卒業論文　　2004年度(平成16年度)

# マルチユーザ・コンテキストアウェア サービス環境のための 柔軟な機器排他制御手法


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

米澤拓郎

# 卒業論文要旨　　2004 年度 (平成 16 年度)

# マルチユーザ・コンテキストアウェア サービス環境のための 柔軟な機器排他制御手法

近年，ユビキタスコンピューティング環境の普及に伴い，ユーザの機器の利用形態が多様化している．センサが空間に埋め込まれ，環境状況に基づいた機器制御を行うコンテキストアウェアサービスの利用が可能になると，ユーザが機器に対して要求を行う機会はさらに増加する．この環境では，同一機器に対して複数ユーザからの要求が同時に発生する状況が多数起こりえる．同時に複数ユーザの要求を満たすことのできないテレビ等の情報機器では，誰の命令を優先すべきか考慮して機器の排他制御を行う必要が生じる．

通常，この要求の衝突はユーザ同士の話し合いで解決されることが多い．しかし，ユビキタスコンピューティング環境には多数の機器が存在するため，要求が衝突した度にユーザの話し合いを必要としてはユーザの負担が大きい．よって，機器の管理者側でどのユーザの要求を優先するかというポリシを設定して排他制御を行うことが今後必要となる．

本論文では，管理者が機器を使用する複数ユーザの関係性をポリシとして容易に反映し，柔軟な排他制御を行える環境の実現を目的とする．この目的を達成するために，実際の生活環境におけるユーザの関係性を反映した**マルチレベル優先順位決定ポリシ**による排他制御手法を提案し，そのポリシを多数の機器に対して容易かつ適切に設定し，管理者のコストを軽減した排他制御環境を実現するシステム **DAREGAYA** の設計と実装を行う．また，DAREGAYA を定量的，定性的に評価し，システムの有効性を示した．

キーワード：
排他制御，マルチユーザ，ユビキタスコンピューティング，コンテキストアウェア，デザイン

**慶應義塾大学　環境情報学部**
**米澤 拓郎**

# Abstract of Bachelor's Thesis

# Adaptable Exclusion Control for Multi-user Context-aware Service Environment

Recently, with the spread of ubiquitous computing environment, user can use appliances through various methods. With sensors embedded into space, user can use context-aware services. Therefore the opportunity of to use appliance will increase. In this environment, situations in which multiple users access the same appliance will occur often. So, mutual exclusion of requests will be needed for appliances which cannot fill demands of two or more users simultaneously, such as television.

Currently, this conflict of demand is solved through user's communication. However, there are so many appliances in ubiquitous computing environment, that it incurs high cost for users if they need to communicate whenever the demands collide. So the policy to automatically determine which demand prevails needs to be deployed by appliance's administrator in ubiquitous computing environment.

This paper aims to realize the environment where an administrator can perform flexible mutual exclusion, with policy which reflects the relation of two or more users who use appliances. To archive this purpose, this paper proposes **multi-level priority determination policy** - the exclusion control technique which reflects a user's relation in an actual living environment. We also design and implement **DAREGAYA** system in which realize mutual exclusion environment which administrator set up the policy easily and appropriately to many appliances. DAREGAYA can mitigate administrator's costs. This paper evaluate and shows the validity of DAREGAYA.

Keywords:
mutual exclusion, multi-user, ubiquitous computing, contex-aware, design

**Takuro Yonezawa**

**Faculty of Environmental Information**
**KEIO University**

# 目 次

# 図 目 次

# 表 目 次

# 第1章

# 序論

本章では，まず本研究の背景を述べる．そして本研究の概要について述べる．

## 1.1 本研究の背景

情報技術の発展により，高度な計算能力を持った情報機器が登場し，それらがネットワークで相互に接続されるようになった．情報機器を利用するユーザの生活空間は，Mark Weiser が提唱したユビキタスコンピューティング環境 [3] へと急激に変化しつつある．この環境では，ユーザは様々な操作方法で機器に対して命令を行うことができる．機器のボタンによる直接操作，赤外線を用いたリモコン操作，ネットワークを通じた遠隔利用はすでに一般家庭で日常的に使われている．例えば東芝の FEMINITY シリーズ製品 [1] や，National の暮らしネット製品 [2] では，外出先からネットワークを利用して冷蔵庫の中身を確認したり，帰宅の際に前もってエアコンの温度をネットワークを利用して調節することができる．

一方，センサ技術も進歩している．Mote [4] や Smart-Its [5] 等の超小型センサノードは温度や湿度，加速度等を計測できるセンサを搭載し，ユーザはそれらのセンサ情報をネットワークを介して得ることができる．センサが認識できる環境の状態も多様化しており，Active Bat [6] や Cricket [7] 等のセンサでは，屋内でのユーザや物の位置情報を把握することができる．

今後，情報機器の普及に加えこれらのセンサが環境側に埋め込まれると，環境の温度や湿度変化、ユーザや物の動きに応じて情報機器を制御できるコンテキストアウェアサービス環境が実現されることになる．今までは「環境変化→ユーザの操作→情報機器制御」という３つのプロセスを経るのに対し，コンテキストアウェアサービス環境では予め環境変化に対しての機器の挙動を定義することで,「環境変化→情報機器制御」という２つのプロセスで機器制御を実現できる．環境変化に対するユーザの要求が生じる度にユーザ自身に機器を操作させる手間を省き，ユーザが予め設定した通りの自動機器制御が実現できる．

## 1.2 本研究の概要

上述したようにユビキタスコンピューティング環境では，情報機器の利用形態が多様化される．利用形態の多様化に伴い，ユーザの機器に対する命令が従来よりも増加することが考えられる．しかし，情報機器は複数のユーザで共有されることが多い．一度に複数のユーザの要求を満たすことができない機器（例えばテレビやライト）に対して複数のユーザから要求が行われると，誰の要求を実行するかということが大きな問題となる．複数のユーザからの要求から，一人のユーザの要求を選んで機器を制御する一連のプロセスを，本研究では**排他制御**と呼ぶ．

本研究では，複数のユーザがコンテキストアウェアサービスを利用する環境を**マルチユーザ・コンテキストアウェアサービス環境**と定義し，この環境における機器の排他制御を実現する．排他制御は，どのユーザの要求をどのような基準で優先して実行すべきという排他制御ポリシに基づき行われるが，本研究ではこのポリシを設定する主体として機器の管理者に注目する．機器は管理者によって提供されるため，まず第一に管理者の意図を反映したポリシによる排他制御が実現されるべきである．環境に適した排他制

御ポリシを設定するためには，機器の管理者がその機器を使用する複数ユーザをどう扱うかという意図を，ポリシとして容易に反映できなければならない．

本研究では，マルチユーザ・コンテキストアウェアサービス環境において，機器の管理者がユーザの性質を考慮した柔軟な排他制御のためのポリシを定義し，排他制御が行える環境を目指す．この環境を実現するためにマルチレベル優先順位決定ポリシによる排他制御手法を提案し，空間に存在する多数の機器へ容易にポリシを適用できるシステムを実現した．

## 1.3 本論文の構成

第２章では，まず情報機器利用形態の変化について述べ，排他制御の必要性を説明する．そして，マルチユーザ・コンテキストアウェアサービス環境における排他制御について考察を行う．排他制御を行うための問題点を挙げ，本研究の目的について詳しく説明する．第３章では，本研究の目的を実現するための手法として，マルチレベル優先順位決定ポリシによる排他制御手法を提案し，そのポリシを多数の機器にどう適用するかについて述べる．第４章では，マルチレベル優先順位決定ポリシによる排他制御機構 DARE とマルチレベル優先順位決定ポリシの機器への適用機構 GAYA からなるシステム，DAREGAYA の設計について述べる．第５章では DAREGAYA の実装について述べる．第６章では，DAREGAYA の評価を行う．最後に第７章で結論・今後の課題を述べ，本論文を締めくくる．

# 第2章

# 情報機器の排他制御

本章では，情報機器の利用形態の変化について述べ，それに伴う機器の排他制御の必要性について説明する．次に排他制御を実現するために必要な要素について述べ，それを実現する上での問題点を上げる．最後に，本研究の目的を述べ，問題の解決手法を提案する．

## 2.1 排他制御の必要性の高まり

まず，情報機器の利用形態の変化について説明し，その結果必要となる機器の排他制御について述べる．そして，本研究が対象とするマルチユーザ・コンテキストアウェアサービス環境について説明する．

### 2.1.1 機器利用形態の変化

情報技術の進歩により，機器利用形態が大きく変化している．従来は，機器を利用するには対象の機器に直接触れるか，専用のリモコンで操作する方法が主であった．現在，ネットワーク接続性を持つ情報機器が普及し，外出先の携帯電話からインターネットを利用して機器を操作できるようになった．そして，さらにセンサが環境に埋め込まれ温度や湿度、人の位置情報等の環境状態が把握できるようになると，環境状態と情報機器の機能を結びつけたコンテキストアウェアサービスと呼ばれる機器の利用形態が普及することが考えられる．この利用形態が普及した環境を，コンテキストアウェアサービス環境と定義する．本研究で予測する将来の機器利用形態も含めて，機器利用形態の変化を表 2.1 にまとめた．

| | 過去 | 現在 | 将来（予測） |
|---|---|---|---|
| 機器環境 | 機器の普及 | 情報機器の普及 | 情報機器とセンサの普及 |
| 利用形態 | 触れる，専用リモコンでの操作による利用 | ネットワークを通じた操作による利用 | コンテキストアウェアサービスの利用 |

表 2.1: 環境と利用形態の変化

コンテキストアウェアサービス環境では，情報機器に対する個人個人の細かな要求が実現可能となる．例えば,「私が踊ったら，ＣＤプレイヤーから音楽を流してほしい」という要求や,「私がテレビから離れると音量を大きくしてほしい」という要求が簡単に実現される．これらのサービスは，環境に置かれた機器の管理者が提供した機能を，サービス利用者が環境変化に応じた要求を行うことで実現される．

### 2.1.2 本研究が対象とする環境

本研究が対象とする環境は，コンテキストアウェアサービスが利用できるユビキタスコンピューティング環境である．ユビキタスコンピューティング環境では，センサや情報機器が空間に遍在し，ユーザの行動を支援する．ユビキタスコンピューティング環境を実現した空間の例として，Active Space [8]，Easy Living [9]，Smart Space Lab [10] 等が挙げられる．本研究では，このユビキタスコンピューティング環境を実現した，オフィスや研究室等の建物を対象環境とする．この対象環境を図 2.1 に示す。情報機器にはそれぞれ管理者（もしくは複数人からなる管理者グループ）が存在し，複数ユーザに機器の利用環境を提供している．

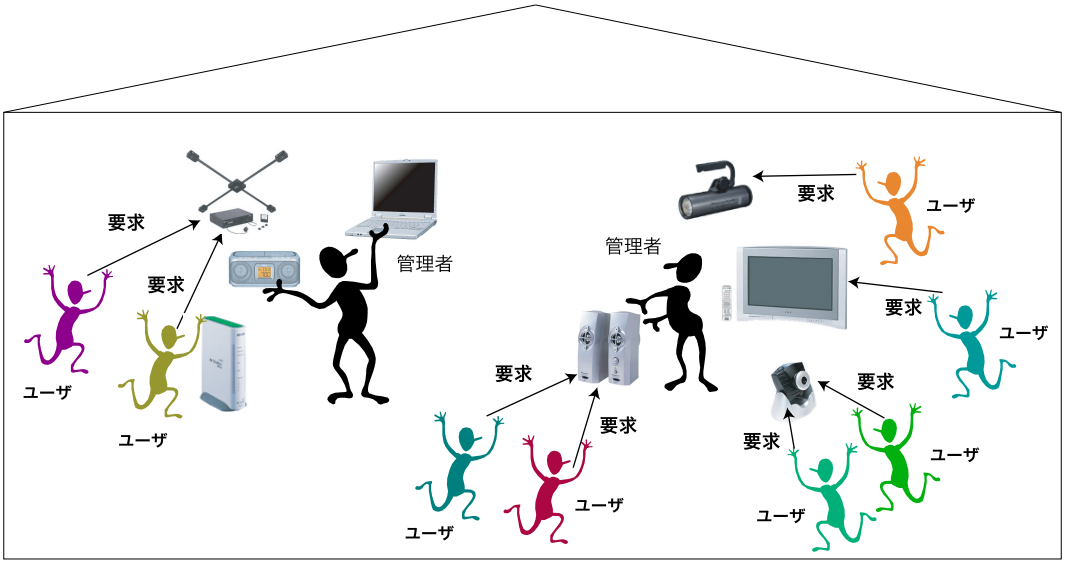


図 2.1: 想定環境

### 2.1.3 排他制御の必要性

生活環境においては複数のユーザが情報機器を共有する場面が多い．例えばテレビ、ビデオ、CD コンポ、ライトやエアコン等の情報機器は複数ユーザによって使用される．しかし，情報機器の中には，複数の機器制御要求を同時に処理できないものがある．例えば，ライトはＯＮとＯＦＦの制御要求を同時に満たすことはできない．また，テレビは表示できるチャンネルが 1 つだけなので，4ch に変更する，6ch に変更する，という制御要求を同時に満たすことができない．

複数ユーザで共有された情報機器で，複数ユーザの要求を同時に満たすことができない機器に対し，排他制御が必要となる (図 2.2)．本研究では，機器のアクセスコントロールは対象としない．アクセスコントロールとは，機器の使用を適切な資格を持つ人に制限するために用いられる権利の付与を意味し，排他制御はその資格を持つ者同士に対してどちらに機器の制御権を与えるかを意味する．本研究は，複数の要求を同時処理できない機器に対して，アクセスコントロールが行われた上で，複数ユーザからの要求が同時に行われた状況での排他制御を対象とする．機器制御における，排他制御の位置づけを図 2.3 に示す．

将来，複数のユーザがコンテキストアウェアサービスを利用する環境では，この排他制御の必要性が高まることが予想される．複数ユーザがコンテキストアウェアサービスを利用するマルチユーザ・コンテキストアウェアサービス環境では，環境情報の変化による細かな機器状態の変化が可能であるため，同一機器に対して複数ユーザからの要求が多く発生することが考えられるからである．マルチユーザ・コンテキストアウェアサービス環境において，機器に対して複数のユーザが機器制御を要求することで，排他制御が必要となる例を 2 つ挙げる．

例 1: A さんは部屋に入るとテレビが ON になって 4 チャンネルに設定されるように，B

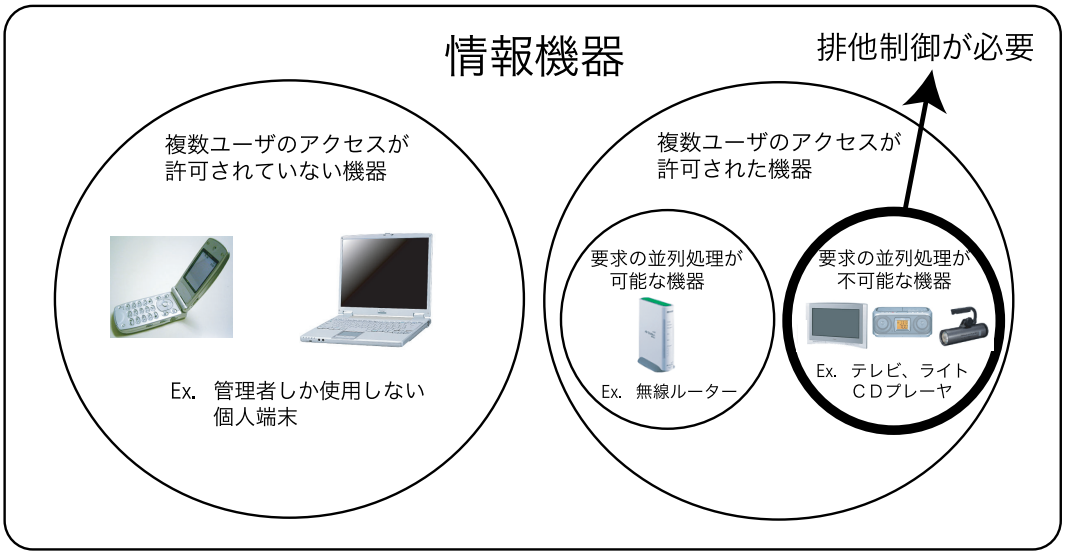


図 2.2: 排他制御が必要な機器の分類

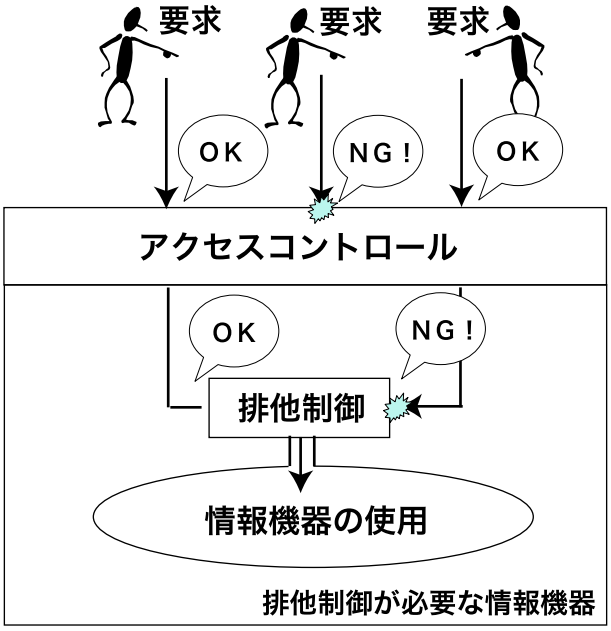


図 2.3: 排他制御の位置づけ

さんは ON になって 6 チャンネルに設定されるようにサービスを構築した．もし 2 人が同時に部屋に入ってきた場合，どのユーザの要求を優先するかという問題が生じる．

例 2: A さんはソファに座ると音楽プレイヤーからロック音楽が流れるように，B さんはクラシック音楽が流れるようにサービスを構築した．もし 2 人が同時にソファに座った場合，どのユーザの要求を優先するかという問題が生じる．

上の例は，どちらも同一機器に対する要求の衝突が発生しているため，どちらの要求を実行するか決定する必要がある．マルチユーザ・コンテキストアウェアサービス環境では，環境状態の変化が生じるたびに機器に対して自動的に制御要求が発生し，このような同一機器に対する複数ユーザの要求の衝突が頻繁に起こりえるため，排他制御の必要性が生じる．

## 2.2 マルチユーザ・コンテキストアウェアサービス環境における排他制御

本節では，マルチユーザ・コンテキストアウェアサービス環境において，管理者によって排他制御のポリシを設定する必要性を述べる．このポリシ設定を実現するための問題点と，本研究の目的について述べる．

### 2.2.1 管理者による排他制御ポリシ設定

同一機器に対し、複数のユーザの制御要求が同時に起こった場合，ユーザ同士の議論でどちらが機器の使用権を得るのか決定できる．しかしながら，マルチユーザ・コンテキストアウェアサービス環境においては，空間に多数の機器が存在し，それらの機器に対して要求の衝突が頻繁に発生する．要求の衝突が発生する度に処理の中断が発生し，ユーザの議論を必要するのでは，ユーザの大きな負担となる．さらに，ユーザの数が増加すると，誰が同時に要求を行っているのか把握することが困難となり，議論が行えない場合も考えられる．このような状況は，ユーザに意識させずに支援を行うことを目的とするユビキタスコンピューティング環境に反する．ユーザに意識させない排他制御を実現するためには，機器側で何らかのポリシにそって排他制御を行う必要がある．しかしながら，機器側で単一のポリシで排他制御を行うだけでは，管理者がどういうポリシで機器を提供するかという意図が反映できない．よって，排他制御のポリシを機器の管理者が設定できる環境が必要である．ここで，管理者がどのようなポリシで排他制御を行うのかを想定し，具体例を挙げて説明する．

#### 要求到着順での排他制御

最も単純な方法として，最も早く要求を行ったユーザを優先するポリシが考えられる．例えば，A さんがテレビに要求を行ってテレビを使用している状態では，後から到着した B さんの要求は受け入れない．これは，その機器を使用するユーザが全て対等な関係にある場合に多く用いられると考えられる．

#### 明確な優先順位をつけた排他制御

機器の管理者が誰の命令を実行すべきか決定する方法の１つに，ユーザの明確な優先度に基づいた排他制御方法が挙げられる．この明確な優先度を用いた排他制御の例を以下に述べる．機器を利用するユーザとしてＡとＢが存在し，ＡはＢより優先される (以下、Ａ＞Ｂと表現) という優先順位が設定されているとする．今，ある情報機器に対し，２人が異なる制御要求を行おうとしている．その要求を行う場面として，3通り考えられる．

- 誰もサービスを利用してなく，どちらか１人がサービスに対し要求する場合
  - 誰もサービスを利用していない状況で，ＡもしくはＢがサービスの要求を行ったとする．この場合は，ＡもしくはＢの要求が実行されることになる．
- Ａがサービスを利用していて，Ｂがそのサービスに対しＡと異なる要求する場合
  - Ａがサービスを元々利用している場合，この情報機器のサービスはＡに占拠されている状態である．この状況でＢが新しくサービスに異なる要求をしたとする．優先順位はＡ＞Ｂなので，Ｂの要求は保留される．
- Ｂがサービスを利用していて，Ａがそのサービスに対しＢと異なる要求する場合
  - Ｂがサービスを元々利用している場合，この情報機器のサービスはＢに占拠されている状態である．この状況でＡが新しくサービスに異なる要求をしたとする．優先順位はＡ＞Ｂなので，Ｂが受けていたサービスは停止し，Ａの要求が満たされるようにサービスが提供される．

このような明確な優先順位づけに基づく排他制御は，機器を利用する全ユーザの優先順位が静的で，明記できる場合有効である．

#### その他のポリシに基づく排他制御

管理者によっては，独自のポリシに基づき排他制御を行うことが考えられる．例えば，多く機器を使用したことのあるユーザは使用したことのないユーザより優先順位を下げて排他制御を行ったり，時間帯に応じて優先されるユーザを変更して排他制御を行う．または，そのポリシ自体を利用するユーザ同士で予め決定させて排他制御を行うことも考えられる．

### 2.2.2　問題点と目的

排他制御を行うために複数ユーザの要求のうち誰の要求を実行するかというポリシには，まずその機器の管理者がどのような基準で機器を提供するか、という意図が反映されるべきである．第 2.2.1 項に挙げたように，管理者の設定したいポリシは様々である

ため，予め全てのポリシを想定することは不可能である．よって，管理者が望み通りの排他制御を行うためには，管理者自身でそのポリシを定義できなければならない．また，管理者が所有する機器は複数存在することが想定されるため，その複数の機器に対しポリシを容易に適応できなければ，管理者の負担が大きくなる．

本研究の目的は，

- 排他制御を行うためのポリシを管理者自身が柔軟に定義でき，
- 定義したポリシを空間に存在する多数の機器に容易に設定できる環境

の実現である．この目的を実現するため，管理者が実際の生活環境におけるユーザの関係性を柔軟に反映できるマルチレベル優先順位決定ポリシによる排他制御手法及びそのポリシ適用手法を提案し，実装を行う．

## 2.3 本章のまとめ

本章ではまず，ユビキタスコンピューティング環境における，排他制御の必要性について述べた．次に，排他制御が必要な情報機器を明確にし，排他制御は機器の管理者自身のポリシに基づき行われることを説明した．最後に問題点と目的を述べた．

# 第3章

# マルチレベル優先順位決定ポリシ手法の提案

本章では，マルチレベル優先順位決定手法による柔軟な機器の排他制御を提案し，説明する．また，その手法を複数の機器にどう適用していけば容易に運用が可能か述べる．

## 3.1 マルチレベル優先順位決定ポリシによる排他制御手法

管理者が柔軟な排他制御を実現するために，本研究ではマルチレベル優先順位決定ポリシ (以下，**MPDP**: **M**ulti-level **P**riority **D**etermination **P**olicy) による排他制御手法を提案する．MPDP は，階層的に表現される複数のユーザグループに対し，異なる優先順位決定アルゴリズム（以下 **PDAlgo**: **P**riority **D**etermination **Algo**rithm）を適用したポリシである．MPDP は各情報機器上に存在し，情報機器は MPDP に基づいて排他制御を行う．本手法を用いることで，機器の管理者は柔軟なユーザの優先順位づけを定義することができる．

### 3.1.1 MPDP の有効性

ユーザの生活空間において，内部に様々なレベルのグループが存在することが多い．例えば大学の研究室においては，教員，博士課程の学生，修士の課程学生，学部生というように，同じ研究室というグループの中にも細分化されたグループが存在する．一般家庭においても，子供，青年，成人，お年寄りというグループが存在しうる．このように，内部にグループを持つユーザ全てに対し，管理者がすべて同じポリシで排他制御を行うことは考えにくい．MPDP を用いることで，管理者はユーザを複数のグループとして管理し，そのグループ内で優先順位の決定方法を適用できる．

MPDP を用いると，研究室の例では次のような排他制御が可能となる．まず管理者は，教員＞博士課程の学生＞修士課程の学生＞学部生と明確な優先度を決定して排他制御を行える．そして，博士課程の学生，修士課程の学生，学部生の同一グループ内ではそれぞれのユーザを対等に扱い，要求到着順に優先権を与えることができる．MPDP は多階層でグループを管理することが可能で，それぞれのグループの中を更にグループに分け，それぞれのグループ対し優先順位決定の基準を与えられる．

このような排他制御が実現可能なことから，MPDP を用いた排他制御は，現実世界のユーザ同士の関係を反映するのに適した柔軟な排他制御方法と言える．

### 3.1.2 MPDP の構成

MPDP の構成を図 3.1 に示す．MPDP は階層化されたグループ，グループを形成するユーザ，グループごとに割り当てられた PDAlgo の 3 要素から成り立っている．それぞれの構成要素について，以下に説明する．

#### 構成要素

- ユーザ
  ユーザは，機器へのアクセス権限を持つ個人を指す．

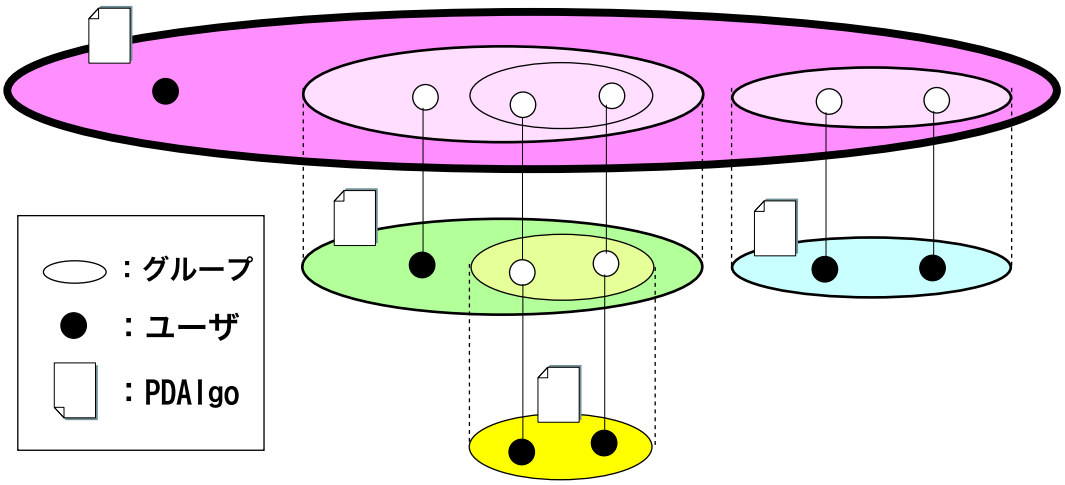


図 3.1: MPDP の構成

- グループ
  グループは，一人以上のユーザから構成される論理的な集合である．グループ内部にグループを持つことができ，その関係性を階層的に成立できる．その機器へのアクセス権を持つユーザが全て含まれたルートとなるグループを基本グループと呼ぶ．MPDP は最低この基本グループから構成される．グループ内部で構成された一階層下のグループは，そのグループに存在するユーザと同等に扱われる．

- PDAlgo（優先順位決定アルゴリズム）
  PDAlgo は，グループを構成する複数のユーザの優先順位の決定基準となるアルゴリズムである．グループ毎にこのアルゴリズムは定義される．このアルゴリズムの方法としては，明確な優先度づけであったり，機器制御要求到着順であったりする．これは，管理者が独自に定義できる．

### MPDP による排他制御実行プロセス

ここでは，設定された MPDP による排他制御実現のための実行プロセスについて述べる．MPDP は，階層的なグループ毎にそれぞれの PDAlgo を用いてユーザの優先順位を比較し，最終的に要求を実行するユーザを決定する．実行プロセス例を図 3.2 に示す．ユーザの機器制御要求が複数発生した場合，PDAlgo によって他のユーザとの優先順位の比較検討が行われる．基本グループの階層で比較検討が行われた結果，選ばれたユーザが機器制御実行権限を持つことになる．このマルチレベルでの優先順位の比較は，新しいユーザがサービスの要求を行うごとに行われる．

MPDP では管理者が自由にユーザをグループに登録でき，階層的に構成できる．そして，作成した管理者のそれぞれのグループに対して異なるアルゴリズムで適応できるため，あるユーザとグループ同士の関係性，グループ内部でのユーザ同士の関係性を考慮した柔軟な排他制御機構を構築できる．

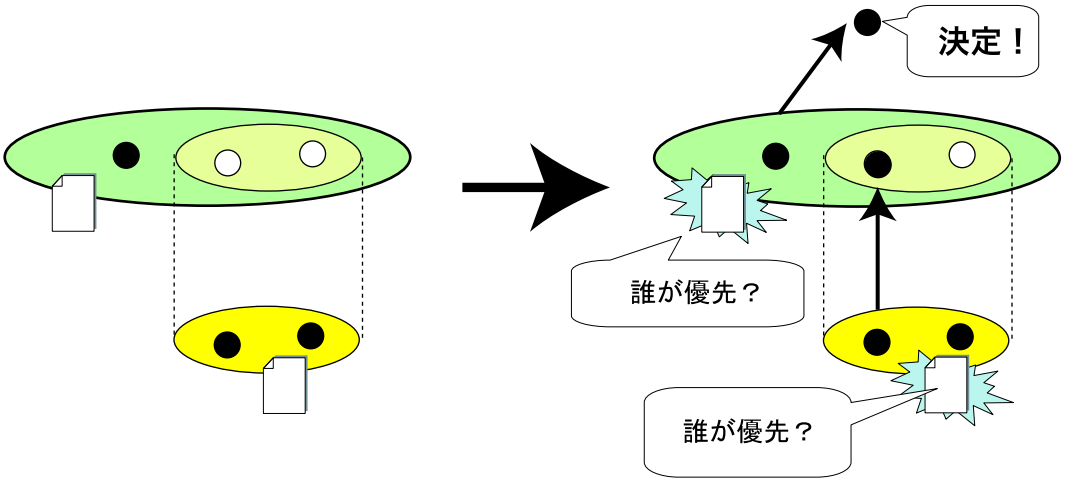


図 3.2: MPDP の排他制御実現プロセス

#### グループオーナへの記述権限の付与

機器の管理者は作成した全てのグループに対して PDAlgo の記述権限を持つ．一方で，管理者はそれぞれのグループに対してグループオーナを設定でき，オーナが自由に内部にグループをつくり，グループ内で PDAlgo を設定することを許可できる．これは，それぞれのグループ内では自律的にポリシを決定させるという管理者の意向を実現するものである．この場合は，管理者がそのグループで初期状態で使用する PDAlgo をはじめに設定し，記述が許可されたグループオーナが書き換えられる．グループオーナは，管理者と同様にそのグループ内部に作成したグループに対してオーナを設定でき，階層化されたグループのそれぞれでオーナがポリシを設定できる．

### 3.1.3 MPDP を用いた排他制御シナリオ

MDPD の設定例を図 3.3 に示す．6 人のユーザ（A，B，C，D，E，F）が機器のアクセス権を得ている．もしこのうちの何人かのユーザが同時にサービスを要求した場合，排他制御を行う必要がある．まず，基本グループとして（A，B，C，D，E，F）のグループが存在する．基本グループの内部で，(B，C，D）が含まれるグループ 1，(E，F）が含まれるグループ 2 が構成されている．それぞれのグループに対して，管理者は異なる PDAlgo を適用する．基本グループには，ユーザ A，グループ 1，グループ 2 の並び順を優先順位とする PDAlgo が適用されている．グループ 1 には，要求到着順が優先順位とする PDAlgo が適用されている．グループ 2 にはランダムに優先順位が決定される PDAlgo が適用されている．

次に，それぞれのユーザがサービスを要求した場合に誰が優先されるのかを説明する．

ケース 1: A と E が同時に要求を行っている場合

この場合，基本グループの PDAlgo によって優先度が決定される．この PDAlgo

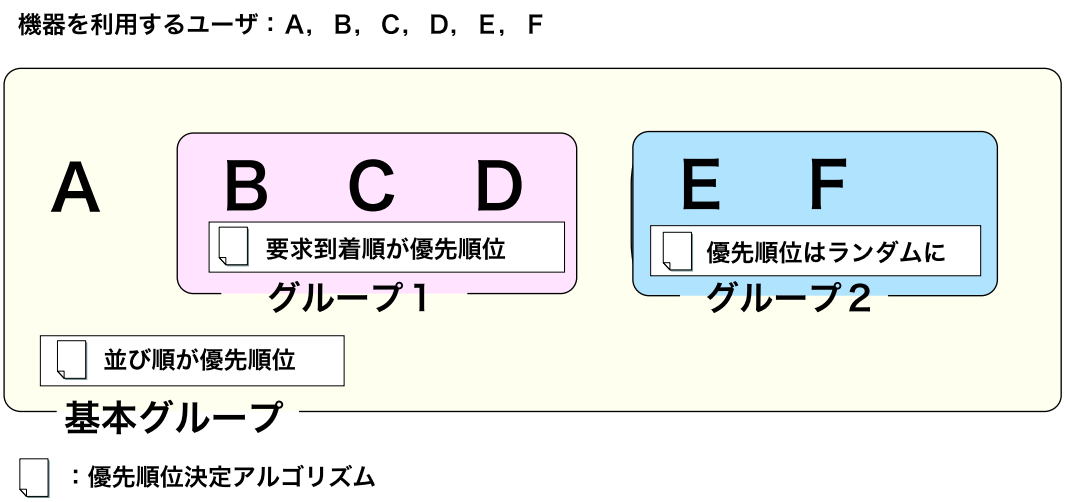


図 3.3: MPDP の設定例

では，A ＞グループ 1 ＞グループ 2 と明確な優先度が決定されているため，A の要求が優先される．

ケース 2: B と C が同時に要求を行っている場合
この場合，お互いグループ 1 に所属しているため，グループ 1 の PDAlgo によって優先順位が決定される．サービス到着順に優先順位が決定されるため，B の方が先にサービスを要求していれば B が優先され，C の方が先にサービスを要求していれば C が優先される．

ケース 3: E と F が同時に要求を行っている場合
この場合，お互いグループ 2 に所属しているため，グループ 2 の PDAlgo によって優先順位が決定される．ここでは，ランダムに優先順位が決定される．

## 3.2 MPDP 適用手法

機器の管理者が設定した MPDP を効果的に機器に適用し，機器に対して運用していく手法が必要である．まず最も単純な手法をのべ，その後に本研究が提案する手法を説明する．

### 3.2.1 機器ごとに MPDP を設定する手法

最も単純な方法として，空間内に存在する機器一つ一つに対してポリシの設定を行う方法が考えられる（図 3.4）．ユビキタスコンピューティング環境では，管理者は多数の機器を同時に管理していることが想定されるため，管理者に負担がかかる．また，新たなユーザが機器を利用する場合や，機器が移動したり新たに設置された時にポリシを変

更したい場合，再び一つ一つ設定しなければならない．この方法では,「それぞれの機器に対して明確にポリシを決定できる」という利点があるが，反面「コストが高い」と「環境変化に適応できない」という欠点がある．この問題点を解決するための手法を次に提案する．

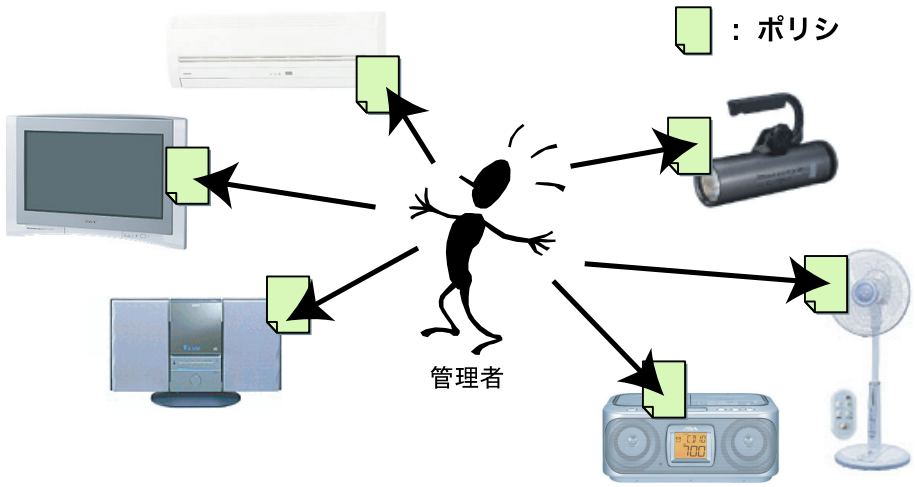


図 3.4: 一つ一つの機器に対し，ポリシを設定する手法

### 3.2.2　空間単位で複数の機器に MPDP を設定する手法

特定の空間に所属する複数の機器は，同じポリシに設定されることが多いと想定される．例えば，リビングルームという空間に存在する機器に対しては，父親が優先されることが考えられる．また，兄弟の部屋では，兄が優先されることが考えられる．

このように，機器を使用するユーザの優先順位づけは，その機器が所属する空間の特質に関係することが多い．よって，空間ごとにポリシを設定して，その空間に存在する全ての機器にそのポリシを適用する手法について検討する（図 3.5）．管理者は空間ごとにポリシを設定するだけでよいため，設定に必要なステップ数が減り，管理者のコストを抑えられる．また，空間を移動した機器や新しく出現した機器にも，その空間のポリシが動的に適用されるため，環境変化に対して適応性を持つ．

この手法は，一つ一つの機器に対してポリシづけを行うより,「コストが低く」,「環境変化に適応する」という利点を持つ．一方で,「一つ一つの機器に明確なポリシが設定できない」という欠点を持つ．よって本研究では，特定の機器に対して固有のポリシをつける手段を残しつつ，空間単位でポリシを設定する手法を提案する．ユビキタスコンピューティング環境では多数の機器が存在するため，まず空間単位でポリシを適用することで管理者の設定コストを抑え，空間の中で明示的に異なるポリシを設定したい機器に対しては手動で設定する手法が適している．

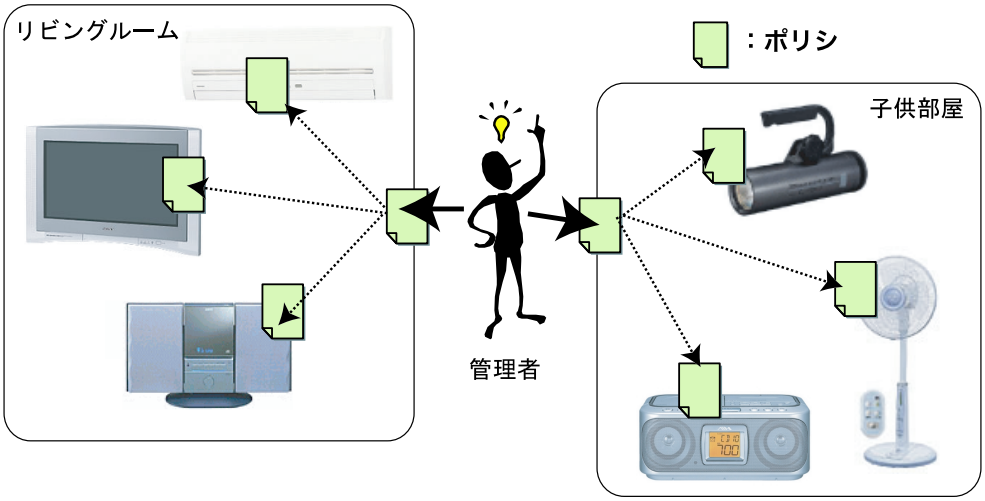


図 3.5: 複数の機器に対し，空間単位でポリシを設定する手法

## 3.3 関連研究

本手法の有効性を確認するために関連研究を挙げ，マルチユーザ・コンテキストアウェアサービス環境でそれぞれの手法が適しているか，検討する．

### 3.3.1 ラウンド・ロビン・スケジューリング

一つの CPU をどのプロセスが使用するか決定する方式の一つとして，ラウンド・ロビン・スケジューリング [11] による排他制御方法が挙げられる．これは，各プロセスを公平に捉え，それぞれのプロセスに対してクォンタムと呼ばれる実行許可時間が割り当てられる．あるプロセスのクォンタムを終了すると，別のプロセスに CPU がそのクォンタム分割り当てられる．マルチユーザ・コンテキストアウェアサービス環境でのサービスは，CPU のように分割して各ユーザが使用できないものが多いため単純にこの方法を用いることは適さない．例えばテレビは続けて番組を見ることに意味があり，ユーザの要求の数だけチャンネルが順次変更されては，どのユーザも満足できない．

### 3.3.2 優先順位スケジューリング

ラウンド・ロビン・スケジューリングが全てのプロセスを公平に見なしているのに対し，優先順位スケジューリング [12] はプロセスごとに優先順位をつけ，プロセスごとに割り当てるクォンタムを変更したり，他のプロセスの実行を中断させる手法をとっている．マルチユーザ・コンテキストアウェアサービス環境では，その環境によって優先されるユーザが存在することが考えられるため，この手法による排他制御は有効であることが多い．また，プロセスを優先順位のクラスごとにグループ化し，クラス間では優先

順位スケジューリングを用い，それぞれのクラス内ではラウンド・ロビン・スケジューリングを使用する方法も存在する．このモデルは本研究が提案するモデルと類似しているが，このモデルだけでは 2.2.1 で挙げたような管理者の様々な意図を反映することができないため，柔軟性に欠けている．

### 3.3.3　その他のポリシによる排他制御方法

他にも最短ジョブ優先スケジューリング [13] 等、排他制御のためにどの要求を優先して実行するかという基準のポリシが存在するが，全ての環境において最適な排他制御方法は存在しない．なぜならその環境ごとに存在するユーザの性質，管理者のサービス提供方式が異なるからである．

## 3.4　本章のまとめ

本章では，階層的にユーザグループを構成し，それぞれに対し異なる優先順位決定アルゴリズムを適用できるマルチレベル優先順位決定ポリシによる排他制御を提案・説明し，記述方法について述べた．また，このポリシを管理者が効果的に機器に適用できる空間単位の設定手法を提案・説明した．また，最後に関連研究について述べた．

# 第4章

# DAREGAYAの設計

本章では，MPDP を用いた排他制御機構及び空間単位 MPDP 配布機構から構成されるシステム，DAREGAYA の設計について述べる．

## 4.1 DAREGAYAの概要

DAREGAYA は，MPDP を用いた排他制御機構 DARE と，空間単位 MPDP 配布機構 GAYA から構成されるシステムである．マルチユーザ・コンテキストアウェアサービス環境において，管理者による柔軟な排他制御を容易に行うことができるシステムである．DAREGAYA の全体概要図を図 4.1 に示す．ユーザはコンテキストアウェアサービスを実現するソフトウェアを用いて，環境状況に応じた機器の制御要求を行う．DARE は MPDP を参照して機器の排他制御を行う．GAYA は管理者が設定した MPDP を空間単位で配布する．本章では，まず MPDP の記述方法を設計し，次に DARE と GAYA について説明する．

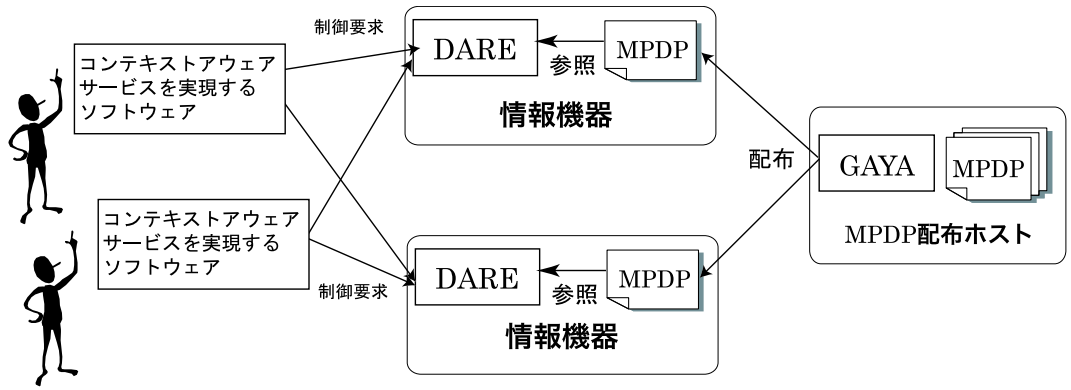


図 4.1: DAREGAYA の全体概要図

## 4.2 MPDPの作成方法

MPDP は XML による構造化されたドキュメントとして表現される．XML を用いる利点は，構造化言語処理系の実装の容易性である．汎用 XML パーサーを用いることで，字句解析，構文解析，意味解析を容易に行える．また，汎用 XML パーサーは，Java，C/C++，Perl をはじめ様々な言語に実装されているため，処理系の実装も多様な言語による実行環境上で可能となる．このような理由で，XML による記述が適している．

MPDP は，管理者及びグループオーナが記述した XML 形式の MPDP Element ファイルを MPDP Maker が読み込むことによって作成される．MPDP Maker により MPDP を作成する設計とした理由は，MPDP 内部の記述に対して作業の分離が可能であるからである．管理者とグループオーナは，自分の管理する範囲をそれぞれ記述できる．MPDP Maker は管理者の記述した root.xml を読み込み，root.xml の内部にグループオーナの記述があった場合にはグループ名.xml を読み込み，MPDP を作成する．root.xml 内部にグループオーナの記述がない場合には root.xml をそのまま MPDP として用いる．MPDP Maker の処理図を図 4.2 に示す．

次に，管理者またはグループオーナが記述する MPDP Element 及び MPDP Maker から

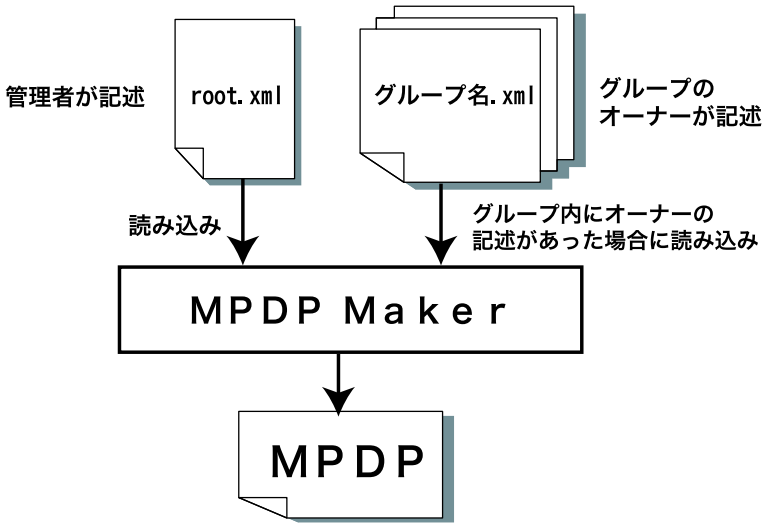


図 4.2: MPDP Maker の処理

作成される MPDP の記述形式について説明していく．

### 4.2.1 MPDP 及び MPDP Element の宣言

MPDP 及び MPDP Element は XML による構造化された文章であるため，XML 文章の宣言をファイルの先頭で行う．また，MPDP 及び MPDP Element であることを示すために，<MPDP>タグ，<MPDPElement>を記述する．

```
<? xml="version1.0" ?>
<MPDP>
</MPDP>
```

```
<? xml="version1.0" ?>
<MPDPElement>
</MPDPElement>
```

### 4.2.2 グループの記述

グループは，<group>タグを用いて記述される．属性として，グループの識別名を記述する．<group>タグの内側には，PDAlgo，ユーザ，内部グループが記述される．

```
<group name="move! research group">
</group>
```

### 4.2.3 ユーザの記述

ユーザは，<user>タグを用いて記述される．<user>タグの内側には，そのユーザを表す名前が記述される．

```
<user> takuro </user>
```

### 4.2.4 グループオーナの記述

管理者もしくはグループオーナは，自分が管理するグループに所属するユーザがグループ内にグループを作成したり，PDAlgo を自分たちで設定できるように許可できる．この場合は，グループオーナの存在を記述し，オーナが設定したグループ内の MPDP を読み込むことができる．グループオーナは<owner>タグを用いて記述される．この記述があった場合は，group タグの name 属性を参照し，グループ名.xml を参照する．

```
<owner>skk</owner>
```

### 4.2.5 PDAlgo の記述

PDAlgo は，<pdalgo>タグを用いて記述される．<pdalgo>タグの内側には，そのグループに適用する PDAlgo の名前が記される．システム内部にはこの名前を持つ PDAlgo が存在し，タグの内側に指定した PDAlgo を参照することを意味する．

```
<pdalgo> Order_of_demand_arrival </pdalgo>
```

### 4.2.6 MPDP の作成例

MPDP は管理者とグループオーナが記述した MPDP Element から構築される．MPDP Element の記述例を以下に示す．機器のアクセス権を持ったユーザは ngt，skk，takuro，mics，yukio，micchie の 6 人である．この 6 人のうち，同時に要求が発生した時のために管理者は MPDP Element の記述を行う．管理者が記述した MPDP Element である root.xml の例を図 4.3 示す．

基本グループの中には，Master グループと，Bachelor グループが含まれている．基本グループでは，ngt（ユーザ）＞Master（グループ）＞Bachelor（グループ）という明確な優先順位づけによる PDAlgo が設定されている．Master グループと Bachelor グループ内部では，要求到着順に優先権を与えるという PDAlgo が設定されている．さらに Bachelor グループにはグループオーナが設定されており，このオーナはグループ内部の MPDP Element を記述することができる．以下に，B4 グループのオーナが記述する Bachelor.xml の例を図 4.4 示す．

```
<?xml version="1.0" ?>
<MPDPElement>
    <group name="move! research group">
        <pdalgo> Order_of_a_row </pdalgo>
        <user> ngt </user>
        <group name="Master">
            <pdalgo> Order_of_demand_arrival </pdalgo>
            <user> duh </user>
            <user>  ide </user>
        </group>
        <group name="Bachelor">
            <owner> skk </owner>
            <pdalgo> Order_of_demand_arrival </pdalgo>
            <user> skk </user>
            <user> takuro </user>
            <user> mics </user>
            <user> yukio </user>
            <user> micchie </user>
        </group>
    </group>
</MPDPElement>
```

図 4.3: root.xml の記述例

Bachelor グループの内部には skk（ユーザ）＞B4（グループ）＞B3B2（グループ）という明確な優先順位づけがされており，B4 グループ内部では要求到着順に優先権が与えられ，B3B2 グループ内部ではランダムに優先権が割り当てられるように PDAlgo が設定されている．図 4.3 の root.xml と図 4.4 の Bachelor.xml から，MPDP Maker は MPDP を作成する．この２つの MPDP Element から作成された MPDP を図 4.5 に示す．root.xml で記述された Bachelor グループ内部が，Bachelor.xml で記述された内容に置き換わっている．機器はこの MPDP に基づき機器の排他制御を行う．

## 4.3 DARE: MPDP を用いた排他制御機構の設計

DARE のシステム概要図を図 4.6 に示す．DARE は主に排他制御モジュール，ユーザ・グループバインドモジュール，PDAlgo 決定モジュール，優先ユーザ決定モジュール，MPDP の５つの要素からなる．管理者が決定した MPDP を用い，DARE は排他制御を行う．以下，各モジュールについて説明する．

```
<?xml version="1.0" ?>
<MPDPElement>
        <group name="Bachelor">
            <pdalgo> Order_of_a_row </pdalgo>
            <user> skk </user>
            <group name="B4">
                <pdalgo> Order_of_demand_arrival </pdalgo>
                <user> takuro </user>
                <user> mics </user>
            </group>
            <group name="B3B2">
                <pdalgo> Random </pdalgo>
                <user> yukio </user>
                <user> micchie </user>
            </group>
      </group>
</MPDPElement>
```

図 4.4: Bachelor.xml の記述例

### 4.3.1 排他制御モジュール

排他制御モジュールは，機器の制御要求が複数のユーザから行われた場合に，実際に排他制御を行うモジュールである．機器制御要求を行っているユーザのリストを内部に保持し，どのユーザの要求を実現するか処理を行うため，ユーザ・グループバインドモジュールにユーザリストを伝える．そして，優先ユーザ決定モジュールから得たユーザの機器制御要求を実現するために，機器制御実行モジュールに決定されたユーザの要求を伝える．

機器制御要求ユーザリストはユーザの要求がある度に更新され，その度にどのユーザを優先するか確認する．もし現在実行しているユーザと同じユーザに優先権が決定されると，現在の処理を継続する．もし異なると，新しく決定されたユーザの要求を実行する．

### 4.3.2 ユーザ・グループバインドモジュール

ユーザ・グループバインドモジュールは，その機器に設定された MPDP を参照し，排他制御モジュールから得た機器制御要求ユーザリストからユーザとグループのバインドを行い，ユーザとグループが結びつけられた UG リストを PDAlgo 決定モジュールに渡す．UG リストは，要求を行っているユーザがどのグループに所属しているかという情報が記述されている．

```
<?xml version="1.0" ?>
<MPDP>
    <group name="move! research group">
        <pdalgo> Order_of_a_row </pdalgo>
        <user> ngt </user>
        <group name="Master">
            <pdalgo> Order_of_demand_arrival </pdalgo>
            <user> duh </user>
            <user>  ide </user>
        </group>
        <group name="Bachelor">
            <pdalgo> Order_of_a_row </pdalgo>
            <user> skk </user>
            <group name="B4">
                <pdalgo> Order_of_demand_arrival </pdalgo>
                <user> takuro </user>
                <user> mics </user>
            </group>
            <group name="B3B2">
                <pdalgo> Random </pdalgo>
                <user> yukio </user>
                <user> micchie </user>
            </group>
        </group>
    </group>
</MPDP>
```

図 4.5: MPDP Maker が作成した MPDP

### 4.3.3 PDAlgo 決定モジュール

PDAlgo 決定モジュールは MPDP を参照し，ユーザ・グループバインドモジュールから渡されたリストから優先順位を決定するために使う PDAlgo を決定する．決定した PDAlgo と UG リストを優先ユーザ決定モジュールに渡す．

### 4.3.4 優先ユーザ決定モジュール

優先ユーザ決定モジュールは，PDAlgo 決定モジュールで決定された PDAlgo に基づいて，UG リストから優先するユーザを決定する．決定されたユーザは排他制御モジュールに渡される．

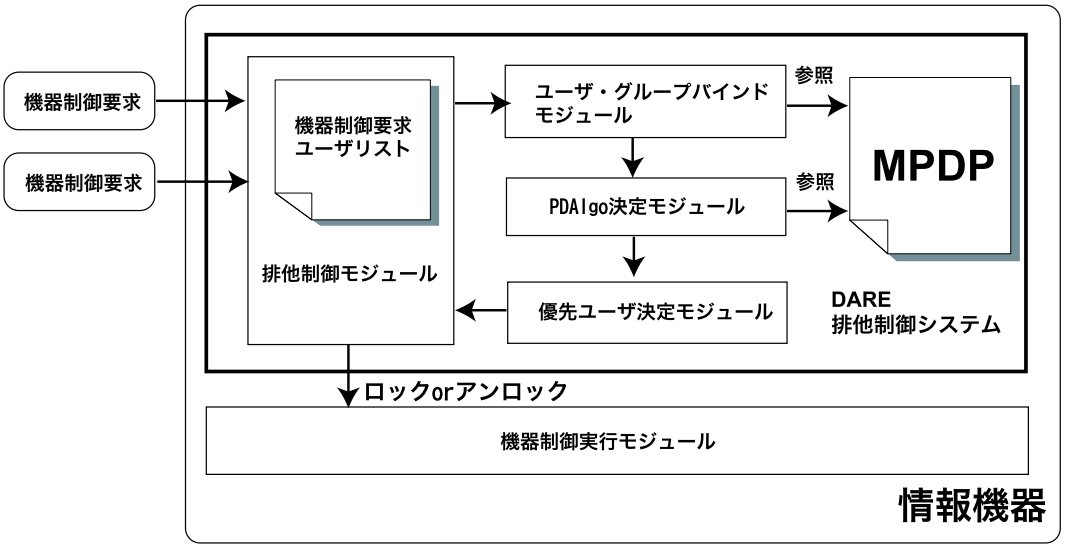


図 4.6: DARE システム概要図

## 4.4 GAYA: 空間単位 MPDP 配布機構の設計

空間単位 MPDP 配布機構は，空間単位で MPDP を配布するシステムである．本機構では，機器所属空間をどう認識する手法が重要となってくる．本設計では，RF タグや位置情報認識センサ，もしくは Smart-Its 等の小型センサノードを利用して機器所属空間を認識する機構を構築し，認識した空間情報を用いて MPDP を配布する．

MPDP 配布機構は，主に機器所属空間認識モジュール，GAYA クライアントモジュール，GAYA サーバーモジュールの 3 要素から成る．それぞれの機器上には，GAYA クライアントモジュールが実装されている．環境側には機器を認識できる何らかのセンサが備え付けられてあり，機器所属空間認識モジュールが実装されている．また，管理者が空間単位で MPDP の配布を行うための GAYA サーバーモジュールが存在する．それぞれはネットワークに接続されている．システムの概要図を図 4.7 に示す．

以下，各要素について説明する．

### 4.4.1 機器所属空間認識モジュール

機器所属空間認識モジュールは，様々なセンサを用いて機器がどういう空間に所属しているかを認識する．認識できる空間の粒度は用いるセンサによって様々なので，管理者が最も運用しやすいと思う空間の粒度で機器を認識できるようにセンサを選ぶ必要がある．管理者は認識できる空間に対し名前付けを行い，機器所属認識モジュールは機器がその空間に所属したことを確認したら，機器上の GAYA クライアントモジュールに所属空間情報を渡す．通達に必要な機器の IP アドレスは，GAYA クライアントモジュールが広告する機器情報を参照して認識する．

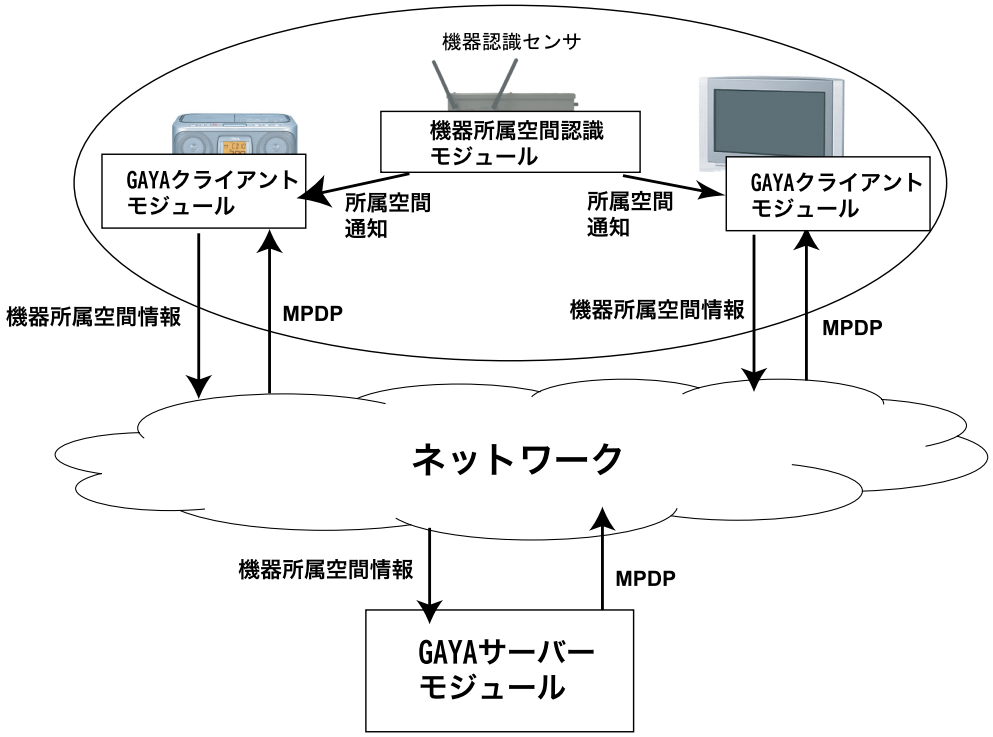


図 4.7: GAYA のシステム概要図

### 4.4.2 GAYA クライアントモジュール

GAYA クライアントモジュールは情報機器上に存在し，主に 2 つの要素からなる．一つ目は，機器情報・所属空間情報広告部．二つ目は MPDP 受け取り部である．内部構成図を図 4.8 に示す．

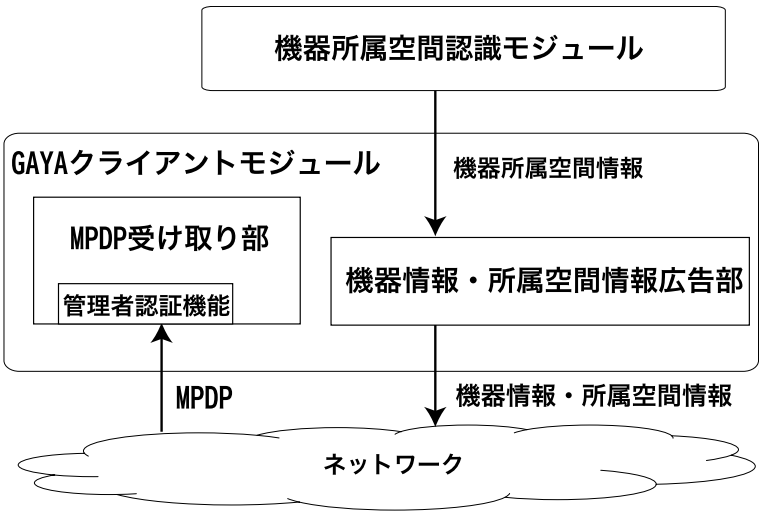


図 4.8: GAYA クライアントモジュール内部構成図

機器情報・所属空間情報広告部は，機器情報及び所属空間情報をマルチキャストを用いてネットワークに広告する．広告される情報は，機器の名前，所属している空間情報，その機器に割り当てられた IP アドレス，MPDP 配布を必要とするかである．所属している空間情報は，機器所属空間認識モジュールによって通知される．もし，自分の所属空間が確認ができない場合は `null` となる．機器によっては，空間単位でなく，固有に MPDP を設定したい場合があるため，４つ目の要素が用意されている．true の場合は空間単位の配布を許し，false の場合は許さないを意味する．例を以下に示す．この広告は定期的に行われる．

```
Appliance( SQNY TV, SmartSpaceLab, 133.27.170.90, true )
```

MPDP の受け取りは，ネットワークを通じて行われる．GAYA クライアントモジュールが MPDP 配布を必要としている場合，GAYA サーバーモジュールが機器の所属空間を認識すると，GAYA クライアントモジュールの MPDP 受け取り部に対して MPDP を配布する．MPDP 受け取り部は MPDP を配布する相手が管理者であることの認証を行い，その認証が成功すると MPDP を受け取る．機器の所属する空間が変更された場合には，再び新しい MPDP が配布される．

### 4.4.3 GAYA サーバーモジュール

GAYA サーバーモジュールは，機器状況認識部，MPDP 配布部の主に２つのモジュールから成る．GAYA サーバーモジュールの内部構成図を図 4.9 に示す．

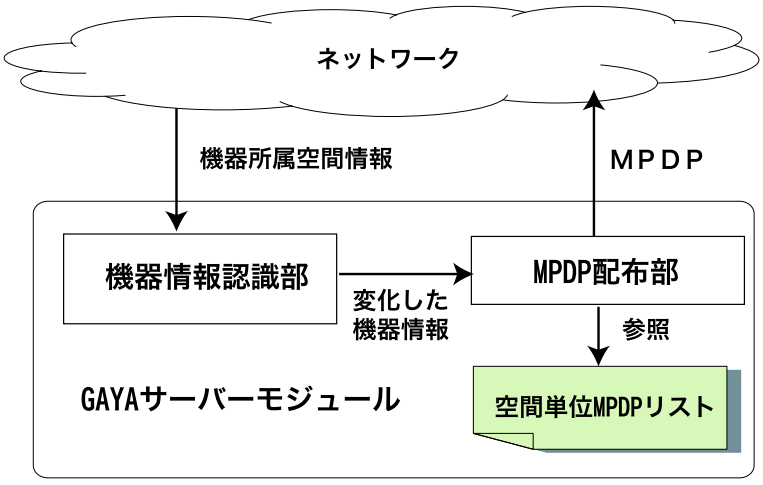


図 4.9: GAYA サーバーモジュール内部構成図

機器状況認識部は，複数の GAYA クライアントモジュールから送られてきた機器所属空間情報を参照し，機器が所属する空間に変更があった場合に MPDP 配布部に配布命令

を送る．MPDP 配布部は，空間単位 MPDP リストを参照し，管理者がその空間に設定した MPDP を GAYA クライアントモジュールにパスワード認証を行った上で，送信する．空間単位 MPDP リストには，この空間にはこの MPDP を適用する，という情報が記述されている．このリストも MPDP と同様の理由で，XML で記述する．

#### 空間単位 MPDP リスト記述方法

- 空間単位 MPDP リストの宣言

  MPDP は XML による構造化された文章であるため，XML 文章の宣言をファイルの先頭で行う．また，空間単位 MPDP リストであることを示すために，<MPDPatSPACE> タグを記述する．

```
<? xml="version1.0" ?>
<MPDPatSPACE>
</MPDPatSPACE>
```

- 空間

  空間は，<space>タグを用いて記述される．属性として，空間の識別名を記述する．<space>タグの内側には，適用する MPDP が記述される．

```
<space name="SSLab">
</space>
```

- MPDP

  MPDP は，<mpdp>タグを用いて記述される．<mpdp>タグの内側には，その空間に適用する MPDP が記述される．

```
<mpdp> For_SSLab.mpdp </mpdp>
```

#### 空間単位 MPDP リストの例

管理者が記述する空間単位 MPDP リストの例を図 4.10 示す．

この XML で記述された空間単位 MPDP リストを参照し，空間に適用する MPDP を決定し，空間に存在する各機器に MPDP を配布する．

## 4.5 本章のまとめ

本章では，MPDP の記述形式を定義し，MPDP を用いた排他制御機構 DARE と，空間単位 MPDP 配布機構 GAYA から構成されるシステム，DAREGAYA の設計について述べ

```
<?xml version="1.0" ?>
<MPDPatSPACE>
    <space name ="SSLab">
        <mpdp>For_SSLab.xml</mpdp>
    </space>
    <space name="t41">
        <mpdp>For_t41.xml</mpdp>
    </space>
    <space name="o208">
        <mpdp>For_o208.xml</mpdp>
    </space>
</MPDPatSPACE>
```

図 4.10: 空間単位 MPDP リスト例

た．DAREGAYA を用いることで，空間内部でのユーザグループ単位に対するポリシ設定，空間単位でのポリシ設定が可能となり，柔軟な機器排他制御が可能となる．

# 第5章

# DAREGAYAの実装

本章では，DAREGAYA の実装について述べる．

## 5.1 実装概要

本章では，MPDP Maker 及び DAREGAYA のプロトタイプ実装の概要について述べる．DAREGAYA は DARE と GAYA から構成される．DAREGAYA の実装構成図を図 5.1 に示す．DARE 及び GAYA クライアントモジュールは情報機器上に実装する．GAYA サーバーモジュール及び機器所属空間認識モジュールにはそれぞれ専用のマシンを用意する．それぞれの実装概要について述べる．

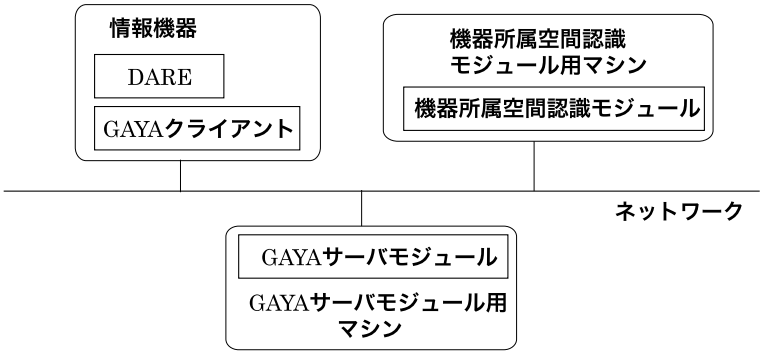


図 5.1: DAREGAYA 実装構成図

### 5.1.1 MPDP Maker の実装概要

MPDP Maker では複数の MPDP Element から MPDP を作成する機能を実現した．管理者が作成した root.xml を読み取り，グループオーナーの存在が記述されているとそのグループの MPDP Element を参照する．

### 5.1.2 DARE 及び GAYA の実装概要

DARE では機器排他制御モジュール，ユーザ・グループバインドモジュール，PDAlgo 決定モジュール，優先ユーザ決定モジュールを実装した．また，PDAlgo として明確な優先順位による優先アルゴリズム，要求到着順優先アルゴリズム，ランダムに優先ユーザが決定されるアルゴリズムの 3 つを実装した．DARE では管理者が設定した MPDP によって要求を優先するユーザを決定し，排他制御を行う機能を実現した．

GAYA では機器所属空間認識モジュール，GAYA クライアントモジュール，GAYA サーバーモジュールを実装した．また空間単位 MPDP リストを XML で記述し，空間単位 MPDP リストを解析するパーサーを実装した．GAYA では空間単位で MPDP を自動配布する機能を実現した．

### 5.1.3 実装環境

MPDP Maker 及び DAREGAYA は，表 5.1 に示す開発環境で実装した．DAREGAYA の実装言語として 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インタフェースクラスの利用が挙げられる．

| 項目 | 環境 |
|---|---|
| CPU | Intel Pentium M 1.80GHz |
| memory | 1GB |
| OS | Windows XP Professional SP2 |
| JDK | Java 2 Platform Standard Edition 5.0 |

表 5.1: MPDP Maker 及び DAREGAYA 開発環境

#### プラットフォーム非依存性

DARE，GAYA とも多様な機器上で動作するため，様々なプラットフォームで実行できる必要がある．Java 言語では，JavaVM がプラットフォームの差異を吸収し，Java プログラムの中間コードを実行する．JavaVM がインストールされた機器上であればプラットフォームを気にすることなく実行が可能であるため，DARE 及び GAYA の実装に適している．

#### インタフェースクラスの利用

それぞれの機器用に最適な排他制御を実行するために，管理者自身が PDAlgo を記述する場合が考えられる．また，GAYA において空間認識モジュールとして様々なセンサの利用が考えられ，用いるセンサによって管理者自身の実装が必要な場合がある．Java 言語ではインタフェースクラスを作成し，それを実装することで特定の機能を備えるクラスを実現できる．このため，PDAlgo や空間認識モジュールの実装時において，実装の方針を示すことができる．

## 5.2 MPDP Maker の実装

MPDP Maker は表 5.2 のクラス群から構成される．ElementReader は，まず管理者が設定した root.xml を読み込み，内部にグループオーナーの記述があれば，グループ名.xml を再帰的に読み込む．そして，MPDPPublisher が MPDP を作成する

| クラス名 | 機能 |
|---|---|
| ElementsReader | MPDP Element を読み込む |
| MPDPPublisher | 複数の MPDP Element から MPDP を作成する |

表 5.2: MPDP Maker 構成クラス

## 5.3 DARE の実装

本節では，DARE の実装の詳細について述べていく．DARE は表 5.3 のクラス群から構成される．

| クラスまたはインタフェース名 | 機能の説明 |
|---|---|
| ExclusionControl | 排他制御の実行 |
| Binder | ユーザ名とグループの結びつけ |
| Matcher | MPDP の解析 |
| Demand | ユーザとユーザの要求を管理 |
| PDAlgo | PDAlgo を実装するためのインタフェース |
| Order_of_a_row | 明確な優先順位づけによる PDAlgo |
| Order_of_demand_arrival | 要求到着順に優先順位を決定する PDAlgo |
| Random | ランダムに優先順位を決定する PDAlgo |

表 5.3: DARE 構成クラス

### 5.3.1 排他制御モジュール

排他制御モジュールとして`ExclusionControl`クラスを実装した．`ExclusionControl`クラスの`setDemand`メソッドでユーザの要求を受け取って管理し，`isNeedChoice`メソッドで排他制御の必要性を判定する．排他制御の必要性があれば，優先するユーザを選択するモジュールへユーザ情報を渡し，優先ユーザが決定されれば，`doDemand`メソッドで機器に命令を渡す．`ExclusionControl`クラスのうち，要求を実行するユーザを選択し，排他制御を実行する部分のソースを図 5.2 に示す．

ユーザの要求は，厳密には全く同時に起こりえることはない．先に要求を実行しているユーザがいる時に，他人の要求が到着した場合に排他制御が必要となり，その２人を対象として優先ユーザを決定する．複数のユーザが要求を行ってきた場合に，この処理を必要な数だけ実行するように実装を行った．また，優先されたユーザが機器の制御権限を解放する場合には，`removeUserDemand`メソッドが処理を行う．

### 5.3.2 ユーザ・グループバインドモジュール及び PDAlgo 決定モジュール

ユーザ・グループバインドモジュール及び PDAlgo 決定モジュールの機能を`Binder`クラスで実装した．`Binder`クラスは内部に`Matcher`クラスを持ち，`Matcher`クラス

```
if(isNeedChoice(user) == true){
        Binder binder = new Binder(access_usrs);
        PDAlgo pdalgo = binder.getPDAlgo();

        Demand d_usr = pdalgo.selectUser((Demand)access_usrs.elementAt(0),user);
        doDemand(d_usr);

}else{
        doDemand(user);
}
```

図 5.2: 排他制御実行部のソース

が MPDP のパーサーとして動作し，`Binder` クラスの各メソッドの処理に必要な MPDP の要素を解析する．

`Binder` クラスの `u_g_bind` メソッドでそれぞれのユーザが所属するグループを認識し，`decidePDAlgo` メソッドで用いる PDAlgo 名を決定する．用いるそれぞれの PDAlgo はクラスとして実装し，`decidePDAlgo` メソッドで得た PDAlgo 名の `PDAlgo` クラスのインスタンスを生成する．この処理を行う `getPDAlgo` メソッドのソースを図 5.3 に示す．

```
public PDAlgo getPDAlgo(){

        PDAlgo pd;
        Class pdalgo;

        try{
                pdalgo = Class.forName("jp.ac.keio.sfc.ht.daregaya.dare.pdalgo."+r_pdalgo);
                pd = (PDAlgo)pdalgo.newInstance();
                return pd;

        }catch(Exception e){
                e.printStackTrace();
        }

        return null;
}
```

図 5.3: getPDAlgo メソッド

### 5.3.3 優先ユーザ決定モジュール

優先ユーザ決定モジュールは，PDAlgo インタフェースを実装した PDAlgo 実装クラスで実装される．このクラスは，要求到着順で優先ユーザを決定したり，ランダムで決定したりする PDAlgo を実現したクラスである．図 5.4 に PDAlgo インタフェースを示す．PDAlgo インタフェースは，ユーザとユーザ要求を内部に保持する Demand クラスのインスタンスを 2 つ引数にとり，そのどちらかのインスタンスを返値として返す selectUser メソッドを実装することを規定する．

```
public interface PDAlgo{

        public Demand selectUser(Demand usr1,Demand usr2);

}
```

図 5.4: PDAlgo インタフェース

PDAlgo 実装クラスとして，MPDP に記述された優先順位順に優先ユーザが決定される Order_of_a_row クラスと，要求到着順に優先ユーザが決定される Order_of_demand_arrival クラスと，ランダムに優先ユーザが決定される Random クラス，過去の複数ユーザの要求履歴を見て，多数決で優先する要求を決定する Majority_decision を実装した．このうち，Order_of_a_row クラスを図 5.5 に示す．

## 5.4 GAYA の実装

本節では，GAYA のプロトタイプ実装について述べる．

### 5.4.1 機器所属空間認識モジュール

機器所属空間認識モジュールを構成する主要なクラスとインタフェース一覧を表 5.4 に示す．

| クラスまたはインタフェース名 | 機能の説明 |
|---|---|
| DetectApplianceSpace | 機器の所属空間認識モジュールのためのインタフェース |
| ApplianceInEventListener | 機器の空間所属を通知するイベントリスナ |
| ApplianceSpacePerceptioner | 機器所属空間認識モジュールの起動 |
| ApplianceInfoReceiverForAd | 機器情報の受信 |
| AdminAppliance | 機器情報の管理 |
| SendSpaceInfo | 機器所属空間情報の送信 |

表 5.4: 機器所属空間認識モジュール構成主要クラス

```
public class Order_of_a_row implements PDAlgo {

	public String selectUser(String usr1, String usr2) {

		DocumentBuilderFactory dbfactory;
		DocumentBuilder builder;
		Document doc;
		Element root;
		NodeList list;

		try{
			dbfactory = DocumentBuilderFactory.newInstance();
			builder = dbfactory.newDocumentBuilder();
			doc = builder.parse(new File("./MPDP.xml"));
			root = doc.getDocumentElement();
			list = root.getElementsByTagName("group");
			Element element = (Element)list.item(0);
			NodeList userList = element.getElementsByTagName("user");

			for(int t=0;t<userList.getLength();t++){

				Element userElement = (Element)userList.item(t);
				String user = userElement.getFirstChild().getNodeValue();

				if(user.equals(usr1)||user.equals(usr2)){
					return user;
				}
			}

		}catch(Exception e){
			e.printStackTrace();
		}
		return null;
	}
}
```

図 5.5: Order_of_a_row クラス

機器の所属空間を認識するセンサとして，RF-CODE [14] 社製の RFID を用いた．機器にタグをつけ，環境側に設置されたリーダで読み取る．機器とタグ ID は機器所属空間認識モジュール内で結びつけられており，機器がリーダで検知できる範囲に入ったイベントが確認されると，その機器に対して所属空間を通知する．機器が所属する空間を認識できるセンサとしては他にも位置情報センサ等の使用が考えられるため，汎用的にモジュールが使用できるようにセンサ周りの実装のために `DetectApplianceSpace` インタフェース（図 5.6）を定義した．

`DetectApplianceSpace` インタフェースの実装クラスは，機器が空間に所属したことを通知するイベントソースの役割を果たす．`ApplianceSpacePerceptioner` クラスは，`ApplianceInEventListener` インタフェースを実装し，機器の空間所属が確

```
public interface DetectApplianceSpace {

        public void addApplianceInEventListener(ApplianceInEventListener listener);
        public void removeApplianceInEventListener(ApplianceInEventListener listener);
        public void applianceIn();
        public void applianceOut();

}
```

図 5.6: DetectApplianceSpace インタフェース

認されると `applianceIn` メソッドが呼び出され，`SendSpaceInfo` クラスに所属空間を機器に対して通知させる．機器の IP アドレス等の情報は，`ApplianceInfoReceiverForAd` クラスが受け取り，`AdminAppliance` クラスが管理する．図 5.7 に `ApplianceSpacePerceptioner` クラスのソースを示す．

```
public class ApplianceSpacePerceptioner implements ApplianceInEventListener{


        AdminAppliance aa = new AdminAppliance();

        public static void main(String args[]){
                ApplianceSpacePerceptioner as = new ApplianceSpacePerceptioner();
        }

        public ApplianceSpacePerceptioner(){
                ApplianceInfoReceiverForAd airf = new ApplianceInfoReceiverForAd(aa);
                ApplianceInEventByRFID ai = new ApplianceInEventByRFID();
        }

        public void applianceIn(ApplianceInEvent e) {
                aa.sendSpaceInfo(e.getSpaceName(),e.getApplianceName());
        }

        public void applianceOut(ApplianceInEvent e){
                aa.sendSpaceInfo(e.getSpaceName(),e.getApplianceName());
        }

}
```

図 5.7: ApplianceSpacePerceptioner クラス

| クラスまたはインタフェース名 | 機能の説明 |
|---|---|
| GayaClient | GAYA クライアントモジュールの起動 |
| GetSpaceInfo | 機器所属空間情報を受信 |
| ApplianceInfoAdvertiser | 機器情報・機器所属空間情報を広告 |
| GetMPDP | MPDP の受信 |

表 5.5: GAYA クライアントモジュール構成クラス群

### 5.4.2 GAYA クライアントモジュール

GAYA クライアントモジュールを構成するクラス一覧を表 5.5 に示す．GetSpaceInfo クラスが機器所属空間認識モジュールから機器が所属する空間情報を受け取り，その空間情報と機器情報をセットにして `ApplianceInfoAdvertiser` クラスがネットワークに対して広告を行う．GetMPDP クラスは GAYA サーバーモジュールから MPDP を受け取る．

### 5.4.3 GAYA サーバーモジュール

| クラスまたはインタフェース名 | 機能の説明 |
|---|---|
| GayaServer | GAYA サーバモジュールの起動 |
| ApplianceInfoReceiver | 機器情報・所属空間情報受信 |
| DecideSendMPDP | MPDP を配布することの決定 |
| BinderSpaceToMPDP | 空間単位 MPDP リストから，送信する MPDP を決定 |
| SendMPDP | MPDP を送信する |

表 5.6: GAYA サーバーモジュール構成クラス群

GAYA サーバーモジュールを構成するクラス一覧を表 5.6 に示す．`ApplianceInfoReceiver` クラスが GAYA クライアントから広告された機器情報・所属空間情報を受け取り，管理する．`DecideSendMPDP` クラスが機器に対し MPDP の配布の必要性を判定すると，`BinderSpaceToMPDP` クラスが空間単位 MPDP リストから管理者が設定したその空間に対しての MPDP をとりだす．`SendMPDP` クラスがその MPDP を GAYA クライアントに配布する．

## 5.5 本章のまとめ

本章では，MPDP Maker の実装をまず行い，DAREGAYA のプロトタイプとして DARE と GAYA をそれぞれ実装した．DARE では MPDP を用いて優先ユーザを決定し，排他制御を行うことができる機能を実装した．また，`PDAlgo` インタフェースを定義し，実装クラスとして 3 クラス実装した．GAYA では RFTag/Reader を用いた機器所属空間認識モジュールを実装し，管理者が別のセンサを用いても実装できるように `DetectApplianceSpace`

インタフェースを定義した．GAYA クライアント，サーバーモジュールをそれぞれ実装し，空間ごとに MPDP を配布する機能を実装した．

# 第6章

# DAREGAYAの評価

本章では，DAREGAYAの評価について述べる．まずDAREGAYAを定量的に評価し，次に定性的に評価する．

## 6.1 DAREGAYA の定量的評価

本節では，DAREGAYA の定量的評価を行う．まず DARE の評価を行い，次に GAYA の評価を行う．

### 6.1.1 DARE の定量的評価

本項では，DARE の定量的評価について述べる．DARE の評価方針として，MPDP による排他制御を行うためのオーバーヘッドを計測する．DARE に複数ユーザの要求が生じてから，PDAlgo を用いた排他制御が実行されるまでの時間を計測する．

#### 評価環境

PC 上で音楽プレイヤーを実装し，音楽プレイヤーの音楽再生サービスに対する排他制御機構として DARE を用いた．複数ユーザはそれぞれ異なる音楽を再生する要求を出し，その要求に対して排他制御を行う．評価環境は表 5.1 に示したものと同様である．MPDP は 4.2.6 で示した例を使用した．PDAlgo として，明確な優先度づけによる優先ユーザを決定する Order_of_a_row，要求到着順に優先ユーザを決定する Order_of_demand_arrival，ランダムに優先ユーザが決定される Random が用いられる．

#### 排他制御実行時間

ユーザの要求を同時に発生させて，優先されるユーザが決定されるまでの時間を測定する．この測定時間は，複数ユーザの要求を受けてから，MPDP を参照してユーザを比較するための PDAlgo を選択し，PDAlgo による排他制御が行われる処理時間を意味する．PDAlgo ごとの実行時間を計測できるように，制御を要求するユーザを調整した．測定を５０回行い，平均実行時間を算出した．図 6.1 に PDAlgo ごとの測定時間を示す．

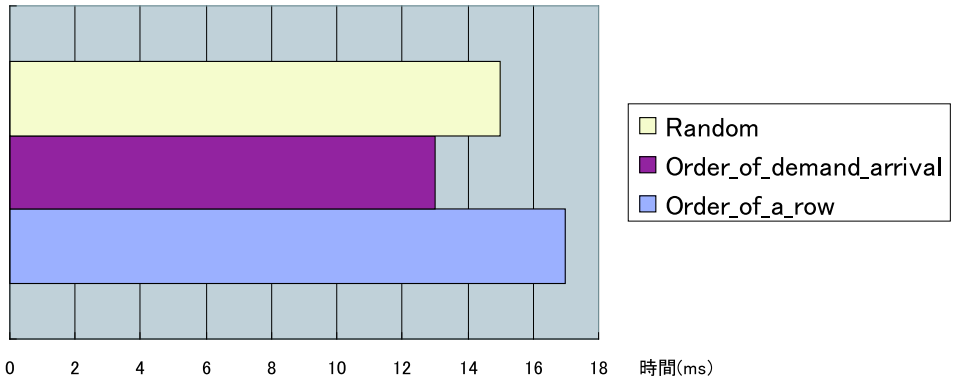


図 6.1: 各 PDAlgo ごとの排他制御実行時間

それぞれの PDAlgo に対する平均排他制御実行時間は，Order_of_a_row が 17 ミリ秒，Order_of_demand_arrival が 13 ミリ秒，Random が 15 ミリ秒であった．優先するユーザを選択するアルゴリズムによって細かい差はあるが，実際の実行時間としてはどれも短く，実際の使用に耐えうるものと評価できる．

### 6.1.2 GAYA の定量的評価

本項では，GAYA の定量的評価について述べる．GAYA の評価方針として，機器の数に応じた MPDP 配布の時間変化について考察する．

#### 評価環境

GAYA サーバモジュールを表 5.1 のマシン上で起動させ，GAYA クライアントモジュールを８台のマシン上で起動させた．GAYA サーバは 100BASE-TX でネットワークに接続されている．GAYA クライアントモジュールを起動させるそれぞれのマシンスペックを表 6.1 に示す．

| ノード No | OS | CPU | memory | 接続形態 |
|---|---|---|---|---|
| 1 | WindowsXP SP2 | Intel PentiumM 1.30GHz | 760MB | IEEE 802.11g |
| 2 | WindowsXP SP2 | Intel PentiumM 1.60GHz | 1.0GB | IEEE 802.11g |
| 3 | WindowsXP SP2 | Intel PentiumM 1.7GHz | 1.0GB | IEEE 802.11g |
| 4 | WindowsXP SP2 | Intel PentiumM 2,1GHz | 2.0GB | IEEE 802.11g |
| 5 | WindowsXP SP2 | Intel PentiumM 1.7GHz | 1.0GB | IEEE 802.11g |
| 6 | WindowsXP SP2 | Intel Pentium3 1.2GHz | 760MB | IEEE 802.11b |
| 7 | MacOS 10.3 | PowerPC G4 1.25GHz | 1.5GB | 54Mbps AirMacExtreme |
| 8 | MacOS 10.3 | PowerPC G4 Dual1.25GHz | 768MB | 100BASE-TX |

表 6.1: GAYA クライアントモジュール起動ノード

#### 機器数の変化に対する MPDP 配布実行時間の変化

空間に存在する機器数を変化させながら，それぞれの機器が空間に所属し，GAYA サーバーが機器情報・所属空間情報を受け取ってから空間単位 MPDP リストを参照し，その機器に適した MPDP の配布が終了するまでの実行時間を計測した．機器が所属する空間を２つ用意し，それぞれの空間に対して設定する MPDP のサイズは 608 バイトと 419 バイトである．それぞれ５０回測定を行い，その平均時間を算出した．図 6.2 に測定結果を示す．

機器数の増加に対して，ゆるやかな時間増加が確認された．機器数を $x$ 軸にとり，MPDP 配布時間を $y$ 軸にとると，$y = 0.88x + 25.4$ という式に近似できた．この式を用いると，機器が１００台あった時の予測 MPDP 配布時間は $y = 0.88 \times 100 + 25.4 = 113.4$ ミリ秒となり，MPDP 配布機構として実際の使用に耐えうるものと評価できる．

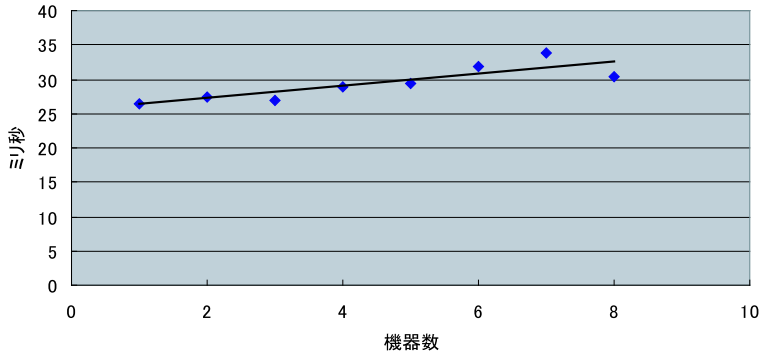


図 6.2: 機器数の変化に対する MPDP 配布実行時間の変化

## 6.2 DAREGAYA の定性評価

3.3 章で挙げたような，特定のポリシによる固定的な排他制御手法と DAREGAYA を定性的に比較する．比較する項目は，環境適応性と導入簡易性である．比較の結果を表 6.2 に示す．

| 比較項目 | DAREGAYA | 特定ポリシによる固定的な排他制御 |
|---|---|---|
| 環境適応性 | ○ | × |
| 導入簡易性 | △ | ○ |

表 6.2: 排他制御手法の比較

### 6.2.1 環境適応性

環境適応性とは，その機器が使用される環境に適した排他制御方法であるかということを指す．ラウンド・ロビン・スケジューリング方式や優先順位スケジューリング方式による固定的な排他制御では，様々な環境に対して適応できない．DAREGAYA では，MPDP によって機器を使用するユーザの性質を考慮して管理者が複雑なポリシを設定できる上，機器が所属する空間の性質によってそのポリシを動的に変更することができるため，環境適応性を満たしている．

### 6.2.2 導入簡易性

導入簡易性とは，管理者が設定したポリシによる排他制御機構の導入が容易であることを指す．予め機器に固定的なポリシを組み込んでおく場合は，管理者が設定する部分がないため，導入簡易性の面では優れている．DAREGAYA による排他制御では，柔軟な排他制御が可能な分，管理者の設定を必要とするため，トレードオフとして導入の手間がかかる．しかし，DAREGAYA は空間単位での MPDP 配布機構を備え，その導入の手間を減少させている．

## 6.3　本章のまとめ

本章では，DAREGAYA の定量的評価，定性評価を行った．定量的評価では，MPDP を参照して排他制御が実行される時間，機器数の変化による MPDP 配布時間について考察した．定性評価では，固定的なポリシによる排他制御方法に対する優位性を示した．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

本章では，今後の課題と結論について述べる．

## 7.1 今後の課題

本節では，DAREGAYA が今後考慮すべき課題として，多様な PDAlgo の考案・実装と不特定多数のユーザが機器利用を行う際の対応を挙げる．

### 7.1.1 多様な PDAlgo の考案・実装

本論文では，要求到着順による優先ユーザ決定，明確なプライオリティづけによる優先ユーザ決定，ランダムな優先ユーザ決定を行う PDAlgo を実装した．今後，管理者がより柔軟性を持つ MPDP を記述するために，多種多様な PDAlgo を考案し，実装しておくことで，管理者が自ら PDAlgo を記述する手間を省くことができる．この例として，センサを用いて環境状況に応じて動的に優先順位を変更する PDAlgo や，過去のユーザの使用頻度をみて動的に優先順位を変更する PDAlgo 等が期待される．

### 7.1.2 不特定多数のユーザが機器利用を行う際の対応

本論文はオフィスや研究室等の狭いコミュニティを対象としており，機器を使用するユーザが把握できている状況を扱っている．今後，公共空間に様々な人が使用する公共端末が増加した場合に，不特定多数のユーザが機器利用を行うことが想定される．この状況においても柔軟に対応できるように，MPDP ポリシの記述力を高め，DAREGAYA を改良していく必要がある．

## 7.2 結論

本論文では，マルチユーザ・コンテキストアウェアサービス環境において柔軟な排他制御を行うために，マルチレベル優先順位決定ポリシによる排他制御方法を提案した．また，マルチレベル優先順位決定ポリシに基づく排他制御機構及びポリシ配布機構を実現する DAREGAYA を実装し，評価した．

DAREGAYA は，機器の管理者は環境に存在するユーザの性質を考慮した柔軟な排他制御を実現する．本論文では，マルチレベル優先順位決定ポリシの記述形式を定義し，管理者もしくはグループオーナが階層的にグループ及びグループに適用される PDAlgo を記述できるようにした．また，管理者が独自のポリシによってグループ内の排他制御を行えるようにインタフェースを定義し，排他制御に柔軟性を持たせた．さらに，管理者が様々なセンサを用いて空間を認識できるようにインタフェースを定義した．本論文では，評価として DAREGAYA を用いた排他制御実行時間とポリシの配布時間の計測を行い，既存の固定的な排他制御に対して環境に適応した柔軟な排他制御を行えることを示した．

# 謝辞

本研究を進めるにあたり，懇切丁寧な御指導を賜わりました，慶應義塾大学環境情報学部教授徳田英幸博士に深く感謝いたします．また，政策・メディア研究科助教授の高汐一紀博士を始め，本論文執筆にあたって多方面に及び相談にのっていただいた慶應義塾大学徳田研究室の諸先輩方に感謝いたします．

慶應義塾大学徳田研究室で励ましあってきた 4 年生の皆さんと，影ながら支えてくれた 1，2，3 年生の皆さんにもお礼いたします．特に，共に研究生活を送った move!プロジェクトグループの小泉健吾氏，榊原博氏とは切磋琢磨して貴重な刺激を頂くことができました．ACE プロジェクトグループの松倉友樹氏には，元気の源を数多く頂きました．本多道夫氏には研究に対する数多くの指摘，激励を頂き，また手料理を多くご馳走になりました．NTTDoCoMo ネットワーク研究所の永田智大氏には，本研究に対して数多くの助言をいただきました．

そして，私を温かく支えてくれる親，兄弟，親戚に感謝します．最後に，本論文執筆中に逝去した祖母にこの論文を捧げます．

平成 16 年 12 月 29 日
米澤拓郎

# 参考文献


[1] 東芝ネットワーク家電 フェミニティ
*http://feminity.toshiba.co.jp/feminity/*

[2] ホームネットワークシステム「くらしネット」
*http://national.jp/appliance/product/kurashi-net/*

[3] Weiser, M.: The Computer for the 21st Century, *Scientific American*, Vol. September, pp. 94-100, 1991.

[4] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David E. Culler, and Kristofer S. J. Pister. System architecture directions for networked sensors. *In Architectural Support for Programming Languages and Operating Systems*, pp. 93-104, 2000.

[5] L.E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl and H.W. Gellersen. Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts, *Proc. of UBICOMP 2001* Atlanta, GA, USA, Sept. 2001.

[6] Mike Addlesee, Rupert Curwen, Steve Hodges, Joe Newman, Pete Steggles, Andy Ward, Andy Hopper. Implementing a Sentient Computing System. *IEEE Computer Magazine*, Vol. 34, No. 8, pp. 50-56, August 2001.

[7] Adam Smith, Hari Balakrishnan, Michel Goraczko, Nissanka Priyantha,Tracking Moving Devices with the Cricket Location System, *Proc. 2nd USENIX/ACM MOBISYS Conf.*, Boston, MA, June 2004.

[8] Manuel Roman, Christopher K. Hess, Renato Cerqueira, Anand Ranganathan, Roy H. Campbell, and Klara Nahrstedt. Gaia: A Middleware Infrastructure to Enable Active Spaces. *In IEEE Pervasive Computing*, pp. 74-83, Oct-Dec 2002.

[9] Easy Living
*http://research.microsoft.com/easyliving/*

[10] Okoshi, T., Wakayama, S., Sugita, Y., Iwamoto, T., Nakazawa, J., Nagata, T., Furusaka, D., Iwai, M., Kusumoto, A., Harashima, N., Yura, J., Nishio, N., Tobe, Y., Ikeda, Y. and Tokuda, H.: Smart Space Laboratory Project: Toward the Next Generation Computing Environment, *IEEE Third Workshop on Networked Appliances (IWNA)2001*, 2001.

[11] Andrew S. Tanenbaum: MODERN OPERATING SYSTEMS SECOND EDITION, pp.142-143

[12] Andrew S. Tanenbaum: MODERN OPERATING SYSTEMS SECOND EDITION, pp.143-144

[13] Andrew S. Tanenbaum: MODERN OPERATING SYSTEMS SECOND EDITION, pp.146

[14] RF Code - Real-Time Asset and Personnel Identification, Tracking and Location Solutions Utilizing Active RFID Technology
`http://www.rfcode.com/`