修士論文 **2003** 年度 **(**平成 **15** 年度**)**

コンテキストアウェアコンピューティング環境におけるコンテキストの研究

慶應義塾大学大学院 政策・メディア研究科

堀江 裕隆

修士論文要旨 2003 年度 (平成 15 年度)

コンテキストアウェアコンピューティング環境における コンテキストの研究

論文要旨

近年の技術革新により，情報家電機器がユビキタスコンピューティング環境で活躍するようになり，それらを制御することが困難となってきた．しかし，人やその周囲の情報を検知するセンサ技術の発達により，人の行動に応じて機器を適応的に動かすコンテキストアウェアアプリケーションの研究が盛んに行なわれて始めた．

コンテキストアウェアアプリケーションは様々な研究者によって研究され，コンテキスト自体が何を示した情報で，どのように表記し，検知すべきか議論されている．

そこで本研究はコンテキストをユビキタスコンピューティング環境で扱うコンテキストアウェアコンピューティング環境を提案し，この環境を構築するために必要なフレームワークを提案する．

このフレームワークを用いることで，異なる利用者が作成したコンテキストの検知手法を，他の利用者がコンテキストをメタ情報を比較することで，評価しながら使えるようになった．また，コンテキストに登録したアプリケーションサービスの挙動が矛盾した場合に，それを動かしたコンテキストに問題がある事を検知し，修正できるようにした．

コンテキストアウェアコンピューティング環境のためのフレームワークに基き運用するコンテキスト検知機構を設計しこれを活用するために必要なコンテキスト情報と XML を用いたディレクトリサービスを容易した．そしてコンテキスト検知機構の中枢であるコンテキスト変換機構と事象解析メカニズムを実装した．

本研究で検討されたフレームワークを用いることで，他の利用者が作成したコンテキストを共有しコンテキストアウェアコンピューティング環境を利用できるようになった．また，そのコンテキストが利用者の思いと矛盾が生じた場合そのコンテキストを必要としない利用者が選別して使わずに済むようになった．これを実際にコンテキスト情報を用いて構築することで示した．

コンテキストアウェアコンピューティング環境は，本フレームワークを用いて構築すると，利用者間でコンテキストの検知手法を共有できるようになった．

キーワード :
1 コンテキストアウェアアプリケーション, 2 コンテキストアウェアコンピューティング環境, 3 コンテキスト，4 状況認知

慶應義塾大学大学院 政策・メディア研究科

堀江 裕隆

Abstract of Master's Thesis
Academic Year 2003

Study of the Context in Context-Aware Computing Environment

Summary

Rapid development of Ubiquitous Computing Environment has brought an active production of diverse information appliances to the world. This improvement brought us not only the ability to control numerous appliances from the Internet, but also the complexity of handling numerous types of devices. Therefore improvement of sensor has technology accelerated and created a concept of matching the humans context to activate his/her numerous devices, which we now call Context-Aware application.

Context-Aware Application are studied all around the world and researcher are all trying to find out how context could be produced, be written, and can capture.

In this thesis, we defined the Context-Aware Computing Environment, where Context-Aware Application's functions goes ubiquitous in Ubiquitous Computing Environment. We will create a framework for using this environment to allow multiple user to create and share the context for his or her own use. This is done by allowing a Context Information to be quarried to Context Information Transition Engine which directs the information queried to Context Analyzing Engine to analyze the data capturing from Data Management Engine.

The Context Information Transition Engine stores Context Information which were successfully changed to context. This process is required since other user, who is unfamiliar with crating the context capturing rule may search the Context Information Transition Engine to find the context capturing rule of his/her request.

By creating the framework for Context-Aware Computing Environment and implementing the Context Information Transition Engine, users have started to receive an option to share the context capturing method.

Keywords :
1 Context-Aware Application, 2 Context-Aware Computing Environment,
3 Context, 4 Environment Acknowledgment

Keio University Graduate School of Media and Governance

Hirotaka Horie

目次

第 1 章　序論 **1**

1.1 研究背景 2

1.1.1 ユビキタスコンピューティング環境に対応した情報家電機器 . . . 2

1.1.2 センサ技術の現状 3

1.2 本研究の目的 5

1.3 本論文の構成 5

第 2 章　コンテキストアウェアコンピューティング環境 **6**

2.1 コンテキストアウェアコンピューティング環境 7

2.1.1 コンテキスト 7

2.1.2 アウェアアプリケーション 8

2.1.3 コンテキストアウェアアプリケーション 8

2.1.4 C-AApp のシナリオ 10

2.1.5 C-AComp 環境の概要 11

2.1.6 コンテキストアウェアコンピューティング環境のまとめ 13

2.2 関連研究 13

2.2.1 TEA 14

2.2.2 C-MAP 14

2.2.3 Personal Positioning System 14

2.2.4 Context Information Service 15

2.2.5 Context-Toolkit 15

2.2.6 Life Patterns 16

2.2.7 SensorML 16

2.2.8 関連研究のまとめ 16

2.3 C-AComp 環境における課題 17

2.3.1 機構と機器の発見 17

2.3.2 収集するデータ 18

2.3.3 再検知できる事象解析手法の考案 19

2.3.4 コンテキストの抽象化 20

2.3.5 コンテキスト検知機構の評価 21

2.3.6 コンテキストと AS 22

2.3.7 C-AComp 環境における課題のまとめ 22

2.4 C-AComp 環境のまとめ 22

第 3 章 C-AComp 環境のためのフレームワーク 24
3.1 C-AComp 環境の為のフレームワークの概要 25
3.2 ディレクトリサービス 26
3.3 データ管理機構 28
3.4 コンテキスト検知機構 30
3.4.1 事象解析メカニズム 30
3.4.2 コンテキスト変換機構 31
3.4.3 コンテキスト検知機構のまとめ 35
3.5 コンテキスト-欲求バインド機構 35
3.6 C-AComp 環境の為のフレームワークに対するまとめ 36

第 4 章 コンテキスト検知機構 37
4.1 コンテキスト検知機構の構築までの想定環境 38
4.2 コンテキスト情報 39
4.2.1 コンテキスト情報:ContextInformation 39
4.2.2 ディレクトリサービス:DirectoryServiceInformation 41
4.2.3 データ管理機構:DataManagementEngine 42
4.2.4 事象解析メカニズム:ContextAnalyzingEngine 43
4.2.5 コンテキスト検知のルール:contextCapRule 43
4.2.6 コンテキスト:contextMeaning 45
4.2.7 コンテキスト情報のまとめ 45
4.3 XML ディレクトリサービス 46
4.3.1 他のディレクトリサービスとの連携 46
4.3.2 データ管理機構との連携 47
4.3.3 事象解析メカニズムとの連携 47
4.3.4 コンテキスト変換機構との連携 47
4.3.5 ディレクトリサービスとの通信プロトコル 48
4.3.6 XML ディレクトリサービスのまとめ 49
4.4 コンテキスト変換機構 CITE:
Context Information Transition Engine 49
4.4.1 CITE の概要 50
4.4.2 コンテキストの検知要求 50
4.4.3 コンテキストの検知成立 52
4.4.4 コンテキストの検索 52
4.4.5 コンテキスト変換機構 CITE のまとめ 52
4.5 事象解析メカニズム:SPICeS 53
4.5.1 パターン解析による事象解析 53
4.5.2 スライディングタイムウィンドウを用いた特徴抽出 54

4.5.3 事象解析メカニズム SPICeS の実装例 55
4.5.4 事象解析メカニズム:SPICeS のまとめ 56
4.6 事象解析メカニズム:TinyDBClient 56

第 5 章 評価 **58**
5.1 定性的評価 59

第 6 章 結論 **62**
6.1 今後の課題及び展望 63
6.1.1 コンテキスト-欲求バインド機構との連携 63
6.1.2 共通のスキーマ 63
6.1.3 各種ルールの自動生成 63
6.1.4 より高度な事象解析メカニズムの検討 63
6.1.5 CITE の出力する情報からサブコンテキストの生成 64
6.1.6 今後の課題及び展望のまとめ 64
6.2 まとめ 64

付 録 A 付録 **68**
A.1 コンテキスト情報 69

図目次

1.1 ユビキタスコンピューティング環境と情報家電機器の図 3
1.2 無線機能を有した小型センサ:Berkley Mote 4

2.1 コンテキストアウェアアプリケーションの図 8
2.2 C-AComp 環境の図 13
2.3 「人が椅子に座る」コンテキストの検知 21

3.1 C-AComp 環境における各種機構と利用者の位置付けの図 25

4.1 特徴のある数値情報の推移 54
4.2 スライディングタイムウィンドウによるデータの管理 54

表 目 次

4.1 名称略称 48

5.1 定性的評価 60

第1章

序論

本章では，本研究の意義および本論文の内容構成について述べる．

1.1 研究背景

近年，インターネットの基盤整備が行われたため，インターネットはどこでも使えるようになった．これにより，人間だけで無く，機器自体もいつでもどこでも相互に通信できる環境を手に入れた．市場では，この通信基盤を前提とした情報家電機器の製造が始まり，多種で多量の機器が遠隔から同時に操作できるようになった．ネットワークに対応した機器は増え続ける一方で，それらの制御方法は複雑さを増している．そのため，このままでは使いたい家電機器の使いたい機能を探すだけで疲労し，自由に動かすことができなくなってしまう．そこで近年では，これらの機器を利用者の環境に応じて自動的に動作させるコンテキストアウェアアプリケーションの研究が注目を浴びている．

本節では，このコンテキストアウェアアプリケーションが動作させうる情報家電機器と，動作内容を判断する情報の収集技術の現状を解説する．

1.1.1 ユビキタスコンピューティング環境に対応した情報家電機器

近年の爆発的なインターネットの普及により，パーソナルコンピュータの利用が一般的となった．また，コンピュータはIEEE1394やBluetoothと言った家電機器との接続口を用意し始め，インターネットと接続できるコンピュータと繋がるようになった．接続したコンピュータを通し家電機器はインターネットと接続できるようになり，単体でも接続されるようになった．このようにインターネットと接続する機器を情報家電機器と呼び多くの製品が製造され販売され始めた．これらの具体的な例を下記に述べる．

- 東芝 FEMINITY シリーズ[1]
  - エアコン
  - 冷蔵庫
  - オーブンレンジ
  - ホームラウンドリー
  - 照明
- ネットワーク対応 HD ビデオレコーダ[2]
- ネットワークハンディカム[3]
- 携帯電話

これらの機器はインターネットに接続され，いつでもどこでも，それらの持つ情報を閲覧したり，またそれらの操作が可能となった．これら情報家電機器と利用者の関係を図 1.1 に示し解説する．

[1]東芝 IT アクセスポイントを経由してインターネットに接続
[2]SONY Cocoon
[3]SONY Network HandyCam

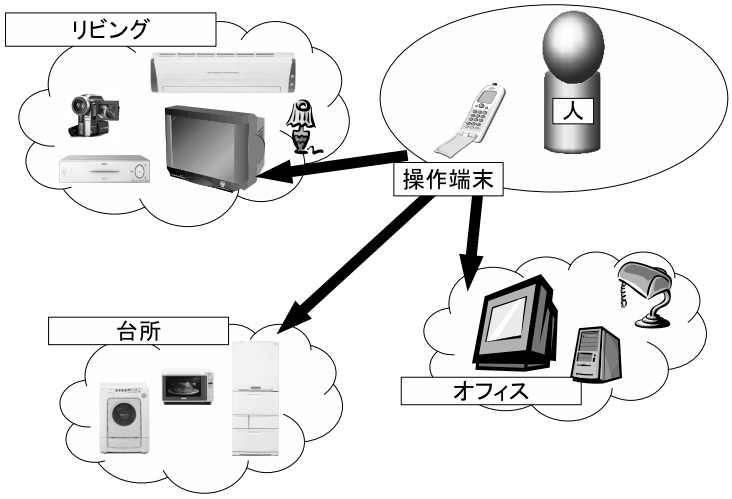

図 1.1: ユビキタスコンピューティング環境と情報家電機器の図

利用者は情報家電機器と通信を行うため，インターネットに接続できる通信端末を利用する．この端末は携帯電話や PDA と言った小型無線端末であっても，パーソナルコンピュータであっても構わない．そして，この端末からインターネットを通して情報家電を操作，もしくはそれが持つ情報の取得を行う．具体的には，冷蔵庫から内部に保存している牛乳の賞味期限を得る事や，エアコンの温度調節を遠隔から行う事や，また，HD ビデオレコーダあればビデオ予約とその確認が行える．

このように，利用者が持つ操作端末がインターネットにさえ接続させれば，いつでもどこでも様々な家電機器を利用できるようになった．

1.1.2 センサ技術の現状

現在，周囲の情報を収集するために様々なセンサが存在する．周囲の熱や水分を計るだけのセンサや，機器に埋めこまれた情報とその位置を検知するセンサ，または顔の特徴を検知するものまで作成された．また，センサが通信機能を持ち，インターネットへ接続し，他の機器と情報をやり取りできるようになった．このように，センサからの情報収集技術は革新してきた．そこで，本部ではネットワーク対応の小型センサ技術と，複雑な情報を入手する強力な識別センサ技術について具体例を述べ解説する．

ネットワーク対応の小型センサ技術

センサの出力するデータをインターネットで利用するために，Smart-IT [14] や Berkley-Mote [9] と言った小型無線通信基盤が開発された．これら小型センサは独自の通信規格を用いて情報収集し，コンピュータた繋がる事でその情報をインターネットへと配信できる．これら自身が小型の計算機であるため，それらには電力の許す限りセンサを乗せ

られる．このセンサのうち，Berkley Mote を図 1.2 に示す．これには 温度，照度，音量，

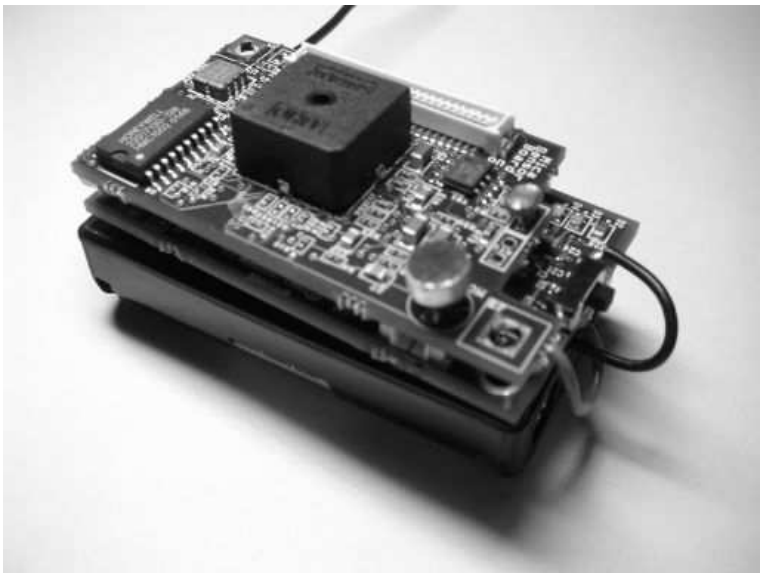

図 1.2: 無線機能を有した小型センサ:Berkley Mote

加速度，磁気を計測するセンサが搭載されている．上部に取りつけられた基盤を変更することで上述した以外のセンサも搭載できる．

このように，小型センサを用いること環境からの情報を収集し，インターネットへ流すことができるようになった．これにより，環境に対する情報を容易に収集し解析できるようになった．

強力な識別センサ技術

センサから受信したデータを内部で演算することで，より高度な情報を生成する強力なセンサが開発された．屋外で位置情報を識別するためには，個体の位置座標を緯度，経度と高度で詳細に検知できる GPS が開発された．また，屋内では部屋に対する座標情報を詳細に入手できる ActiveBadge [1] が開発された．また，物体をその物体がもと特有な情報から識別する画像認識技術も開発され，その例として人間の持つ瞳の情報から個体を識別する，バイオメトリクス [11] が開発された．また，個体に機器を設置し，それに識別子を振ることで，その個体自身の名称とそれがいるおおまかな位置を検知する RF-ID [12] が開発された．

このように様々な強力な識別センサ技術が確立したため，人間以外でも個体の識別や場所の検知を行え，特定の場所に設置されている物体を認知できるようになった．

1.2 本研究の目的

本研究の目的は，ユビキタスコンピューティング環境で氾濫し始めている様々な機器を，人間が毎回コントロールせずとも，その場に応じて最適な動作を自動的に実行する，コンテキストアウェアコンピューティング環境を構築することだ．この目的を達成するために，コンテキストを複数人で共有しあえるための，コンテキストアウェアコンピューティング環境のフレームワークを提案する．そして，このフレームワークで扱うコンテキストの表記方式を決め，それを生成するコンテキスト検知メカニズムを実装する．また，このコンテキストを複数人で活用できるよう，検知できるコンテキストの検索機構を構築する．

1.3 本論文の構成

本論文は全6章より構成される．次章では，コンテキストアウェアコンピューティング環境について概説し，その環境を構築する上で生じる研究課題について言及する．本論文では，この研究課題のうち，コンテキストが分散環境でも作成でき，複数の利用者が定義したコンテキストの検知手法を共有するフレームワークを提案し，実際にコンテキストを生成する機構を作成する．第3章では，コンテキストアウェアコンピューティング環境のためのフレームワークを提案し第4章にてそのフレームワークに沿ったコンテキストを検知するコンテキスト検知機構の設計の詳細を説明必要となる機構と共に実装する．第5章において性能評価を行う．第6章では，まとめと今後の課題について述べ，本論文を締めくくる．

第2章

コンテキストアウェアコンピューティング環境

本章では，コンテキストアウェアコンピューティング環境を解説するために必要な用語の定義をし，コンテキストアウェアアプリケーションを用いたシナリオと，コンテキストアウェアコンピューティング環境の概要を紹介する．そして，既存の関連研究を提示し，コンテキストアウェアコンピューティング環境における課題を提示する．

2.1 コンテキストアウェアコンピューティング環境

コンテキストアウェアコンピューティング環境 (以下 C-AComp 環境[1]) とは，ユビキタスコンピューティング環境にてコンテキストアウェアアプリケーションの機能が分散設置され，それらが利用者の嗜好に応じたに利便性を連携して提供するコンピューティング環境である．このコンテキストアウェアアプリケーションとは，コンテキストと言う情報をアウェアし動作するアプリケーションである．

本節ではこのコンテキストとアウェアアプリケーションが何であるかを定義し，その組合せであるコンテキストアウェアアプリケーションについて解説する．そして，シナリオを用いてその需要と利便性を解説し，最後に，コンテキストアウェアアプリケーションの機能が分散した C-AComp 環境の概要を解説する．

2.1.1 コンテキスト

コンテキストの意味を辞典 [13] から以下に引用する．

> context: The interrelated conditions in which something exists or occurs. [2]

これはつまり，認めることができる関係が相互になりたち，なにかが有るか起きていればそれを context として定義できる．この言葉自体は非常に明解で何が必要かを示しているがその反面，必要である事項は何でも良いとも言えてしまう．そのため，その対象となる範囲を狭め定義することは非常に難しい．

しかし，この情報は非常に有益なため多くの研究者が定義を試みている．コンテキストの研究者の一人である Annid K. Dey は [2] にてコンテキストを以下のように定義している．

> Context: any information that can be used to characterize the situation of entities(i.e. whether a person, place or object) that are considered relevant to the interaction between a user and an application, includeing the user and the application themselves. Context is typically the location, identity and state of people, groups and computational and physical objects. [3]

本研究では Dey 氏の定義するコンテキストをサブコンテキストとして定義する．そして，このサブコンテキストとは相互連関する対象と，その対象の事象が常に組合せられた情報で示される．また，このサブコンテキストの集合をコンテキストと呼びそれにも相互連関する対象と，その対象の事象が組合せられて示される．つまり，コンテキスト自身もまたサブコンテキストとして扱われる．

---

[1] Context-Aware Computing Environment

[2] 相互関係のある状態が有るか起きること．

[3] コンテキスト:人 (利用者) やアプリケーションに相互連関すると考慮され，誰なのか，どこなのか，また何なのかと言ったある状況や立場や特徴を示す情報の構成要素をコンテキストと呼ぶ．なお，相互関連する情報は人やアプリケーション自身であっても構わないず大抵は場所や，人・組織・計算機器・物体のステート (身分や状態) や識別情報 (身元や名称) である．

2.1.2 アウェアアプリケーション

アウェアアプリケーションとは，何か対象となる物や事象をアウェアすることで動作するアプリケーションである．このアウェア (aware) とは何かを理解するため辞典 [13] から意味を引用する．

aware: having or showing realization, perception, or knowledge [4]

このように，アウェア (aware) すると言うことは，何かを認知し認識すると言う意味を持つ．つまり，アウェアアプリケーションとは，何かを認知し，その情報を基に挙動を起すアプリケーションである．このアプリケーション自身は内部の行動情報を把握し，それと外部からの情報を融合し動作する．

2.1.3 コンテキストアウェアアプリケーション

コンテキストアウェアアプリケーション (以下 C-AApp) とは，利用者に関するコンテキストを検知することで最適な機能を駆動させるアプリケーションである．このアプリケーションには様々な機構が存在し，コンテキストの検知から機器の制御までの流れを作り出している．それらと機構を図 2.1 に示し利用者との関係を説明する．

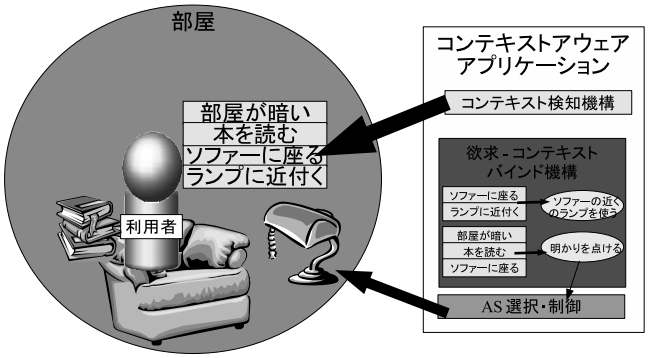

図 2.1: コンテキストアウェアアプリケーションの図

利用者

アウェアアプリケーションを利用する立場の存在を利用者と定義する．この利用者は人間や動物と言った生物以外にも，自動車やテレビと言った機器でも構わない．また，仮

[4]認識，知覚，知見できること，またそれを見せること

想空間上に存在するオブジェクトであっても，自身を含むアプリケーションであっても構わない．つまり，アウェアアプリケーションに” お世話” になる存在が利用者である．図 2.1 では利用者が暗い部屋のソファーに本を持って座った所が描かれている．

アプリケーションサービス

アプリケーションとは機器を制御するプログラムである．そして，アプリケーションは機器を制御することによって，利用者が行わなければいけない仕事を肩代りするか，利用者に有益な情報を提供する．このアプリケーションの行為をサービスの提供と呼び，アプリケーションがサービスを提供することをアプリケーションサービス[5](以下 AS) と呼ぶ．

この AS は情報家電を通し提供されていて，一つの情報家電が一つの AS だけしか提供しないとは限らない．図 2.1 では，照明を AS の提供者として表現し，その明りを照らす機能を AS として表記している．

コンテキスト検知機構

コンテキストは第 2.1.1 部にて定義したよう，利用者に関わる状況や状態の情報と名称である．図 2.1 では,「部屋が暗い」,「ソファーに座る」,「本を読む」,「ランプに近付く」としてコンテキストを表現している．コンテキストアウェアアプリケーションはこのようなコンテキストを検知する機構を最低でも一つ供えている．本研究ではこの検知機構をコンテキスト検知機構と呼ぶ．

コンテキスト-欲求バインド機構

コンテキストに応じて利用者が特定の AS を利用したいと考えることを，本論文ではコンテキストに応じた欲求が生まれると呼ぶ．この欲求はコンテキストと利用者に応じて変化する．そのため，どのような利用者が，どのようなコンテキストを検知した時に，どのような欲求が生まれるかを無から事前に把握しなければコンテキストに応じた AS の提供ができない．そこで，これらコンテキストと欲求の組合せを事前に保持し，提供する機構は不可欠となり，それを行う機構をコンテキスト-欲求バインド機構と呼ぶ．

図 2.1 では,「部屋が暗い」,「本を読む」,「ソファーに座る」とコンテキストを検知した時に,「明りを付けたい」と言う欲求と組合せるよう用意している．また,「ソファーに座る」,「ランプに近付く」ことで「ソファーの近くのランプを使いたい」と言った欲求になるようにコンテキスト-欲求バインド機構に用意してある．

この欲求は AS を動かすための情報であるが，コンテキストの集合から構成されている．つまりは，この欲求自体もコンテキストとして解釈できる．本研究では利用者がこれらの情報を分別して考慮できるよう，欲求とコンテキストは別の情報であるとして考慮している．

[5]Application Service

2.1.4 C-AApp のシナリオ

本部では三つの C-AApp を用いたシナリオを提示する．

- タイムリーコーヒーメーカー
  諭吉は毎朝出社前にタイムリーコーヒーメーカーから抽出されたばかりのコーヒーを飲んでいた．起きてから一杯飲み，またシャワーを浴びた後にもう一杯と計二杯，抽出して飲んでいた．

  諭吉は出張のため，普段とは違う遠い事務所に行かなくてはならくなったため，普段よりも早く起きなければならなかった．しかし，諭吉はタイムリーコーヒーメーカーの設定を何も変更せず，普段通りにベッドに着きいつもよりも早く起きた．すると，タイムリーコーヒーメーカーからいつも通りにコーヒーが抽出され始めた．また，彼がシャワーを浴び着替えて出てくると，二杯目の分もいつも通りに抽出され始めていた．彼は何食わぬ顔でコーヒーを飲み普段と同じように家を出ていった．

- 遅刻対応テレビ
  太郎は普段，平日の 8 時には帰宅していた．彼は毎週 9 時から始まるドラマを楽しみにし，必ずその時間にソファーで寛ぎながらドラマを見ていた．しかし，急な残業のためドラマの開始時間に間に合うか間に合わないか分らない時間まで家に戻れなくなった．

  太郎の帰路は混雑しており，玄関に着いてみると 9 時 15 分を過ぎていた．しかし，彼は何も焦らず部屋に入り，着替え，リビングのソファーに座ると始まっていた筈のドラマがオープニングから遅刻対応テレビに映し出された．

- 快通携帯電話
  花子は学生で，彼女は友達とのおしゃべりが非常に大好きで，授業中と睡眠中以外は必ずだれかしらと会話や電話，もしくはメールのやり取りをしていた．彼女は通学中，話そうという気持を我慢しメールのやり取りしていた．

  睡眠中やおしゃべりをしている時や電車に乗っている時は，おしゃべり仲間と電話を通した会話を遠慮したかったが遠慮無く電話がかかってきてしまう．花子は相手からの呼び鈴を無視できず,「メールでお願い!!」と毎回通話し頼みこんでいた．

  そこで，彼女はこのように電話の受け取りが，辛い時は自動的に「メールでお願い」と連絡を返せる電話への買換えを検討した．そして，彼女の行動に応じた返信を自動的にする，快通携帯電話を変更した．

  この携帯電話には，専用のイヤリングとネックレスがあり，それを就寝時以外身に着けていると，着けた者の行動に共なって自動的に行動に応じた返答をしてくれるものであった．

  花子はこの携帯電話に行動に共なった対応を設定することで，会話中と電車に乗っている時は通信相手へ自動的にメールでの応対をお願いするようになった．また，

授業中と就寝中では携帯電話からの連絡は呼出し音が通達されず，すべて留守録に残されるようになった．

これらすべてのC-AAppは各々が違うコンテキストを検知し，それに応じて生まれた欲求を対処している．これらのシナリオで出てきたコンテキストと欲求の一覧を以下に纏める．

- タイムリーコーヒーメーカ

  コンテキスト 諭吉が起きる，諭吉がシャワーを出る

  欲求 コーヒーが欲しい

- 遅刻対応テレビ

  コンテキスト 太郎が家に居ない，太郎がドラマを見ていない

  欲求 ドラマをバッファして欲しい

  コンテキスト 太郎がソファーに座っている，太郎がドラマを見ていない

  欲求 ドラマを再生して欲しい

- 快通携帯電話

  コンテキスト 花子が話す，花子が電車で移動する

  欲求 メールで連絡するよう応答して欲しい

  コンテキスト 花子が勉強する，花子が寝る

  欲求 留守録になって欲しい

このように，C-AAppは各々が必要となるコンテキストを検知し，それに応じた欲求の処理をしている．

2.1.5 C-AComp環境の概要

C-AComp環境とは単一の機器であるC-AAppが想定している環境とは違う．この環境では現在のユビキタスコンピューティング環境において，C-AAppが持っていたコンテキスト検知機構とASが様々な機器に遍在し，これらを無数の利用者が，無数のコンテキスト-欲求バインド機構を扱うことでコンテキストに応じたASの動作を利用者の欲求に適応して提供する．このように想定環境が広がったため，無数のコンテキストと無数の機器に遭遇する．そこで，従来とは違いどのような形がC-AComp環境であるかを，第2.1.4部で示したシナリオを用いて，彼等が持つ機器の機能を拡張し御互いが使い会うことを想定してこのC-AComp環境を解説していく．

始めに登場人物を下記に整理する．

- 諭吉
- 太郎
- 花子

彼等は同じ家に住んでいることと想定し今後は話しを進める．彼等の持つ C-AApp はそれ単体では自身のコンテキストに応じて機器の制御が直接行なわれる．しかし，そこで検知したコンテキストは他の二人も検知したく，またそれを用いて他の C-AApp を操作したいのでは無いであろうか？

それぞれの C-AApp の持つコンテキスト検知機構の詳細を下記にまとめる．

- コーヒーメーカ

  起床状態検知機構 諭吉の起床を検知

  シャワー利用検知機構 諭吉のシャワーの利用を検知

- テレビ

  スケジュール管理機能 太郎の時刻に応じたドラマ閲覧を検知

  人体検知機構 太郎の在宅を検知

  家具利用検知機構 太郎のソファーの利用を検知

- 電話

  移動手段検知機構 花子が電車で移動していることを検知

  会話検知機構 花子が話している事を検知

  集中状態検知機構 花子の勉強を検知

  起床状態検知機構 花子の就寝を検知

それぞれの検知機構は独自の手法を用いてそれぞれの事象を検知している．諭吉の起床状態はベッドの加圧に有無があるかで検知できる．また，シャワーの利用はお湯の利用状態により検知できる．また，在宅検知は鍵の空け閉めと rf-id の固有 ID 検知により行なえる [22][1][12]．花子の勉強は音の強度や周囲の変化具合から検知できる [3]．

このように様々な事象はそれぞれの C-AApp が検知し，他の機器へその情報を渡し利用することを想定していない．そのため，これらの機器が御互いコンテキストを共有し，連携して動くことができない．C-AComp 環境ではこれらの情報を共有するため，単体では想定しえなかった機器の挙動ができる．その例を下記に列挙する．

- コーヒーメーカが検知する起床情報と，テレビが管理するスケジュールを確認し，利用者が起きていなければそれぞれの機器が音や匂いを出すことで，通知する起床情報に対応した目覚し時計．

- 携帯電話の持つ集中を検知する機能から，その集中に応じてテレビの音声を下げる遠慮するテレビ．
- 他人が睡眠やソファーで寛ぐなどしている時は音量を遠慮する携帯電話．

このように，様々なコンテキスト検知機構から得られる情報を御互いに共有し，そのコンテキストに応じて機器が動作する．これを下記の図 2.2 に示す．諭吉，太郎，花子は

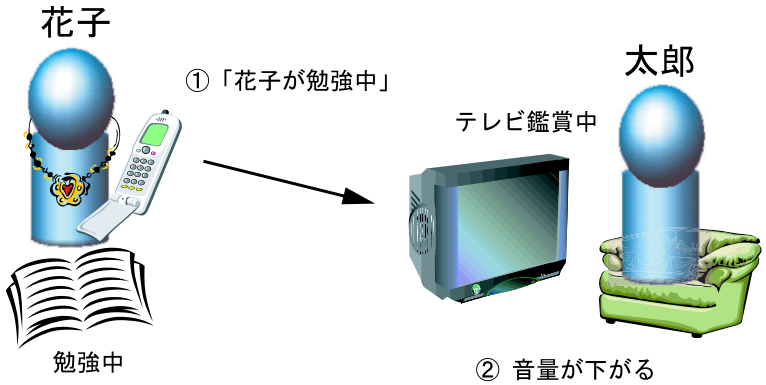

図 2.2: C-AComp 環境の図

それぞれ検知したいコンテキストを自身が検知機構を持つことや，環境から出力されてくるものを選ぶことで利用し，必要な時に応じて利用者に応じて最適な挙動をそれぞれの機器に行なわせるコンピューティング環境を C-AComp 環境と呼ぶ．

2.1.6 コンテキストアウェアコンピューティング環境のまとめ

コンテキストアウェアコンピューティング環境とは，利用者の状況や立場を示す情報を複数のコンテキスト検知機構が検知するコンピューティング環境である．この，コンテキスト検知機構は単一の AS に縛られることなく存在し，利用者に応じたコンテキストを生成する．そして特定のコンテキストが利用者に応じた欲求が生じている時に検知されることで，その欲求を AS を用いて解消する．このように利用者の欲求を計算機がコンテキストを自動的アウェアすることで，解消するコンピューティング環境をコンテキストアウェアコンピューティング環境と呼ぶ．

2.2 関連研究

現在コンテキストアウェアアプリケーションを用いた研究が世界中で盛んに行なわれている．本節では関連する研究がどのようにコンテキストを捉え生成し，扱っているかを解説する．

2.2.1 TEA

Karlsruhe TecO 大学の Albrecht Schmidt は持ち運びが自由で，環境の変動を捉える単調なセンサを複数搭載できる基盤，TEA Board を構築し，そこで Context を検知する TEA System[17] を開発した．Schmidt は TEA Board から出力されるデータからセンサごとに閾値を決め，それがあてはまる時を CUE として検知し，これらの集合をコンテキストとして定義した．CUE の閾値はコンテキストを検知する前に TEA Board を持ち歩き，データの蓄積後，それをニューラルネットワークで解析することで探し出された．

Schmidt は携帯電話と TEA Board を接続し利用者に持たせることで，利用者の行動に応じたコンテキストを生成した．そしてこのコンテキストに応じてコンテキストアウェアアプリケーションである携帯電話の挙動を変動させた．

2.2.2 C-MAP

ATR 知能映像通信研究所の角康之らは，美術館や博物館などの情報を展示する会場で，その展示品に関連する情報を利用者に提供する C-MAP[22] システムを構築した．C-MAP は利用者が興味を引く展示品に対してより詳しい情報を提供する機構だ．この研究の目的は，現実空間に存在する情報をデジタル化されている情報と関連付けることで，利用者に現実空間の実在する情報からデジタル化された情報を入手することだ．そのため，現実空間に実在する情報に利用者の興味が持つことをコンテキストとして表現した．

C-MAP ではこの興味を Active Badge System[1] を用いて利用者の位置情報を把握し，興味対象に近づく事で，そこに関連付けられているデジタル化されたデータが提供される．つまり，特定の対象物に近づく事とそのその対象の名称を興味と言うコンテキストとして捉えている．

このように，C-MAP は Active Badge から検知した利用者の位置情報をもとに，展示している作品に近づく事が利用者の興味を示したと言うコンテキストとして扱い，利用者にその展示品の情報を提供した．

2.2.3 Personal Positioning System

株式会社東芝研究所開発センターの青木恒は，頭部に取り付けられた装着型カメラが取得する画像の特徴を解析することで利用者の位置情報をリアルタイムで特定する，Personal Positioning System(以下 PPS)[24] を開発した．この研究の目的は，GPS の利用できない屋内で環境側にセンサを設置せずとも，その場所を特定することだ．そして，この場所を特定することで，その場所に関連する出来事を保存し，その場所に適したサービスを利用者に提供した．つまり，PPS は画像から検知する位置情報をコンテキストとして捉えている．

PPS は小型カメラが出力する画像のヒストグラムから特徴量を計算し，その量の極立つものだけを場所単語として記憶し，このカメラを持ち歩く利用者の場所を特定する．この際，その場所と建造物の物理的な位置を地図上に保存して行く．利用者が同じ場所

を再度通ることで再度特徴量を算出し，その算出に近い特徴量が算出されていた場所を同じ場所として保存する．これによって，PPS では建造物毎にその地図が必要だが，その情報のみで利用者の位置を検出することができる．そして，その位置に関連付けられたサービスが起動できる．

このように，PPS は人の位置情報を画像の特徴から検知し，それをコンテキストとして捉え利用した．

2.2.4 Context Information Service

Carnegie Mellon University の AURA プロジェクト [4] の一部の研究として Glen Judd を筆頭に行なわれている．Judd は AURA で検知される環境の変化や機材の位置情報をコンテキストとしている．そして，これを遠隔から検索てきるよう Context Information Service(以下 CIS)[5] をデータベースとして構築した．これにより，必要なコンテキストの項目と場所を指定することで，その情報をいつでも引出せるようにしている．また，SQL に似た表記を Context Information Provider に登録することで，コンテキストの検知時にイベントを要求することができる．

CIS ではコンテキストに，その情報の取得に関するメタ情報を記述することの必要性を述べている．このメタ情報とは Accuracy(情報の的確さ)，Confidence(情報の信頼性)，Update time(最終更新時刻)，と Sample interval(コンテキスト生成に利用した収集時間) である．

このように，CIS はコンテキストにメタ情報を付加し，データベースに挿入することで，アプリケーションが必要とする項目を要求するだけで，コンテキストの検索を行えるようにした．

2.2.5 Context-Toolkit

University of California, Berkeley の Anind K. Dey はプログラマが実空間に存在する情報を用いてアプリケーションを構築する際，それら実世界の事象や物体を認識し扱うために，これらの情報を取得し制御する Context-Toolkit[2] を構築した．

Context-Toolkit では，実世界の場所や人，または時，そしてその人が起こしている行動を widget を通して取得できるようにし，それらの情報の取得や解析をアプリケーションプログラマに意識させなくした．センサからの情報の入手を隠蔽しそれを透過的に扱える機構を構築し，その値の変動から作り出せるイベントも Context-Toolkit を用いて扱える．また，入手する情報が正しく無い情報である事も検討され，その情報を複数の提供者や複数回入手することで確認する手法も考慮されている．

このように，Context-Toolkit はプログラマの視点から，実世界の情報をコンテキストとして直接集められるツールキットとして構築された．

2.2.6 Life Patterns

Massachusetts Institute of Technology 大学の Brian Clarkson は，Life Patterns[6][3] を解析するため，自身の行動を 100 日間ウェアラブルコンピュータでモニタした．このウェアラブルコンピュータはバックパックに計算機と 2 台の小型カメラと，マイクとジャイロコンパスを備えることで構築されたいる．Life Patterns の解析は，搭載したセンサが蓄えた数値データを独自のアルゴリズムでクラスタリングし分類した．このクラスタリングされた情報は，特定のパターンが存在していて，その情報が収集した情報の随時に表われていた．つまり，そのパターンが存在する時間帯を検知することで，過去の利用者の行動と同じ行動を行なっていることが示せると証明した．このように Clarkson は特定の人間の生活パターンをコンテキストとして自動的に区分けできるようした．

2.2.7 SensorML

University of Alabama in Huntsville の Mike Botts は広域からセンサの情報を収集するために，付加情報の記述方法として Sensor Model Language (以下 SensorML)[15] を考案した．現在これは GIS にて Discussion Paper として提示されている．この SensorML は構成ハードウェアに異存せず，そのセンサの出力する情報を表記する手法を提案し，それが設置された場所や人や機材の性能を出力されるデータの一部として付加できるようにした．

この付加情報を用いることで，データの入手先が明確となり，分散して提供されていた情報に意味が持たせられた．これによって，特定の地域で起きている事象を解析でき，そのデータの信頼性も確認できるようになった．つまり，特定の範囲対象を SensorML を用いて区切り，その範囲で起きている事象，コンテキストを入手できるようになった．

SensorML の主な利用目的は，地球上の特定の地域を区切りそこの人口や汚染状態などの情報を収集するために考案されている．つまり，広域な自然環境のコンテキストを調べるために考案された．

2.2.8 関連研究のまとめ

現在多くの研究者がコンテキストの検知に関する研究を行なっている．コンテキストが何を示すのか，その見解の統一は行なわれていないが，多くの研究が中心となる利用者に関連する情報をコンテキストとして見ている．このコンテキストの生成方法や利用方法は多く研究されているが，そのコンテキストの多くは特定の環境に異存した情報を用いて構成されるか，その要求方法の検討がつく特定の利用者だけに扱える情報となっている．しかし，このような限定された条件下でのコンテキストの検知は現実化されてきている．

このように，多くの研究者がコンテキストの入手方法を研究し，その回答を各々が独自の手法で編み出している．

[6] 生活パターン

2.3 C-AComp 環境における課題

C-AComp 環境では単一の C-AApp と動作環境の想定が著しく違い，検知対象となるコンテキストや利用対象となる AS の種類が限定されなくなった．そのため，それらの情報を分散環境で管理する必要性が出た．本節では C-AComp 環境において，複数の利用者に関するコンテキストを作り，各々の欲求に対応する機器の AS を選択し，それを提供する利用するにあたりクリアしなければならない課題を解説する．

2.3.1 機構と機器の発見

C-AComp 環境で利用者はコンテキストを利用するために周囲に設置された無数のコンテキスト検知機構からコンテキストを入手し，それを用いて AS を自動的に制御する．この一連の作業を行うために始めに行う作業はこれら検知機構へのアクセス方法の入手と，その検知機構と利用者の関係を知る事だ．つまり，コンテキストを検知するために，コンテキスト検知機構に関するコンテキストを入手しなければならない．

また，コンテキスト検知機構のコンテキストが事前に入手でき，それを用いてコンテキストを入手したとしても，それを用いて利用する AS を探し出さなければならない．AS は小型な機器でも構わないため利用者が常に持ち運ぶことも考慮できるが，C-AComp 環境では利用者が持てる以上の AS が至る所に設置されている．そのため，利用者は自身が持つ AS だけでは満足できるとは限らず，周囲の AS を使う必要性が出てくる．いくらユビキタスコンピューティング環境とは言え，周囲に存在する AS を好き勝手に利用できるわけでも無い．そのため，周囲に存在する AS とそれが扱う機器を動的に探し出し制御する必要がある．

この一連の作業を行う事は一筋縄には行かず，様々な課題を解かなければコンテキスト検知機構を探し出せず，また AS を扱うこともできない．そこで，この課題を下記に列挙する．

- 利用者はその場所でどのような検知機構が存在するかわからない
- 利用者は機器が見えていても通信方法ががわからない
- 利用者は機器や検知機構と通信できても制御方法・命令方法がわからない
- 利用者が機器や検知機構を制御する権限を持たないかもしれない
- 利用者が AS の恩恵を受けられる距離にいないかもしれない
- 利用者が検知機構の出力するコンテキストの検知範囲にいないかもしれない

これらの項目を明確にしなければ利用者はコンテキストを入手することも，そのコンテキストで AS を自動的に動かすこともできない．

このように，利用者が必要とする様々な機器や機構を発見できなければ，C-AComp 環境を使うこと自体ができない．

2.3.2 収集するデータ

C-AComp 環境でコンテキスト検知機構は様々な機器からデータを入手する．そのため，移動した先々において異種の機器から情報を入手し解析することがある．このように解析機構が直結し利用する機器以外から情報を入手すると，機器の出力するデータを扱う前に，そのデータに関する情報を確認する必要がある．本部では確認が必要な事項が種類，単位尺度，出力頻度，意義であるとしその理由を解説する．また，コンテキスト解析機構が必要とするデータを出力する機構をデータ出力機構と呼ぶ．

種類

C-AApp 内部で解析機構が用いるデータは，利用するデータの出力基が常に同じであるため，それがどのような種類の情報であるか解析機構は常に理解できる．そのため，データ自身は極力不要な情報を排除し解析機構が解釈できる最低限の情報だけを出力する．このデータをコンピュータネットワークに直接出力してしまうと，データ自身が何を検知し解析した情報であるかわからず，解析機構はその情報を見付け出せずに，C-AApp で使われていた用に扱えなくなってしまう．これを避けるためには，出力機構が提供するデータの種類を明確にし，それを解析機構へ広告しなければならない．そして，これを必ず行わなければ解析機構は検討違いの情報を利用してコンテキストの解析を行なってしまう．例えば，圧力を計測することで椅子の利用状態を解析する機構が，圧力では無くそこの温度を利用してしまい，検討違いな解析結果が生まれてしまう．このように C-AComp 環境ではデータの種類を出力機構が明確に広告しなければならない．

単位尺度

C-AComp 環境では解析機構が利用するデータの種類が広告されただけではデータの解析はできない．対象となるデータの出力候補とその範囲が明確になり，さらにその候補間の関係が定義されなければ利用できない．ここで示した出力候補とはすなわち出力されるデータの形態を示し，その関係とはそのデータの性質を示す．つまり，出力されるデータが何を示しその示された情報がどのような意味を持つのかを明確にしなければならないと言うことだ．これが数値情報であれば，単位を示す事でそのデータの意味と数値間の尺度を明確にできる．つまり，データの単位尺度を明確にする必要がある．

例えば，温度を出力している機構が存在した場合，データ出力機構は摂氏，華氏，ケルビン，機器独自の単位，そして相対的な数値変動のみを示す数値として単位尺度を表記できる．これらは同じ温度を示す情報だがそれぞれの数値にまったく違う意味を持つため，これらは別物の情報として解析機構は判別し，解析しなければならない．

このように，データの単位尺度を明確にしなければ解析機構はその解析の際に間違った解析結果を生成してしまう．そのため，データ出力機構は自身が出力するデータの単位尺度を明確にしなければならない．

鮮度

コンテキスト解析機構によって必要となるデータの鮮度はまちまちである．しかし，C-AComp 環境に設置された出力機構が常に正しい鮮度の情報，とくに新鮮なデータを提供するとは限らない．C-AApp ではデータを要求するとそのデータが必ず最新の情報を提供することを約束できた．しかし，遠隔に存在する出力機構からデータを入手するとその鮮度が保証されない．出力機構によってはそれ自身のデータ入手が頻繁に行なわれず，提供できるデータの鮮度が保持されていた過去のデータである場合もある．また，コンピュータネットワーク上ではデータ通信に遅延が生じる事があるため，データ転送の遅延によって入手が遅れてしまう場合がある．また，出力機構はこのデータの鮮度を常に，その出力データの一部として出力しているとは限らない．そのため，入手したデータ自身の鮮度の確認が行えない場合がある．このように，C-AComp 環境で入手できるデータには鮮度の保証が無い．

意味

コンテキスト解析機構が扱うデータは，主にアプリケーションの挙動に関する情報やセンサが出力する情報を扱う．様々な機材からこのデータを入手するが同じ製造元が造り出した同系統の機種から情報を入手することもある．計算機上では通常，同じ機種から入手した情報は同じように処理できる．しかし，C-AComp 環境では，例え同じ機種から入手した情報であっても，そのデータを同じように扱うことはできない．これは，設置対象と設置場所に応じて出力されてくるデータの意味がまったく変ってしまうためだ．例えば温度のデータを入手していた場合，人の温度を検知すれば健康状態や快適指数を探りだせるが，それを台所で検知すれば調理状態や火災の有無を探り出せる．このように，データ出力機構が出力するデータの検知対象とその設置場所を考慮しなければ，C-AComp 環境ではデータを扱うことができない．

2.3.3 再検知できる事象解析手法の考案

コンテキストが指し示す対象が広範囲であったり，抽象的な事象であったりするため，コンテキストがどのような情報で構成されているかその明確な定義は無い．そして，特定の要素を検知する事で汎用的にどこでも同じコンテキストを入手できる手法も存在しない．これは，コンテキストが異存する情報は場所に応じて必要な事もあるからだ．そのため，コンテキストを検知するためには，どのような機材が必要でどこに設置すれば良く，その機材から得られるどのような情報を用いて解析すれば良いのか考案しなければならない．この作業を一人の利用者が移動した先々の環境で把握することは不可能である．特に移動先の環境に設置されているデータ出力機構のデータの出力特性など想定ができるわけも無い．そのため，この他人が考案したコンテキストを暫し活用する必要がある．

例えば，利用者がオフィスで「仕事をしている」と言うコンテキストを表記する場合，

オフィスにどのような出力機構が存在し，そこからどのようなデータが出てくるかは，常時活用するものにしかわからない．このオフィスの場合，椅子に加圧計，机に照度計，そして筆記用具に振動検知機が設置されており，椅子に重圧が掛り，机が電灯によって照らされ，筆記用具に振動が常に起きている場合は「仕事をしている」としてコンテキストを入手できる．つまり，事象解析手法を，機器に上述したようにセンサが仕込まれている事を理解しているものならわかるが，それを理解しないものはわからない．そこで，これを知っている A が検討した「仕事をしている」コンテキストを，それを知らない同僚 B が使いたいと感じるのは自然である．しかしここで，どのようにこの事象解析メカニズムを譲渡すれば良いか考慮する必要性が出てくる．

このように，コンテキスト解析機構にどのようなデータを期待させ，それをどのように事象解析メカニズムが解析すれば良いのか考慮する必要がある．そして，そのデータを他人が扱えるような手法も考慮する必要がある．

2.3.4 コンテキストの抽象化

C-AApp のコンテキスト解析機構はデータ出力機構と直結され，常に予期できるデータだけを受け取りコンテキストを生成する．そして，このコンテキストから欲求を生成し AS を動作させる．つまり，コンテキストが直接欲求となっており，一連の作業が完結しているため，そのため，コンテキストの生成過程で情報の抽象化を考慮する必要が一切無い．しかし C-AComp 環境では，生成されたコンテキストが様々な欲求に変化され，様々な AS によって利用される．そのため，抽象的にかつ汎用的にコンテキストを表記しなければならない．しかし，コンテキストの抽象化には利用者の主観と嗜好が大きく影響する．そのため，コンテキスト検知機構が生成する結果は特定の利用者の主観や嗜好で抽象化されている．つまり，このコンテキストを複数の利用者と共有できるよう抽象化することが難しい．

異種の検知機構で命名されたコンテキストを利用者が同種として検知できても，計算機はそれを同種として検知できない．つまり，命名規則が統一されていないため，その命名が違うだけでそのコンテキストを同じ意味を持つことを検知できない．例えば，椅子が利用者によって座られている状態と使われている状態はほぼ同じである．しかし，このほぼという微妙なニュアンスを計算機は理解することができない．そして，この命名は利用者によって異なって付けられる．そのため，利用者は命名規則を公開し，その意味が他の命名規則とどのような関係を示しているか提示しなければ，他の利用者はそれを理解できず扱えない．

また，例えこの命名規則が公開され統一されても問題がある．利用者は主観によってこの抽象化と命名を行うため，他の利用者によっては命名の意味に意義を唱える事がある．つまり，これは同じ検知機構から得られた結果であっても，複数の利用者がその結果を見た場合，矛盾した結果として受け取られる．この矛盾を図 2.3 を用いて解説する．椅子が人に使われているかを確認するコンテキスト検知機構 A と B が存在するとする．コンテキスト検知機構 A は椅子に加速度計からの振動による加速度変動から，椅子の空き状態を見る．また，コンテキスト検知機構 B は座蒲団に仕掛けられた加圧計の圧力に

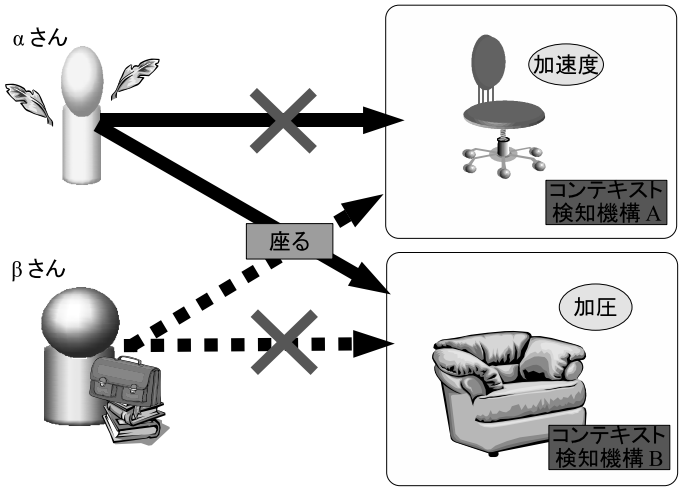

図 2.3: 「人が椅子に座る」コンテキストの検知

応じて，椅子の空き状態を見る．このように，異種のコンテキスト検知機構から同種の抽象化されたコンテキストが生成される事がある．このように生成されたコンテキストはどちらも必ず正しく見えるが，利用者によって片方しか使えない場合がある．α さんは，座る時にまったく椅子に振動を与えないため，A のコンテキスト検知機構ではコンテキストが検知できない．また，β さんは椅子に荷物を常に乗せてしまう人なため，B のコンテキスト検知機構ではコンテキストが検知できない．このように，両検知機構は抽象化された同義のコンテキストを生成していても，利用者によっては抽象化した意味に矛盾が生じてしまう．ここでは，両者は現状を御互いにとって正しく認識はしているが，相手の抽象化に賛同はできず矛盾が生じる．

このように，命名規則を統一し抽象化されたコンテキストであっても，利用者によってその意味が異なり同一のコンテキストとして扱うことができなくなる．

2.3.5 コンテキスト検知機構の評価

コンテキスト検知機構は様々な事象解析のメカニズムによって作られている．この解析メカニズムは一人の利用者がすべてのメカニズムを考慮して造られているわけでは無い．そのため無数の主観によって構成されたものを利用者は使わなければならない．つまり，コンテキスト検知機構によって利用者の主観とは似わないものが生じることが常識となる．そこで，出力されるコンテキストを利用者が取捨選別し，賛同が得られるコンテキストだけを利用者が扱える評価指標の導入が C-AComp 環境では重要となってくる．

コンテキストを評価し取捨選別する理由は他にも存在する．コンテキストは利用者にとって，制御権の有する AS を自動的に挙動しえる情報である．そのため，悪意を持ったコンテキスト検知機構が故意に利用者へ AS を制御するであろう欲求を生成しそうなコンテキストを送りこむことが懸念される．このコンテキストを善意なコンテキストと

して扱ってしまうと，利用者が制御権を持つ AS を悪意を持ったコンテキストの配送者が乗っ取れてしまう．または，AS の利用を不能にする攻撃が行なえてしまう．このような自体を避けるためにも，C-AComp 環境で利用者はコンテキストを評価し扱わなければならない．

2.3.6 コンテキストと AS

コンテキストは利用者の考慮するバインドの規則により，コンテキスト欲求バインド機構にて欲求へと変換され AS の提供を促す．この変換法則は利用者によって違うため，利用者の考案した変換法則は，他人に提供する場合，コンテキストと同様，その特定の利用者の推奨法則でしか無い．そのため，AS への挙動に矛盾を生じさせる法則を登録できてしまう．これは一人の考案した法則だけでも矛盾することがある．その例を解説する．まず，老人が「暖房の制御」と言う AS を扱え，「消す」欲求と「点ける」欲求が発行できる．この際，コンテキストとして「犬がいる」こと「子供がいる」ことを検知できる．老人はこれを用いて，「犬がいる」時は「暖房を消す」こととし，「子供がいる」時は「暖房を点ける」よう欲求を AS と組合せた．すると，「犬」と「子供」が交互に老人に近付くと問題は起きないが，同時に来てしまうと，矛盾した欲求が発行されてしまう．このような矛盾した欲求が発行されないよう，また発行されても対処できるよう，C-AComp 環境で A-S は対処できなければならない．

2.3.7 C-AComp 環境における課題のまとめ

本節では C-AComp 環境の構築における課題を考慮しそれを述べた．C-AComp 環境を構築するためには機構と機器の発見できるようになり，それらから出力されるデータの詳細を詳細に入手できなければならない．また，このデータから再検知できる事象解析手法を考案し，他人と協調できる命名規則を考案することでコンテキストの抽象化し，利用者によって生じる矛盾の中てもコンテキストを利用者へ提供できるようにならなければならない．また，C-AComp 環境で利用者は他人の考慮したコンテキストも使わなければならず，それの分別を行えなければならない．そして，コンテキストと欲求をバインドし，AS を受けられなければならなければならない事を述べた．

2.4 C-AComp 環境のまとめ

本章では，C-AComp 環境が利用者の状況や立場をコンテキストとして表記し，それ検知することで利用者を支援する AS が自動的に提供されるコンピューティング環境が有用である事を示した．そして，複数の関連研究を紹介することで多くの研究者がコンテキストの表記を研究し多くはその表現方法よりも，その入手方法を独自な解析手法で汎用性が無く作りだしてしまっている事を示した．そして，最後に C-AComp 環境を実現する上で，人の考案したコンテキストは他人が扱うと問題が起きる事を述べ，それでも，

他人の考慮したコンテキストを使えなければならない事を示した.

第3章

C-AComp環境のためのフレームワーク

本章では，コンテキストを汎用的に利用するために**C-AComp**環境のためのフレームワークを提案する．

3.1 C-AComp 環境の為のフレームワークの概要

C-AComp 環境を利用するためには，第 2.1.3 部にて解説した C-AApp が持つ AS とコンテキスト検知機構とコンテキスト-欲求バインド機構だけでなく，様々な場所に遍在するデータや，コンテキストや，AS を処理する機構が必要となる．そこで，本研究で提案するフレームワークではこれら機構がそれぞれの仕事を行えるよう新たな機構を設置する．本節では新設される機構の概要を解説しそれらと利用者が C-AComp 環境でどのような関係にあるかを図 3.1 を用いて解説する．

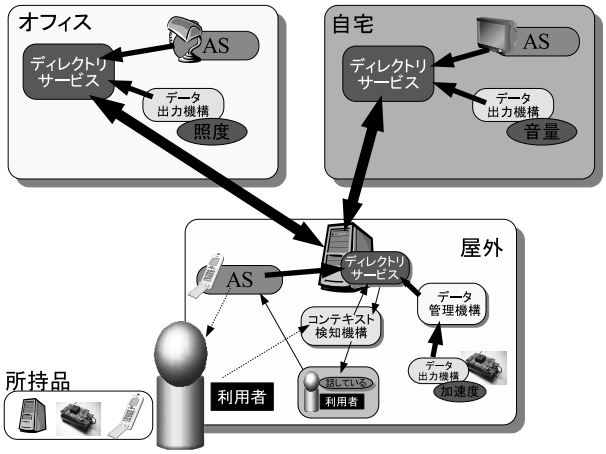

図 3.1: C-AComp 環境における各種機構と利用者の位置付けの図

利用者は様々な機構を持ち運ぶため，それら機構が御互い情報を発見しやりとりできるよう，それらの通信情報を一箇所にまとめる必要がある．C-AComp 環境ではこの情報をディレクトリサービスに登録し扱う．つまり，すべての作業はこのディレクトリサービスが必要となる情報を持つ各種機構との通信先を探す事から始まる．図 3.1 にも示されているよう，すべてのアクセス情報はディレクトリサービスの集約される．そして，利用者はこのディレクトリサービスを必ず持つ必要がある．このディレクトリサービスがコンテキストの検知に必要となる各種機構を発見し，さらに，そのコンテキストに応じて生じる欲求を処理する AS をも発見する．

コンテキスト検知機構はコンテキストを生成するためにデータ出力機構からデータを収集する．しかし，このデータは利用者が持つディレクトリサービスに存在しないかもしれない．そこで，環境側に設置されたデータ出力機構から情報を入手する必要性が出てくる．しかし，この機構をディレクトリサービスが移動した先々で直接探し出すことは難しい．そこで，環境側にもディレクトリサービスを設置し，このディレクトリサービス同士が通信することで環境に設置された出力機構からデータを入手する．しかし，まだ問題が存在する．他のディレクトリサービスに登録されたデータ出力機構は利用者が持つ通信機構だけでは，第 2 層の時点で通信機器とのプロトコルが違いデータの入手

が行えないかもしれない．そのため，それを吸収するデータ管理機構が各ディレクトリサービスと共に存在しなければならない．図 3.1 に示すようにデータ管理機構をディレクトリサービスに登録することで，利用者の持つデータ出力機構からでも，環境に設置されたデータ出力機構からでもデータを入手できるようになる．

これにて，コンテキストは生成されるが，ここからコンテキストに応じた挙動を AS に行なわせなければならない．図 3.1 に示すように，AS の発見はディレクトリサービスにそれ自身を登録することで行われる．しかし，コンテキストに応じた欲求は複数のコンテキスト欲求バインド機構から作られてしまうと矛盾した欲求だれけになってしまう．そこで，この C-AComp 環境の利用者が常にコンテキスト欲求バインド機構を持ち運び，この機構だけを使うことで一つのコンテキストから矛盾した欲求が生じなくなる．

このように利用者はディレクトリサービスを中心に各種機構を発見し制御する事で必要となるコンテキストを検知し，そのれから利用者の欲求を生成し，その場に存在する AS を起動させる．次節から上記で解説した機構の特性と，それが管理し通信すべき情報に関して解説する．

3.2 ディレクトリサービス

C-AComp 環境では利用者が必要とするコンテキストや AS を提供する機構が様々な機器に遍在し，それらの活躍が利用者に恩恵を与える．しかし，第 2.3.1 部にて解説したよう，これらの存在自体を直接発見し，それを制御することは難しい．そこで，C-AComp 環境では特定の機構の発見から，その機構との通信までを支援する機器情報管理機構，ディレクトリサービスの存在が必須となる．ディレクトリサービスには ASAMA [21] や JINI のルックアップサービス [23] が存在し，利用者が必要となる機材や機構はこれらを通して発見する．

このディレクトリサービスは地理的に区切られた空間を一つの単位として管理する．そして，その空間に存在する不動な機構，つまりそこで空間外に移動しない AS やデータ出力機器や，またコンテキスト解析機構を，このディレクトリサービスに登録する．この登録作業はその空域を管理する管理者が行い，それら機材に関する情報を登録する．ここで登録の必要がある情報を定めそれを下記に示し，その理由を解説する．

- 本機構に関する情報
  - アクセス方法
  - アクセス権限
  - 位置情報
  - 管理責任者
- 本機構以外のディレクトリサービス
  - アクセス方法

- AS
  - 名称
  - 種類
  - 利用方法
  - アクセス方法
  - 設置場所
- データ管理機構
  - データ管理機構の種類の名称
  - データ管理機構へのアクセス方法
  - データ出力機構の情報
    - データ出力機構へのアクセス方法
    - データ出力機構の設置場所
    - データ出力機構の検知対象の名称
    - データ出力機構の検知対象の性能
- コンテキスト検知機構に関する情報
  - 事象解析メカニズム
    - 種類
    - アクセス方法
  - コンテキスト変換機構
    - アクセス方法
    - メインとして利用しているディレクトリサービスへの一意な名称
- コンテキストにおける状態の出力候補

ディレクトリサービスは自身の詳細な情報とそのディレクトリサービスと通じて接続できている様々な機構の情報を保持している．つまり利用者はどこか一つのディレクトリサービスに利用したい機構の情報を探しだすことによって，その利用者が通信できる範囲で利用できる機構を見付けだすことができる．

ディレクトリサービスが自身で抱えている情報は，それ自身へのアクセスに関する情報以外に，そのディレクトリサービスの位置情報と，そのディレクトリサービスの管理責任者がある．位置情報は本部にてすでに解説したよう，そのディレクトリサービスを指定することにより利用したい機構の範囲を限定するために存在する．また，管理者の情報はそこに接続されているディレクトリサービス以外の各種機構の信頼性を計るために用いる．この本研究では特定の AS，データ管理機構，そしてコンテキスト検知機構は必

ず一つのディレクトリサービスに属し，そのディレクトリサービスを通してでしかそのアクセス方法を探し出せないようになっていることを想定している．つまりは，ここで指定した管理者にしか各種機構を設置する権限を与えられるていない．そのため，この管理者を確認することで，その情報を本当に信頼して良いかを判断する指標になる．もちろん，各種機構が発する情報の信頼性はまた別の問題である．この管理者は，そこにその機構が本当に登録されていることを保証する保証人の名称であるだけだ．

本機構では自身以外のディレクトリサービスの情報は，それへのアクセス方法以外保持しない．ディレクトリサービスとコンテキスト検知機構の一部は動的にその情報が更新されるが，それ以外は静的であることを想定している．そのため，この頻繁に更新を受ける情報の量を減らす目的として，自身以外のディレクトリサービスのアクセス以外に関する情報は搭載しない．また，動的となるコンテキスト検知機構の一部とはコンテキスト変換機構を指す．この機構が動的となる理由は第 3.4.2 部にて後述する．

本機構では各種機構が物理的に存在する場合，それらの位置情報を保持している．物の位置情報を表記するための研究は豊富にあり，それを指定する手法は固定することができないため，本研究ではその情報を記述できるとして言及し，それを名称のみで表記する．この位置情報の名称はディレクトリサービスの管理範囲内で一意の名称でなければならない．また，ディレクトリサービス自体の位置情報は他の未知のディレクトリサービスと動的に連携することを想定しているため，統一化された汎用的なフォーマットで記述されなければならない．つまり，人間が生活する 3 次元空間上で一意に示せる位置情報を表記することが推奨される．本研究ではこの情報に緯度，経度，高度とその名称を表記することを推奨する．つまり，このディレクトリサービスの管理空域をこの位置情報で指定し，利用者がそのディレクトリサービスを扱うか考慮できるようにする．

本機構では各種機構が持つ固有の識別情報である名称や，各機構特有の情報を保持する．利用者が持つコンテキスト検知機構とコンテキスト-欲求バインド機構は本機構にこれらの情報を基に，各種機構がそのディレクトリサービスの管理空域に存在することを検知し，それらへのアクセス方法を通達する．

このようにディレクトリサービスによって管理範囲を限定しそれらの情報を管理する事で，各機構が扱うべき他の機構を限定できるようになる．そして，ディレクトリサービス同士が通信しあう事で，他の機構が動的な発見機能を考慮する必要がなくなる．このようにディレクトリサービスを通し各種機構が必要な他の機構の情報を入手することで，コンテキストに応じた AS の提供を移動した先々で扱えるようになる．

3.3 データ管理機構

C-AComp 環境では様々なデータ出力機構が存在し，それらからデータを直接入手すること難しい．これは，第 2.3.2 部で述べたよう，データ自身に様々な情報を付加しなければ C-AComp 環境では役に立たないデータとなるからだ．そのため，この付加しなければならない情報をも含むデータをすべてのデータ出力機構に強制するのでは無く，付加情報を付加し限定された空間のデータを常時管理するデータ管理機構を用いることが有

用となる．そして，このデータ管理機構を第 3.2 節で解説したディレクトリサービスとペアで設置することで，限られた空間内のデータをデータ管理機構を通すことでアクセスできるようになる．

このデータ管理機構が保証し，提供しなければならない付加情報とは，データの種類と，単位尺度と，鮮度と，設置対象と設置場所である．また，データ出力機構からデータを要請するための命令をも管理し，管理機構自身がデータを一時的に保持，またはそれを介すことで，データを入手できるようにならなければならない．つまり，データ管理機構は下記に列挙する項目の情報を，各データに対して保持できなければならない．

- 出力機構の性能
  - 型・単位
  - 出力頻度
- 取得時刻
- 取得場所
- 取得対象

コンテキストの解析には，利用する情報が同じ種類の情報でなければならない．しかし，C-AComp 環境におけるデータの種類とは，単純にデータ型や単位だけでは無い．これだけでは C-AApp のようにすべてが直結された解析機構でしか活用できない．そのため，出力機構に応じてそのデータの鮮度と，それを計測している対象とその場所も付加しなければならない．また，この単位は出力されてくるデータの候補を示す情報となる．そのため，この単位に関する解説も行えなくてはならない．この解説によって利用者が求めているデータと同じ情報を出力しているのか判定できる．

データの鮮度を確認するためには，その出力桔梗がデータを提供できる頻度と取得された時刻が重要となる．取得された時刻だけでは，解析機構が同じ要素の情報を特定の時間内に集め解析する手法を用いていた場合，次に予測されるデータの出力時刻の検討がつかず，解析自体行えなくなってしまう．また，データの出力頻度だけでは，データ管理機構にそのデータ自身が保持されていた場合，その頻度の間の何時に入手した情報であるかを正確に把握できなくなってしまう．つまり，頻度が一日のような出力機構であった場合，取得時間が確認できなければ，一秒前に取られたばかりの情報なのか，それとも一日近く前に取られた情報なのかを確認できない．単一のデータを用いて解析する解析メカニズムである場合，一秒前に取られた情報であれば活用できるだろうが，一日前に入手された情報では活用できない可能性が高い．そのためこの出力頻度と取得時刻はペアで取得できなければならない．

データの取得場所は，この管理機構が登録されているディレクトリサービスを検索する事で大まかな所属は確定できる．しかし，さらに粒度の細かい位置情報を確定したい場合，特定のディレクトリサービスに属している，つまりは特定の場所に設置されたグループに所属していると言った情報だけでは利用者が満足できるとは限らない．そのた

め，より精度の高いコンテキストを入手するため，解析機構は粒度のさらに細かい場所の情報を要求する．この粒度の細かい情報を用いる事で利用者はその情報をコンテキスト解析を行う際に扱うか否かを確認できる．

データの取得対象はディレクトリサービスによって管理された空間の一部の情報を示す．この対象の名称はディレクトリサービスで一意に特定できる名称として保持される．つまり，データ出力機構はすべてディレクトリサービスに一意な名称で登録される．また，この対象の情報は二種類の情報によって構成される．一つはその一般的な名称．そしてもう一つがその名称をディレクトリサービスの管理範囲内で同一名称のものを一意に識別できる番号．この二つの情報をもってデータの取得対象とし保存される．

このように，出力機構が出力するデータにこれらの情報すべてを揃えることで，それを同じ種類としてコンテキスト解析機構はみなし扱えるようになる．

3.4 コンテキスト検知機構

コンテキスト検知機構はデータ管理機構とディレクトリサービスからデータと，そのデータの付加情報を入手することで，利用者に関するコンテキストを生成する機構である．つまり，コンテキスト検知機構とは，コンテキスト生産機構が生成したコンテキストを扱う機構である．

本フレームワークで構築するコンテキスト検知機構は二つに分かれる．一つはコンテキストの解析をデータ管理機構からデータを入手し解析する事象解析を行う，事象解析メカニズムである．また，もう片方は事象解析で解析された結果をコンテキスト-欲求バインド機構が扱い易い汎用的な形式に変換するコンテキスト変換機構である．

本節ではこれら二つの機構がコンテキスト検知機構の中でどのような活躍するかを解説する．そしてコンテキスト検知機構が，ディレクトリサービスとデータ管理機構と，コンテキスト-欲求バインド機構とどのような関係にあり，生成されるコンテキストと利用者がどのようにな関係にあるかも解説する．

3.4.1 事象解析メカニズム

事象解析メカニズムとは，出力機構が出力するデータを用いて特定の事象をコンテキストとして解釈する解析メカニズムである．ここで解析する事象とは，特定の対象がどのような場所にあり，どのような状態にあるかを示す情報である．

この事象解析メカニズムは第 3.4.2 部にて後述するコンテキスト変換機構が指定したデータを完全に信頼するため，事象解析メカニズム自体はその事象が利用者にとって正しいかは判断しない．しかし，その事象がデータ管理機構が出力するデータのみから正しいことを示す．つまり，上記の事象の候補を事象解析メカニズムは提案し，それを他機構が判断する．また，事象解析メカニズムはコンテキスト変換機構よりデータ以外にそのデータの解析ルールを受け取ることができる．これによって，データの性質によって解析手法を変更する必要があるメカニズムが存在する場合，それをこの解析ルールに

よって適応することができる．

事象解析メカニズムは単一のデータのみでなく複数のデータを同時に入手しそれを処理する．つまり，事象解析メカニズム自身はデータを一定期間溜めようが，ルールによって指定された過去の情報と参照しようが，外部の機構と連携しようが何をしても構わない．つまり，上述する対象と状態を判別するための情報さえ出力すればどのような事を行なっても構わない．そして，この事象が本当に利用者にとって正しいかは，事象解析メカニズムが用いない付加情報をコンテキスト変換機構が検証することで判断する．

このように事象解析メカニズムの解析手法とその事象の認証を分けることで，無数の利用者が無数の事象解析メカニズムを C-AComp 環境で扱えるよう考案できる．

3.4.2 コンテキスト変換機構

コンテキスト変換機構は利用者が必要とするコンテキストの要求を処理し，そのコンテキストを検知するにあたり必要となる事象解析メカニズムを割当て，その結果を要求者の望む形態に変換し提供する機構である．利用者は本機構に下記の項目を申請しコンテキストの生成を要求する．

- コンテキスト情報の公開情報
- ディレクトリサービスの情報
  - 管理責任者
  - 位置情報
  - ディレクトリサービスへのアクセス方法
- コンテキスト情報の検討者名称
- コンテキスト
  - 検知の対象の名称
  - 検知対象の位置情報
  - 検知対象の状態
  - 検知時刻
- コンテキストの表記ルール
  - 名称の表記ルール
  - 位置情報の表記ルール
  - 状態の表記ルール
  - 検知時刻の表記ルール

- データ管理機構
  - データ管理機構の種類の名称
  - データ管理機構へのアクセス方法
  - データ出力機構の種類の名称
  - データ出力機構へのアクセス方法
- 事象解析メカニズム
  - 事象解析メカニズムに与えるルール
  - 事象解析メカニズムの名称
  - 事象解析メカニズムへのアクセス方法
- 事象解析結果-コンテキスト変換ルール

これらすべての情報を本機構に申請することで，コンテキスト変換ルールに示された名称と，位置情報と，状態をコンテキストとして入手できるようになる．これらの情報を本研究ではコンテキスト情報と呼ぶ．

利用者はこのこのコンテキスト情報を二通りの手法で作成する．一つはアクセス方法以外の項目を明記し，残りの情報をディレクトリサービスから補完し作成する方法である．また，もう一つは，他人が一つ目の方法で作成したコンテキスト情報を複数の項目の条件を指定することで探し出す方法である．

前者は利用者がディレクトリサービスの空域で管理されているデータ出力機構や事象解析メカニズムを熟知していて，その解析手法を考慮できる立場の人間が行う．これで，コンテキストを生成するためのコンテキスト情報が用意される．そして，このコンテキスト情報をもとに本機構にコンテキストの検知を要請することで，その結果をコンテキストとして入手する．

後者は，前者が作ったコンテキスト情報を探し出す．しかし，前者の探したすべてのコンテキスト情報が本機構に残っているわけでは無い．前者の方法で本機構のコンテキストの検知を要請する際，その情報を他者に公開して良いかを確認する．公開できる場合，そのコンテキスト情報で指定した各種ルールと条件がマッチした場合，コンテキストが生成され，その結果が本機構に残るようになる．そして，この残されたコンテキスト情報を下記に示す項目を指定することで検索する．

- ディレクトリサービスの情報
  - 管理責任者
  - 位置情報
- コンテキスト情報の検討者名称
- コンテキスト

  - 検知の対象の名称
  - 検知対象の位置情報
  - 検知対象の状態
  - 検知時刻
- コンテキストの表記ルール
  - 名称の表記ルール
  - 位置情報の表記ルール
  - 状態の表記ルール
  - 検知時刻の表記ルール

もちろんこれらのすべての要素を指定して選ぶ必要は無い．部分的に比較するために特定の要素だけを選び検索することも考慮する．ただし，コンテキストの各要素を指定しない場合，それを指定しない事自体が条件となる．つまり空であると，その要素を考慮し無いと言う条件として扱う．

この作業で選ばれたコンテキスト情報は直接コンテキストの検知要請として扱われるわけでは無い．この後者の手法はあくまで利用者の必要とするコンテキスト情報の作成である．つまり，この後者の手法を扱うことで下記に示す扱うべき機構とそれらへのアクセス方法を探し出す．

- データ出力機構
- データ管理機構
- 事象解析メカニズム

そして，これらの機構の扱い方法を記述した下記の二つの種類のルールを入手する．

- 事象解析メカニズムに与えるルール
- 事象解析結果-コンテキスト変換ルール

このルールと扱うべきデータ出力機構と事象解析メカニズムを探し出す事によって，コンテキストを生成できるコンテキスト情報を作り出せる．そして，この作成したコンテキスト情報を用いて本機構に申請する事で，指示したルールに準じた事象解析のルールを検知した際に，申請したコンテキストを検知したとして扱えるようになる．

ここまで，コンテキストの表記ルールに関して触れずに来たが，これは上記コンテキストを表記する際に用いるルールである．これらの情報の表記は特定のルールに沿い表記する．位置情報の場合は POIX [7] や GML [19] と言った標準化仕様が存在するため，これらに準拠して記述した場合はその旨を表記すれば良い．また，検知時刻に関しては Date and Time Formats [16] と言った標準化が存在するため，それに準拠して表記した場

合はその旨を表記すれば良い．しかし，コンテキストの名称や状態は利用者の主観によって変わり標準化できない．そのため，これらは利用者に応じてその名称や状態の候補を事前に用意し提供する必要がある．名称の標準化は下記に示す情報と共に示す．

- 対象ルール作成者名称
- 対象の情報
  - 対象の名称
  - 同種名称

ディレクトリサービスが管理する空域に存在するデータ出力機構が設置された対象の情報を上記の項目として表記し残す．この作業は管理者が許可し，その情報は各ディレクトリサービスが管理する．この項目はディレクトリサービスが他に公開されると同時に公開される．これによって，ディレクトリサービス内で検知できる対象を，利用者が常に確認できる．しかし，この対象の名称ルールは管理者の独断で書かれてしまい，汎用性が期待できない可能性が高い．そこで，同種の名称の項目に汎用的な名称を用意しその名称に所属してるであろう事を管理者は想定し記述する．もちろん，この項目は利用者の要請により追加される事も検討しなければならない．

また，利用者の状態も名称と同様考慮する必要がある．しかし，対象とは違いこの状態は排他的に候補を選べる形で提供する．これを下記に列挙する．

- 状態ルール作成者名称
- 状態ルールの名称
- 状態の情報
  - 状態の名称候補 (複数)

この状態の候補は，複数の事象解析メカニズムが同時に同じ対象を解析した場合，矛盾が生じる状態を結果として生成してしまった時にそれを検知できるよう用意する．つまり，複数の事象解析メカニズム間で同一の重みを持つ状態を解析した時に，矛盾した結果を事象解析メカニズムの優先度を決められように用意している．そして，この状態のルールは事象解析メカニズムの結果とセットとして扱われる事が検討される．そのため，事象解析メカニズムが設置されている場所においてこのルールは扱われる．つまり，ディレクトリサービス単位でこの情報を管理することはできず，事象解析メカニズムが設置されている範囲で活用できる．そのため，このルールは事象解析メカニズムが活用されている範囲で公開しなければならない．

このように C-AComp 環境では利用者は事前にコンテキストの検知手法とその結果の出力要請を行わなければコンテキストが入手できないが，このような制約を設ける事で他者のコンテキストの検知方法を検索し利用できるようになる．

3.4.3　コンテキスト検知機構のまとめ

本部ではコンテキスト検知機構の部分要素である事象解析メカニズムとコンテキスト変換機構について解説した．事象解析メカニズムは指定されたデータを扱い，指定されたルールに沿って事象を解析する．解析結果はコンテキスト変換機構で扱われ，コンテキスト情報を参照することでその結果からコンテキストを生成しコンテキストの検知を促した．また，その結果のルールを用意する事で，他の機構がその結果を読めるような枠組みを提案した．そして同時に，複数の事象解析機構が矛盾した結果を生成した場合の事を考慮し，その矛盾を発見できるコンテキストの同列の重み付けを解釈できるような枠組みを提案した．

3.5　コンテキスト-欲求バインド機構

本フレームワークで欲求とは，情報家電と言う AS の制御を行う事を指す．コンテキスト-欲求バインド機構はこの欲求をコンテキストと結び付け，AS の始動を利用者からの直接命令されるわけでは無く，コンテキストから行えるようにする機構である．本機構はコンテキストとは違い，物理的，または論理的にその物体が存在しステートが一意に内部で保管されている．そのため外部からの変更に矛盾が生じた場合，それを検知することができる．この矛盾とは，一定時間内に定期的に同一の AS に対し同一の利用者から特定の排他的選択を行うステートの変更が頻繁に行われる事を指す．つまり，ライトのオン・オフを繰り返したり，ビデオの操作を巻戻しては速め，そしてスローモーションで再生する，と言った形で単一の動作しか指定できない操作に対し，それ以上の動作を要求する事を指す．

コンテキストに応じた欲求の組合せは利用者がすべて決める．そのため，自身が矛盾した要請を機械に命令してしまうことがある．そこで，本機構で発見した矛盾を利用者が分けられるよう，以下に区分を提示する．

1. 人為的な勘違いからまったく同じか近いコンテキストに正反対の動作要請を関連付けてしまった場合

2. コンテキスト生成に用いた要素同士に強い関連性があり，それに気が付かなかった場合

3. 関連性が存在するが，それを検知できず，偶然同じコンテキストであるのに反対の動作要請を関連付けてしまった場合

これらを機械的に区分するには，コンテキストの生成に利用された両方の構成要素の関連性を，検知できなければならない．この時点で，関連性を構成要素から見出すのは苦難である．そのため，利用者にそれぞれ関連があると思われる同種の構成要素を提示するなどし，このコンテキストが上記のどれに当てはまるか指示を仰ぐ．1 である場合は，両方のコンテキストを抽象化し同義に扱えるようにする．2 や 3 である場合は利用者の嗜好により，どちらかの嗜好を優先させ，再検知時に両者の優劣が分かるよう履歴を残す．

このように矛盾する動作を本機構で検知することでコンテキストの自身の作成した欲求の矛盾を検知したり，コンテキスト情報に記述された事象解析メカニズムとコンテキスト変換ルールに矛盾がある事を検知できる．

3.6 C-AComp 環境の為のフレームワークに対するまとめ

本章では C-AComp 環境を扱うためのフレームワークを提案した．そして，このフレームワークではディレクトリサービスと，データ管理機構と，コンテキスト検知機構と，コンテキスト-欲求バインド機構が遍在している環境で，利用者がディレクトリサービスと，利用者が検知したいコンテキストのルールを表記したコンテキスト情報と，その結果を欲求へ変換するルールを持ち歩き，各機構にそれらを挿入する事でコンテキストに応じた AS を制御することを提案した．そして，このフレームワークを用いることで，コンテキスト情報を利用者が持た無い場合は，それを C-AComp に設置されたコンテキスト生成機構に必要な項目を通達するだけで入手できることを示した．

第4章

コンテキスト検知機構

本章では，**C-AComp** 環境においてコンテキスト検知機構の概要を解説する．そして，コンテキスト検知機構がどのように各種機構と連携し，自身が持つコンテキスト変換機構と事象解析メカニズムを用いてコンテキストを検知するか解説する．また，それらを連結するコンテキスト情報と **XML** ディレクトリサービスを解説する．そして，コンテキスト変換機構**:CITE** を設計・実装しそれが扱う，事象解析メカニズムの一例である事象解析メカニズム機構 **SPICeS** と **TinyDBClient** の設計と実装をする．

4.1 コンテキスト検知機構の構築までの想定環境

本研究ではコンテキストを生成するまでを研究の対象としているため，コンテキストの生成以降の作業は他の研究に任せる．つまり，AS との連携自体は他の研究にまかせる．本研究では，コンテキストを作るまでを実現する．

本フレームワークで解説した機構には，既存の機構を用いる事を想定している．本節ではその機構を述べ解説する．

- データ出力機構
  - BerkleyMote
- データ管理機構
  - TinyDB

BerkleyMote からはそれが持つ固有 ID，加速度計，照度計，温度計，磁力計，音量計のデータをデータ管理機構に送信する能力を備えているため利用する．

TinyDB[18] は BerkleyMote へデータの要求を投げ，それを取得できる BerkleyMote の管理機構である．TinyDB は本フレームワークで述べた機能要件を満たしているため利用する．

各種機構へのアクセスはすべてインターネットプロトコルを用いることを想定とする．また，各種機構の所持情報は XML によって記述し，その情報の交換はテキスト情報を扱うことを想定する．

また，本研究がこの想定環境の C-AComp 環境を実現する上で実現されていなく，その具現化が必要と考える機構を下記に述べる．

- コンテキスト情報の表記方法
  - XML による表記
- ディレクトリサービス (コンテキスト用)
  - XML ディレクトリサービス
- コンテキスト変換機構
  - CITE
- 事象解析メカニズム
  - SPICeS
- 事象解析メカニズム
  - TinyDBClient

本章ではこれらの機構の設計及び実装を解説して行く．

4.2 コンテキスト情報

コンテキスト情報はコンテキストを生成する上で無くてはならない情報である．本研究では，このコンテキスト情報を XML を用いて表記する．コンテキスト情報は，コンテキスト変換機構において要求，検索，通知のすべてで利用される．付録 A.1 に具体的な XML を示してあるので，それを参照して欲しい．この表記を順を追って部分要素を解説する．

4.2.1 コンテキスト情報:ContextInformation

コンテキスト情報はすべて *ContextInformation* タグで囲まれる．

```
<ContextInformation
security="公開情報"
citype="コンテキスト情報の種類"
>…</ContextInformation>
```

この，*ContextInformation* タグには二つの属性が存在する．それは *security* と *citype* だ．*security* は利用者が挿入したコンテキスト情報を他者が検索できるよう，コンテキスト解析機構に残すかどうかを指定する．指定できる項目とその効力を下記に述べる．

- **security**

  public 他人がこのコンテキスト情報を扱う許可を与える

  private 要請者だけがコンテキスト情報を扱う

- **citype**

  query コンテキスト情報として，コンテキストの検知要請を行う

  search コンテキスト情報の入手として，コンテキスト情報の検索要請を行う

  searchR コンテキスト情報の入手として，コンテキスト情報の検索要請結果

  ci コンテキストの検知要請の結果としてそのコンテキストとして情報を伝える

このように，公開対象とコンテキスト情報自体の意義をこれらの属性は定義している．*ContextInformation* には下記に示す要素が含まれている．

```
<ContextInformation>
<Requester><![CDATA[処理の要求者]]></Requester>
<Authenticator match = "比較方法"
><![CDATA[考案者名称]]><Authenticator>
<reqTime>要請時刻</reqTime>
<objectXMLPointer><![CDATA[URI]]></objectXMLPointer>
<statustXMLPointer><![CDATA[URI]]></statustXMLPointer>
<DirectoryServiceInformation>…</DirectoryServiceInformation>
<DataManagementEngine>…<DataManagementEngine>
<ContextAnalyzingEngine>…<ContextAnalyzingEngine>
<contextCapRule>…<contextCapRule>
<contextMeaning>…<contextMeaning>
</ContextInformation>
```

Requester はコンテキスト情報の処理を以来している者を一意に判別できる名称を記述する．この一意な名称には E-Mail アドレスを用いる．

Authenticator はコンテキスト情報の考案者の一意に判別できる名称を記述する．これは *Requester* と同様，E-Mail アドレスを用いる．*Authenticator* はコンテキスト情報をコンテキスト変換機構でコンテキストを検索する際にこの情報が判断基準の一つとなる．そのため，このタグには属性情報として *match* を用いて比較方式を記述する．なお，この *match* は様々な要素に付加される．これら属性情報で指定できる項目と，その効力を下記に述べる．

- **match**

  any 要素の情報が何であっても正として認識する

  none 要素の情報が空である場合のみ正として認識する

  direct 指定した要素内の情報と完全一致する正として認識する

  regex 指定した要素内の情報を用い正規表現で比較し得た情報を正として認識する

この *match* の属性は，要素が記述されている場合は *direct* として扱われ，記述されていない場合 *any* であるとして扱われる．空要素を指定したい場合はは *none* を指定しなければならない．

reqTime はこの情報が利用者によって発行された時刻が記述される．記述方式は W3C の勧告する [16] の方式を使う．

objectXMLPointer と *statustXMLPointer* はコンテキストの情報を指定する時の名称と状態の名前空間を検討する時に扱う情報へのポインタである．これらの情報自体は XML で表記し，ここではそれらへの URI を指定する．

DirectoryServiceInformation と，*DataManagementEngine* と，*ContextAnalyzingEngine* と，*contextCapRule* と，*contextMeaning* は次部以降で解説する

4.2.2 ディレクトリサービス:DirectoryServiceInformation

DirectoryServiceInformation の要素ではコンテキスト情報を扱う際に利用するディレクトリサービスの情報を記述する．本要素では以下のように示される．

```
<DirectoryServiceInformation>
<DirectoryServiceLocation myself = "デフォルトのDSか否か">
<hostname>対象となるホスト名</hostname>
<portNum>接続するポート番号</portNum>
</DirectoryServiceLocation>
<DirectoryServiceRealLocation match = "比較方法"
><![CDATA[位置名称]]></DirectoryServiceRealLocation>
<Maintainer match="比較方法"
><![CDATA[nandu@ht.sfc.keio.ac.jp]]></Maintainer>
<timestampRlx at = "init"
tComp = "時刻比較方法" >時刻</timestampRlx>
<timestampRlx at = "lastMod"
tComp = "時刻比較方法" >時刻</timestampRlx>
</DirectoryServiceInformation>
```

DirectoryServiceInformation には利用すべきディレクトリサービスの情報が記述されている．つまり，ここで指定されたコンテキスト情報で足りない要素を探し出す際に 利用するディレクトリサービスがここで規定されている．この情報はそれが存在するコンピュータネットワーク上の位置と物理的な位置で示され，さらに，そのディレクトリサービスを管理する管理者の名称を記述している．また，ディレクトリサービス自体の更新時刻の情報も保持される．

ディレクトリサービスのコンピュータネットワーク上の位置情報は *DirectoryServiceLocation* によって表記している．この要素が持つ *hostname* と *portNum* によってディレクトリサービスのネットワーク上の位置情報が規定されている．ここに通信する事でコンテキスト情報の不足している情報の補完を行う．*DirectoryServiceLocation* には *myself* と言う属性情報が付加される．この情報は利用者が使うコンテキスト検知機構の変換機構が常に使うディレクトリサービスであるかを示す情報である．この属性はブーリアンで指定する．

DirectoryServiceRealLocation には接続先のディレクトリサービスの位置情報が記述される．この情報には *match* 属性が存在し，コンテキストの検知の際に扱われる．

Maintainer とはこのディレクトリサービスの管理者である．この利用者の情報は *Authenticator* と同様 E-Mail アドレスで示し，またこの要素にも *match* の属性が存在する．

timestampRlx とは このディレクトリサービスの起動時刻や更新時刻などの事項情報を指す．これには二つの属性が存在する．それを下記に示す．

- **at**

init ディレクトリサービスの始動時刻

lastMod ディレクトリサービスの更新時刻

- **tComp**

  now 表記された時刻である事

  before 指定した時刻より前である事を要求する

  after 指定した時刻より後である事を要求する

利用者は，*Maintainer* と，*DirectoryServiceRealLocation* と，*timestampRlx* の要素を確認する事で，このディレクトリサービスに登録された様々な機構からの情報を，コンテキストの検知時に扱うか否かを検討する．これらの情報は検索や要求の際，条件として指定できる．

4.2.3 データ管理機構:DataManagementEngine

DataManagementEngine の要素ではコンテキスト情報の検知に共ない扱うデータ管理機構へのアクセス方法を記述した情報である．この情報はコンテキスト情報につき一つとは限らない．複数のデータ管理機構から情報を扱う可能性が存在するため複数記述できる．この情報は検索時には通常記述されず，また要求時には必ず記述されていなければならない．検索時に扱いたいデータ管理機構が確定している場合は *match* 属性を *direct* とするが，通常これは無記入の *any* として扱う．本要素は以下のように示される．

```
<DataManagementEngine security="public">
<typeName
match = "比較方法"
><![CDATA[対象となるデータ管理機構の種類の名称]]></typeName>
<hostname>対象となるホスト名</hostname>
<portNum>接続するポート番号</portNum>
</DataManagementEngine>
```

この要素の構成は *DirectoryServiceLocation* と酷似している．しかし，決定的に違う点があり *myself* の属性が無い事と *typeName* の要素が存在している事と，公開対象を選択できる点だ．コンテキスト検知機構はデフォルトで扱うデータ管理機構と言う概念が存在しない．そのため，デフォルトの機構が存在しない．そして，C-AComp 環境ではデータ管理機構が単一の種類だけで賄え，単一のプロトコルで動くとは到底考えられない．そのため，管理機構毎にデータの要請プロトコルを変更できるよう，そのデータ管理機構の種類を明記しなければならない．それが，この *typeName* だ．それ以外は，インターネットを想定環境としている以上，接続先のホスト名とポート番号が必要となる．

4.2.4 事象解析メカニズム:ContextAnalyzingEngine

ContextAnalyzingEngine とは事象解析メカニズムを指す．事象解析メカニズム自身はその計算量が莫大になる事があるため，必ずしもコンテキスト検知機構の内部に存在しているとは限らない．そのため，事象解析メカニズムへの接続先を用意してやらねばならない．複数の事象解析メカニズムを混在させ，活用する事は想定していないため，*ContextAnalyzingEngine* は常に一つしか記述しない．本要素の記述例を下記に示す．

```
<ContextAnalyzingEngine security="public">
<typeName
match = "比較方法"
><![CDATA[対象となる事象解析メカニズムの種類の名称]]></typeName>
<hostname>対象となるホスト名</hostname>
<portNum>接続するポート番号</portNum>
</ContextAnalyzingEngine>
```

この要素の構成は基本的にデータ管理機構のものと何も変らない．事象解析メカニズムの種類を一意に示せる名称と，それへの接続方法が記述される．

4.2.5 コンテキスト検知のルール:contextCapRule

contextCapRule は二つのルールから構成されている．このルールとは，コンテキストの基となる事象を検知するために事象解析機構へ適応するルールと，事象からコンテキストに変換するためのルールである．

これらの情報は検索時には一切記述されない．これらを入手する事が検索を行う一番の理由であるからだ．また，コンテキストの要求時には必ず記述しなければ，コンテキストは入手できない．下記にこの記述方法を示す．

```
<contextCapRule>
<dataHandler><![CDATA[データ管理機構]]></dataHandler>
<analyzer><![CDATA[事象解析メカニズム]]></analyzer>
<profileType><![CDATA[解析ルール表記手法]]></profileType>
<profile><![CDATA[解析ルール]]></profile>
<resultHandler rowT = "同時刻の事象解析結果の合計"
reqAmountT = "返答の合計回数">
<dataElement
rowN = "列の番号" reqAmount = "解析の回数"
meanNum="コンテキスト変換ルールの番号"
match = "比較方法"
><![CDATA[比較すべき情報の雛形]]><dataElement>
</resultHandler>
</contextCapRule>
```

dataHandler はデータ管理機構の種類の名称を指し，そこからデータを取得する事を指定する．そして，*analyzer* は事象解析メカニズムを指し，これを指定する事で利用する解析機構を明確する．

profileType は事象解析メカニズムに提供する解析ルールの記述方法を規定し，*profile* に事象解析メカニズムに渡す解析ルールを提供する．この事象解析メカニズムのルールは一つのコンテキスト情報につき，一つしか記述できない．事象解析メカニズムはその結果を，同時刻に解析し終えた複数の要素として提供する．

resultHandler では，事象解析メカニズムが出力する情報をコンテキストとして扱えるようにその情報を変換するルールを記述する．*resultHandler* 自身には *rowT* と言う要素数の合計を自然数で表わす．また，*reqAmountT* で指定した回数だけ事象解析の返答を行う．ここでは零以下を指定する事で，下記で指定するすべてのコンテキストを入手するまでその処理を終らせなくできる．

そして，*dataElement* で記述した比較すべき情報の雛形を，そこに記述した属性情報に沿い処理する．この属性情報に関して下記に示し解説する．

rowN 自然数で指定した比較すべき要素の番号

meanNum 自然数で指定した比該当時に示すべきコンテキストの番号

match 雛形との比較方法

reqAmount 要求する回数

rowN で指定されて番号の事象解析メカニズムから出力される要素のデータを扱い，比較すべき情報の雛形と比較する．比較する手法は *match* の属性を基に行う．該当情報が真として出て来た場合，*meanNum* で指定した番号の，第 4.2.6 部にて後述するコンテキ

ストを利用者に提供できるようにする．なお，このコンテキストは直接示さず，検知した際に用いたコンテキスト情報の項目を多少変更するだけで返答する．これに関してはコンテキスト変換機構にて後述する．この事象解析は *reqAmount* で指定した回数だけ通達する．この回数が零になると *resultHandler* 回数に残量があろうと，このコンテキスト情報の処理は終了する．なお，この変換ルールは解析ルールとは違い複数個用意しても良い．

4.2.6 コンテキスト:contextMeaning

contextMeaning とは利用者にとって理解できるコンテキストである．コンテキスト-欲求バインド機構はこの情報を基にコンテキストを欲求に変更すべきだ．もちろん，コンテキスト情報に表記した位置情報や更新時刻と言った他の情報を基に欲求へと変換しても良いが，このその時には，このコンテキストを考慮すべきである．このコンテキストの表記方法を下記に記す．

```
<contextMeaning num = "コンテキスト変換ルールの番号">
<capTime>変換時刻・検知時刻</capTime>
<contextObjectPointer
match = "比較方法" ><![CDATA[対象の名称]]></contextObjectPointer>
<contextStatusPointer
match = "比較方法" ><![CDATA[対象の状態]]></contextStatusPointer>
</contextMeaning>
```

contextMeaning には *num* と言う属性があり，この番号を指定しなければならない．この番号は上述したよう *dataHandler* で指定される．つまり，この番号に検知されたコンテキストが要求者に通達される．検索を行う際は，この番号が無視される．そのため，記述してはならない．

このコンテキストが生成された時にその時間は *capTime* に記述する．

contextObjectPointer と *contextStatusPointer* にはコンテキストとして表記する情報を記述する．これらの情報は両者共に *match* の属性を保持しているが，その比較はコンテキストの検知時では無く，コンテキストの検索時に扱う．

4.2.7 コンテキスト情報のまとめ

このようにコンテキスト情報は数々の情報を内包している．具体的な XML は付録 A.1 に記述しているためそれを参考にして欲しい．このコンテキストで必ず指定しなければならない情報はあくまで，*DirectoryServiceLocation* と，*contextObjectPointer* と，*contextStatusPointer* だけである．しかも，デフォルトのディレクトリサービス以外の *DirectoryServiceLocation* はデフォルトのディレクトリサービスが情報を持っているため，そ

こから自動的に補完できる．つまり，利用者がかならず指定しなければならない情報とは，*contextObjectPointer* か *contextStatusPointer* の比較方法だけで，それに関する，もしくはそのコンテキストの利用方法が検索できるようになる．

4.3 XML ディレクトリサービス

コンテキスト生成機構がコンテキストを生成するためには，第 4.2 節で述べたコンテキスト情報の中に，様々な機構の情報を補完しなければならい．そこで，この情報を補完する機構が XML ディレクトリサービスである．

本節では，この XML ディレクトリサービスが C-AComp 環境において各種機構とどのような情報をやりとりしているかを解説し，その情報の通信プロトコルを解説する．

4.3.1 他のディレクトリサービスとの連携

他のディレクトリサービスとはインターネットを介してデータの通信を行う．しかし，これではすべてのディレクトリサービスからデータの入手ができてしまし，すべてのディレクトリサービスからデータの確認を行わなければならなくなる．そのため，ディレクトリサービス自身がその位置情報を確認し，他のディレクトリサービスと近づいた場合にのみ，そのデータの共有を行う．これ自身莫大な研究テーマとなるため，これは他の研究が実現した者を使うと想定している．既存のものでは GPS や AcitiveBadge[1] などがある事を解説している．そのため，位置情報は御互い把握していることを前提としている．

利用者は粒度を指定することで，第 4.2.2 部で解説した XML の部分要素を要請に応じて返答できなけばならない．そのため，本機構は自身の位置情報，コンピュータネットワーク接続情報，管理者情報と更新時刻情報以外に，自身の周囲で通信できるディレクトリサービスの通信先情報を持っていなければならない．他のディレクトリサービスの情報は動的に変更され，変更が起きる事があるため，それらへのアクセス方法しか管理しない．それ以上の情報が必要な場合，そのディレクトリサービスに接続し情報を入手すれば良い．

このようにディレクトリサービス同士自身の情報を登録できないといけないため，ディレクトリサービが他のディレクトリサービスに登録するための共通プロトコルが必要となる．これは，ディレクトリサービス発見以外のプロトコルとして用意し，そこにすべての外部からの機構の情報を受けとれるようにしなければならない．そして，ここで受け取るデータは各種 XML であるため，ディレクトリサービス自身が XML の検証を行えなければならない．

このように，ディレクトリサービスは他のディレクトリサービスと接続する手法を常に管理し，要請に応じてそれへの接続手法を伝達する．

4.3.2 データ管理機構との連携

データ管理機構は特定のディレクトリサービスに常に属し，そこにしか自身の情報を公開しない．例え外部のディレクトリサービスと遭遇してもその情報を新しく遭遇したディレクトリサービスに通達するわけでは無い．これは管理範囲をディレクトリサービス単位で分ける事によって，権限の分担が完全に分けられるからである．そして，何より，データ管理機構が管理するデータ出力機構の位置情報やそれらが設置された対象をデータ管理機構が管理するわけでは無いからだ．この情報はディレクトリサービスが管理し，その空域を管理する管理者がその情報を入力する．この情報は，第 4.2.1 部にて解説した，*objectXMLPointer* に記述する．これによって，対象の名称を利

用者が扱えるようになっている．もちろん，この対象の名称をいかに考慮するかはこの空域の管理者の腕にかかってしまっている．

本機構は第 4.2.3 部で解説した XML データをその要請に応じて返答できなければならない．そのため，管理者が *objectXMLPointer* の情報をディレクトリサービスに登録すると同時に，データ管理機構へのアクセス方法と種類を登録しなればならない．そして，この登録された情報を基に，本機構が要請に応じたデータ管理機構に関する XML データを提供する．

4.3.3 事象解析メカニズムとの連携

事象解析機構もデータ管理機構と同様に，それ自身は常に特定のディレクトリサービスに属す．そして，その事象解析メカニズムが解析した結果生成するコンテキストのうち，*contextStatusPointer* の情報をデータ管理機構同様管理者が登録したいが，これは管理者の意思では無く事象解析メカニズムを考案しそのルールを作成した者がその事象に対する情報を作るため，*objectXMLPointer* の様に登録されるわけでは無い．事象解析メカニズムのルールを考案した者はそれの出力する *contextStatusPointer* と同じ重み付けのある情報を考慮し，それを公開しなければならない．そして，それをコンテキスト情報に記述しなければならない．そのため *statusXMLPointer* はディレクトリサービには保管されている必要性は無い．しかし，この *statusXMLPointer* に記述する情報を公開したく無く限定された空間でのみ使われて欲しい場合に限り，管理者がその情報をディレクトリサービスに登録し，扱える．

本機構は第 4.2.4 部で解説した XML データをその要請に応じて返答できなければならない．そのため，管理者が事象解析メカニズムを運用させると同時に，この機構へのアクセス方法と種類を登録しなればならない．そして，この登録された情報を基に，本機構が要請に応じた事象解析機構に関する XML データを提供する．

4.3.4 コンテキスト変換機構との連携

コンテキスト変換機構は利用者が常にもち運ぶ．そして，ディレクトリサービスもこのコンテキスト変換機構と対で活動する．つまり，ディレクトリサービスはコンテキス

ト変換機構の一部であると言える．そのため，コンテキスト変換機構から大量の検索要求を受ける．

この要求は通常単一のコンテキスト変換機構からだけだが，環境に設置されたディレクトリサービスなどは利用者が持つコンテキスト変換機構から情報の要請が行われる事もある．しかし，逆に利用者の持つディレクトリサービスには他者のコンテキスト変換機構から接続されて欲しく無い場合がある．そのため，信頼できるコンテキスト変換機構を登録し，それからだけ情報の要請を受けられようにならなければならない．

そのためコンテキスト変換機構へのアクセス方法とその利用者の名称を保持しそこからだけアクセスできるような利用者の認証が行えるようにしなければならない．

4.3.5 ディレクトリサービスとの通信プロトコル

ディレクトリサービスは様々な機構から情報の登録要請が行なわれる．この登録は自動的に各種機構が行うのが望ましいが，それでは管理範囲を規定する手法が無くなってしまう．そのため，他のディレクトリサービスとコンテキスト変換機構からの要請以外は，管理者の登録によって行なわれる．この登録はTCP/IP を用いて行う．また，本機構であるXML ディレクトリサービスはTCP コネクションを受け付ける．推奨するポート番号を 10323 とする．

各種機構の名称は長いためそれらの略称を用いる．下記の表4.3.5 に各種命令で使われる名称と略称の対応表を記述する．

名称	プロトコル内略称
ディレクトリサービス	DS
事象解析メカニズム	CAE
データ管理機構	DME
コンテキスト変換機構	CITE
検知対象の名称に関する XML	OXMLP
検知対象の状態に関する XML	SXMLP

表 4.1: 名称略称

この略称を指定する事で各種情報の登録と入手を行う．入力するデータはすべてXML であるため，データの入出力はコマンドの後にすべての改行コードを外したファイルのサイズを指定し，その改行コードを外した情報を改行コードまで送受信する．

下記にXML ディレクトリサービスにおけるデータの入出力プロトコルを下記に示す．

- 認証が必要な情報の入力及び変更

  setMyDSMaintainer 自身の管理者に関する情報の XML を受け付ける

  setMyDSIXML 自身のディレクトリサービスに関する情報の XML を入力する

setMyDSIRLXML 自身のディレクトリサービスに関する位置情報の XML を入力する

setMyDSILXML 自身のディレクトリサービスに関するネットワーク情報の XML を入力する

setCITEXML 自身が信頼するコンテキスト変換機構の情報の XML を入力する

setXML:name CAE，DME，OXMLP，SXMLP を指定することでその情報の XML を入力する

- 認証の必要が無い情報の入力及び変更

  putDSIXML 他機構がディレクトリサービスに関する情報の XML を入力する

- 情報の出力

  getFileNameAndNumberOf:name ディレクトリサービスに保存されている CAE，DME，OXMLP，SXMLP を name で指定することで，種類の名称と保存されている数を CSV で出力する

  getXML:name:number:typeName ディレクトリサービスに保存されている各種機構の情報の XML を機構の名称，番号，そして種類があれば種類を指定する事で入手する

情報の入力には認証が必要な情報が存在するが，その出力には認証が存在しない．なぜなら，他者に出力させたくない情報はそれの *security* を *private* としており，*setCITEXML* で登録したコンテキスト変換機構にしか公開させていないからだ．これ以上複雑な認証は分散環境であるがゆえ行わない．

このようになプロトコルを用意する事で各種機構と情報のやりとりを行う．

4.3.6 XML ディレクトリサービスのまとめ

XML ディレクトリサービスは指定された空域に存在する様々な機構を管理する．多くの機構の登録はそれを管理する管理者が行い，その利用は特定のコンテキスト変換機構からしか許可しないか，すべてに公開するかの二択を選択させた．そして，プロトコルを規定し他の機構の情報を保持できる事を示した．

4.4 コンテキスト変換機構 CITE: Context Information Transition Engine

コンテキスト変換機構は，事象解析メカニズムが検知した事象を複数の利用者が扱えるよう，コンテキストとして情報を変化させる役割も持つ．また，コンテキスト情報を用いた利用者からのコンテキスト要請とコンテキスト情報の検索要請に答える役割りを担っ

ている．本節ではこのコンテキスト変換機構を Context Information Transition Engine(以下 CITE) として構築する．本節では CITE が他機構とどのように連携しているかを解説し，送られて来るコンテキスト情報からどのような処理を行いコンテキスト情報の返信か，コンテキストの返信を行うか解説する．それがコンテキスト情報をどのように扱い各種機構と連携するか述べる．

4.4.1 CITE の概要

CITE は第 3.4.2 部のフレームワークにて解説したコンテキスト変換機構の一例である．本機構はこれを第 4.2.1 部にて指定したコンテキスト情報と第 4.3 節で解説した XML ディレクトリサービスと，既存に存在するデータ管理機構の一例である TinyDB と事象解析メカニズムを用いて構築する．本機構は利用者が用いたコンテキスト情報でコンテキストが生成できたもののみを保管することで，他人がこのコンテキスト情報を検索できる．

本機構は利用者がコンテキスト情報を本機構に挿入する所から始まる．ここで受け取るコンテキスト情報とは検知要求か検索要求であり，その処理方法はまったく違う．そこで，それらの処理を次の部から解説して行く．

4.4.2 コンテキストの検知要求

コンテキストの検知要求はコンテキスト情報を CITE に送信する事から始まる．この際，コンテキスト情報のメイン要素である *ContextInformation* の *citype* が持つ属性を *query* と指定することで要求が行なわれる．コンテキストの検知に必要な情報はこの *ContextInformation* の中にすべ記述されなければならない．

この際，より誤検知が少いコンテキストを検知させるには，コンテキスト情報のすべての要素と属性の情報を記述する事が望ましい．すべての条件を埋める事で，すべての要素が完全一致しない限りそのコンテキスト情報で指定されたコンテキストは検知されなくなる．つまり，検知される可能性が低くなるため，誤検知が少なくなるとも取れる．この逆は必ずしも真とは言えない．それは，このコンテキスト情報を完全に指定しそれを検知したとしても，その情報が必ずしも利用者が頭の中で描いたコンテキストであるとは言い切れないからだ．そのため，誤検知が多少増えようとも，より柔軟な表記の方が良い．そのため，コンテキスト情報の中でどの情報が必要で，どの情報は不要かを明確にする．

CITE にコンテキストを要求する際，一概に特定の要素は記述しなくても良いと言うわけでは無い．CITE ではコンテキスト情報を確認する順序がある．CITE がコンテキスト情報を受けとり始めに確認する事項は，*contextMeaning* と *contextCapRule* の項目である．これらの項目がすべて埋まっており，*contextMeaning* の *num* 属性にダブリが無い事を確認し，*contextCapRule* の *dataElement* の *meanNum* が確認された *num* の番号のどれかに一致しているかを確認する．項目が足りない場合 *MISS_ELE:* とし，その後に足りない項目の名称が記述され，コンテキスト情報の要請者に通達される．また，*DataManagementEngine*

と *contextCapRule* 内部の *dataHandler* が矛盾していないかを確認する．また上記と同様，*ContextAnalyzingEngine* と *contextCapRule* 内部の *analyzer* が矛盾していないかを確認する．これらが矛盾している場合，*conflict:* とし，その後に足りない項目の名称が記述され，コンテキスト情報の要請者に通達される．

これらの項目が正しいとして確認されると次に利用する XML ディレクトリサービスが確認される．C-AComp 環境ではすべての作業は利用するディレクトリを決める事から始まる．つまり，下記に示す *DirectoryServiceInformation* の事項の情報がコンテキスト情報に記述されているかを確認する作業から始まる．

- *myself* 属性
  デフォルトのディレクトリサービス
- *DirectoryServiceLocation*
  ディレクトリサービスのネットワーク上の位置情報
- *DirectoryServiceRealLocation*
  ディレクトリサービスの位置情報
- *Maintainer*
  ディレクトリサービスの管理者名称
- *timestampRlx*
  ディレクトリサービスの始動・更新情報

これらの情報で異なる情報がある時点で指定されたコンテキスト情報は使われず，コンテキスト情報の要請者に *NF_DS* と返信を返す．空要素である場合，その項目は「何でも良い」と言うことなので比較は行わない．空要素を指定したければ明示的に指定しなければならに．

では比較対象となるディレクトリサービスの情報は何時入手るのか？それは，上記の項目がすべて CITE が利用しているデフォルトの *myself* ディレクトリサービスで無い事を確認し次第，デフォルトのディレクトリサービスから情報接続できる情報を入手する．ここで入手した入手したディレクトリサービスの数だけ検索はスレッド化し，利用すべきディレクトリサービスかを確認する．該当するディレクトリサービスである場合処理が続き，該当しないディレクトリサービスの場合，自身のスレッドが残り何番目の生存しているスレッドかを要請者に通達し，そのスレッドを終了する．項目が該当する場合処理は続く．

次に接続先のディレクトリサービスから *OXMP* を入手し *contextMeaning* の *contextObjectPointer* の項目が存在するか確認する．ここで *OXMP* が存在しなければ *OXMP_ObjectNF* を返すすスレッドを終了する．もちろん，この終了の際には他にあと何本の検知スレッドが走っているかを返す．*contextObjectPointer* に指定されている要素が存在しない場合は，*cop_ObjectNF* を返し，*OXMP_ObjectNF* を返す時と同様にスレッドを終了する．

そして，最後にこれらのデータ管理機構と事象解析メカニズムをこのディレクトリサービス内で探し，見付けたら事象解析メカニズムにデータ管理機構の場所を伝え，*profile*

を挿入する．データ管理機構が見付からなかった場合は，*DME_NF* を返し，事象解析メカニズムが見付からなかった場合は，*CAE_NF* を返す．挿入が成功した場合挿入が成功した旨を伝え，それが指定されたスレッドの何番目であったかも伝える．

これによって事象解析機構にデータ管理機構との連携作業は任せ，処理されて来たデータを *contextCapRule* の *dataElement* に応じて比較し続ける．比較が該当する場合，コンテキストとなる情報を *querySuccess* を利用者に返答し，*reqAmountT* と *reqAmount* で指定された回数に応じて処理を続ける．これにてコンテキストの検知が成立する．

4.4.3 コンテキストの検知成立

コンテキストの検知が成立した時に，*querySuccess* を送る時のコンテキスト情報には，*reqAmount* の情報を減らしてやり，*ContextInformation* の *citype* を *ci* とし，この情報をもたらした各種機構の情報を書き入れ，*contextMeaning* の *capTime* を更新し返信する．その時，*ContextInformation* の *security* 項目を確認しそれが *public* である場合，その情報を公開しても良いとみなし，CITE 内の共有コンテキスト情報保存ディレクトリにこのコンテキスト情報のコピーを起く．この項目が空欄である場合，それは *private* であるとみなしコピーしない．これによって，この共有ディレクトリにコンテキスト情報が溜まる．この仕組みは善意に基づきなりたっているが，外部に自身のコンテキスト情報を晒したく無い場合は，自身の信頼するディレクトリサービスだけを指定し，利用すれば良いだけだ．

4.4.4 コンテキストの検索

次にコンテキストの検索を行う．この作業を行うには，*ContextInformation* の *citype* が持つ属性を *search* とし，最低限下記のどちらかの項目が記述していれば良い．

- contextObjectPointer
  検知対象の名称
- contextStatusPointer
  検知対象の状態

要素を指定していない場合はどのような条件でもマッチすると見做すため，コンテキストの検知要求と同じ順序で各種項目を検索し，ディレクトリサービスに保存され，該当するコンテキスト情報の項目を *ContextInformation* の *citype* が持つ属性を *searchR* と変更するだけで要求者に返答する．

4.4.5 コンテキスト変換機構 CITE のまとめ

このように，CITE はコンテキストをコンテキスト情報の項目を指定する事で要請できる．

4.5 事象解析メカニズム:SPICeS

本節ではコンテキストの事象解析メカニズムの一例である SPICeS[1] [6] を解説する．SPICeS は特定のデータ出力機構から出力される数値変動のパターンを特徴として捉え，それを特定の事象として扱うメカニズムである．本節ではこのパターンとは何かを解説しその解析手法を解説する．そして，その解析手法を用いて実装例を解説する．

4.5.1 パターン解析による事象解析

事象解析メカニズムはデータ管理機構の管理するデータ出力機構からデータを入手しそれを解析する．この解析とは，データにおけるパターンを検知し，そのパターンを特定の事象として扱う事を示す．現在，パターン認識は心理学の分野で広く研究されている．心理学では文字，絵，図形，音声などを利用して鋳型照合モデルか特徴抽出モデルにてパターン認識を行う．事象解析機構も入手したデータをこのパターンとして捉え扱えるようにする．このパターンの検知方法には特徴抽出モデルと鋳型照合モデルがある．これらのモデルを下記で説明する．

> 鋳型照合モデル:
> 入力パターンを，あらかじめ貯蔵されている典型的なパターン (鋳型) と直接的な照合によって認識すると仮定する．
>
> 特徴抽出モデル:
> パターンを構成する基本的な特徴 (示差的特徴とよばれる) をまず抽出して，それを組み合わせて認識すると仮定する．

特徴抽出モデルを利用しデータ管理機構が管理する特定の種類のデータから，特定の数値の変動を捉える．そして，その変動の中でも特徴のあるものだけを捉える．この特徴のあるデータの例を図 4.1 に示す．このようにデータの変位に一環した特徴が見られることでその区間を特徴的であるとして切り出す．つまり，限定をした時間内で入手したデータサンプル内で生じる数値変動を常に監視することで，その範囲内で生じた変化を任意のアルゴリズムで捉えられる．この変動を捉えることで，その時間範囲内で特徴がある変動が生じたとして分別できる．このように，データが起こす数値の経緯にパターンを求めることで特徴が抽出できる．

しかし，この手法では簡易なアルゴリズムを利用すると莫大な量の特徴が抽出され，結果自体は特徴のあるパターンとは言えなくなる．逆に，複雑なアルゴリズムを利用すると，その計算の絶対量が増え，演算に時間が掛り実時間内に計算を終らせる事が難しくなってしまう．そのため，計算量が複雑過ぎず，計算対象の量が多す，しかし検知アルゴリズムが簡単過ぎないものを考慮しなければならない．

[1] Situation Pattern Information CapturE and Sample

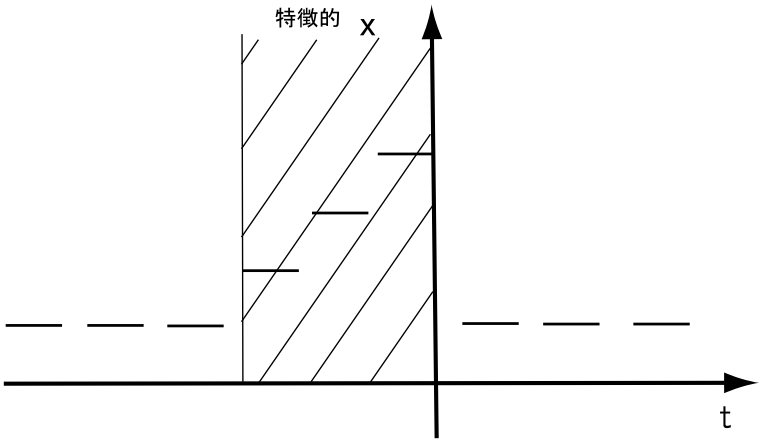

図 4.1: 特徴のある数値情報の推移

4.5.2 スライディングタイムウィンドウを用いた特徴抽出

スライディングタイムウィンドウとは，TCP/IP で用いられるスライディングウィンドウが如く逐次入手するデータを一定の時間で区切った一時領域であるウィンドウに溜め解析する解析手法である．なお，指定した時間まではデータを溜めなければならなく，それまで解析はできない．この指定した時間をウィンドウのサイズと呼ぶ．また，指定した時間以上の時間が立つとしてされた粒度において過去のデータがウィンドウから消え，その分入手する新しいデータが溜められて行く．この様子を下記の図 4.2 に示す．

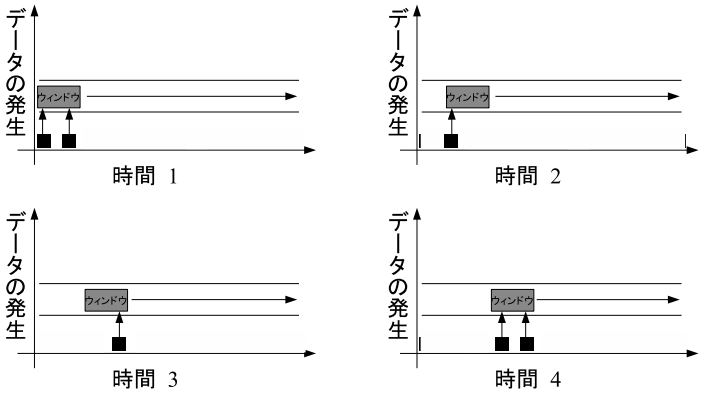

図 4.2: スライディングタイムウィンドウによるデータの管理

事象を解析するにはこの解析手法が有効となる．このウィンドウ内で生じる情報から，それを特徴として保存し，再度同じウィンドウを同じ種類のデータにかけることで，同じ特徴が出ると期待できる．そこで，下記に示す情報をそれぞれのウィンドウで検出し

これを特徴として残す.

- 初期データ入力値
- 最終データ入力値
- 最大値
- 最小値
- 平均値
- データの個数
- 経過時間

初期値と最終値が最大値と最小値である場合，そのウィンドウ内のデータの推移の方向が上昇傾向な事象やか下降傾向の事象か取れる．また最大値，最小値が共に同じであるか，または平均値から指定された閾値からの範囲に両者があれば変動が無い事象を検知できる.

この事象の解析手法は非常に単純であるため，その事象自体何の意味を持たない．しかし，この事象を複数個の異る範囲の時間を指定したウィンドウを使い，これらの情報を残すことでベクトルの方向と，そのデータ変動の範囲が残せるようになる．このベクトルと範囲を残す事が SPICeS の解析手法である.

検知するできる事象はウィンドウのサイズとその個数を変更する事で大幅に変動してしまうため，時間内に入手したサンプル数はできる限り同じである事が好ましい.

また，このデータの変動はデータ出力機構が設置された場所に強く異存する．そのため，この解析手法は様々なデータに対して適応できるが，出力傾向が変動しすぎ，一定の期間同じベクトルの情報が出ない機構には向かない.

このように，ウィンドウを規定する事で実時間で限定された情報量の計算しか行わない事が保証される．そのため，解析にかかる時間と計算量を事前に予測でき，事象解析が実時間でできるか予測がたてられる.

4.5.3 事象解析メカニズム SPICeS の実装例

本節では 2002 年度に開かれた ORF で展示た事象解析メカニズム SPICeS について解説する．SPICeS で捉えた事象を再検知可能なコンテキストとして扱い，AS である仮想 iPAQ 電話の動作モードを変動させた.

仮想 iPAQ 電話は以下の構成で構築した.

- PC:iPAQ 3660 x2
- OS:Familiar Linux

- Network: IEEE802.11b

この仮想電話には 3 つのモードが存在し留守録，通常，自動転送と備え，利用者の行動によってそれをそれぞれの行動に関連付けた．

利用者の事象は事象解析機構 SPICeS を用いて捉えた．なお，ORF2002 では状況適応型アプリケーション支援機構 SPICeS として命名していた．SPICeS は以下の環境で構築した．

- PC:ThinkpadX20
- OS:FreeBSD4.x
- Sensor:adxl202
- Network:IEEE802.11b

adxl202 とは加速度を計測するチップであり，それをシリアル上の基盤に乗せ，そのデータ変動に応じて事象を解析した．実際に利用者の足にこのチップを装着させ移動中のデータを複数のウィンドウに溜めさせた．そして，利用者が仮想電話のモード切り替えを押すと，そのモードを切り替えた時の事象を利用者行動時の欲求として保持し，同時にウィンドウ内に溜めた事象の特徴も保存した．そして，この特徴と同じ傾向のある動作を起した時に，切り替えで指定したモードに自動的に変更させた．具体的に検知された事象は，以下に示す三つの行動状態である．

- 静止している
- 歩いている
- 走っている

この行動状態を SPICeS は確認できる事を実証した．

4.5.4 事象解析メカニズム:SPICeS のまとめ

事象解析メカニズムは数値変動のパターンをスライディングタイムウィンドウを用いる事で解析するメカニズムである．このメカニズムを使うことで，静止する，歩く，走ると言った行動のパターンを事象解析の結果取れる事を実証した．

4.6 事象解析メカニズム:TinyDBClient

TinyDBClient は TinyDB が備えているデータの処理機能を扱い，その処理結果を CITE が扱える形式に変換する機能である．TinyDB へ単純な SELECT で利用したいセンサのデータを選ぶだけでは無く，IF や符号を用いた制御文に従い条件付けされた情報の結果を出す．

この機構は SPICeS とは違い，同刻に出力された複数のデータを解析する事に優れている．本機構は検知すべき対象に付随された mote のノードを指定し，検知対象の特定を行う．本機構は TinySQL で指定した SQL 文を CITE に渡すだけの機構である．しかし，この受渡しを行わない限り，複数人が同一の解析結果から相反するコンテキストの情報を生成することなどできない．そのため，TinyDBClient が生成された情報を利用者に応じて扱えるよう CITE に受渡す．

この受渡しに使われる情報は JAVA の VECOTR 型である．そして，中身は常に数値となっている．この解析結果を CITE で扱うことで，コンテキストを TinyDB から生成できるようになる．

第5章

評価

本章では，本研究で構築したコンテキスト検知機構を用いた C-AComp 環境のためのフレームワークの肝であるコンテキスト変換機構を定性的に評価する．

5.1 定性的評価

本研究で提案したフレームワークを既存のコンテキストアウェアアプリケーションのフレームワークと比較し，シナリオを用いてそれが実現できる定性的な見解で評価する．

始めに比較に用いるシナリオを解説する．そして，比較すべき項目を提示しその比較結果を提示する．最後に，この比較内容からの考察をの述べる．

- 評価用のシナリオ

  A，B，Cの三人の利用者がオフィスを活用し，全員の環境はほぼ同じだとする．このような環境下で，コンテキストに応じたASの制御をそれぞれが行いたいと考える．この際，AとBは同じデスクワークを行うため行動パターンが非常に似ている．しかし，Cは接客業を行うため行動パターンがまったく違う．

  そして，Aはセンサの設置場所を把握しており，この部屋の機器情報をすべて管理しているため様々なルールを考案でき，足りない機材を導入することもできる．Bは機械音痴でまったくコンテキストの検知ルールが作れず，他人の作ったルール，特にAの作った検知ルールを流用している．Cは中途採用で以前他の環境でAと同じように機器を管理していたことがあるため，自身でルールを作れる．

  このような想定の場合，AとBのコンテキストはCものとは違い，Cは独自のコンテキスト検知ルールを次々と考案する．Aはコンテキストを既存の機器から作る事は少なくなり，より多くの機材を導入して，コンテキストAに最適なコンテキストを作り出す．Cは設置された機材の情報を吟味し，それに応じてコンテキストのルールを作る．BはCが来る前までは環境に残されたルールを使うだけだったが，Cの作ったルールが肌に合わず，困惑するようになる．

この例は非常に抽象的で具体的なコンテキストを何一つ書いていない．しかし，特定のコンテキストの検知ルールとは特定の人にしか使えなく，その項目を評価する事はおかしい．そこで，上記のような環境を想定した場合に起きる検討項目を下記まとめ，それらの事象が想定されて研究されたかを評価項目として取り挙げる．

- 評価項目

  A 他人が作り出したコンテキストの生成ルールを共有すると想定しているか

  B 利用者によって相反するコンテキストが存在する事を想定しているか

  C コンテキストの自動生成ができるか

  D コンテキストが常に正しい情報では無いと想定しているか

  E プログラマーにコンテキストの利用インタフェースを提供しているか

  F 単体で事象の解析が行えるか

下記に評価対象となる関連研究を提示する．

- 評価対象の関連研究

1 TEA

2 C-MAP

3 Personal Positioning System

4 Context Information Service

5 Context-Toolkit

6 Life Patterns

7 SPICeS(単体)

8 TinyDB

9 本機構:CITE

これらの評価項目に基き関連研究と定性的な評価を表 5.1[1] に述べる．

表 5.1: 定性的評価

	A	**B**	**C**	**D**	**E**	**F**
1	○	×	○	×	×	○
2	○	×	×	×	×	○
3	○	×	○	×	×	○
4	○	△	×	×	◎	○
5	○	×	×	○	◎	○
6	×	×	◎	×	×	○
7	×	×	△	×	×	○
8	○	△	×	×	◎	○
9	○	○	×	○	○	×

以降評価項目の考察を行なう．

- 他人が作り出したコンテキストの生成ルールを共有すると想定しているか．
  コンテキストの解析手法を考案するのは難しい．そのため，できる限り他人が作ったコンテキスト検知機構を使い，その考案労力を減らしたい．そのため，コンテキストの結果がより汎用的に扱えるよう検討された研究ほど，そのルールの共有が容易に行える．殆どの関連研究は複数人で作られた機構を使う事を想定している．しかし，SPICeS や LifePattern は利用者に特化し，その人だけが扱える過去の履歴情報を蓄え，その人にだけ役に立つコンテキスト生成している．本機構はこの項目を実現している．

[1] ×を出来ない．△をできるが歪曲した解釈，○を出来る，◎を非常に優れている．

- 利用者によって相反するコンテキストが存在する事を想定しているか.
  通常計算機科学で矛盾する答えば許されない．そのため殆どの関連研究は矛盾するコンテキストの存在を考慮して作っていない．TinyDB と CIS はその構造上，複数人が同じ解析を行い，その結果の捉え方は各自に任せられているため，その任せられた部分で矛盾が生じられる．本機構はこの要件を実現し，複数人が同じ事象解析を行っても，その結果を使うべき利用者を指定する事で，選べるため，矛盾の中から扱いたいコンテキストだけを取り出せる．

- コンテキストの自動生成ができるか
  PPS や，Life Patterns や，SPICeS や，TEA はすべて利用者がセンサを持ち運ぶ事でコンテキストを生成する機構である．そして，それらは特定のコンテキストを自動的に生成する事を目的としているため実現している．本機構はこの機能を実現していない．代わりに，他人が作成したルールを検索する形で利用者が知らないコンテキストの検知方法を探す手法を提案している．

- コンテキストが常に正しい情報では無いと想定しているか
  これは利用者によって相反する情報では無く単にコンテキストの誤検知の考慮を示している．Context-toolkit と本機構はこの事を危惧している．他機構は入手した情報が必ず正しい情報であるとしている．

- プログラマーにコンテキストの利用インタフェースを提供しているか
  CIS と TinyDB は既存の SQL 記述方式に酷似した方式をインタフェースとして提供しているため非常に扱い易い．また，Context-Toolkit もプログラマーにコンテキストを使ったプログラミングを行えるよう研究しているため，JAVA によってコンテキストを検知する API が作成されている．本機構は XML データを提出するに留まっているため，上記手法ほど充実したインタフェースを提供していない．また，何よりも独自の XML と，外部で動作する事象検知メカニズムに情報を渡す手法までこの XML に記述しなければコンテキストを入手できないため，扱い易いインタフェースとは言えない．

- 単体で事象の解析が行えるか
  本機構は単体だは何もできない．本機構は様々な機構と連動して動く事で効力を発揮する構成であるためこのような結果となる．

- 評価のまとめ本章ではコンテキスト関連研究も提示して本機構を定性的に評価した．本機構を用いることで矛盾が生じるような利用者が同一の環境でコンテキストを利用できる事を想定している研究が少ない事を示し，その想定に対応した既存研究がほぼ無い事を示した．また副産物てきな結果として，利用者に密着し解析する機構はコンテキストの自動生成に強いと判明した．

第6章

結論

本章では，今後の課題および展望について触れ，本研究についてまとめる．

6.1 今後の課題及び展望

C-AComp 環境において，複数の利用者が相反する事象をそれぞれの観点のコンテキストとして表記できるようになった．しかし，これによって，新たな課題が生まれて来た．本節ではこの課題及び展望に関して解説する．

6.1.1 コンテキスト-欲求バインド機構との連携

本研究のフレームワークではコンテキスト-欲求バインド機構に関してその構築方法を解説した．しかし，実際の構築は本研究では行っていない．そこで，今後の課題としてこのコンテキスト欲求バインド機構を構築し，利用者の作り出したコンテキストの矛盾を見付けだせるコンテキスト-欲求バインド機構を構築し，それと連携しコンテキストを作れるようにしなければならない．

6.1.2 共通のスキーマ

現在，コンテキストは事象の名称と状態をコンテキストとして扱っており，それらの名前は利用者が個々に名付けている．そのため，管理範囲が限定され，利用者が限られている場合は良いが，それがより多くの者に公開されると，名前に嗜好が出すぎて問題が起きる．特に異文化の考え方を持つ利用者がそのコンテキストを使い始め，CITE に登録し始めると棲み分けができるとは言え共通事項の多くを異種の情報として扱ってしまう．そのため共通意味を物情報間の繋がりを作れるよう共通の異文化間でも扱える共通のスキーマを考慮しなければならない．

6.1.3 各種ルールの自動生成

現在コンテキストの検知ルールは利用者が考案し，それを使った場合 CITE に残る．これではその考案ができる利用者が多大のルールを書いて行かなければより多くのコンテキストの検知ルールが作れない．SPICeS はその補助を行い限定された行動のコンテキストを取得するが，それでは利用者のすべてのコンテキストは捉えられない．そのため，コンテキスト変換ルールは事象解析ルールの自動生成を行う機構，もしくはその保持機構の構成を検討して行かなければならない．

6.1.4 より高度な事象解析メカニズムの検討

事象解析機構が非常に単純なものだけを導入しているので，さらに複雑な解析機構の導入を検討しなければならない．しかし，逆に複雑すぎる機構であると，それ単体で事象では無くより高度なコンテキストを検知できるようになる．そのため，より高度な事象解析機構との連動を視野に入れ機能拡張を解こさなければならない．

6.1.5　CITE の出力する情報からサブコンテキストの生成

CITE の出力する情報はそれ自身で完結してしまっているため，その情報を再度 CITE 自身が活用し，さらに高度なコンテキストの生成が行えるよう機能拡張して行かねばならない．しかし，この機能拡張を行うと，無限にループしてしまうルールを構築できてしまう．例えば,「暑い」と「蒸している」ならば「苦しい」と検知し，さらにそこに「雨が振っている」と検知すると「蒸している」として検知されてしまいかねない．このようなループになってもその解析を続けないような仕組みを導入できなければ，自身の出力する情報を再帰して扱えない．

6.1.6　今後の課題及び展望のまとめ

本研究を通して他人の検討したコンテキストのを共有できるようになったが，コンテキストの検知の問題は解決されていない．今後はより多くの検知手法を考案し，その情報をまとめ，共通化できなければ，利用者により多くのコンテキストを提供して行くこ事は難しい．

6.2　まとめ

本論文を読み通すことで，今後のコンピューティング環境には，コンテキストを活用したコンテキストアウェアコンピューティング環境の必要性を理解しただろう．しかし，この環境を作るには数々の課題をこなすため，本論文にて解説したフレームワークの重要性も理解していただけたと思う．その中でも，他人をコンテキストを共有することの難しさは，顕著に現れていただろう．

また，コンテキスト検知機構を用いたコンテキストの生成課程を見てもわかると思うがコンテキストは人間にとって非常に単純な情報を検知するために，計算機は多大な労力をかけて構築している．

本論文を読むことで，現在のユビキタスコンピューティング環境からコンテキストアウェアコンピューティング環境へ移ることを想像して頂けただろう．また，今後の課題に述べたように，さらなる事象解析機構の考案やその発展と標準化，そして，それらの結果をさらに解析する事でより人間の検知しているコンテキストに近い情報を生成して行くだろう．そして，C-AComp 環境をだれもが使い，それが機械によって支援されている事が気が付かなくなるほどこの分野は発展して行くであろう．

謝辞

本論文の執筆にあたり，御指導頂いた慶應義塾大学環境情報学部教授の徳田 英幸博士を始め，本論文の副査としてご助言頂いた慶應義塾大学環境情報学部教授の清木 康博士，慶應義塾大学大学院政策・メディア研究科助教授高汐 一紀博士 に深謝する．

特に，慶應義塾大学大学院政策・メディア研究科専任講師の中澤 仁博士には研究の議論のために時間を割いて頂いた．また，慶應義塾大学大学院政策・メディア研究科博士課程の永田 智大氏，同大学院同研究科博士課程の岩本 健嗣氏，慶應義塾大学環境情報学部の榊原 寛氏には絶えざる励ましと御指導を賜わった．ここに感謝の意を表し記し残す．

2004 年 1 月 14 日

堀江 裕隆

参考文献

[1] Andy Harter, Andy Hopper. A Distributed Location System for the Active Office. In *IEEE Network*, January 1994.

[2] Anind K. Dey, Daniel Salber and Gregory D. Abowd. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Anchor article of a special issue on context-aware computing in the Human-Computer Interaction*, Vol. 16, No. 2-4, pp. 97–166, 2001.

[3] Brian Clarkson. *Life Patterns: structure from wearable sensors*. PhD thesis, Massachusetts Institute of Technology, 2002.

[4] D. Garlan and D. Siewiorek and A. Smailagic and P. Steenkiste. Project aura: Toward distraction-free pervasive computing, 2002.

[5] Glenn Judd and Peter Steenkiste. Providing Contextual Information to Ubiquitous Computing Applications. Technical report, Carnegie Mellon University, 2002.

[6] Hirotaka Horie. 状況適応型アプリケーション支援機構 SPICeS, 2002.

[7] Hiroyuki Kanemitsu. POIX: Poin Of Interest eXchange Language Specification, 1999. World Wide Web Consortium.

[8] Hot Node, 2001. http://www.i-node.co.jp/.

[9] J. Hill and D. Culler. A wireless embedded sensor architecture for system-level optimization, 2001.

[10] Jin Nakazawa, Yoshito Tobe, Hideyuki Tokuda. On Dynamic Service Integration in VNA Architecture. *IEICE Transaction on Muti-dimensional Mobile Information Networks*, pp. 1610–1623, 2001.

[11] John Daugman. How iris recognition works. Technical report, Iridian Technologies Inc., 2001.

[12] Katsunori Shindo, Noboru Koshizuka, and Ken Sakamura. Ubiquitous Information System for Digital Museum using Smart Cards. In *SSGRR*, January 2003.

[13] Merriam Webster. *Merriam-Webster Dictionary*. Merriam Webster, 2003.

[14] Michael Beigl and Hans Gellersen. Smart-Its: An Embedded Platform for Smart Objects.

[15] Mike Botts. Sensor Model Language(SensorML) for In-situ and Remote Sensors. Discussion Paper, Open GIS Consortium, 2003. Open GIS Consortium.

[16] Misha Wolf, and Charles Wicksteed. Date and Time Formats, 1997. World Wide Web Consortium.

[17] S. Albrecht,K.A. Aidoo, T. Antti, T. Urpo, L.V. Kristof, V.V. Walter,. Advanced Interaction in Context. In *1st International Symposium on Handheld and Ubiquitous Computing (HUC99)*, pp. 89–101, 1999.

[18] Sam Madden, and Wei Hong. TinyDB:In-Network Query Processing in TinyOS. Technical report, Intel-Research Berkeley and UC Berkeley, 2002.

[19] Simon Cox,Paul Daisey, Ron Lake, Clemens Portele, Arliss Whiteside. OpenGIS Geography Markup Language(GML) Implementation Specification. Discussion Paper, Open GIS Consortium, 2003. Open GIS Consortium.

[20] Tadashi Okoshi,Shirou Wakayama,Yousuke Sugita, Soko Aoki, Takeshi Iwamoto, Jin Nakazawa, Tomohiro Nagata, Daichi Furusaka, Masayuki Iwai, Akihiko Kusumoto, Noriyuki Harashima, Jun'ichi Yura, Nobuhiko Nishio, Yoshito Tobe, Yasushi Ikeda, and Hideyuki Tokuda. Smart Space Laboratory Project. In *International Workshop on Networked Appliances*, 2001.

[21] 永田智大, 西尾信彦, 徳田英幸. サービス利用状況の変化に対する適応支援機構. 情報処理学会 システムソフトウェアとオペレーティングシステム研究会, pp. 1–8, 6 2002.

[22] 角 康之, 江谷 為之, Sidney Fels, Nicolas Simonet, 小林 薫, 間瀬 健二. C-MAP: Context-aware な展示ガイドシステムの試作. 情報処理学会論文誌, Vol. 39, No. 10, pp. 2866–2878, 10 1998.

[23] 荒川 弘熙（編）. Jiniって何だ？ Java がもたらす近未来のネットワーク技術, 第 1 章. 株式会社 カットシステムス, 1999.

[24] 青木 恒. ウェアラブル・コンピュータ向けリアルタイム Personal Positionin System. 情報処理学会論文誌, Vol. 41, No. 9, pp. 2404–2412, 9 2000.

[25] 石崎 峻. 自然言語処理 , pp. 68–69. 情報系教科書. 昭晃堂, 1998.

[26] 堀江裕隆, 岩本健嗣, 永田智大, 西尾信彦, 徳田英幸. Wearable Network における状況パターン抽出機構. 情報処理学会 第 61 回 平成 12 年度後期全国大会 講演論文集, 第 3 巻, pp. 445–446, 東京, 2000.

付 録A

付録

A.1 コンテキスト情報

デフォルトのディレクトリサービスを用いたコンテキストの要求としてのコンテキスト情報.

```
<?xml version="1.0" encoding="UTF-8"?>
<ContextInformation security="public" citype="query">
<Authenticator><![CDATA[nandu@ht.sfc.keio.ac.jp]]></Authenticator>
<Requester><![CDATA[nandu@ht.sfc.keio.ac.jp]]></Requester>
<reqTime>2003-12-13T03:02:18.889</reqTime>
<DirectoryServiceInformation myself="true"/>
<DataManagementEngine>
<typeName><![CDATA[TinyDB]]></typeName>
</DataManagementEngine>
<ContextAnalyzingEngine>
<typeName><![CDATA[TinyDBClient]]></typeName>
</ContextAnalyzingEngine>
<objectXMLPointer><![CDATA[http://libra.ht.sfc.keio.ac.jp/~nandu/xml/object.xml]]></objectXMLPointer>
<statustXMLPointer><![CDATA[http://libra.ht.sfc.keio.ac.jp/~nandu/xml/status.xml]]></statustXMLPointer>
<contextCapRule>
<analyzer><![CDATA[TinyDBClient]]></analyzer>
<dataHandler><![CDATA[TinyDB]]></dataHandler>
<profileType><![CDATA[TinySQL]]></profileType>
<profile><![CDATA[SELEXT nodeid FROM sensors WHERE noise>600 AND light<400 AND nodeid=2]]></profile>
<resultHandler rowT="2" reqAmountT="1">
<dataElement
rowN="2"
meanNum="1"
match="direct"
reqAmount="1">2
</dataElement>
</resultHandler>
</contextCapRule>
<contextMeaning num="1">
<contextObjectPointer><![CDATA[chair]]></contextObjectPointer>
<contextStatusPointer><![CDATA[inUse]]></contextStatusPointer>
</contextMeaning>
</ContextInformation>
```

コンテキストの検索用のコンテキスト情報.

```
<?xml version="1.0" encoding="UTF-8"?>
<ContextInformation citype="searh">
<Authenticator match="direct"><![CDATA[nandu@ht.sfc.keio.ac.jp]]></Authenticator>
<Requester><![CDATA[nandu@ht.sfc.keio.ac.jp]]></Requester>
<reqTime>2003-12-13T03:02:18.889</reqTime>
<DirectoryServiceInformation>
<DirectoryServiceLocation myself="false">
<hostname>libra.ht.sfc.keio.ac.jp</hostname>
<portNum>10323</portNum>
</DirectoryServiceLocation>
<DirectoryServiceRealLocation match="direct"><![CDATA[NanduOffice]]></DirectoryServiceRealLocation>
<timestampRlx at="init" compare="now">2003-12-13T03:03:14.29</timestampRlx>
<Maintainer match="direct"><![CDATA[nandu@ht.sfc.keio.ac.jp]]></Maintainer>
</DirectoryServiceInformation>
<objectXMLPointer><![CDATA[http://libra.ht.sfc.keio.ac.jp/~nandu/xml/object.xml]]></objectXMLPointer>
<statustXMLPointer><![CDATA[http://libra.ht.sfc.keio.ac.jp/~nandu/xml/status.xml]]></statustXMLPointer>
<contextMeaning>
<contextObjectPointer match="direct"><![CDATA[desk]]></contextObjectPointer>
<contextStatusPointer match="direct"><![CDATA[inUse]]></contextStatusPointer>
</contextMeaning>
</ContextInformation>
```

コンテキスト検索の結果としてのコンテキスト情報.

```
<?xml version="1.0" encoding="UTF-8"?>
<ContextInformation security="public" citype="searchR">
<Authenticator><![CDATA[nandu@ht.sfc.keio.ac.jp]]></Authenticator>
```

```
<Requester><![CDATA[nandu@ht.sfc.keio.ac.jp]]></Requester>
<reqTime>2003-12-13T03:02:18.889</reqTime>
<DirectoryServiceInformation>
<DirectoryServiceLocation myself="false">
<hostname>libra.ht.sfc.keio.ac.jp</hostname>
<portNum>10323</portNum>
</DirectoryServiceLocation>
<DirectoryServiceRealLocation match="direct"><![CDATA[NanduOffice]]></DirectoryServiceRealLocation>
<timestampRlx at="init" compare="now">2003-12-13T03:03:14.29</timestampRlx>
<Maintainer match="direct"><![CDATA[nandu@ht.sfc.keio.ac.jp]]></Maintainer>
</DirectoryServiceInformation>
<DataManagementEngine>
<typeName><![CDATA[TinyDB]]></typeName>
</DataManagementEngine>
<ContextAnalyzingEngine>
<typeName><![CDATA[TinyDBClient]]></typeName>
</ContextAnalyzingEngine>
<objectXMLPointer><![CDATA[http://libra.ht.sfc.keio.ac.jp/~nandu/xml/object.xml]]></objectXMLPointer>
<statustXMLPointer><![CDATA[http://libra.ht.sfc.keio.ac.jp/~nandu/xml/status.xml]]></statustXMLPointer>
<contextCapRule>
<analyzer><![CDATA[TinyDBClient]]></analyzer>
<dataHandler><![CDATA[TinyDB]]></dataHandler>
<profileType><![CDATA[TinySQL]]></profileType>
<profile><![CDATA[SELEXT nodeid FROM sensors WHERE noise>600 AND light<400 AND nodeid=4]]></profile>
<resultHandler rowT="2" reqAmountT="1">
<dataElement
rowN="2"
meanNum="1"
match="direct"
reqAmount="1">4
</dataElement>
</resultHandler>
</contextCapRule>
<contextMeaning num="1">
<contextObjectPointer><![CDATA[chair]]></contextObjectPointer>
<contextStatusPointer><![CDATA[inUse]]></contextStatusPointer>
</contextMeaning>
</ContextInformation>
```

コンテキスト要請の結果としてのコンテキスト

```
<?xml version="1.0" encoding="UTF-8"?>
<ContextInformation citype="ci">
<Authenticator><![CDATA[nandu@ht.sfc.keio.ac.jp]]></Authenticator>
<Requester><![CDATA[nandu@ht.sfc.keio.ac.jp]]></Requester>
<reqTime>2003-12-13T03:02:18.889</reqTime>
<DataManagementEngine>
<typeName><![CDATA[TinyDB]]></typeName>
</DataManagementEngine>
<ContextAnalyzingEngine>
<typeName><![CDATA[TinyDBClient]]></typeName>
</ContextAnalyzingEngine>
<DirectoryServiceInformation myself="false">
<DirectoryServiceLocation>
<hostname>libra.ht.sfc.keio.ac.jp</hostname>
<portNum>10323</portNum>
</DirectoryServiceLocation>
<DirectoryServiceRealLocation><![CDATA[NanduOffice]]></DirectoryServiceRealLocation>
<timestampRlx at="init">2003-12-11T00:00:00.00</timestampRlx>
<timestampRlx at="lastMod">2003-12-13T03:03:14.29</timestampRlx>
<Maintainer><![CDATA[nandu@ht.sfc.keio.ac.jp]]></Maintainer>
</DirectoryServiceInformation>
<objectXMLPointer><![CDATA[http://libra.ht.sfc.keio.ac.jp/~nandu/xml/object.xml]]></objectXMLPointer>
<statustXMLPointer><![CDATA[http://libra.ht.sfc.keio.ac.jp/~nandu/xml/status.xml]]></statustXMLPointer>
<contextMeaning>
<contextObjectPointer><![CDATA[desk]]></contextObjectPointer>
<contextStatusPointer><![CDATA[inUse]]></contextStatusPointer>
</contextMeaning>
</ContextInformation>
```