修士論文　　2002年度(平成14年度)

自律的な情報家電機器制御機構の構築

慶應義塾大学大学院　政策・メディア研究科
青木 崇行

修士論文要旨　　2002年度(平成14年度)

自律的な情報家電機器制御機構の構築

論文要旨

本論文では、ホームユビキタスコンピューティング環境に適した自律的な情報家電機器制御機構の構築について述べる。環境にコンピュータや様々な機器が遍在し、ユーザが様々な形態でそれらを利用可能になる一方で、機器やサービスの利用方法の複雑化や高機能化も同時に起こっている。同様の現象は家庭内の情報家電機器環境にも起きており、様々なユーザがシステムリソース、ネットワーク接続、機能、操作方法の異なる様々な機器を、単体もしくは複数同時に制御する必要が出てきている。

上記のような問題点を解決するために、本論文では、前向き推論アルゴリズムを利用した自律的な情報家電機器制御機構であるAONA(Autonomic Operation of Networked Appliances)を構築した。AONAを利用することにより、ユーザは状況認識、利用判断、機器利用の各ステップを制御機構に任せられる。

AONAの設計においては、自律的な情報家電機器制御に必要な自律的機能、制御方法の検討、実時間処理、セキュリティ等の設計方針について述べ、AONAの主要な4つのモジュールであるSensing Module、Planning Module、Operating Module、User Interfaceについて検討する。特にPlanning Moduleについては、制御プラン作成アルゴリズム、制御プラン記述フォーマット、制御プラン競合回避機能、制御プラン整合機能について考慮し、設計する。また実装については、利用したセンサ、情報家電機器について述べ、さらに、各モジュールのアーキテクチャについて相互関係、提供するメソッド、サンプルコードなどを示す。評価では、AONAの実用性を示すために、定量的評価を行う。また、関連研究との違いを明確にし、優位性を示すために定性的評価を行い、Plangent、HIVE、ADS、Autonomic Compuitngと比較する。

AONAの実現により、ユーザは状況認識、利用判断、機器利用の各作業をする必要がなくなり、制御機構が自律的な情報家電機器制御を実現する。AONAはホームユビキタスコンピューティング環境に自律型コンピューティングを適用したシステムとなる。

キーワード：

1 自律型コンピューティング　2 ユビキタスコンピューティング　3 情報家電コンピューティング　4 推論システム　5 機器制御

慶應義塾大学大学院　政策・メディア研究科

青木 崇行

Abstract of Master's Thesis

Academic Year 2002

Autonomic Operation of Networked Appliances

Summary

This paper proposes an autonomic operation system for home ubiquitous computing environment that consists of heterogeneous networked devices and appliances. Currently users are required to operate appliances one by one using their remote controllers or web enabled cellular phones to conduct home ubiquitous computing. In this operation, users are required to perform three actions: sensing, planning, and operating. First, they sense the environment or appliances condition. Then, they think how to operate the appliances. Finally, they operate the appliances. Recent development of networked appliances with rich functions are increasing users' burden to have premise knowledge on appliances and to perform above-mentioned three actions.

To address this problem, I have designed and implemented AONA system which eliminates the users' burden by enabling an autonomic behavior of the home ubiquitous computing environment. AONA system is composed of three modules: sensing module, planning module, and operating module.

The sensing module subrogates the sensing action of users by acquiring the environmental information through networked sensors such as a timer, a location sensor, and a weather sensor. The planning module produces operation plans by using reasoning algorithms. The reasoning algorithm retrieves the operation plans from the rule sets registered on ahead. When a certain condition which users registered in advance occurs, the system autonomously detects the condition and produces operation plans. Operating module operates the networked appliances or services.

By using the AONA system, the users are no longer required to check the condition of environment, to think about operation of appliance, and to operate each appliances by themselves. The AONA systems performs these actions in stead of users and realize autonomic computing in the home ubiquitous computing environment.

Keywords:

1 Autonomic Computing 2 Ubiquitous Computing 3 Networked Appliances Computing
4 Reasoning System 5 Device Control

Keio University Graduate School of Media and Governance
Soko Aoki

目 次

第 1 章 序論 **1**
1.1 本研究の背景 1
1.2 本研究の目的と概要 1
1.3 本研究の特徴 2
1.4 本研究の構成 2

第 2 章 ユビキタスコンピューティングと情報家電機器環境 **3**
2.1 ユビキタスコンピューティング環境 4
2.1.1 ユビキタスコンピューティング 4
2.1.2 ホームユビキタスコンピューティング 4
2.1.3 自律型ユビキタスコンピューティング 5
2.2 問題点と解決方法 5
2.2.1 問題点 6
2.2.2 本研究での解決方法 7
2.3 自律的な情報家電機器制御機構 7
2.3.1 家電機器利用時に必要な作業の分類 7
2.3.2 自律的な情報家電機器制御機構の概要 8
2.3.3 自律的な情報家電機器制御機構の機能要件 8
2.4 関連研究 9
2.4.1 Plangent 9
2.4.2 HIVE 10
2.4.3 ADS 10
2.4.4 Autonomic Computing 11
2.4.5 関連研究の比較 12
2.5 本章のまとめ 12

第 3 章 自律的な情報家電機器制御機構 (AONA) の設計 **13**
3.1 設計方針 14
3.1.1 自律的機能 14
3.1.2 集中制御と分散制御の比較 15
3.1.3 実時間処理 16

3.1.4 セキュリティ . . . 17
3.2 システム構成 . . . 18
3.2.1 ハードウェア構成 . . . 18
3.2.2 ソフトウェア構成 . . . 18
3.3 Sensing Module の設計 . . . 18
3.3.1 State-Full と State-Less の比較 . . . 19
3.4 Planning Module の設計 . . . 20
3.4.1 制御プラン作成アルゴリズム . . . 20
3.4.2 制御プラン記述フォーマット . . . 22
3.4.3 制御プラン競合回避機能 . . . 23
3.4.4 制御プラン整合機能 . . . 24
3.5 Operating Module の設計 . . . 24
3.5.1 メッセージ通信制御と移動コード制御の比較 . . . 24
3.5.2 制御結果のフィードバック . . . 25
3.6 User Interface の設計 . . . 25
3.6.1 GUI の設計 . . . 26
3.6.2 GUI 表示ハードウェアの検討 . . . 26
3.7 本章のまとめ . . . 26

第 4 章 自律的な情報家電機器制御機構 (AONA) の実装 27
4.1 実装方針 . . . 28
4.2 実装概要 . . . 28
4.3 Sensing Module の実装 . . . 28
4.3.1 実装に利用したハードウェア . . . 29
4.3.2 Sensor クラスとそのサブクラス . . . 30
4.3.3 StateUniter クラス . . . 30
4.4 Planning Module の実装 . . . 31
4.4.1 Brain クラス . . . 31
4.4.2 Container クラス . . . 32
4.4.3 PlanContainer クラス . . . 32
4.4.4 RulebasedPlan クラス . . . 34
4.5 Operating Module の実装 . . . 34
4.5.1 実装に利用したハードウェア . . . 34
4.5.2 実装したソフトウェア . . . 36
4.6 User Interface の実装 . . . 37
4.7 動作手順 . . . 37
4.8 アプリケーション例 . . . 38
4.8.1 習慣的制御モデル . . . 38
4.8.2 複数機器同時制御モデル . . . 39
4.9 本章のまとめ . . . 39

第 5 章 自律的な情報家電機器制御機構 (AONA) の評価 **41**
5.1 定量的評価 42
5.1.1 測定環境 42
5.1.2 測定方法 42
5.1.3 測定結果 42
5.2 定性的評価 44
5.2.1 自律性の評価 44
5.2.2 環境親和性の評価 45
5.2.3 簡便性の評価 45
5.3 本章のまとめ 46

第 6 章 結論 **47**
6.1 今後の課題 47
6.1.1 学習アルゴリズムの導入 47
6.1.2 ルックアップサービスの導入 47
6.1.3 実時間処理の検討 48
6.1.4 セキュリティモデルの検討 48
6.2 本論文のまとめ 48

参考文献 **51**

図 目 次

2.1 従来の制御と、AONA による制御 9
2.2 Plangent 10
2.3 HIVE 11
2.4 ADS 11

3.1 従来からの家電機器制御との比較 14
3.2 集中制御と分散制御 16
3.3 ハードウェア構成図 18
3.4 ソフトウェア構成図 19
3.5 State Full / State Less な Sensing Module 19
3.6 一段階推論と多段階推論 22
3.7 プラン記述フォーマット 23
3.8 BNF 記法による制御プラン記述フォーマット 23
3.9 BNF 記法による制御プラン記述フォーマット使用例 23
3.10 User Interface のステートチャート図 26

4.1 慶應義塾大学徳田研究室 SSLab 28
4.2 実装概要 29
4.3 Sensing Module クラス図 30
4.4 Planning Module クラス図 31
4.5 PlanContainer クラスの一部 33
4.6 RulebasedPlan クラスの一部 35
4.7 Operating Module クラス図 36
4.8 User Interface クラス図 37
4.9 User Interface スクリーンダンプ 38
4.10 習慣的制御モデルのプラン例 39
4.11 複数機器同時制御モデルのプラン例 40

5.1 測定 1 ：全体的な処理時間 43
5.2 測定 2 ：プラン数変化による処理時間の測定 43
5.3 AONA シーケンス図 44

表 目 次

2.1 関連研究の比較 12

4.1 実装に利用したセンサ 29
4.2 Sensor クラスのメソッド一覧 30
4.3 StateUniter クラスのメソッド一覧 31
4.4 Brain クラスのメソッド一覧 32
4.5 Container クラスのメソッド一覧 32
4.6 実装に利用した家電機器 36
4.7 Appliance クラスのメソッド一覧 37

5.1 測定環境 42
5.2 関連研究と AONA の比較 44

第 1 章

序論

1.1 本研究の背景

情報処理能力やネットワーク接続機能を持った機器が環境に遍在する、ユビキタスコンピューティング環境が登場し始めている。 また、その環境に存在するユーザも携帯電話や PDA（小型情報端末）を持ち運び、ネットワークへの接続性を保持できている。 このユビキタスコンピューティング環境における様々なサービスも登場し、様々なユーザが利用している。

家庭内にも同様の変化は起きており、様々な家電機器が情報処理能力やネットワーク接続機能を持ち始め、情報家電機器と呼ばれるようになり、機器の持つ機能が増加、複雑化している。またそれら機器の持つ機能を利用するための操作形態も多様化し、例えば、リモートコントローラを利用したり、携帯電話を利用してインターネット経由で利用したりする。さらに、現在の家電機器は機器を単体で利用すると同時に、複数機器の同時利用（機器間コラボレーション）、多様なメディアタイプへの対応（例えば、ステレオが CD、MD、MP3 などに対応している ）などが可能となっている。

このように、近年の家電機器ユーザは、情報家電機器制御のために、従来と比較して、より多くの操作方法、機器の機能などを学ぶ必要が出てきている。しかし、すべてのユーザにすべての機器の機能、操作方法を学ばせるのは現実的には難しい。

1.2 本研究の目的と概要

本研究の目的は、情報家電機器制御におけるユーザの負担を減らすために、自律的な情報家電機器制御機構を構築することにある。

本研究における自律的な情報家電機器制御機構は、情報家電機器制御において、これまでユーザが自ら行っていた、環境認識、制御判断、実際の制御の各作業を、ユーザに置き換わってシステムが行う。 環境認識においては、ネットワーク接続されたセンサーや家電機器より、ユーザの制御状態、環境情報などを取得し、プログラムで利用可能なフォーマットに情報を変換する。また、制御判断においては、与えられた情報から、ユーザの嗜好にあった家電機器制御プランを生成する。さらに制御において、

作成された制御プランに従い、各種機器をシステムが制御を行う。ユーザは初期設定さえ行えば、上記作業中は、特に入力や出力の確認を求められず、制御機構がユーザとは自律的に制御を行う。

1.3　本研究の特徴

本研究の特徴は、大きく分けて3点ある。まず第1に、ホームユビキタスコンピューティング環境に自律型コンピューティングの考えを取り入れ、これまではユーザの逐次制御が必要だったところに、前向き推論アルゴリズムを利用し、複数の機器を自律的に制御可能な制御機構を構築したことである。第2に、ホームユビキタスコンピューティング環境について、その機器、機能、ユーザ等について考察し、これまでは複数の機器を使用する場合は複数の制御機器を用いる必要があったところを、統一的なインターフェイスを設計、実装し、ユーザにとってより簡便な情報家電機器利用できる機能を提供したことにある。そして第3に、現在存在するセンサ機器、情報家電機器を利用した、プロトタイプを設計、実装、評価し、実機とともに利用可能なソフトウェア群を提供したことにある。

1.4　本研究の構成

本論文では、まず2章でユビキタスコンピューティングと情報家電機器環境について説明し、情報家電機器制御機構の機能要件を述べる。また、他の研究機関、企業で研究開発が行われているエージェントシステム、制御システムなどの関連研究について述べ、本研究の意義を考察する。3章では、自律的な情報家電着機器制御機構 AONA について、その設計方針、システム構成、各ソフトウェアのモジュールが採用したアルゴリズムやアーキテクチャを含めた設計について述べる。4章では、情報家電機器制御機構の実装について述べ、5章で評価として、定量的評価、定性的評価を行う。最後に6章で、今後の課題を述べ、本論文をまとめる。

第 2 章

ユビキタスコンピューティングと情報家電機器環境

本章では、本研究の背景となったユビキタスコンピューティング、情報家電機器環境について概観し、本研究の目的である自律的な情報家電機器制御機構について考察する。また、関連研究として、Plangent、HIVE、ADS、Autonomic Computing について触れ、本研究との関連性、違いについて論じる。

2.1 ユビキタスコンピューティング環境

本章では、本研究の背景、想定環境となるユビキタスコンピューティングについて、一般的なユビキタスコンピューティング、ホームユビキタスコンピューティング、自律型ユビキタスコンピューティングについて述べる。なお、近年、ユビキタス（Ubiquitous:遍在）コンピューティングについて、パーベイシブ (Pervasive:拡散) コンピューティングやインビジブル（Invisible:不可視）コンピューティング等、様々な呼び方が存在するが、本論文では、ユビキタスコンピューティングの基本的コンセプトを作り上げた Mark Weiser[21] に基づきユビキタスと呼ぶ。

2.1.1 ユビキタスコンピューティング

環境に計算機が遍在し、あらゆるオブジェクトに情報処理能力やネットワーク接続機能が備わることにより、ユビキタスコンピューティング環境が生まれ始めている。モバイルコンピューティングにおいては、ユーザは計算機を持ち運び、利用する必要があったが、ユビキタスコンピューティングにおいては、計算機はあらゆる場所に存在し、またそれら環境に埋め込まれた計算機と、ユーザの持つ PDA、携帯電話、IC カード等の機器とが、接続され、相互作用し、ユーザのタスクを実行できる。

ユビキタスコンピューティングに関する研究は Xerox PARC の ParcTab[17] から始まり、現在では IBM のパーベイシブコンピューティング [2]、マサチューセッツ工科大学の Oxygen プロジェクト [4]、イリノイ大学の ActiveSpace[6] などがある。それらの多くは、環境に遍在する機器とユーザの持つクライアントデバイスの連携方法、分散する機器やサービスの統合方法などに関するミドルウェア、ネットワークに関する研究開発である。

このユビキタスコンピューティングは、社会にも浸透し、様々なサービスとして形を変えている。例えば、従来の携帯電話には通話機能だけしかなかったが、近年では、電子メールや WEB 閲覧が可能になり、さらに GPS 機能や、環境に設置されている携帯電話用アンテナを利用することにより、位置情報に関するサービスが提供されている。また、駅やカフェに存在するホットスポットサービスも一例として挙げられる。ユーザは自らのノートブックパソコンなどを保持し、ホットスポットサービスが提供されているエリアに到着すると、自動的にそのノートブックパソコンが無線 LAN を通じて、インターネットに接続され、ネットワーク上のサービスが利用可能になる。

2.1.2 ホームユビキタスコンピューティング

家庭内のあらゆる家電機器も、情報処理能力、ネットワーク接続機能を持ち、情報家電機器となり、ユビキタスコンピューティング環境の一部となりつつある。例えば、ステレオコンポが USB により、パーソナルコンピュータ (PC) と接続され、PC 内の音楽ファイルの再生、PC からのステレオコンポの制御などが可能となったり、ハードディスクレコーダーが Ethernet ケーブルにより、直接インターネットに接続され、番組情

報などの動的ダウンロードなどが可能となっている。また、家電機器を機能単位で分割、統合し、仮想的な情報家電機器を構築する VNA[14] なども開発されている。

本論文では家庭内に浸透しつつあるホームユビキタスコンピューティング環境を研究対象環境とし、その主な構成要素に特に着目し、次節以降ではこの環境を情報家電機器環境と呼ぶ。

2.1.3 自律型ユビキタスコンピューティング

これまでのユビキタスコンピューティングの研究開発により、様々な状況で、様々なデバイスが様々なネットワークにより、様々なサービスへのアクセスが可能になってきた。コンテキスト、プロトコル、データフォーマット、デバイスの差異を吸収し、相互接続、シームレスな相互利用などが可能となった。これらの研究の多くは、ユビキタスコンピューティング環境における機器の発見、接続、統合などをフォーカスしており、環境側に知性、自律性を付加したものは多くなかった。ここでは、ユーザのタスクを環境側が認識し、ユーザに代って行う環境、インフラストラクチャとそれに必要な技術について述べる。

ユーザのタスク処理を助けるためには、タスク処理は、ユーザにとって簡単にでき、自らの体を明示的に使用せずに行える必要がある。ワシントン大学の Portolano[5] は、システムを user interfaces、distributed services、networking infrastructure と分類し、ネットワークとそこを流れるデータに知性を持たせ、ユーザのタスク処理を助ける。また、慶應義塾大学の Active Authentication[12] は、図書館やスーパーマーケットでの貸し出し処理、会計処理を、環境に設置されているセンサとコンピュータがユーザとそのオブジェクトを自動認識し、自動認証するシステムである。これら二つの研究に共通するのは、システムはユーザには明示的な入力を求めず、環境側がユーザの要求を察知して、ユーザのタスクを自動的に処理する点にある。

以上のように、ユビキタスコンピューティング環境を最大限に利用し、環境に遍在するハードウェアやソフトウェアが、自律的に動作し、ユーザを補助する必要がある。これにより、ユーザの明示的なサービス利用がなくとも、ユーザの嗜好にあったサービスが環境から提供される。したがって、今後は、ユーザは他のことをしながら、自らが求めるサービスが自律的に行われたり、ある状況におかれたら自動的にサービスが起動される様な、自律性を持った知的なユビキタスコンピューティング環境の出現が予想される。本研究では、このような自律型ユビキタスコンピューティングを情報家電機器環境に適応させ、ユーザの家電機器利用に関する負担を減少させる。

2.2 問題点と解決方法

本章では、現状の情報家電機器環境とそのユーザについて、まず問題点について考察し、そして本研究で採用する解決方法について述べる。

2.2.1 問題点

問題点として、情報家電機器環境における問題点とユーザにおける問題点の二つの視点から分析する。

情報家電機器環境の問題点

情報家電機器環境が持つ問題点として、従来からの家電機器環境が持つ問題点と、家電機器が情報家電機器に進化することにより発生する問題点の 2 種類がある。以下では、従来からの家電機器環境が持つ問題点として、単一機器上での高機能化や複雑化について述べ、さらに情報家電機器環境の実現により発生する問題点として、複数機器のコラボレーション的利用と制御方法の多様化について述べる。

- 単一機器上での高機能化や複雑化
  科学技術の発展にともない、家電機器の持つ機能、システムリソースが増えてきた。通常の家電機器は、単一目的製品であるにも関わらず、様々な付加機能が追加されている。この高機能化は、しばしば機器利用における複雑化につながり、家電機器利用における問題点の一つになっている。

- 複数機器のコラボレーション的利用
  従来の家電機器では、それぞれの機器を単独に使用していたのと比較して、ネットワーク接続機能が付加された情報家電機器では、複数機器のコラボレーション的利用が可能となってきた。これまでもテレビとビデオは同時に利用されていたが、今後はさらに多くの機器がコラボレーション的に利用され、単一機器においても高機能化や複雑化が進んでいる機器が、複数同時に制御をする必要が出てくることにより、さらに制御が困難になると予想される。

- 制御方法の多様化
  従来の家電機器では、その機器を制御するために、ユーザは機器上のボタンや、機器付属のリモートコントローラを用いて、ユーザ対機器が一対一で制御を行っていた。情報家電機器における情報処理能力とネットワーク接続機能の付加は、これまでの制御方法に加え、インターネット経由、コンピュータや携帯電話上のソフトウェア経由での制御も可能になり、制御方法が多様化している。この多様化は、ユーザによっては制御方法の複雑化に繋がる。

ユーザの問題点

情報家電機器環境におけるユーザに関わる問題点を、コンピュータのユーザと比較して、特にその知識や経験と、利用スタイルの 2 点について考察する。

- 知識や経験
  家電機器のユーザは、コンピュータのユーザと比較して、新しい技術や機能に対しての知識や経験が少ない場合が多い。コンピュータは元々、研究機関やオフィ

スでの使用を前提としていたため、その使用方法の習得についても学習を要するものであったとしても、問題はなかった。しかし、家電機器環境では老若男女様々なユーザがおり、必ずしもすべてのユーザが情報機器に対して知識や経験を持たなく、情報家電機器の制御は困難になっている。

- 利用スタイル
  情報家電機器の利用スタイルとコンピュータの利用スタイルについて比較するとオープンな利用方法か、クローズな利用方法かという違いが出てくる。コンピュータの利用は、個人で行うことが多く、そのスクリーンを複数人が覗き込んだり、その入力を複数人がすることは多くない。しかし、家電機器は複数人で同時に利用する。従って、情報家電機器環境では、この家電機器利用におけるオープンな利用スタイルに対応する必要がある。

2.2.2 本研究での解決方法

上記のような情報家電機器環境とユーザにおける問題点を考慮し、本研究では、以下のような解決手法を取る。まず、ユーザが制御する機器が単数であっても、複数であっても統一的なインターフェイスを提供し、さらに複数機器の同時利用の設定についてもボタン操作などで可能にする。つまり、コンピュータ利用時のようなキーボード入力や、多数のウィンドウ操作などをなくすと同時に、従来の家電機器利用時のような複数のリモートコントローラの利用も必要としない。また、ユーザに代ってソフトウェアで実現された制御機構が自律的に家電機器の制御を行う。それにより、ユーザの直接的且つ逐次的な機器利用の手間を省く。

2.3 自律的な情報家電機器制御機構

本章では、本論文で提案する自律的な情報家電機器制御機構について、まず現状の情報家電機器の利用方法を分析し、構築する制御機構の概要について述べ、そして機能要件を挙げる。

2.3.1 家電機器利用時に必要な作業の分類

高機能化、複雑化、多様化した情報家電機器の制御において、ユーザには以下のような作業が必要である。

1. 認識
   家電機器利用時の前に、環境や機器についての認識をする

2. 判断
   認識により得られた情報に基づき、家電機器の利用方法について判断する

3. 利用
   判断に基づき、実際に家電機器を利用する

具体例を挙げる。例えば、部屋内に人がおり、温度が 20 度以下になった場合に、ストーブと加湿器のスイッチを入れるという制御を行いたい。従来の方法だと、ユーザはまず人と温度を認識し、その温度になることを検知しなければいけない。20 度以下になった場合、制御方法を考えながら、ストーブと加湿器のある場所まで移動し、ボタンを押し、スイッチを入れる。また、温度が上がりすぎた場合は、ストーブと加湿器の温度を下げなければならない。

本研究では、上記の認識、判断、利用の 3 種類の作業について、制御機構がユーザに代って行い、自律性を実現する。

2.3.2 自律的な情報家電機器制御機構の概要

本論文では、情報家電機器制御におけるユーザの負担を減少させるために、自律的な情報家電機器制御機構 (AONA:Autonomic Operation of Networked Appliances) を設計、実装、評価する。

AONA の実現により、上述のストーブの例は以下のように解決される。人と温度の認識はセンサにより行われ、ユーザの事前設定により、20 度以下かつそこに人間がいる場合は、AONA が自動的にストーブと加湿器のスイッチを入れる。

図 2.1 で示すように、AONA はユーザに代わり、情報家電機器制御に必要な、状況認識、制御プランの作成、制御の各種作業を行う。状況認識では、ネットワークに接続されたセンサ、家電機器から環境、操作状況等の情報を取得し、プログラムで利用可能なフォーマットに変換する。また、制御プランの作成では、ユーザからの初期設定と状況認識の結果に基づき、情報家電機器制御プランを作成する。そして、その制御プランに従い、実際に機器を制御する。

2.3.3 自律的な情報家電機器制御機構の機能要件

情報家電機器を自律的に制御し、ユーザの制御時の負担を減らすために必要な AONA の 3 つの機能要件について、説明する。

- 自律性
  自律性とは、ユーザに状況認識、判断、制御をさせることなく、制御機構がユーザに代って行う機能である。自律性を実現するためには、状況認識、判断、制御の各機能をシステムが自律的機能として実現すると同時に、各段階において、ユーザからの入力、ユーザへの出力確認を求めない、非同期性も同時に満たす必要がある。

- 環境親和性
  環境親和性とは、提供する情報家電機器制御機構が、情報家電機器環境に適する性質である。第 1 に、提供する機構、システムが情報家電機器制御向けに考察され

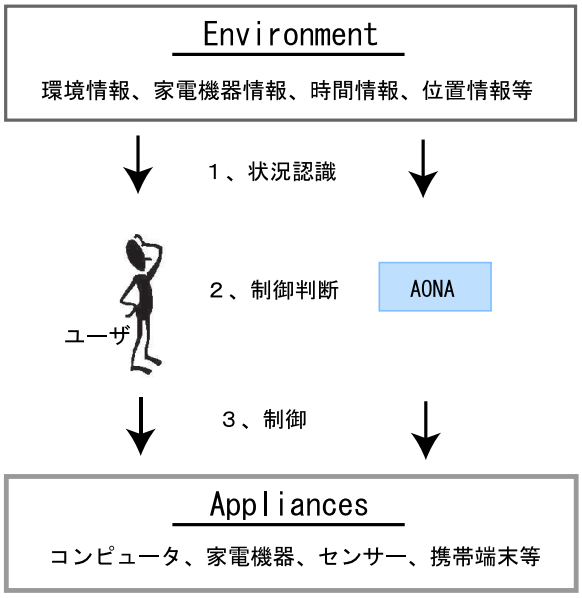


図 2.1: 従来の制御と、AONA による制御

たものである必要があり、第 2 にそのシステムの採用しているアプローチやアーキテクチャがヘテロジニアスな家電機器が存在する環境に向いたものである必要がある。

- 簡便性
  簡便性とは、ユーザにとって情報家電機器制御機構が扱いやすいという性質である。ミドルウェアやフレームワークの提供では、他のプログラマが情報家電機器向けにアプリケーションソフトウェアとして、制御機構を提供する必要が出てくる。したがって、本研究では、家庭内のエンドユーザ向けにシステムを提供すると同時に、そのユーザインターフェイスがシンプルで扱いやすい必要がある。

2.4 関連研究

本研究では、状況認識、制御プラン作成、制御の各機能を実現し、ネットワーク接続された家電機器の自律的な制御を行う。本節では、自律的機能やネットワーク上に分散したリソースの統合などの本研究と関連するテーマを取り扱った研究について考察し、関連性、違いを述べる。

2.4.1 Plangent

Plangent[15] は、東芝研究開発センターで開発された、モバイル機能、プランニング機能、プラン実行機能を保持した、知的モバイルエージェントシステムである。ユーザ

から与えられるゴールについて、プランニング機能の後ろ向き推論により、モバイルエージェント自らが意思決定をする。また、モバイルエージェントであることより、複数ホスト間に跨るタスク処理の場合に、非接続処理が可能となったり、負荷分散が可能となる。しかし、Plangent を動作させるためには、各ホストにエージェントプラットフォームが必要であり、エージェントの動作を決定するアルゴリズム部分では Plangent 独自の PlangentScript を利用する必要があるため、情報家電器機環境においては不向きである。また、モバイルエージェントシステムにおけるセキュリティ問題 [11] についても検討が必要である。Plangent のシステム概要図を図 2.2 に示した。

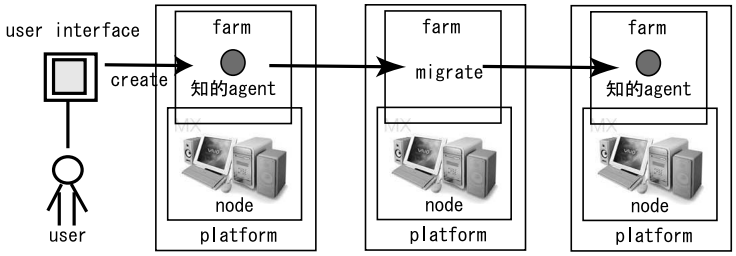


図 2.2: Plangent

2.4.2 HIVE

HIVE[13] は、マサチューセッツ工科大学で開発された、ローカルのシステムリソースを統合するための、分散エージェントプラットフォームである。Media Lab. の Things That Think プロジェクトの一環として開発されている。HIVE では、ネットワークに分散的に接続されたノードを cells、そのノードのシステムリソース、デバイスを shadows、そしてそれらを統合する計算実体を agents と抽象化し、アプリケーション開発者にとって、わかりやすい API の提供を実現している。また、エージェントプラットフォームの提供とともに、それを利用するためのユーザインターフェイスも提供し、マウス操作だけで、ネットワーク上のリソースを接続可能にしている。その一方で研究の目的が、分散エージェントプラットフォームの研究開発であるため、システム全体としての自律性やモバイルエージェントシステムである利点等について考察が弱い。Hive のシステム概要図を図 2.3 に示した。

2.4.3 ADS

Appliance Data Services(ADS)[7] は、スタンフォード大学で開発された家電機器コンピューティング用のアプリケーションフレームワークである。ADS は、AV 機器やデジタル機器とコンピュータのシームレスな融合を目指しており、様々なデバイス、ファイルフォーマットを統一的に扱えるフレームワークとアプリケーションを提供している。アプリケーションフレームワークとして、大きく分けて Data Receive、Application

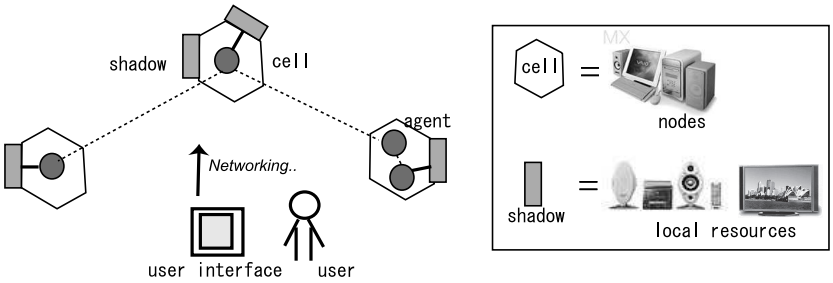


図 2.3: HIVE

Control、Services Execution の 3 段階に別れており、機器接続、データ変換、アプリケーションの自動的選択と起動などの機能を提供するが、機器の自律的制御、ユーザとの非同期性等については考慮されていない。ADS のシステム概要図を図 2.4 に示した。

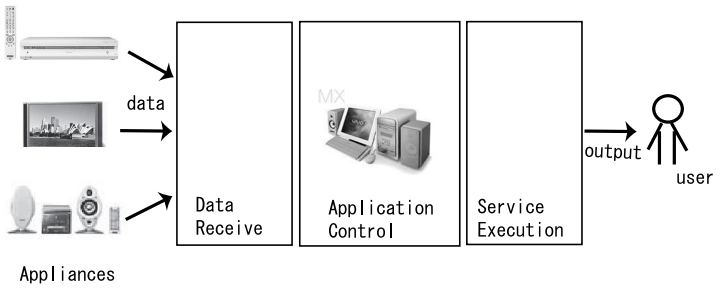


図 2.4: ADS

2.4.4 Autonomic Computing

Autonomic Computing[8] は、IBM で進められている自律型コンピューティング実現に関する取り組みの総称である。Autonomic Computing とは、人間の体が体温や心拍数などの自律的な調節機能を備えているように、コンピューティングシステムに自己管理機能を搭載し、人手がかかる作業を最小限に抑え、情報システムをとりまく環境の複雑化に対応するための包括的なアプローチである。Autonomic Computing を実現するために必要な 4 つの要素として、自己構成、自己修復、自己防御、自己最適化が挙げられており、主に IBM が開発、提供しているサーバーマシンにこれらの技術が適用されている。この取り組みは、企業や組織における情報システムの管理の自律化を目指すものであり、家庭内の情報家電機器環境への適用については、触れられていない。

2.4.5 関連研究の比較

既存のエージェントシステム、自律的コンピュータシステムを用いることにより、自律的な情報家電機器制御機構の一部は実現可能である。しかし、上述の性質要件をすべて満たすような既存研究は存在しない。表 2.1 に関連研究の比較を示す。

表 2.1: 関連研究の比較

項目	Plangent	HIVE	ADS	Autonomic Computing
自律性	○	×	×	○
環境親和性	×	○	○	×
簡便性	×	○	○	×

関連研究の持つ性質を大まかに分類すると、自律性は持つが、エンドユーザ、情報家電機器環境向けではない研究と、自律性は持たないが、エンドユーザ、情報家電機器環境向けの研究の、２つに分けられる。本研究では、この自律性、エンドユーザ（簡便性）、情報家電機器環境（環境親和性）のすべてを考慮した制御機構の実現を目指す。

2.5 本章のまとめ

本章では、ユビキタスコンピューティングについて、一般的な意味、ホームユビキタスコンピューティング、自律型ユビキタスコンピューティングを挙げ、本研究の研究対象環境について説明した。また、現状の情報家電機器環境における問題点とそれに対する解決方法を示し、本研究で提案する自律的な情報家電機器制御機構について概要、機能要件を述べた。最後に、関連研究について比較、考察し、本研究との関連性、違いについて考察した。

第 3 章

自律的な情報家電機器制御機構 (AONA) の設計

本章では、AONA の設計について述べる。まず最初に設計方針として、自律的機能、制御方法の検討、実時間処理、セキュリティについて検討する。また、AONA のシステム構成についてハードウェア、ソフトウェアの両側面から述べ、その後 AONA の４つの主要構成要素である、Sensing Module、Planning Module、Operating Module、User Interface の設計について、詳述する。

3.1　設計方針

本節では、実際に家庭内で利用可能な自律的な情報家電機器制御機構 AONA について、その設計方針について述べる。特に自律的機能、制御方法の比較、実時間処理、セキュリティについて、議論する。

3.1.1　自律的機能

本論文では、情報家電機器制御機構における自律性を、ユーザの入力、思考、出力確認などの各作業を減少させ、制御機構がユーザとは非同期的に制御を行うことと定義する。図 3.1 に、従来からの家電機器制御と AONA での自律的な家電機器制御の比較を、シーケンス図を利用して示す。

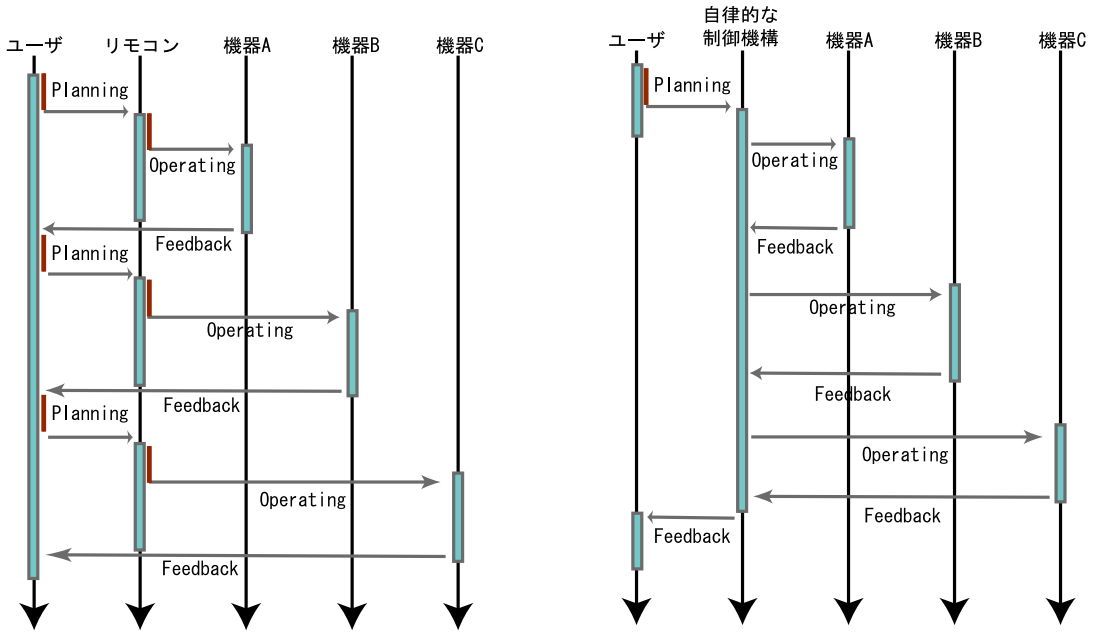


図 3.1: 従来からの家電機器制御との比較

従来の家電機器制御では、ユーザは制御前後と制御中に家電機器と同期的なインタラクションを取る必要があった。しかし、自律的な情報家電機器制御においては、制御機構の持つ自律性や、機器の持つ情報処理能力やネットワーク接続機能により、ユーザは家電機器とは離れ、制御機構がユーザに代わり、状況認識、利用判断、利用の各作業を行う。したがって、制御機構がユーザとは非同期的に機器を制御し、ユーザに対する逐次入力、思考、逐次確認を求めない。

上記のような自律性や非同期生を実現するためには、制御機構は Sensing 機能、Planning 機能、Operating 機能の 3 つの機能を保持する必要がある。

Sensing 機能

Sensing 機能は、ユーザを取り巻く環境、ユーザ、情報家電機器の状態などの環境情報を取得し、その情報をプログラムで利用可能なフォーマットに変換、保持、通知する機能である。ユーザを取り巻く環境はネットワークに接続された各種センサにより検知、認識され、ユーザ情報は各種センサの利用とともにユーザの入力により取得でき、また情報家電機器情報については機器ベンダーの初期設定、ネットワーク経由の監視などにより情報を得られる。また、それらの各種環境情報は、プログラムで利用可能なフォーマット、粒度、単位に変換し、他のプログラムから要求された場合に送信する。Sensing 機能の実現により、ユーザの自らの環境認識、機器認識などの作業が省略され、情報家電機器制御機構の自律的な情報収集、環境認識が可能となる。

Planning 機能

Planning 機能は、Sensing 機能により得られた情報を基に、情報家電機器の制御プランを作成する機能である。プラン作成については、ユーザの基本的な初期設定から、プランを策定し、状況の変化、機器利用の変化に伴い、推論、プランの適用を行う、前向き推論アルゴリズムを採用する。情報家電機器制御では、予想外の機器制御を防ぐ必要があり、また頻繁な設定の変更、プランの変更が可能となる理由から、前向き推論アルゴリズムを選択した。推論アルゴリズムの詳しい説明、選択理由については、次章以降で詳しく述べる。Planning 機能により、ユーザの自らの思考、判断の作業が省力され、情報家電機器制御機構の自律的な制御プラン作成が可能となる。

Operating 機能

Operating 機能は、Planning 機能により作成された制御プランに従い、ネットワーク経由で情報家電機器を制御する機能である。制御方法としては、情報家電機器が Ethernet などにより直接インターネットに接続されている場合は、ネットワーク経由のコマンド送信の制御になり、情報家電機器が Tini[3] や Duonus[22] と言った小型コンピュータや工業用コンピュータにシリアルケーブルなどで接続されている場合は、ネットワーク経由で接続されている小型コンピュータにコマンドを送信し、そのコンピュータ上でコマンドを変換し、情報家電機器に送信する。家電機器がネットワーク接続機能を持たない場合においても、その機器がリモートコントローラにより制御可能な場合は、ネットワークに接続された PDA や Crossam[23] などのユニバーサルリモコン経由での制御が可能となる。Operating 機能により、ユーザの自らの操作、制御が省力され、情報家電機器制御機構のネットワーク経由の自律的な機器制御が可能となる。

3.1.2 集中制御と分散制御の比較

情報家電機器制御機構の設計にあたり、制御機構のアーキテクチャとして、図 3.2 で示すように、集中制御と分散制御の 2 種類があり得る。それぞれについて以下で詳述する。

- 集中制御
  集中制御とは、ホームサーバの様な中央のコンピュータに制御ソフトウェアを配置し、そのサーバからそれぞれの家電機器を制御する方法である。集中制御方法を採用するメリットとしては、制御機構の整備、管理の容易性、既存の情報家電機器環境への親和性等が挙げられる。既存の情報家電機器環境では、PC に情報処理能力が集中しており、各家電機器は情報処理能力を持たないか、持ったとしても少量のシステムリソースである。また、デメリットとしては、中央のサーバへの負荷の集中がある。

- 分散制御
  分散制御とは、集中制御とは反対に、各情報家電機器上に制御機構を配置し、制御する方法である。分散制御方法のメリットとしては、負荷分散の実現とそれによる制御機構のスケーラビリティの保持等がある。しかし、デメリットとしては、整備、管理が複雑になり、また、現状の情報家電機器の持つ機能、システムリソースを考慮にいれると、まだ現実的ではない。

以上を考慮にいれ、本研究では、特に現状の情報家電機器環境への適応性を重視し、一家庭で一つのホームサーバのような PC があることを想定し、現実的な解である集中制御を採用する。

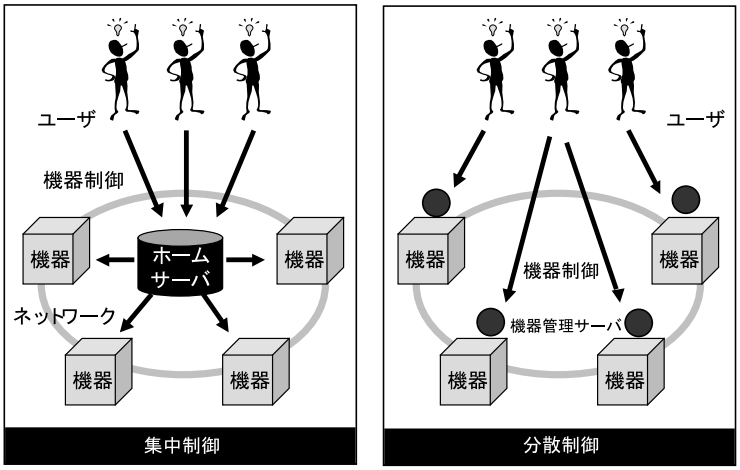


図 3.2: 集中制御と分散制御

3.1.3 実時間処理

設定された時間内に処理を終わらすことを実時間処理 (Real Time Operation) と呼ぶが、情報家電機器制御においても、機器によっては実時間処理が必要になる。既存のコンピューティング環境においては、様々なサービスにおいて、実時間処理が必要と

なっており、例えば、銀行口座関連のトランザクション処理においてであったり、OSにおけるReal Time Operating Systemの存在がある。

しかし、以下の理由から本研究では、実時間処理機能を今回の制御機構には含めない。実時間処理を実現した情報家電機器制御機構を構築するには、被制御対象である情報家電機器と制御機構が配置されるコンピュータなどのハードウェアでの実時間処理機能と、制御機構のソフトウェアとそれが利用する実行環境、オペレーティングシステムにおいても同時に実時間処理を考慮しなければいけない。従って、自律的且つ実時間処理機能を実現した情報家電機器制御機構の実現には、解決すべき研究課題が広範囲に及んでしまうためである。

3.1.4 セキュリティ

情報家電機器制御機構におけるセキュリティを考えるにあたり、ユビキタスコンピューティングにおけるセキュリティ[18]を参考にした。一般的なコンピュータセキュリティを考える際には、機密性、保全性、可用性の3つが指標として使われる。

- 機密性
  機密性とは、情報が部外者には閲覧、知覚されないことである。機密性は、盗聴などで侵され、暗号化などで防がれる。情報家電機器環境においては、家庭内の家電機器構成やその制御が、外部の人間に漏れないようにし、機密性を確保する。

- 保全性
  保全性とは、情報が部外者によって、破壊、改竄されないことである。保全性は、クラッカによる情報に対する不正アクセスにより発生するが、クラッカの不正アクセスを防止すると同時に、不正アクセスされたことを情報利用者に知らせる電子証明等を用いて、被害を食い止める。情報家電機器環境においては、家電機器やそれが保持する情報が破壊、改竄されることを防ぐ必要がある。

- 可用性
  可用性とは、情報が常に利用者にとって利用可能である状態を言う。可用性は、DOS(Denial of Service)アタックにより侵される。情報家電機器環境では、家庭内の機器がクラッカや一般ユーザによって長時間使用され、他人が使用できなくなることを防ぐ必要がある。

本研究での自律的な情報家電機器制御機構は、家庭内の個人的な利用を想定しているが、インターネット経由の制御機構の利用などを考慮すると、機密性、保全性、可用性すべてに対応する必要が出てくる。これらセキュリティについては、一般的な家庭内常時接続を前提としたコンピュータ利用の際に発生する問題、研究課題と同様のため、本論文では研究対象に含めない。

また、現状の家電機器環境では、家庭内の誰もがどの機器でも利用可能になっているが、今後は各ユーザごとの機密性、保全性等を確保するために、制御機構において、ユーザという概念を導入する必要がある。これについては今後の研究課題とする。

3.2 システム構成

本節では、AONA の概要をハードウェア、ソフトウェアの両面から概観する。

3.2.1 ハードウェア構成

AONA のハードウェア構成は、各種センサ機器、コンピュータ、各種情報家電機器の 3 種類の構成要素から成っている。各種センサ機器は、環境情報やユーザ情報を取得するために用いられ、コンピュータはセンサからの情報の計算、制御プランの作成、家電機器制御コマンドの呼び出し等に用いられ、各種情報家電機器はコンピュータから制御される。図 3.3 にハードウェア構成図を示す。

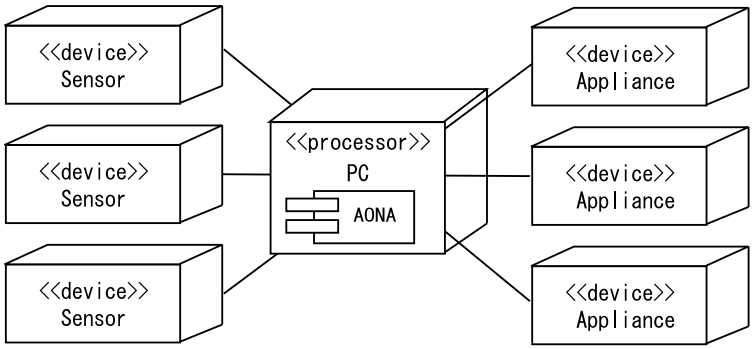


図 3.3: ハードウェア構成図

3.2.2 ソフトウェア構成

AONA は、上述の Sensing 機能、Planning 機能、Operating 機能を実現するために、Sensing Module、Planning Module、Operating Module の 3 つのモジュールとユーザのための User Interface から構成される。Sensing Module が各種センサから情報を取得し、Planning Module が、その情報を基に情報家電機器制御プランを作成し、Operating Module が各種家電機器を制御する。各ソフトウェアモジュールの設計詳細については次項で述べる。図 3.4 にソフトウェア構成図を示す。

3.3 Sensing Moduleの設計

Sensing Module は、各種センサから情報を取得し、適切なフォーマットに変換後、Planning Module に送信する。各種センサは、コンピュータと USB や RS232C により接続され、コンピュータ上にはセンサが接続されているポートに送られてくる情報を常に Read している Sensing Module が用意されている。その後、取得した情報を Planning Module で利用できるフォーマット、文字列に変換し、Planning Module に送信する。

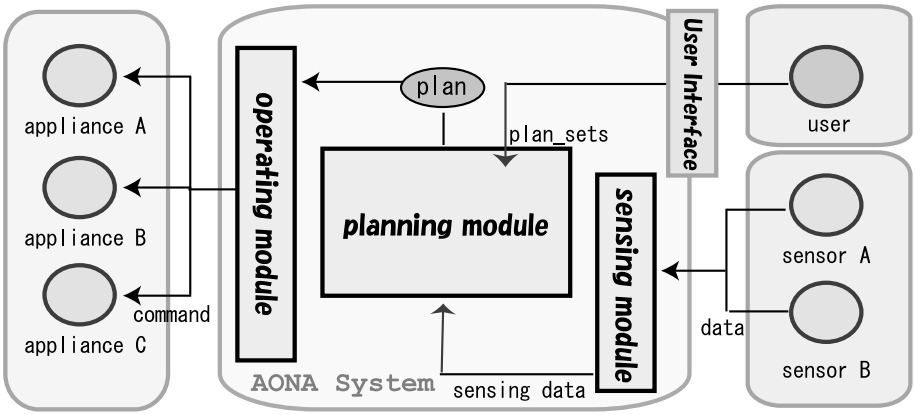


図 3.4: ソフトウェア構成図

3.3.1 State-Full と State-Less の比較

Sensing Module が各種センサの情報を監視するにあたり、そのセンサ情報の保存方法、送信方法には、大きく分けて二つの手法がある。一つは、すべてのセンサ情報を 1 個所に集め、保存する State-Full な方法、もう一つはセンサ情報は保存することなく、常に垂れ流しにしておく State-Less な方法である。 図 3.5 に State-Full な環境監視と State-Less な環境監視の違いを示した図を示す。

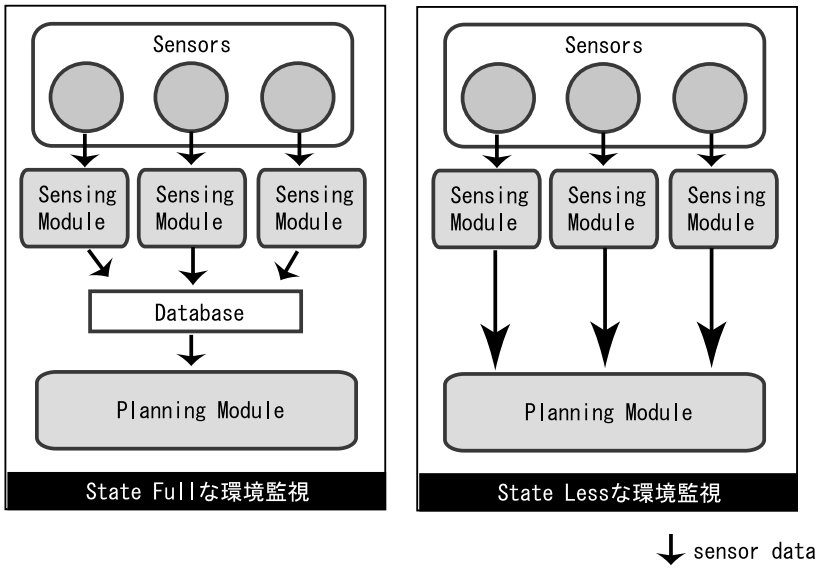


図 3.5: State Full / State Less な Sensing Module

- State-Full な環境監視
  State Full な環境監視では、すべてのセンサはデータベースやソフトウェアモジュールに接続される。そのデータベースやソフトウェアモジュールでは、各センサの最新情報を保持する。Planning Module からセンサ情報の問い合わせがあった場合にだけ、各センサの情報一覧もしくはその部分的情報を送信する。この方法だと、複数のセンサの情報が常に整理され、保存されているため、複数のセンサ情報を統一的に扱いたい場合に有用である。一方で、センサ情報を保持するデータベース用のリソースが必要になる。

- State-Less な環境監視
  State Less な環境監視では、すべてのセンサは直接 Planning Module が配置されているコンピュータにセンサ情報を送信する。この間に、複数のセンサ情報を整理、統合する機構はない。したがって、複数のセンサを統合的に扱いたい場合などは、別に情報を整理する機構を作成する必要がある。しかし、センサ情報を保存し、保持しないため、そのデータベース用のリソースは不要になる。

AONA では、設計、実装の容易さ、システムのスケーラビリティの保持の観点から State-Less な環境監視方法を採用している。

3.4 Planning Moduleの設計

Planning Module は、Sensing Module とユーザから与えられた情報により、情報家電機器制御プランを作成する。ここでは、制御プランを作成する際に必要な、制御プラン作成アルゴリズム、制御プランの記述フォーマット、複数の制御プランが生成された場合の制御プラン競合回避機能、そして制御プラン整合機能について述べる。

3.4.1 制御プラン作成アルゴリズム

ここでは、人工知能研究における主要なプランニングアルゴリズムである、推論と学習について述べ、本研究で採用した推論についてより詳しく分類、整理し、説明する。

推論アルゴリズムと学習アルゴリズム

AONA における Planning Module の設計にあたり、推論アルゴリズムと学習アルゴリズムについて、考察する。

- 推論アルゴリズム
  推論アルゴリズムとは、与えられた知識をもとに、新しい事実、結論を得るアルゴリズムである。情報家電機器制御機構で例を述べると、制御機構には事前知識としてユーザからルールセットが与えられる。情報家電機器環境に変化が起こった場合、制御機構がルールセットを参照し、新しい事実、結論を導きだし、その

事実を基に、家電機器の制御を行う。初期設定はユーザが行えるため、パターン化された家電機器制御の設定、ユーザの予測できない制御プランの作成の防止等が可能になる。

- 学習アルゴリズム
  学習アルゴリズムとは、与えられた情報をもとに、将来使えそうな知識を見つけるアルゴリズムである。情報家電機器制御機構で例を述べると、ユーザの家電機器利用状況を履歴として蓄積し、その利用履歴情報から利用パターンや、ユーザの機器利用に関する嗜好を抽出する。そして、その情報を次回の家電機器利用時に利用する。推論アルゴリズムと比較すると、一般的に学習アルゴリズムの方がより深い自律性を実現できるが、アルゴリズムが複雑になったり、ユーザの予想外の制御が行われてしまう場合がある。

ユーザによる基本設定、パターン化された制御プランの作成、予想外の制御の防止実現のため、本研究では推論アルゴリズムを制御プラン作成アルゴリズムに選んだ。

前向き推論と後ろ向き推論

推論アルゴリズムには、大きく分けて前向き推論と後ろ向き推論の 2 種類の推論プロセスがある。

- 前向き推論
  前向き推論は、与えられたデータから新しい事実を引き出すもので、データ駆動型推論プロセスである。ユーザが事前情報として、ルールを設定し、そのルールと Sensing Module から与えられる情報を基に、次の家電機器制御プランを作成する。前向き推論には、一段階推論と多段階推論がある。これらについては次項で述べる。

- 後ろ向き推論
  後ろ向き推論は、ゴールを与えられ、そのゴールに必要な前段階を引き出す、ゴール駆動型の推論プロセスである。最初に明示的にゴールを与えられるため、推論の焦点は前向き推論と比較すると、定まる。しかし、推論途中にユーザに質問をし、追加データの入力を求める必要が多くなるため、制御機構の自律性や非同期生が損なわれる場合がある。

したがって、本研究で目指す、ユーザと制御機構のインタラクション減少のためには、前向き推論の方が向いている。

一段階推論と多段階推論

本論文では、前向き推論において、事前に与えられるデータから一段階の推論を行うものを一段階推論と呼び、与えられたデータから新たな事実を引き出し、さらにそこから複数回の推論を行うものを、多段階推論と呼ぶ。図 3.6 に一段階推論と多段階推論の違いを示す。

- 一段階推論
  一段階推論は、即応型推論とも呼べ、推論から作成される制御プラン自体は単純なものになるが、推論に必要な時間、システムリソースは少なくなり、制御プラン作成に関する効率は高くなる。

- 多段階推論
  多段階推論は熟考型推論とも呼べ、一般的に作成される制御プランはより最適で複雑なものができるが、作成に必要な時間、リソースは多くなる。

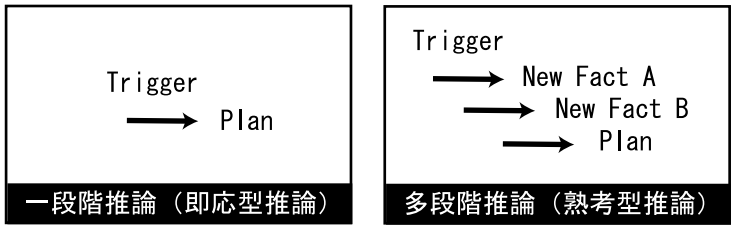


図 3.6: 一段階推論と多段階推論

本論文では、以下の二つの理由から一段階推論（即応型推論）を選択した。一つ目の理由は、推論アルゴリズムをシンプルなものにし、推論に必要なリソースを減少させるため。二つ目の理由は、今回の実装では、事前に与えられるデータを各種センサ機器から与えられるデータとしており、センサからの情報は複数同時に得られるため、多段階推論を採用して、あるセンサの値から別のセンサの値を推論する必要はないためである。

ただし、多段階推論を採用して、導き出される新たな事実を、ユーザの置かれている状況を示した抽象的なメタ情報にすることにより、多段階推論を活用できる場合もある。この場合には、状況抽出機能、メタ情報抽出機能など新たな機構が必要になるので、本研究での研究対象とはしない。

3.4.2 制御プラン記述フォーマット

制御プランには、センサ名、センサ値、機器名、機能の 4 つの変数を用意する。本論文では、センサ名、センサ値の組をトリガ、機器名、機能の組をオペレーションと定義する。例えばセンサ A が、あるセンサ値を取得した場合（トリガ）、機器 B のある機能を実行する（オペレーション）といった具合である。また、一つずつのトリガとオペレーションの組合わせをプランと呼び、それらが複数存在することをルールベースと呼ぶ。図 3.7 に制御プラン記述フォーマットを整理した。

センサ名とセンサ値を結ぶものとして、$sensor = value$、$sensor > value$、$sensor < value$、$sensor \neq value$ 等が挙げられる。それぞれ、ある sensor の値が、value と同じか、それより大きいか、小さいか、違うかを表現する。

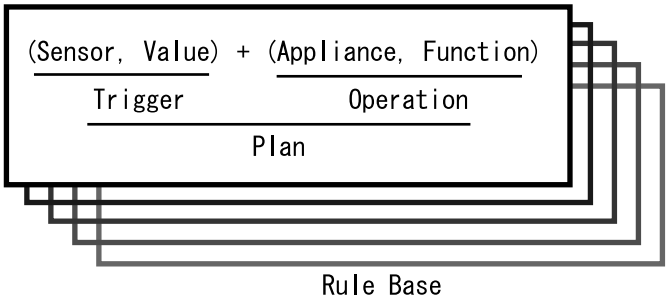


図 3.7: プラン記述フォーマット

制御プラン記述フォーマットについて、図 3.8 にフォーマットを、図 3.9 に、温度センサとプラズマディスプレイを用いた使用例を、BNF(Backus Nauer Form) 記法により例示する。

```
<plan> ::= if <trigger> then <operation>
<trigger> ::= <sensor> + <value>
<operation> ::= <appliance> + <function>
```

図 3.8: BNF 記法による制御プラン記述フォーマット

```
<sensor> ::= ‘‘TempSensor’’
<value> ::= ‘‘0’’ | ‘‘10’’ | ‘‘20’’ | ‘‘30’’
<appliance> ::= ‘‘PlasmaDisplay’’
<function> ::= ‘‘on’’  | ‘‘off’’ | ‘‘input1’’ | ‘‘input2’’
```

図 3.9: BNF 記法による制御プラン記述フォーマット使用例

3.4.3 制御プラン競合回避機能

与えられたプランから制御プランを作成する際に、複数の制御プランが検索される場合がある。その場合、複数の制御プランから実際に実行する制御プランを選択し、他の制御プランを廃棄する必要がある。この複数制御プランから一つの制御プランを選択する機能を、制御プラン競合回避機能と呼ぶ。本研究では、ある一つのトリガには一つのオペレーションしか設定できないようにし、制御プランが競合することを防ぐ。

今後、一つのトリガに複数のオペレーションを設定可能にした場合のプラン競合回避機能にはいくつかの回避アルゴリズムが考えられる。一つ目は最も単純で、検索に一番最初に掛ったものを実行する制御プランとする方法である。実装は単純になるが、アルゴリズムとして知性は低い。二つ目はプランとの検索の際に、最も特定的な制御プランを選択する。プラン検索の際に用いる条件文が多ければ多いほど、その制御プランは特定的であると言え、他のプランより優先して実行する。また、3つ目は各プランにユーザIDを付加して、各ユーザのプライオリティを設定し、実行プランを選択する方法もある。

3.4.4 制御プラン整合機能

推論により導き出した事実が明らかに間違えていた場合、その事実を撤回する必要が出てくる。この事実の修正が可能な推論を非単調推論、事実の修正が不可能な推論を単調推論と呼ぶが、本研究ではその事実をプランごと消去できることより、非単調推論を可能にする。

3.5 Operating Moduleの設計

Operating Moduleは、Planning Moduleにより生成された制御プランに従い、情報家電機器を制御する。これまでの家電機器制御は、ユーザにより機器上のボタンや機器付属のリモートコントローラを用いて行われていたが、AONAでは、Operating Moduleにより、ネットワーク経由で制御を行う。

3.5.1 メッセージ通信制御と移動コード制御の比較

情報家電機器制御を実現するにあたって、実現手法として、大きく分けて2種類の手法について述べる。その2種類は、メッセージ通信制御と移動コード制御であり、それらの違いは情報家電機器制御機構から家電機器へ送る制御コマンドの記述方法にある。

- メッセージ通信制御
  情報家電機器制御機構から家電機器へ送る制御コマンドが、String型やInt型などのメッセージで記述されている手法である。メッセージ通信は、クライアントサーバアーキテクチャで良く用いられる方法であり、インテリジェントな処理や複雑な処理は、クライアントもしくはサーバに配置されたプログラムに記述される。

- 移動コード制御
  情報家電機器制御機構から家電機器へ送る制御コマンドが、オブジェクトやモバイルエージェントなどのプログラムコードで記述されている手法[1]である。移動コードは、Java言語を用いたAppletを利用したシステム、Servletを利用したシステム、またはモバイルエージェントシステムにおいて利用されている。制御

コマンド自体にインテリジェントな処理や複雑な処理を記述することができ、コンピュータ間を移動し実行可能である反面、制御コマンドの容量増加によるネットワークへの付加増大、移動コードの安全性などの問題がある。

なお、AONA では、メッセージ通信制御手法を採用した。

3.5.2 制御結果のフィードバック

機器制御処理結果のユーザへのフィードバック手法について検討する。従来の家電機器環境では、ユーザは家電機器のすぐ近くにおり、自分の目で、家電機器を見ることにより、家電機器の動作を確認できた。しかし、情報家電機器環境においては、必ずしも、ユーザが家電機器の近くにいることは想定できないため、ネットワーク経由のフィードバック手法を用意する必要がある。

- 情報家電機器制御機構上での GUI 表示
  情報家電機器制御機構をユーザの持つ、PC や PDA 上で起動した場合に、そのディスプレイ上に、制御結果を GUI により表示する手法である。実装コストは安価であるが、ユーザの持つ機器が GUI を表示するだけのシステムリソース、プログラム実行環境を保持する必要がある。

- 遠隔地への制御結果メッセージ送信
  ユーザが遠隔地から家庭内の情報家電機器制御機構を操作し、その制御結果がユーザの持つ PDA や携帯電話に、メールやメッセージとして送信される手法である。現在、社会に普及している携帯電話ではメールの送受信が可能なため、現実的な手法ではあるが、実際の家電機器の動作を確認できないため、ユーザが不安になる場合がある。

- 遠隔地への制御結果画像送信
  上述のメッセージ送信における、メールやメッセージといったテキスト情報の代わりに、家電機器の動作を写真もしくはビデオとして撮影し、そのイメージファイルをユーザに送信する手法である。ユーザは、実際の家電機器の動作をマルチメディアファイルとして直感的に確認できるため、送信されてきたフィードバックに確信を持てる。

本研究では、情報家電機器制御機構上での GUI 表示を制御結果のフィードバック手法として採用する。

3.6 User Interface の設計

User Interface は、ユーザに初期設定である、プランの設定を可能にする GUI(Graphical User Interface) である。情報家電機器の多様なユーザに対応するため、シンプルで簡便なデザインを採用する。また、プランの設定、確認、消去の機能も同時に付加する。

3.6.1 GUIの設計

GUIの設計として、図3.10にステートチャート図を示す。GUIが簡便性を保持するために、ボタン、機能などは必要最低限に抑え、パーソナルコンピュータのウィンドウによく用いられるタスクバー上のメニューは設置しない。また、ユーザにキーボードでの入力は要求せず、マウスのクリック操作だけで操作可能にする。

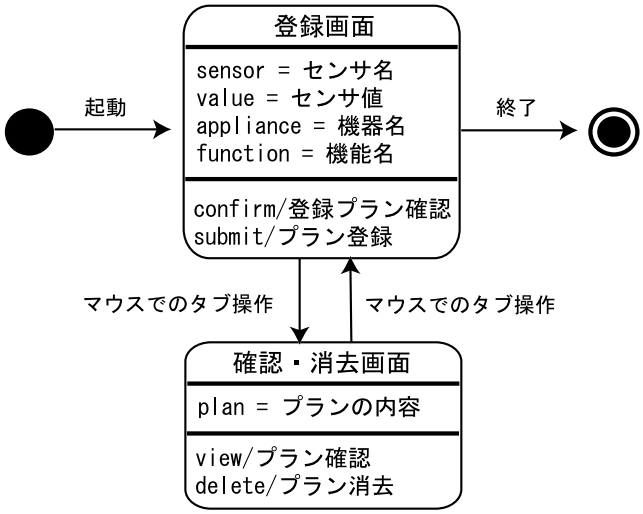


図 3.10: User Interface のステートチャート図

3.6.2 GUI表示ハードウェアの検討

本研究では、GUIをパーソナルコンピュータ上に表示し、マウスにより操作をさせる。将来的には、携帯電話やPDA上でも表示可能にするために、GUI描画言語を携帯電話やPDAで動作可能な言語を採用する。

3.7 本章のまとめ

本章では、まず情報家電機器制御機構AONAの設計方針について、その自律的機能、集中と分散制御の違い、実時間処理、セキュリティについて述べた。そして、システムの構成について、ハードウェア、ソフトウェアの両側面から説明した。また、ソフトウェアの設計についてSensing Module、Planning Module、Operating Module、User Interfaceについて詳述した。

第 4 章

自律的な情報家電機器制御機構 (AONA) の実装

本章では、AONA の実装について述べる。まず AONA の実装方針、実装概要について述べる。その後、AONA を構成する各モジュールについて、実装に用いたハードウェア、実装したソフトウェアについて、説明する。また、最後には、動作手順と AONA を用いたアプリケーション例を示す。

4.1 実装方針

AONA システムの実装には、Java 言語 [9] を利用した。Java 言語は、プラットフォーム独立性を持ち、近年では、携帯電話や各種組み込み機器用の Java 実行環境も整備され、情報家電機器環境に向いている。また、被制御機器、センサは、図 4.1 に示す慶應義塾大学湘南藤沢キャンパス徳田研究室にある Smart Space Laboratory(SSLab)[16] に設置されている各種機器を用いた。

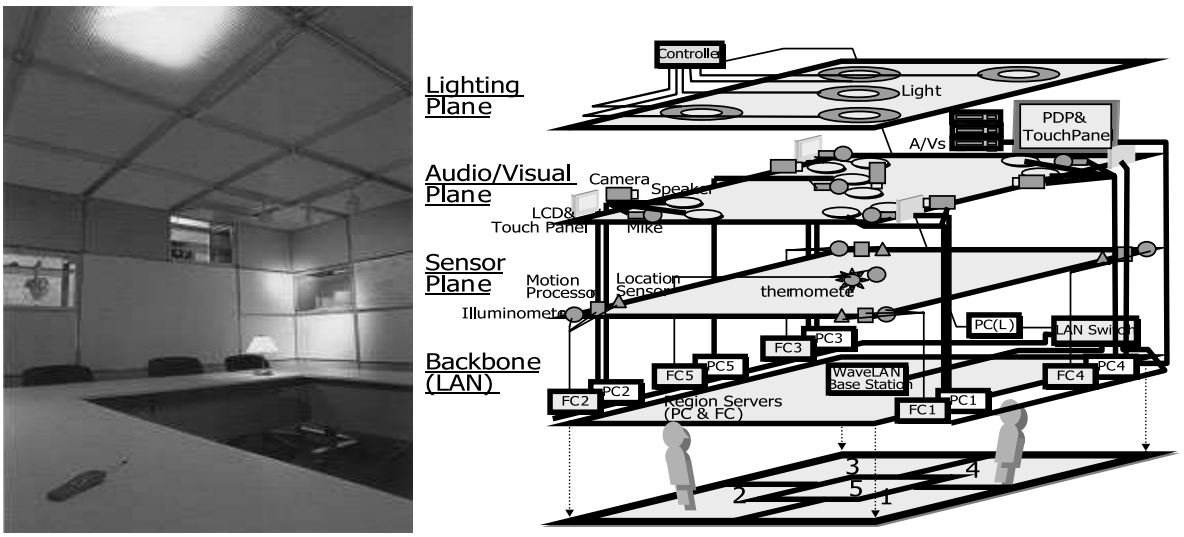


図 4.1: 慶應義塾大学徳田研究室 SSLab

4.2 実装概要

AONA の全体構成を図 4.2 に示す。Sensing Module には、各種センサからの情報リーダの役割を持つ、センサ特定のクラスとそれらを統合するクラスからなる。Planning Module は、Sensing Module から情報を取得し、プランを作成し、トリガが発火されるのを待つクラスを含む。また、Operating Module は、発火されたトリガに対応する、オペレーションを各種情報家電機器に伝える機器特定クラスとそれらを統合するクラスを含む。さらに、User Interface は、プランの登録、確認、消去をユーザに可能にするためのグラフィカルユーザインターフェイスを提供するクラスを持つ。

4.3 Sensing Moduleの実装

Sensing Module について、実装に利用したハードウェアの詳細、実装したソフトウェアの詳細について述べる。

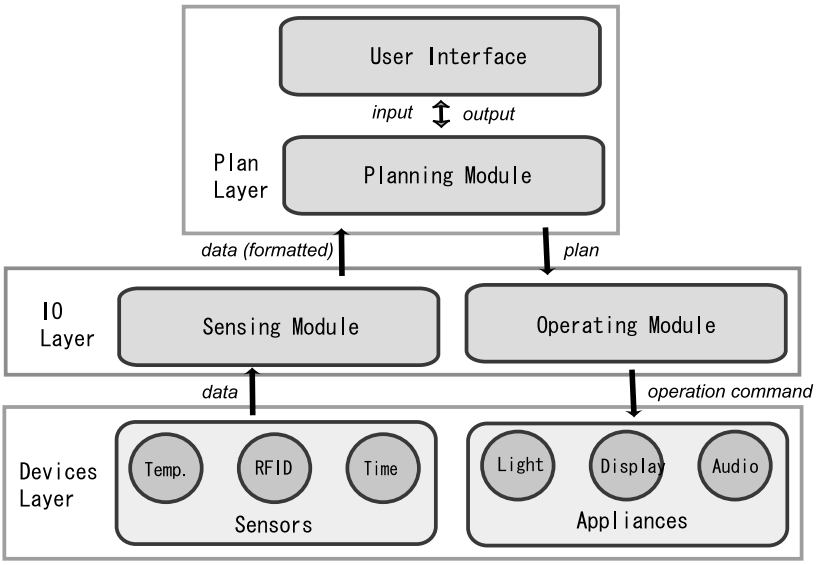


図 4.2: 実装概要

4.3.1 実装に利用したハードウェア

実装では、表 4.1 で示すように、RFID、Davis、Time の 3 種類のセンサを用意した。RFID(Radio Frequency Identification) は、無線通信を使う識別技術であり、RFID タグと RFID リーダから構成される。実装では、毎秒 75 個のタグを認識可能な RFCode 社の Spider を利用した。また、Davis は、Davis 社の気象系センサ群であり、温度、照度、湿度、風力、風向、雨量の各情報を取得する機能を持つ。Time は、パーソナルコンピュータ上の時間を取得する。他とは違いセンサとしてのハードウェアはなく、パーソナルコンピュータを仮想センサとして利用している。

表 4.1: 実装に利用したセンサ

製品	**RFID**	**Davis**	**Time**
製品名	RFCode Spider	Davis センサ群	時間取得センサ
対応クラス	RFCode	Davis	Time
機能	RFID リーダ	気象系センサ	時間取得機能
接続形態	RS232C	RS232C	PC を利用

4.3.2 Sensor クラスとそのサブクラス

Sensing Module は、抽象クラスである Sensor クラスと、3 種類のセンサ用に実装された具体クラスである RFCode クラス、Davis クラス、Time クラスからなる。図 4.3 に Sensing Module のクラス図を示す。

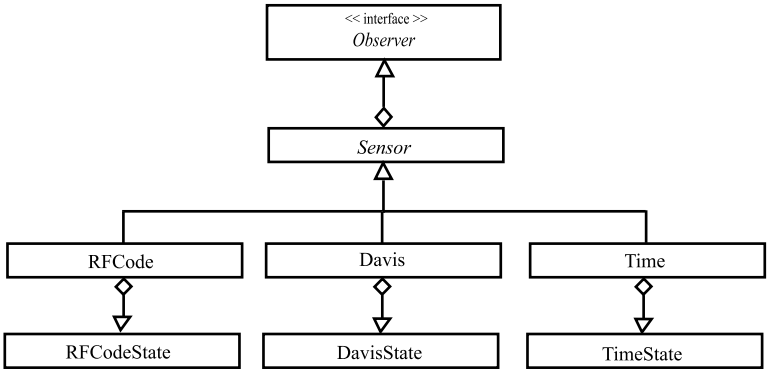


図 4.3: Sensing Module クラス図

Sensor クラスは、表 4.2 で示すメソッドを定義し、RFCode クラス、Davis クラス、Time クラスは、それらメソッドをオーバーライドする。

表 4.2: Sensor クラスのメソッド一覧

public String id();
public State state();
public void addObserver(Observer observer);
public void removeObserver(Observer observer);
protected void notifyObservers();

id メソッドは、センサの個体番号である id を返す。state メソッドは、そのセンサが持つ、現在のセンサ情報を State オブジェクトとして返す。addObserver メソッド、removeObserver メソッド、notifyObserver クラスは、センサ情報が更新された際に、Planning Module に通知するための Observer クラスの登録、消去、通知を行う。

4.3.3 StateUniter クラス

StateUniter クラスは表 4.3 で示すメソッドを定義し、仮想的なセンサを実現し、既存の複数のセンサの情報を統合する。Sensor クラスのサブクラスは、それぞれのセンサ情報を State オブジェクトとして保持するが、StateUniter クラスを用いることにより、複数のセンサの情報を一つの State オブジェクトとして保持が可能になる。

表 4.3: StateUniter クラスのメソッド一覧

public String id();
public State state();
public void unite(State one, State two);

id メソッドは、センサの個体番号である id を返す。state メソッドは、そのセンサが持つ現在のセンサ情報を State オブジェクトとして返す。unite メソッドが引数として取った one と two の二つの State オブジェクトを一つに統合する。

4.4 Planning Module の実装

Planning Module は、主に、Brain クラス、Container クラス、PlanContainer クラス、RulebasedPlan クラスからなる。図 4.4 に Sensing Module のクラス図を示す。以上、４つのクラスについて以下で説明する。

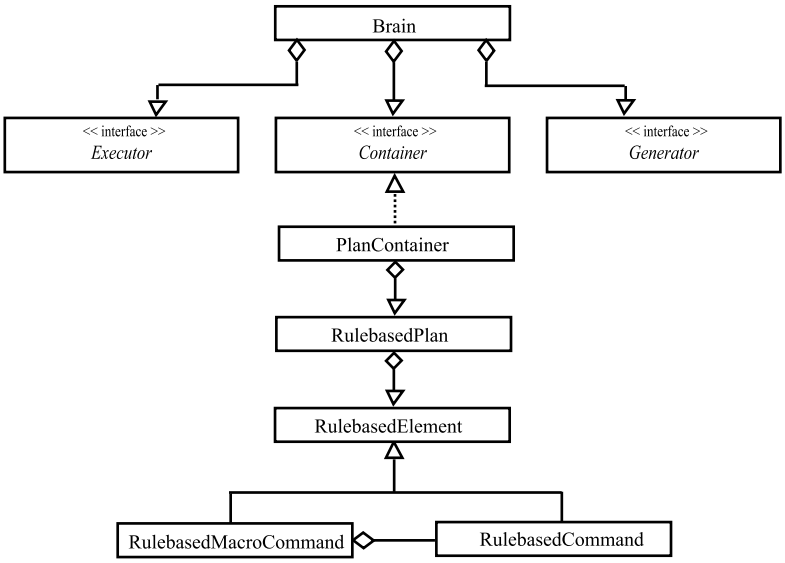


図 4.4: Planning Module クラス図

4.4.1 Brain クラス

Brain クラスは、Container クラス、Executor クラス、Generator クラスのインスタンスを保持し、表 4.4 で示すメソッドを実装する。

表 4.4: Brain クラスのメソッド一覧

public Brain instance() throws InstantiationException;
public Brain instance(Generator, Container) throws InstantiationException;
public void actuated(State state);
public void updated(State state);
public Iterator getPlans();
public void putPlan(RulebasedPlan plan);
public void removePlan(String trigger);

instance メソッドは、Brain クラスのインスタンスを返す。actuated メソッドと updated メソッドは、Sensing Module が取得するセンサ情報に変化があった際に呼び出され、その変化に従って、プランを検索し、機器の制御を Excutor クラスのインスタンスを通じて行う。getPlans メソッド、putPlan メソッド、removePlan メソッドはそれぞれプランの全一覧取得、登録、消去を、Container クラスのインスタンスを通じて行う。

4.4.2 Container クラス

Container クラスは、PlanContainer クラスのインスタンスを保持し、表 4.5 で示すメソッドを定義する。

表 4.5: Container クラスのメソッド一覧

public Iterator iterator();
public boolean has(State state);
public RulebasedPlan get(State state);
public void put(RulebasedPlan plan);
public void remove(String trigger);

Container クラスは、インターフェイスであるため、これらのメソッドを定義し、実装はサブクラスである PlanContainer クラスが行う。

4.4.3 PlanContainer クラス

PlanContainer クラスは、Container クラスを実装したもので、Container クラスで定義されたすべてのメソッドをオーバーライドしている。図 4.5 に PlanContainer クラスのサンプルコードを示す。

```
public class PlanContainer implements Container {

    static private Hashtable planlist = new Hashtable();

    \\制御プランを取得する
    public Plan get(State state){
        String sid = (String)state.get("ID");
        String value = (String)state.get("DATA");
        String trigger = sid + value;
        Plan plan =(Plan)planlist.get(trigger);
        return plan;
    }
    \\該当する制御プランがあるか調べる
    public boolean has(State state) {
        boolean flag = false;
        String sid = (String)state.get("ID");
        String value = (String)state.get("DATA");
        String trigger = sid + value;
        flag = planlist.containsKey(trigger);
        return flag;
         }
    \\制御プランを登録する
    public void put(Object trg, Plan plan){
        State obj = (State)trg;
        String sid = (String)obj.get("ID");
        String value = (String)obj.get("DATA");
        String trigger = sid + value;
        planlist.put(trigger, plan);
     }
    \\制御プランを消去する
    public void remove(String trigger){
        planlist.remove(trigger);
     }
}
```

図 4.5: PlanContainer クラスの一部

iterator メソッドは、現在保持している制御プランをすべて返す。get メソッドは、State 型オブジェクトを引数とし、センサ情報に適した制御プランを返す。put メソッドは、ユーザインターフェイスから呼び出され、新しい制御プランを登録する。remove メソッドは、制御プランを消去する。また、has メソッドは、センサ情報に適した制御プランが存在するか判断をする。

前向き一段階推論の実装

AONA に自律性を与えるための前向き一段階推論は、get メソッド内に実装される。センサが取得したセンサ情報は State オブジェクトとして保持され、そこから Trigger オブジェクトが抽出される。get メソッドでは、Trigger オブジェクトをハッシュテーブルの検索のキーとして利用し、センサ情報にあった Operation オブジェクトを検索、取得する。一段階の検索であるため、即応的推論と呼べる。

多トリガ多オペレーションの実装

推論に用いるトリガを複数にする場合は、Sensing Module で用意した State Uniter クラスを利用して、Trigger オブジェクトに複数のセンサの情報を State オブジェクトとして格納する。また、推論結果として実行するオペレーションの対象機器を複数にする場合は、RulebasedMacroCommand クラスを利用し、複数機器制御の設定を行う。これらの多トリガ多オペレーションの実現により、複数のセンサの情報から、複数の情報家電機器の自律的な制御が可能になる。

4.4.4 RulebasedPlan クラス

RulebasedPlan クラスは、制御プランを定義し、制御プランの各種変数を格納する。図 4.6 に RulebasedPlan クラスのサンプルコードを示す。

id メソッドは、制御プランの id を返す。制御プラン中のトリガ部分は、trigger メソッドが返し、オペレーション部分について element メソッドが返す。また decode メソッドは、制御プランの解析を行い、制御プランを sensor 変数、value 変数、appliance 変数、function 変数に分解する。

4.5 Operating Module の実装

Operating Module について、実装に利用したハードウェアの詳細、実装したソフトウェアの詳細について述べる。

4.5.1 実装に利用したハードウェア

実装では、表 4.6 で示すように、SSLabLight、SmartLamp、PlasmaDisplay、Fan の 4 種類の家電機器を用意した。SSLabLight は、慶應義塾大学湘南藤沢キャンパス徳田

```
public class RulebasedPlan{

    private State state;
    private String aid, fid;
    private static final String id = "rulebased";
    private RulebasedCommand command;
    private RulebasedMacroCommand macro;

    public RulebasedPlan(State state, String aid, String fid) {
        this.state = state;
        this.aid = aid;
        this.fid = fid;
        command = new RulebasedCommand();
        macro = new RulebasedMacroCommand();
        decode();
    }
    \\制御プラン id を返す
    public String id(){
        return id;
    }
    \\制御プランの中のオペレーション部分を返す
    public Element element() {
        return (Element)command;
    }
    \\制御プランの中のトリガの部分を返す
    public Object trigger() {
        return state;
    }
    \\制御プランの解析を行う
    public void decode(){
        command = new RulebasedCommand(aid, fid);
    }

}
```

図 4.6: RulebasedPlan クラスの一部

研究室 SSLab の天井に設置された 4 つの蛍光灯と 4 つのスポットライトからなる。これらのライトは横河電気株式会社の工業用コンピュータ Duonus に接続され、制御される。SmartLamp は、慶應義塾大学徳田研究室と株式会社内田洋行で共同開発された、PC から制御可能で点灯色を変更可能なポール型ライトである。PlasmaDisplay は、パイオニア株式会社のプラズマディスプレイであり、4 系統の入力を持つ。Compaq 社の PDA である iPaq をプログラマブルリモートコントローラにし、このディスプレイをコントロールする。Fan は、扇風機であり、横河電気株式会社の電源管理機器である DA100 に接続され、コンピュータから制御可能にしている。

表 4.6: 実装に利用した家電機器

製品	**SSLabLight**	**SmartLamp**	**PlasmaDisplay**	**Fan**
製品名	SSLab のライト	Smart Lamp	プラズマディスプレイ	扇風機
対応クラス	SSLabLight	SmartLamp	PlasmaDisplay	Fan
機能	ライト	色変化ライト	ディスプレイ	扇風機
接続形態	RS232C	RS232C	IrDA	RS232C

4.5.2 実装したソフトウェア

Operating Module は、抽象クラスである Appliance クラスと、4 種類の家電機器用に実装された具体クラスである SSLabLight クラス、SmartLamp クラス、PlasmaDisplay クラス、Fan クラスからなる。図 4.7 に Operating Module のクラス図を示す。

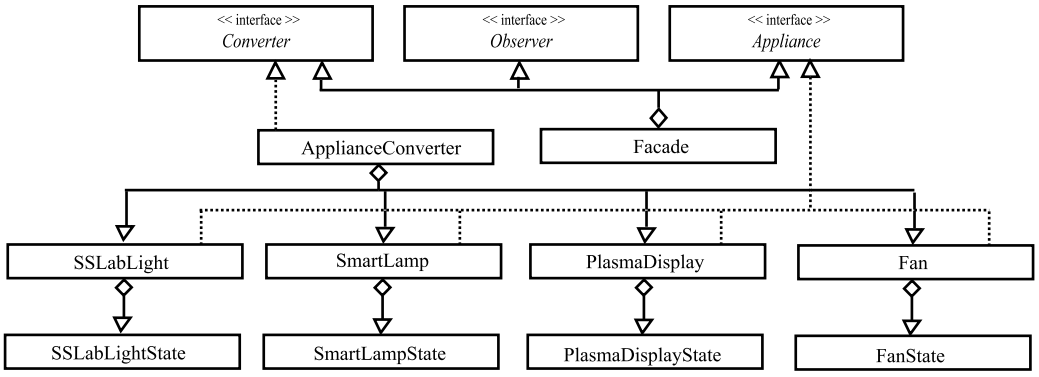


図 4.7: Operating Module クラス図

Appliance クラスは、表 4.7 で示すメソッドを定義し、SSLabLight クラス、SmartLamp クラス、PlasmaDisplay クラス、Fan クラスは、それらメソッドをオーバーライドする。

表 4.7: Appliance クラスのメソッド一覧

public String id();
public State state();
public State actuate(String function) throws IOException;

id メソッドは、家電機器の個体番号である id を返す。state メソッドは、その家電機器が持つ、現在のセンサ情報を State オブジェクトとして返す。actuate メソッドは引数として、String 型の function 変数を持ち、家電機器の機能を実行する。

4.6 User Interfaceの実装

User Interface のクラス図を図 4.8 に、スクリーンダンプを図 4.9 に示す。User Interface では、ユーザの制御プランの設定、確認、消去が行われる。制御プランの設定については、ユーザは、trigger、value、appliance、function の各項目をプルダウンメニューから選択し、登録する。確認については、確認ボタンを押すことにより、現在登録されている制御プラン一覧を表示する。また、制御プランの消去は、確認により表示した制御プラン中から一つプランを選択し、消去ボタンを押すことにより、消去できる。

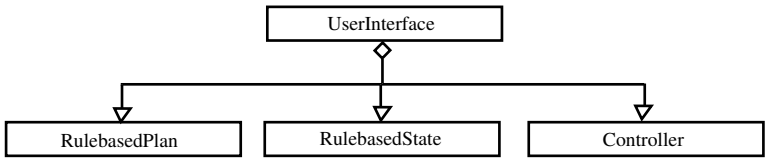


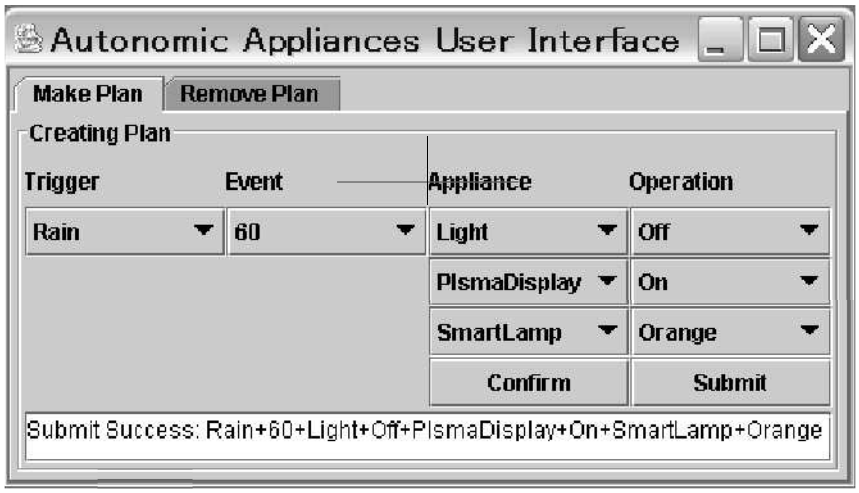
図 4.8: User Interface クラス図

4.7 動作手順

情報家電機器エンドユーザが AONA を利用する際の動作手順を以下に示す。

1. ユーザが User Interface により、プランを入力、登録する
2. Planning Module が入力されたプランを保持する
3. Sensing Module がセンサを監視し、センサ情報を Planning Module に伝える
4. Planning Module がセンサ情報にあったプランを検索する
5. Planning Module が検索されたプランを Operating Module に伝える

プラン登録画面



プラン確認・消去画面

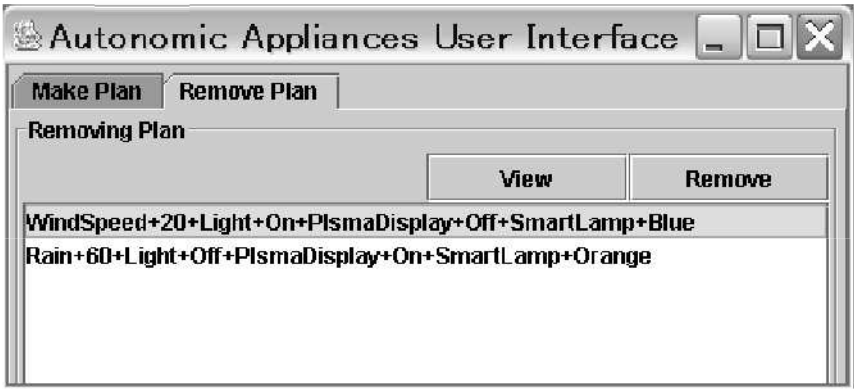


図 4.9: User Interface スクリーンダンプ

6. Operating Module が情報家電機器制御を行う

4.8 アプリケーション例

AONA を利用したアプリケーション例として、習慣的制御モデルと複数機器同時制御モデルを示す。双方とも今回実装に利用したセンサや家電機器だけでなく、その他のセンサや家電機器の利用も含めて、説明をしている。

4.8.1 習慣的制御モデル

ユーザが習慣的に利用する家電機器を AONA で制御させる例を示す。また、図 4.10 に、その際に作成される制御プラン例を疑似コードで示す。

ユーザ A は、毎日帰宅後には、部屋の電気をつけて、テレビのスイッチを入れる。また、部屋が寒い場合には、エアコンのスイッチを入れることにしている。ユーザ A は、AONA を利用し、A が帰宅したら、ライトとテレビのスイッチを入れ、さらに温度が 0 度以下の場合はエアコンのスイッチを入れるように設定した。その次の日から、A は帰宅後、自らライトやテレビやエアコンのスイッチを入れる必要はなくなり、AONA が自動的にユーザの帰宅、温度を検知して制御を行うようになった。

```
if((sensor = RFID)&&(value = userA)){
  ((appliance = Light) && (function = on)) &&
  ((appliance = TV) && (function = on))
  if((sensor = TempSensor)&&(value < 0)){
    (appliance = Aircon) && (function = on)
  }
}
```

図 4.10: 習慣的制御モデルのプラン例

4.8.2 複数機器同時制御モデル

ユーザが AONA に複数の家電機器を制御させる例を示す。また、図 4.11 に、その際に作成される制御プラン例を疑似コードで示す。

ユーザ A は、インターネットから音楽ファイルをダウンロードし、その音楽ファイルを MD に焼いて、ステレオで演奏をしたいと思った。しかし、一つ一つの作業を A がやるのは面倒くさいので、AONA にその作業をするように設定した。AONA は、ユーザに入出力を求めず、自律的にダウンロード、録音、再生の作業を行った。この場合、各作業のトリガとなったのは、一つ前の作業の完了合図であった。つまりダウンロード終了というイベントが、録音開始というオペレーションを呼び出している。

4.9 本章のまとめ

本章では、まず AONA の実装について実装方針と実装概要について述べた。Sensing Module については、実装に利用したハードウェアと実装したソフトウェアについて述べた。Planning Module については、実装したソフトウェアについてサンプルコードとともに説明した。Operating Module については、実装に利用したハードウェアと実装

```
if((sensor = Computer)&&(value = DownloadFinished)){
  ((appliance = AV Device) && (function = Recording))
  if((sensor = AV Device)&&(value = Recorded)){
    (appliance = AV Device) && (function = Play)
  }
}
```

図 4.11: 複数機器同時制御モデルのプラン例

したソフトウェアについて述べた。また User Interface については実装したソフトウェアについてスクリーンダンプとともに説明した。最後に、システムの動作手順とアプリケーション例について、説明した。

第 5 章

自律的な情報家電機器制御機構 (AONA) の評価

本章では、AONA の評価について、定量的評価と定性的評価を行った。定量的評価では、AONA の全体的な処理時間の測定とプラン数の増減にともなう処理時間の測定を行った。また、定性的評価においては、性質要件で述べた項目に基づき、自律性、環境親和性、簡便性について、関連研究との比較をしながら、考察した。

5.1 定量的評価

本節では、AONA の処理時間の測定を行った。測定は、全体的な処理時間の測定とプラン数変化による処理時間の測定の 2 種類を行った。その測定環境、測定方法、測定結果を以下に示す。

5.1.1 測定環境

定量的評価の測定環境として表 5.1 に示す環境を用いた。測定目的を AONA の各モジュールのパフォーマンスやオーバーヘッドの計測としたため、各モジュールはすべて同一計算機上に配置し、ネットワークの利用はしなかった。

表 5.1: 測定環境

項目	環境
CPU	Intel Pentium4 2GHz
Memory	1.00GHz
OS	Windows XP Professional
JDK	JDK 1.3.1

5.1.2 測定方法

測定は、AONA のソースコード内各所にタイプスタンプを記録するコードを埋め込み、各モジュール単位毎に所要時間を計測した。以下の方法により、全体的な処理時間の測定とプラン数変化による処理時間の変化の測定の 2 種類について、測定を行った。

- 測定 1 ：全体的な処理時間の測定
  AONA の 3 つのモジュールである、Sensing Module、Planning Module、Operating Module について、それぞれに必要な処理時間を測定した。3 つのモジュールの処理時間の合計が、AONA 全体として必要な処理時間となる。

- 測定 2 ：プラン数変化による処理時間の測定
  Planning Module において、登録するプラン数を増やすことにより、プランの検索に必要な処理時間の変化を測定した。この測定により、Planning Module のプラン数におけるスケーラビリティがわかる。

5.1.3 測定結果

本節では、前節で説明した方法に基づいて行った測定の結果と考察を述べる。

- 測定１：全体的な処理時間の測定
  全体的な処理時間の測定結果を図 5.1 に示す。それぞれのソフトウェアモジュールで必要な処理時間は 10 ミリ秒以下になっている。Sensing Module と Operating Module については、それぞれ利用するハードウェアにより必要な処理時間が変ってくる。従って、従来のユーザによる直接的な家電機器制御と比較しても、AONA のソフトウェア部分のオーバーヘッドは数十ミリ秒以内に収まるため、現実的なパフォーマンスであると言える。

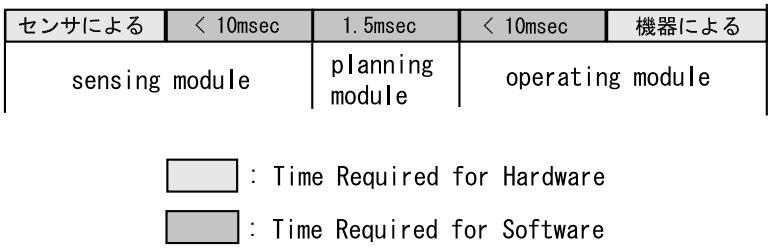


図 5.1: 測定１：全体的な処理時間

- 測定２：プラン数変化による処理時間の測定
  プラン数変化による Planning Module の処理時間の測定結果を図 5.2 に示す。処理時間の単位はミリ秒である。図の通り、プラン数を 100 個まで増やしたところ、Planning Module にて必要な処理時間に大きな変化はなかった。プランの検索については、センサ名とその値によって行われるが、プランはユーザが設定するもので、現在の一般家庭内のセンサ数や、家電機器の利用方法を考慮にいれる限り、100 プランを越えることは少なく、今回の測定結果の処理時間は現実的である。

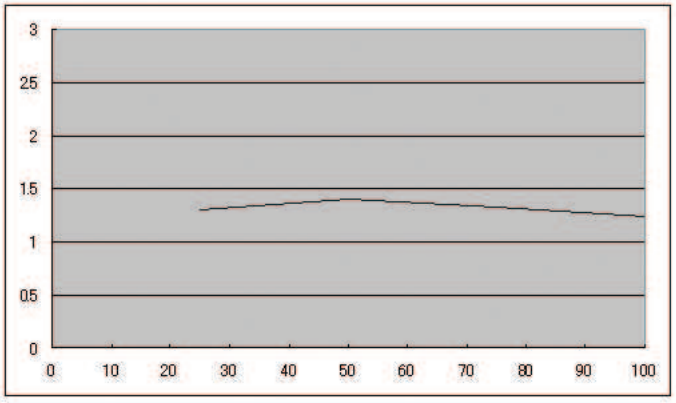


図 5.2: 測定２：プラン数変化による処理時間の測定

5.2　定性的評価

本節では関連研究の頁で挙げた類似システムと、AONA の機能評価と性質評価を行う。評価項目は、機能要件で述べた自律的な情報家電機器制御機構に必要な、自律性、環境親和性、簡便性を利用した。評価結果を表 5.2 に示す。

表 5.2: 関連研究と AONA の比較

項目	Plangent	HIVE	ADS	Autonomic Computing	AONA
自律性	○	×	×	○	○
環境親和性	×	○	○	×	○
簡便性	×	○	○	×	○

5.2.1　自律性の評価

自律性の評価として、ユーザと AONA のインタラクションについて、シーケンス図を図 5.3 に示した。

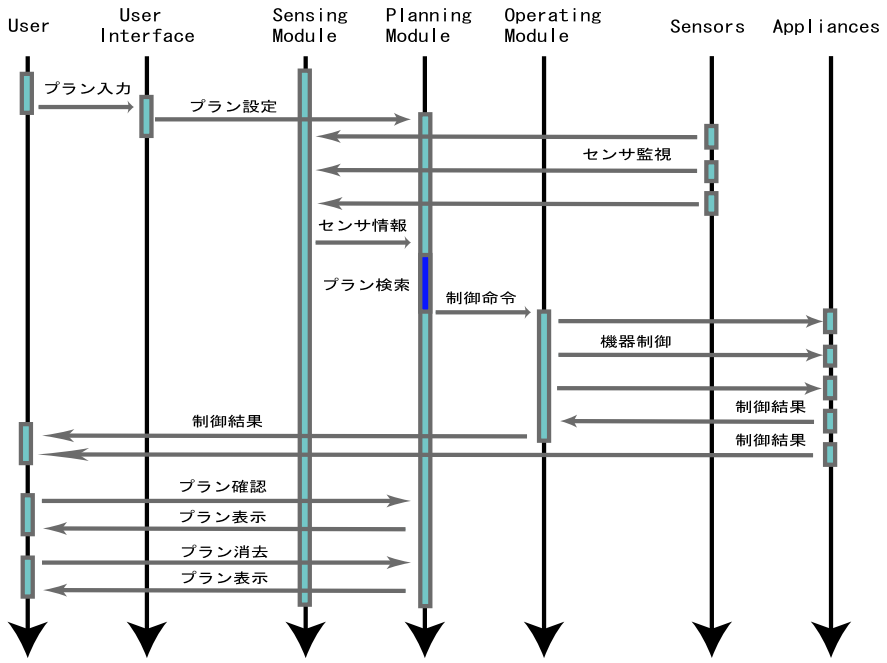


図 5.3: AONA シーケンス図

図のように、ユーザが初期設定としてプランを入力後は、AONA は自律的にセンサの監視、センサからの情報取得、状況にあったプラン検索、情報家電機器の制御を行っ

ている。ユーザが初期設定を終えた後、AONA が自律的に制御を行うため、AONA はユーザとの非同期性も実現している。非同期性の実現により、ユーザは AONA がセンサ監視、プラン検索、機器制御時には、AONA とのインタラクションが不必要になり、別の作業を行える。また、制御結果については、User Interface にエラーを表示するのと同時に、実際の家電機器をユーザが見て、得られる。

5.2.2 環境親和性の評価

環境親和性の評価として、AONA が情報家電機器環境に適した理由を Sensing Module と Operating Module の提供、情報家電機器向け User Interface の提供、Java の利用の 3 点から説明する。

Sensing Module と Operating Module の提供

AONA の実装では、Sensing Module において 3 種類のセンサ用のクラスを、また Operating Module において、4 種類の家電機器用のクラスを用意した。それらはすべて動作可能な機器であり、現在市販されている。従って、現状の情報家電機器環境に適応した様々なクラスを実装できたと言える。

情報家電機器向け User Interface の提供

コンピュータ機器と情報家電機器の操作における大きな違いとして、キーボード入力を求めるか求めないかの違いがある。コンピュータ機器やそれが提供するサービスを利用する場合、サービスはしばしばユーザにキーボードでの文字入力を求める。しかし、情報家電機器制御の場合は、元来ユーザはボタンしか備えないリモートコントローラを用いて制御を行っていたこともあり、文字入力を求めない場合が多い。従って、AONA の実装では、情報家電機器制御に特化したことにより、ボタンだけから構成された情報家電機器向けの User Interface を提供できた。

Java の利用

AONA の実装に Java 言語を利用したことにより、言語自体がプラットフォーム独立性やモジュールの再利用性を持つため、新たな情報家電機器の登場等に対応が可能となった。また、現状の情報家電機器環境において、携帯電話や PDA の様にすでに Java の実行環境を保持した機器が存在するので、現在の環境への親和性も備えている。

5.2.3 簡便性の評価

AONA のユーザインターフェイスでは、ユーザにキーボードでの入力は一切要求せず、すべての入力がマウスで可能になっている。また、ユーザインタフェース上に配置したボタンも必要最低限の数に抑えた。ユーザは数クリックするだけで、制御プランの登録、確認、消去が可能となっている。

5.3　本章のまとめ

本章では、AONA の評価について、定量的評価と定性的評価を行った。定量的評価では、AONA の全体的な処理時間の測定とプラン数の増減にともなる処理時間の測定を行った。また、定性的評価においては、性質要件で述べた項目に基づき、自律性、環境親和性、簡便性について、関連研究との比較をしながら、考察した。

第 6 章

結論

6.1 今後の課題

AONA の今後の課題としては、学習アルゴリズムの導入、ルックアップサービスの導入、実時間処理の検討、セキュリティモデルの検討の 4 点が挙げられる。

6.1.1 学習アルゴリズムの導入

本研究では、制御プラン作成アルゴリズムとして、前向き推論アルゴリズムを採用、実装した。現状の AONA では、制御プラン作成に必要な初期設定はユーザが入力する必要があり、一部のユーザにとっては、その初期設定自体がわずわらしいと感じる場合もある。そのようなユーザには、ユーザ自身の設定をより少なくし、制御機構の自律性をより高める必要がある。より高い自律性のために、ニューラルネットワークを含めた学習アルゴリズムの導入をする必要がある。現在、慶應義塾大学徳田研究室にて研究開発中の学習アルゴリズムを利用したサービス利用システム [10] との統合も検討する。

6.1.2 ルックアップサービスの導入

現状の AONA では、家庭内に存在する情報家電機器は、AONA 提供時に存在する機器と機能をプログラマが AONA のコード内部に記述している。この手法だと、新しい機器や機能が追加された場合や、今まで存在した機器や機能が無くなった場合に対応できない。従って、機器発見、機器登録に関しては、現状はユーザもしくはプログラマが明示的に登録を行わなければならず、その部分で自律性が損なわれている。ネットワーク上に存在する情報家電機器の検索、登録、消去も制御機構が実現するためには、ネットワークサービスの Jini[19] のようなルックアップ機能や、LDAP[20] のようなディレクトリサービスとの連携を考慮する必要がある。

6.1.3　実時間処理の検討

複数機器間のコラボレーション時や、ユーザが時間的制約を抱えている場合に、情報家電機器制御機構が実時間性を満たす必要が出てくる。情報家電機器制御機構の実時間処理の実現は、制御機構であるソフトウェア部分の実時間処理と、非制御対象である情報家電機器のハードウェア部分の実時間処理を同時に考慮する必要がある。また、ソフトウェア部分においても、その制御機構自体と制御機構が動作している実行環境、オペレーティングシステムにおいての実時間処理も考慮に入れる必要がある。

6.1.4　セキュリティモデルの検討

現状の AONA では、ユーザの想定は同一家庭内のメンバーとしているため、セキュリティモデルについては特に検討をしていない。家庭内のメンバーであれば、誰でもどの機器でも利用できるような設定になっている。しかし、例えば家庭内で小さな子供がいて、その子供たちからのある機器への制御制限を行いたい場合は、アクセスコントロールを実現する必要があり、また家庭外からのインターネット経由での制御を行いたい場合などは制御機構への認証、暗号化通信なども必要になってくる。

6.2　本論文のまとめ

本論文では、ホームユビキタスコンピューティング環境に適した自律的な情報家電機器制御機構の構築について述べた。環境にコンピュータや様々機器が遍在し、ユーザが様々な形態でそれらを利用可能になる一方で、機器やサービスの利用方法の複雑化や高機能化も同時に起こっている。同様の現象は家庭内の情報家電機器環境にも起きており、様々なユーザがシステムリソース、ネットワーク接続、機能、操作方法の異なる様々な機器を、単体もしくは複数同時に制御する必要が出てきている。

上記のような問題点を解決するために、本論文では、前向き推論アルゴリズムを利用した自律的な情報家電機器制御機構である AONA(Autonomic Operation of Networked Appliances) を構築した。AONA を利用することにより、ユーザは状況認識、利用判断、機器利用の各ステップを制御機構に任せられる。

AONA の設計においては、自律的な情報家電機器制御に必要な自律的機能、制御方法の検討、実時間処理、セキュリティ等の設計方針について述べ、AONA の主要な 4 つのモジュールである Sensing Module、Planning Module、Operating Module、User Interface について検討した。特に Planning Module については、制御プラン作成アルゴリズム、制御プラン記述フォーマット、制御プラン競合回避機能、制御プラン整合機能について考慮し、設計した。

また実装については、利用したセンサ、情報家電機器について述べたのと同時に、各モジュールのアーキテクチャについて相互関係、提供するメソッド、サンプルコードなどを示した。

評価では、AONA の実用性を示すために、定量的評価を行った。また、関連研究と

の違いを明確にし、優位性を示すために定性的評価を行い、Plangent、HIVE、ADS、Autonomic Compuitng と比較した。

今後の課題として学習アルゴリズム、ルックアップサービス、実時間処理、セキュリティモデルを挙げた。本論文での対象環境は家庭内の情報家電機器に限ったが、AONA の自律型制御機構の基本的概念はより広範囲な分散サービスへの適用も可能である。将来は、今後の課題の解決と同時に、情報家電機器に限らず、街中のホットサービス、WEB サービスなどあらゆるネットワーク経由で利用可能なサービスの自律的利用機構を目指す。

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、本論文の副査として助言を頂いた、慶應義塾大学環境情報学部教授安村通晃博士、同大学大学院政策・メディア研究科助教授高汐一紀博士に深く感謝致します。

さらに、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は感謝いたします。特に慶應義塾大学大学院 政策・メディア研究科博士 3 年の中澤仁氏、同博士 1 年の由良淳一氏、岩井将行氏には絶えざる励ましと御指導を賜わりました。また、情報処理振興事業協会平成 14 年度未踏ソフトウェア創造事業で、一緒にプロジェクトに携わった慶應義塾大学大学院 政策・メディア研究科修士 2 年の桐原幸彦氏、同修士 1 年の中西健一氏、岩谷晶子氏にも感謝致します。

そして，慶應義塾大学徳田研究室で励ましあってきた修士 2 年生の皆さんと，影ながら支えてくれた後輩の皆さんにもお礼いたします。

平成 15 年 1 月 13 日
青木崇行

参考文献

[1] Aoki, S., Nakazawa, J., and Tokuda, H. Autonomous and Asynchronous Operation of Networked Appliances with Mobile Agent. In *Proceedings of 2nd IEEE International Workshop on Smart Appliances and Wearable Computing (IWSAWC 2002)* (July 2002).

[2] Banavar, G., Beck, J., Gluzberg, E., Munson, J., Sussman, J., and Zukowski, D. Challenges: An Application Model for Pervasive Computing. In *Proceedings of 6th ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000)* (Aug. 2000).

[3] Dallas Semiconductor Corp. Introducing TINI: Tiny InterNet Interface, 2000. http://www.ibutton.com/TINI/index.html.

[4] Dertouzos, M. The oxygen project. *Scientific American 281*, 2 (1999), 52–63.

[5] Esler, M., Hightower, J., Anderson, T., and Borriello, G. Next Century Challenges: Data-Centric Networking for Invisible Computing. The Portolano Project at the University of Washington. In *Proceedings of 5th ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 1999)* (Aug. 1999).

[6] Hess, C., Roman, M., and Campbell, R. Building Applications for Ubiquitous Computing Environments. In *Proceedings of 1st International Conference on Pervasive Computing(Pervasive 2002)* (Aug. 2002).

[7] Huang, A., Ling, B., Barton, J., and Fox, A. Making Computers Disapper: Appliance Data Services. In *Proceedings of 7th ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2001)* (July 2001).

[8] IBM Research. Autonomic computing architecture: a blueprint for managing complex computing environments, 2002. http://www-3.ibm.com/autonomic/pdfs/ACwhitepaper1022.pdf.

[9] J.Gosling, B.Joy, and G.Steele. The java language specification, 1999.

[10] Kirihara, Y., Yura, J., Nakazawa, J., and uki Tokuda, H. Prona:a proactive support system for networked appliances. In *IEEE 4th International Workshop on Networked Appliances* (Jan 2002), pp. 145–154.

[11] Makino, S., Okoshi, T., Nakazawa, J., and Tokuda, H. S-agent: The Design of Secure Mobile Agent System. In *Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware 2000)* (Apr. 2000).

[12] Matsumiya, K., Aoki, S., Murase, M., and Tokuda, H. Active Authentication for Pervasive Computing Environments. In *Proceedings of International Symposium on Software Security* (Nov. 2002).

[13] Minar, N., Gray, M., Roup, O., Krikorian, R., and Maes, P. Hive: Distributed Agnets for Networking Things. In *Proceedings of 5th International Symposium on Agent Systems and 3rd International Symposuim on Mobile Agents (ASA/MA '99)* (Oct. 1999).

[14] Nakazawa, J., and Tokuda, H. A Pluggable Service-to-Service Communication Mechanism for Home Multimedia Networks. In *Proceedings of ACM Multimedia 2002* (Dec. 2002).

[15] Ohsuga, A., Nagai, Y., Irie, Y., M.Hattori, and Honiden, S. Plangent: An approach to making mobile agents intelligent. *IEEE Internet Computing 1*, 5 (1997), 50–57.

[16] Okoshi, T., Wakayama, S., Sugita, Y., Aoki, S., 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. In *Proceedings of IEEE Third Workshop on Networked Appliances (IWNA) 2001* (Feb. 2001).

[17] Schilit, B., Adams, N., Gold, R., Tao, M., and Want, R. The PARCTAB Mobile Computing System. In *Proceedings of 4th Workshop on Workstation Operating Systems* (Oct. 1993).

[18] Stajano, F. *Security for Ubiquitous Computing.* John Wiley & Sons, Ltd., 2002.

[19] Sun Microsystems, Inc. Jini Architecture Specification Version1.1, 2000. http://www.sun.com/jini/specs/jini1_1.pdf.

[20] The Open LDAP Project. Open LDAP, 2003. http://www.openldap.org/.

[21] Weiser, M. The computer for the twenty-century. *Scientific American 265*, 3 (1991), 94–104.

[22] 横河電機株式会社. Compact Field Server DUONUS, 2000. http://www.yokogawa.co.jp/DUONUS/Product/Product.htm.

[23] 株式会社スギヤマエレクトロン. 高性能マルチリモコン クロッサム, 2000. http://www.across.or.jp/e_sugi/index.htm.