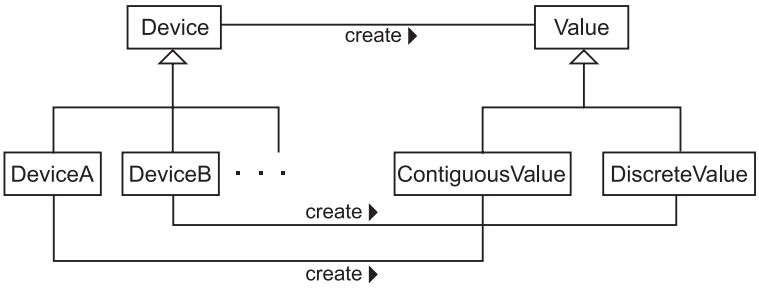

図 5.3: データ取得部のクラス図

図 5.3 に，データ取得部のクラス関係をクラス図で表現する．データ取得部では，5.1.2 項で述べた Device クラスを継承した個々のデバイスを表すクラスから，Value クラスを継承した環境属性を表すデータを取得する．数値データは ContiguousValue クラス，離散データは DiscreteValue クラスとして取得し，履歴管理部に渡す．

5.2.2　履歴管理部

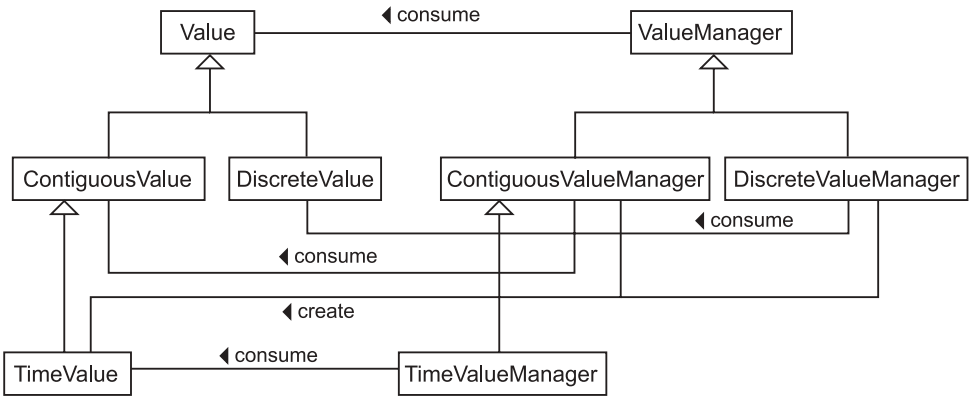

図 5.4: 履歴管理部のクラス図

図 5.4 に，履歴管理部のクラス関係をクラス図で表現する．履歴管理部では，データ取得部から渡される抽象クラスである Value クラスを，同じく抽象クラスである ValueManager クラスで履歴として保持する．ValueManager クラスの概要を図 5.5 に示す．数値データの履歴

```
public abstract class ValueManager {
    //全ての履歴データ
    protected Vector dataset;

    //履歴にデータを加える
    public void addData(Value value) {
        dataset.add(value);
    }

    //全てのデータを離散化するメソッド
    public void classificateValues() {
        for(Enumeration e=dataset.elements();e.hasMoreElements();) {
            classificateValue((Value)e.nextElement());
        }
        .....
    }

    //あるデータを離散化するメソッド
    protected abstract void classificateValue(Value);
    .....
}
```

図 5.5: ValueManager クラス

は ContiguousValueManager クラスで管理され，離散データの履歴は DiscreteValueManager クラスで管理される．

また，データを蓄積する際に ContiguousValue と DiscreteValue のそれぞれに対して，連続してある値を取っている時間の長さを表す TimeValue を生成した．時間の長さは数値データなので，TimeValue は ContiguousData を継承し，ContiguousValue と同じ手法により離散化される．

木作成部で扱うデータは全て離散化済みである必要があるため，データ蓄積後，データ形式に応じて離散化し，木作成部に渡す．DiscreteValue はそのまま渡し，ContiguousValue は図 3.4 に示した，統計を用いた離散化を行った．

5.2.3 木作成部

木作成部では，履歴管理部で離散化が行われたため，全ての Value に対して getClassifiedData メソッドを呼ぶことにより，全てのデータから離散化した値を取得できる．したがって，木作成部では Value クラスを継承した全てのクラスを Value クラスとして統一して扱える．これらのデータを決定木アルゴリズムを実装した TreeProducer クラスに渡すことで，定常状態を表現する木を作成する．作成した木は Tree クラスで表現され，TreeManager クラスに登録に登録する．

5.2.4　判定部

判定部は，Device クラスからのデータ更新があるたびに呼び出され，Device クラスから Value クラスを取得し，ValueManager クラスから TimeValue クラスと Value クラスを離散化する情報を取得し，TreeManager クラスから，Tree クラスを取得する．それぞれの取得した情報から，Value クラスが表現する現在の環境属性が定常か異常かを判定する．

5.3　本章のまとめ

本章では，異常状態を自動的に定義し検知する，SARADA のプロトタイプ実装について述べた．まず，実装の概要について述べ，次に各モジュールについて述べた．

次章では，SARADA の定量的評価，定性的評価を行う．

第6章　SARADAの評価

本章では，異常状態を自動検知するミドルウェアSARADAを評価する．判定基準の自動定義機能について，まず定性的に評価し，次に定量的に評価する．

6.1 定性的評価

本節では，異常判定基準の手動定義と SARADA とを定性的に比較する．手動定義とは，センサや機器などのデバイスに対してユーザやアプリケーション開発者が適切なデバイスを選択し，判定基準を手動で設定する手法である．比較する項目は，環境適応性，定義記述の柔軟性，要求応答性と導入簡易性である．比較の結果を表 6.1.4 に示す．

6.1.1 環境適応性

SARADA は，センサや機器からの情報の履歴を用いて環境に応じた判定基準を設定することが可能である．手動定義する手法では，人間が判断することで，環境に応じた判定基準の設定が可能である．しかし，あらかじめ設定する環境について熟知している必要がある．

6.1.2 定義記述の柔軟性

SARADA では，決定木を応用した手法を取ることで，人間にとって理解可能な記述が可能である．手動定義では，設定を行っているのは人間であるため，定義記述の柔軟性を満たしている．

6.1.3 要求応答性

要求応答性とは，ユーザの要求に応じた異常を検知可能かどうかである．SARADA では自動生成された定常状態の定義に反した状態が，ユーザの要求に合う保障は無い．しかし，異常が検知されたときの判定基準や異常の原因を特定することにより，アプリケーションでフィルタリングをかけることが可能である．手動定義では，ユーザの要求を明示的に記述できるため，要求応答性を満たす．

6.1.4 導入簡易性

導入簡易性とは，状況監視アプリケーション導入が容易であることを指す．SARADA では，環境に既存のセンサや機器に対して，システムが自動的に判定基準を設定するため，状況監視アプリケーション導入の際に手間がかからない．手動定義では，既存のセンサや機器を利用するために，判定基準の設定を行う必要があるため，状況監視アプリケーションの導入に手間がかかる．

表 6.1: 異常状態の定義手法の比較

比較項目	SARADA	手動設定
環境適応性	○	△
定義記述の柔軟性	○	○
要求応答性	△	○
導入簡易性	○	×

6.2 定量的評価

本節では，SARADA に対する定量的評価を行う．まず，評価実験について説明する．次に，定常状態を表現する木が安定するまでの時間を評価し，最後に，木を作成するまでの計算時間を評価する．

6.2.1 評価実験

SARADA の評価実験を行った．まず，実験を行った環境について説明し，次に，実験方法について説明する．

実験環境

図 6.1: ミーティング中の SSLab の風景

SARADA の動作実験を慶應義塾大学徳田研究室の実験施設である，SSLab[21] で行った．SSLab には多様なセンサや機器が設置されており，多種の環境属性を取得できる．ま

表 6.2: 使用したセンサや機器と環境属性

センサや機器	通信方法	環境属性
DA100	RS-232C	室内の温度
WebCamera	USB	照度の変化
RF-Code	100BASE-TX	タグ ID
照明	100BASE-TX	照明の状態

た，SSLab では日常的に会議や実験などの活動が行われている (図 6.1)．そのため，SSLab は SARADA の測定を行う環境として適切である．

使用機材

使用したデバイス，通信方法，扱った環境属性を表 6.2.1 に示す．DA100 は温度センサであり，環境属性として部屋の温度が取得できる．WebCamera は照度の変化を取得するために設置した．RF-Code[20] は，ユーザの入退室情報を取得するために用いた．部屋の照明は，ON と OFF の状態を環境属性として取得した．

定義された木の例

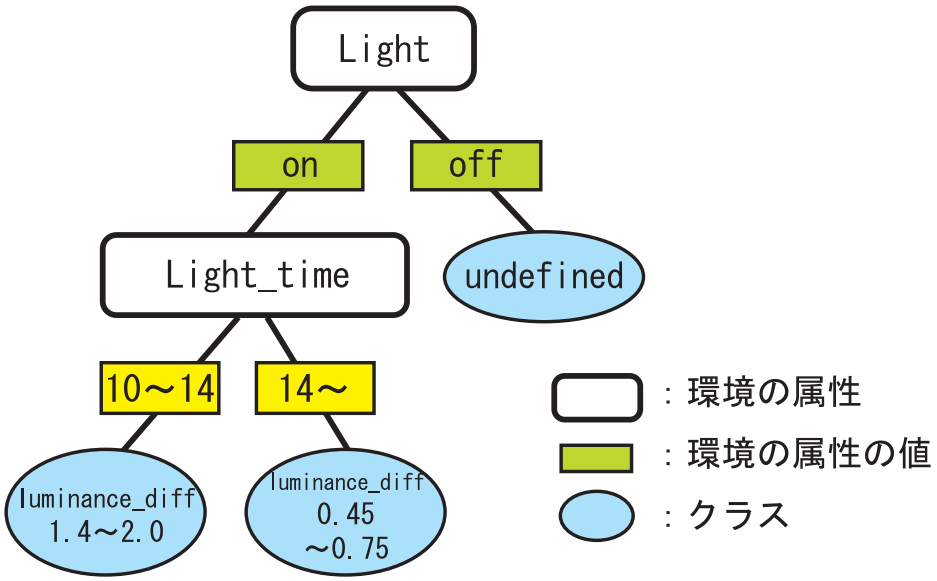

図 6.2: SARADA によって定義された木

定常状態を表現する木の一つを図 6.2 に示す．この木は，カメラによって得られる明るさの差分をクラスとして作成した木である．この木は，ライトが ON になってから 10〜

14 秒後はカメラによって得られる明るさの差分の値が 1.4〜2.0 であり，14 秒以降ではその値が 0.45〜0.75 であれば定常状態であることを示している．

このカメラが設置されていた場所は，ライトの光が届く場所を写していた．そのため，ライトが ON になることによって，明るさの変化に規則性が見られたことで，ライトが ON の木の分岐ができたと考えられる．ライトが OFF の場合には，夜は明るさの変化はほとんど認められないが，昼は日光により明るさが変化する．そのため，明るさの変化にライトからの規則性が見られず，今回使用したセンサや機器からは定常状態が定義することができなかったと考えられる．

6.2.2 定常状態の定義にかかる時間

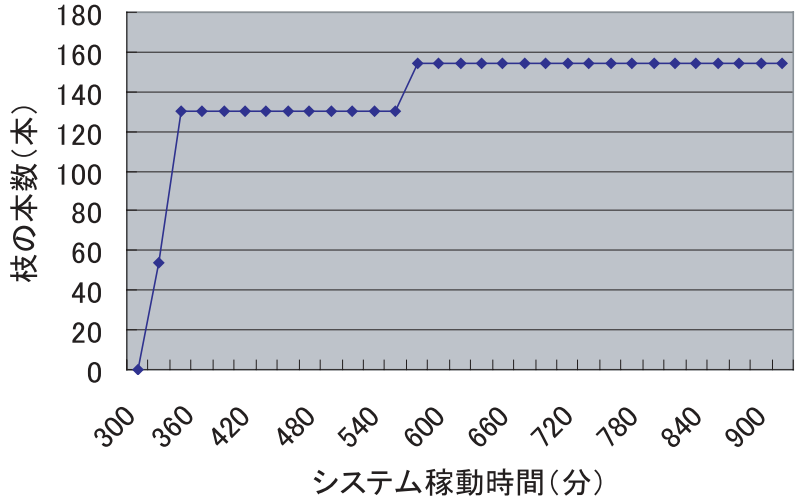

図 6.3: 定常状態を表現する木が持つ枝の本数の推移

SARADA が定常状態を表現する木を作成し始め，定常状態を定義するまでの時間を評価する．図 6.3 は，横軸に SARADA が木の作成を始めてからの経過時間，縦軸に定義した定常状態を表す木に含まれる枝の本数を表したグラフである．

木の作成が始まってから 300 分程度で枝の本数は 154 本となり，その後は，2 日後に計測を終えるまで枝の数に変化は見られなかった．これにより，定常状態が収束することが分かる．

6.2.3 木作成にかかる計算時間の評価

SARADA が木を作成する計算時間について評価する．図 6.4 は，横軸に SARADA が木の作成に利用したデータ量，縦軸に定常状態を表す木を定義するために必要とした時間を表したグラフである．

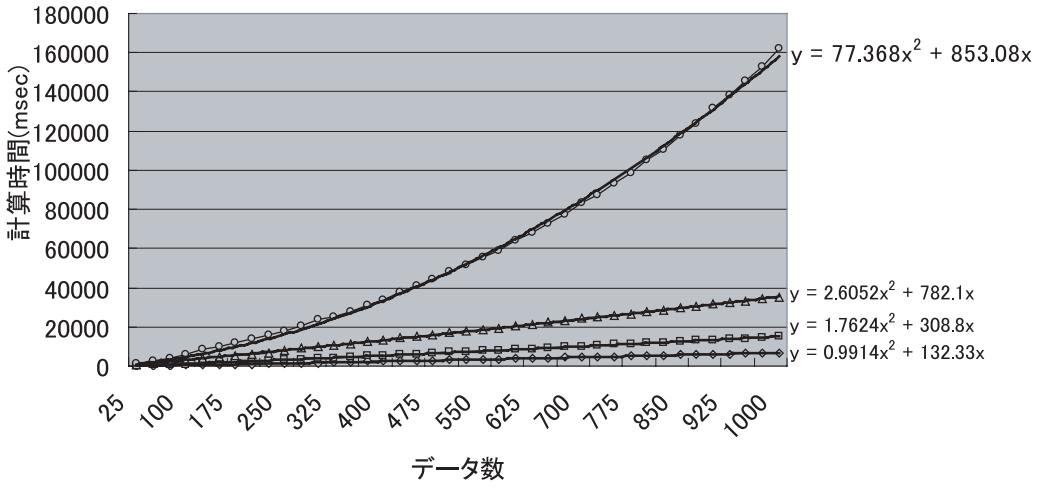

図 6.4: 木作成にかかる時間

４つのグラフは，それぞれ木の作成に用いた環境属性の数が異なるグラフである．上からそれぞれ，26，20，14，10 種類の環境属性を利用したグラフを指す．このグラフを，データの数を独立変数，計算時間を従属変数とした関数と考えると，２次関数で多項式近似が可能である．これは，木の構造が階層構造を取るため，分岐の数だけ処理が複雑になることが理由として挙げられる．

しかし，理論上は分岐の数が指数的に増えると予想される．例として，１階層で平均 N 個の分岐をする場合，分岐数の期待値は，２階層までで N 個の分岐が行われるため N 回，３階層までで，２階層のそれぞれの枝で N 個の分岐が行われるため N^2 回，４階層まででは N^3 回と増えてゆく．

今回の評価で得られた結果の理由として，実際に構築される木が，先に図 6.2 に示したように，未定義の枝が存在したり途中でクラスに到達して終端となることが考えられる．しかし，指数関数と比べると計算量が少ない２次関数とはいえ，データ量が増えた場合に計算量が膨大になることは変わらない．今後，定常状態を損なわないまま履歴からデータを間引く方法や，複数のデータを統一して１つとして扱う方法について検討する．

また，デバイス数が増えた場合にも計算量が増えることがグラフから読み取れる．先ほどと同様に，デバイスの数を独立変数，計算時間を従属変数とした関数と考えると，２次関数で多項式近似が可能である．データ量が同じ場合，作成される木の深度は変わらないと仮定すると，この計算量は理論値と一致する．これは，デバイスの数を N 個とすると，環境属性の数と同じ値であるデバイスの数だけ木が作成される計算量が N に比例し，木を構築する際の分割属性の候補として増える計算量が N に比例するためである．

デバイス数が増えた場合も計算量が膨大になることが予想されるため，今後，意味的に近い環境属性の値を同一に扱う方法を考慮する [29].

6.3　本章のまとめ

本章では，SARADA の定性的評価，定量的評価を行った．定性的評価では，手動設定に対する優位性を示した．定量的評価では，定常状態を表現する木を作成する際に木が安定する時間，木の構築にかかる計算時間に関して考察した．

次章では，今後の課題について述べ，本論文をまとめる．

第7章　結論

本論文では，環境の異常状態を自動定義することにより，状況監視アプリケーションを支援するミドルウェア，SARADAを設計し実装した．本章では今後の課題を挙げ，最後に本論文をまとめる．

7.1　今後の課題

本節では，SARADA が今後考慮すべき課題として，離散化の手法と定常状態の変化への再対応を挙げる．

7.1.1　離散化の手法

今回，SARADA では連続値の離散化に統計的手法を用いた．この方法は，データの平均値が分割の閾値に与える影響が大きいため，データの分布状態によっては閾値の設定が粗過ぎる場合があった．連続値の離散化には様々な手法 [6] が存在するため，今後，それらの計算量や分割数などを考慮し，SARADA への応用を検討する．

7.1.2　定常状態の変化への再対応

定常状態が定義されてから，センサや機器の位置の変化や住人の生活規則の変化により，定常状態が変化した場合，再定義する必要がある．今回，SARADA では古い履歴と新しい履歴を区別せずに扱うため，定常状態を定義し直すには時間がかかる．今後，時間によるデータのクラスタリングや情報の新しさによる重みを考慮し，再定義にかかる時間の短縮を目指す．その際，複数の決定木を作成し，多数決を取るコミッティ学習などを検討している．

7.2　本論文のまとめ

本稿では，ユビキタスコンピューティング環境において，状況監視アプリケーションを支援するミドルウェア，SARADA の実装，評価を行った．SARADA は決定木学習アルゴリズムを応用し，異常状態の判断基準を自動的に定義することで，状況監視アプリケーションの支援を実現する．本論文では，評価として他の定義手法と SARADA との機能比較，判定基準の設定に必要な時間の計測を行い，他の定義手法に対して導入時の負担が軽減することを示した．

謝辞

本研究の機会を与えてくださり，御指導を賜りました慶応義塾大学環境情報学部教授徳田英幸博士に深く感謝致します．また，重要な御助言を頂きました，慶應義塾大学政策・メディア研究科助教授高汐一紀博士の御指導に深く感謝致します．

慶応義塾大学徳田・村井・楠本・中村・南合同研究会の先輩方には折りにふれ貴重な示唆や御助言，御指導を頂きました．特に，徳田研究室の先生方や先輩方，ACE(Active Computing Environmets) 研究グループ，move!研究グループの方々に深く感謝いたします．また，慶應義塾大学政策・メディア研究科修士課程１年の村上朝一氏，同田丸修平氏には，絶えざる励ましや丁寧な御指導をを賜りました．ここに深い感謝の念を表します．また，永田智大氏，岩谷晶子氏，中西健一氏，守分滋氏，榊原寛氏，米沢拓郎氏，小泉健吾氏，大澤亮氏，松倉友樹氏の協力と心遣いに感謝致します．

最後に，研究の日々を共に過ごした，幸田拓耶氏，丸山大佑氏，佐川昭宏氏その他多くの友人に深く感謝し，謝辞と致します．

平成 15 年 12 月 29 日
出内 将夫

参考文献

[1] Aplix Corporation. Jblend.
http://www.jblend.com/.

[2] ZOJIRUSHI CORPORATION. i-pot の特長.
http://www.mimamori.net/service/03.html.

[3] ZOJIRUSHI CORPORATION. みまもりほっとライン.
http://www.mimamori.net/.

[4] ZOJIRUSHI CORPORATION. 象印マホービン株式会社.
http://www.zojirushi.co.jp/.

[5] K. Rommelse D. Heckerman, J. Breese. Decision-theoretic troubleshooting. In *CACM*, pp. 49–57, 1995.

[6] James Dougherty, Ron Kohavi, and Mehran Sahami. Supervised and unsupervised discretization of continuous features. In *International Conference on Machine Learning*, pp. 194–202, 1995.

[7] Chikio Hayashi. The quantitative study of national character: Interchronological and international perspectives. In *International Journal of Cultural Studies*, pp. 91–114, 1998.

[8] J.J. Hopfield and D.W. Tank. Neural computation of decisions opti- mization problems. In *Biological Cybernetics*, pp. 141–152, 1985.

[9] Cory D. Kidd, Robert Orr, Gregory D. Abowd, Christopher G. Atkeson, Irfan A. Essa, Blair MacIntyre, Elizabeth D. Mynatt, Thad Starner, and Wendy Newstetter. The aware home: A living laboratory for ubiquitous computing research. In *Cooperative Buildings*, pp. 191–198, 1999.

[10] UAH Earth System Science Lab. Sensor modeling language.
http://stromboli.nsstc.uah.edu/SensorML/.

[11] Linda B. Lankewicz, Radhakrishnan Srikanth, and Roy George. Anomaly detection using signal processing and neural nets. In *ONDCP International Technology Symposium Proceedings*, 1997.

[12] David Marchette. A statistical method for profiling network traffic. In *First USENIX Workshop on Intrusion Detection and Network Monitoring*, pp. 119–128, Santa Clara, CA, April 9–12, 1999.

[13] Ltd. Matsushita Electric Works. 松下電工.
http://www.mew.co.jp/.

[14] Michael C. Mozer. The neural network house: An environment that adapts to its inhabitants. In *Proceedings of the American Association for Artificial Intelligence Spring Symposium on Intelligent Environments*, pp. 110–114, 1998.

[15] Jin Nakazawa, Tadashi Okoshi, Masahiro Mochizuki, Yoshito Tobe, and Hideyuki Tokuda. Vna: An object model for virtual network appliances. In *Proceedings of IEEE International Conference on Consumer Electronics (ICCE2000)*, pp. ??–??, 6 2000.

[16] J. Ross Quinlan. Discovering rules from large collections of examples: a case study. In *Expert Systems in the Microelectronic Age*. Edinburgh University Press, 1979.

[17] J. Ross Quinlan. Induction of decision trees. In *Machine Learning vol.1*, pp. 81–106. Kluwer Academic Press, 1986.

[18] Ltd. SECOM Co. Secom town ホームセキュリティについて.
http://www.secomtown.com/hs/about/m_gaiyo.asp.

[19] Claude E. Shannon. A mathematical theory of communication. In *The Bell System Technical Journal, 27*, pp. 379–423. Edinburgh University Press, 1948.

[20] InData Systems. Rf code spider rfid system.
http://www.indatasys.comhtml/products.

[21] T.Okoshi, S.wakayama, Y.Sugita, S.Aoki, T.Iwamoto, J.Nakazawa, D.Furusaka, M.Iwai, A.Kusumoto, N.Harashima, J.Yura, N.Nishio, Y.Tobe, Y.Ikeda, and H.Tokuda. Smart space laboratory project: Toward the next generation computing environment. In *Proceedings of IEEE International Workshop on Networked Appliances(IWNA)*, pp. ??–, 2 2001.

[22] M Weiser. The computer for the 21st century. In *Scientific American 256(3),94–104, September 1991*.

[23] R. J. Wilson. グラフ理論. 近代科学社, 1992.

[24] 松田啓史, 山口彰一, 荒川忠洋. 高齢者生活行動モニタリングシステム. Technical Report 82, Matsushita Electric Works, Ltd., 8 2003.

[25] 警察庁. 犯罪統計資料　平成１４年１月～１２月分（前年同期比較）, 2003.
http://www.npa.go.jp/toukei/keiji2/1-12.pdf.

[26] 古川康一, 尾崎知伸, 植野研. 帰納論理プログラミング. 共立出版, 2001.

[27] 国立社会保障・人口問題研究所. 人口統計資料集 (2003 年版) 表７－17　世帯構造別 65 歳以上のいる世帯数：1975〜2001 年, 2003. http://www1.ipss.go.jp/tohkei/Popular/Popular_f.html.

[28] 古川聡, 松田啓史, 萩尾健一, 谷口良, 筒井譲二, 田中智幸. 「ケアモニタ」向けセンサの検知技術. Technical Report 73, Matsushita Electric Works, Ltd., 2 2001. 16-22.

[29] 高田敏弘, 青柳滋己, 栗原聡, 光来健一, 清水奨, 廣津登志夫, 福田健介, 菅原俊治. Proximity mining: センサデータ履歴からの近接性の発見. ユビキタスコンピューティングシステム研究報告, 第 1 巻, 4 2003.