修 士 論 文

動的なアスペクト指向技術を用いた協調動作実現方式の研究

北陸先端科学技術大学院大学
情報科学研究科情報科学専攻

木間 貴行

2010 年 3 月

修 士 論 文

動的なアスペクト指向技術を用いた協調動作実現方式の研究

指導教員　Defago Xavier 准教授

審査委員主査　Defago Xavier 准教授
審査委員　青木利晃 准教授
審査委員　岸知二 客員教授

北陸先端科学技術大学院大学
情報科学研究科情報科学専攻

0810019 木間 貴行

提出年月: 2010 年 2 月

目次

概要

本研究では，機器の持つ特徴をとらえた「観点」を利用して協調動作を定義し，動的なアスペクト指向技術を用いて実行時に，その定義に基づいた制御を行うことで，具体的な機器構成にとらわれない柔軟な協調動作を行う方式を実現する．

「観点」とは，協調動作の実現において，機器または協調動作に横断する特徴のことである．具体的な例を挙げると，明るさに関わる機器には「Light」，音に関わる機器には「Sound」というように，それぞれの特徴に基づいて観点を定義する．こうした観点を利用した定義は，共通の観点を持つ機器全てに適用することができるため，具体的な動作環境や対象機器のバリエーションに依存せずに協調動作を実現することができる．

本研究では，この観点を「基本観点」と「補助観点」の２つに分類した．１つ目の「基本観点」とは，機器の出力特性や機器本来が提供すべき基本機能に着目した観点であり，協調動作の実現においては，様々な機器に横断する特徴として定義・利用する観点となる．２つ目の「補助観点」とは，協調動作の実現において，協調動作の条件判断の定義要素に着目した観点であり，協調動作の実現においては，様々な協調動作に横断する特徴として定義・利用する観点となる．これら２つの観点に分類し，組み合わせて協調動作を定義することで，表現力や柔軟性が向上する．

本研究では，これらの観点を用いた定義方法を動的なアスペクト指向技術を用いて実現した．具体的には，「観点」を用いた協調動作の実現方式を提案し，動的なアスペクト指向技術が利用可能なフレームワークである Spring を用いてプロトタイプを実装し，例題への適用と評価を行った．

第１章　はじめに

1.1 研究背景

近年，ネットワークを通して機器同士を接続し，協調動作を行うシステムが実現されている．具体的な例を挙げると，家庭内にある家電機器を組み合わせて，各家庭でも映画館のような雰囲気で映画を見ることができるサービスなどが実現されている．

現在，これらの協調動作の多くは，集中管理を行う機器，または各機器が，予め定義された協調動作の内容に基づいて動作することで実現されている．

こうした協調動作は，処理の手順や指示方法，対象機器や動作環境などの特性を考慮しなければならない．

1.2 本研究の位置付け

協調動作を行うシステムやサービスは，今後ますます増加すると考えられる．従来のような想定された対象機器や動作環境だけではなく，より人々の生活の身近なところで様々な機器が協調してサービスを提供するようになるため，そうした協調動作の多様性に柔軟に対応できることが求められる．

本研究では，機器の持つ特徴をとらえた「観点」を利用して協調動作を定義し，動的なアスペクト指向技術を用いて実行時に，その定義に基づいた制御を行うことで，具体的な機器構成にとらわれない柔軟な協調動作を行う方式を提案する．

1.3 論文の構成

本論文の構成は以下の通りである．

1 章では，本研究の背景と本論文の章構成について述べる．

2 章では，本研究で解決したい問題点と，問題解決のためのアプローチについて述べる．

3 章では，本研究と関連する既存技術の紹介として，アスペクト指向と動的なアスペクト指向の説明に加えて，本研究で用いる Spring のフレームワークの機能の一部について紹介する．

4 章では，本研究で提案する動的なアスペクト指向技術による柔軟な協調動作の実現方式と観点の概念についての説明を行う．

5 章では，本研究で実装したプロトタイプシステムについて説明する．

6 章では，例題への適用を説明し，その適用結果と評価，考察について述べる．
7 章では，本研究についてのまとめと今後の課題について述べる．

第 2 章　目的

本章では，本研究で解決したい問題点と目的，アプローチについて述べる．

2.1 本研究で解決したい問題点

既存方式での協調動作の実現方法では，動作環境や対象機器に応じて協調動作が定義されている．目的とする協調動作の実現のためには，設計段階において協調動作の設計者が，動作環境や対象機器を想定して，協調動作の内容や条件判断を定義する必要がある．しかし，動作環境や対象機器のバリエーション，様々な制御方法，動作環境の違いなどを考慮して，予め全て定義しておくことは困難である．対象機器や動作環境，協調動作の種類など考慮すべき条件が増加するにつれて，様々な条件の組み合わせについて検討しなければならない．それに加えて，実際の動作環境においては，メーカーやグレードの違いにより，各機器の持つ機能や制御方法も異なる．動作環境においても同様であり，各家庭の機器構成や室内環境も異なる．これらの条件下で，目的とした協調動作を実現するのは非常に困難である．

このような問題が起こる原因の一つとして，協調動作の定義方法が具体的な対象機器や動作環境を想定して定義する必要があることが考えられる．機器や動作環境に応じて協調動作を定義する方法では，対象機器や協調動作，動作環境などの条件が増加するにつれて，必要な協調動作の定義も大幅に増加してしまう．

2.2 研究目的とアプローチ

本研究では，個々の機器ではなく，機器の持つ特徴をとらえた「観点」を利用して協調動作を定義し，動的なアスペクト指向技術を用いて実行時に，その定義に基づいた制御を行うことで，具体的な機器構成にとらわれない柔軟な協調動作を行う方式を提案する．「観点」とは，協調動作の実現において，機器または協調動作に横断する特徴のことである．具体的な例を挙げると，明るさに関わる機器には「Light」，音に関わる機器には「Sound」というように，それぞれの特徴に基づいて観点を定義する．こうした観点を利用した定義は，共通の観点を持つ機器全てに適用することができるため，具体的な動作環境や対象機器のバリエーションに依存せずに協調動作を実現することができる．

第 3 章　関連技術

本章では，提案手法の紹介をする前に，既存技術であるアスペクト指向技術と動的なアスペクト指向技術，本件研究で用いる動的なアスペクト指向技術が利用可能なフレームワークである Spring の一部機能について紹介する．

3.1　アスペクト指向技術

アスペクト指向の考え方の議論には，関心事の分離という概念がよく出てくる．ここでいう関心事とは，扱おうとしているシステム内に散在しているシステムの要素のことである．多くの場合，これらの関心事というのは，システム内の様々なモジュールにまたがり存在する．そのため，これらの関心事は横断的な関心事と呼ばれる．

アスペクト指向技術を用いて，横断的な関心事をシステムの中心的な本体から分離させる理由として，システム自体の保守性，再利用性の向上等の利点があると考えられている．

従来から存在するオブジェクト指向では「関連するデータ」と「処理手続き」の両方を「クラス」という形式の関心事として分離している．オブジェクト指向によって「関心事の分離」はより進んだ形でソフトウェアに適用されることとなり，保守性や再利用性が高いレベルで実現された．しかし，オブジェクト指向の考え方において，まだ「関心事の分離」が不完全であり，その不完全さを補う形でアスペクト指向技術が研究されている．

アスペクト指向技術の考え方が有効な例として，「データの永続性」「ロギング」「セキュリティ」等が挙げられる．ここでは，ロギングの例を用いてアスペクト指向技術の考え方について説明する．ロギングというのは，システムの動作時にログメッセージを画面やファイルに出力をする処理機能のことである．どのようなシステムにおいても，ログを取るという機能は重要な処理である．しかし，ログを取るためには，ログを取りたいタイミングでログの機能を呼び出す必要があり，様々なモジュール中に呼出しのための処理コードを記述する必要がある．そのため，ログを取るという機能がシステム内の複数のもモジュールに横断してしまい，システムの保守性，再利用性は低下する．そこで，図に示すように各モジュールに分散するログ呼出しコードの位置と処理内容を，アスペクトとして本体のコードから分離する．そして，分離したアスペクトにより，必要に応じて各指定場所にアドバイスが適用されることで，ロギングの処理を実現する．

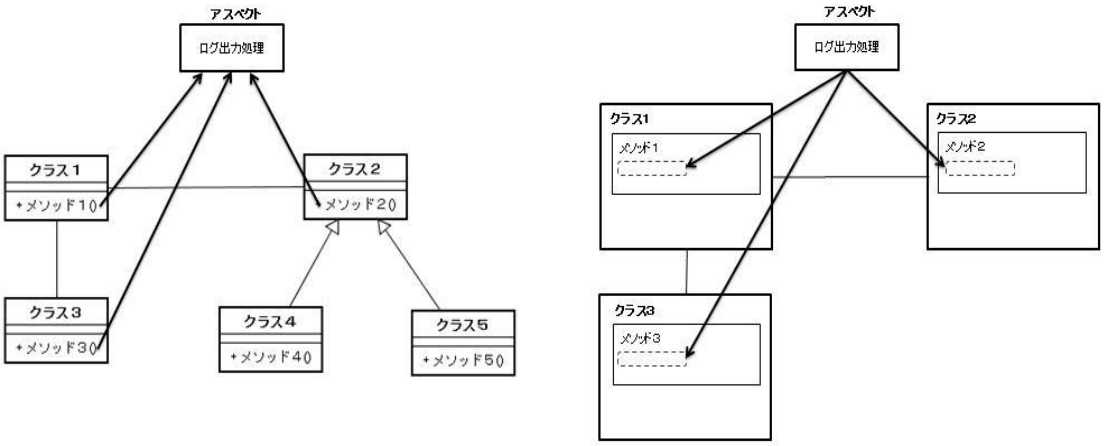

図 3.1：ロギングの例

このように，アスペクト指向技術とは，従来のオブジェクト指向では分離しきれなかった関心事を，いかに分離するかということに焦点を当てた考え方である．

適切なアドバイスの適用が行われるためには，アドバイスがどのようなタイミングで適用されるのかという情報と，適用される側のどこにアドバイスが挿入されるのかという情報を定義しておく事が必要である．そのための仕組みをジョインポイントモデルという．ジョインポイントモデルで用いられる用語について説明する．

- ジョインポイント

ジョインポイントは，アドバイスが持つ振る舞いを割り込ませることが可能なコード上のポイントである．メソッドやコンストラクタの呼出し位置等がジョインポイントとなる．

- ポイントカット

メソッドやコンストラクタの呼出し位置等のジョインポイントのうち，アドバイスを適用したいジョインポイントをまとめたものをポイントカットと呼ぶ．

- アドバイス

ジョインポイントにて実行される処理内容の事である．実行のタイミングとしては，ジョインポイントの事前や事後など，任意のタイミングで実行するタイミングを定義できる．

- アスペクト

関心事が持つ振る舞いと，その横断的な関心事を適用するコード上のポイントをまとめたものであり，ポイントカットとアドバイスを組み合わせたものである．

3.2 動的なアスペクト指向技術

続いて動的なアスペクト指向技術についての説明を行う. 基本的なアスペクト指向の考え方は, 3.1 節と同様である．しかし，3.1 節で紹介した通常のアスペクト指向技術と異なる動的なアスペクト指向技術の特徴の 1 つとして, プログラムの実行を止めずに任意の振る舞いに変更することができる点がある．3.1 節で紹介した通常のアスペクト指向技術が，コンパイル時に分離された処理内容であるアドバイスを指定したポイントカットに適用することができるのに対して, 動的なアスペクト指向技術では，プログラムの実行時にこれらの処理を行うことができる．

これらの動的なアスペクト指向技術が利用可能なフレームワークは，その特徴から大きく 2 種類に大別することができる．どちらの方式のフレームワークにおいても，アドバイスを定義するファイルは，通常の Java のプログラムと同様に記述でき，またアスペクトに関する情報については XML ファイル形式で定義することができる．

- **バイトコードレベルでアドバイスを適用するフレームワーク**

クラスのロード時や実行時にバイトコード変換を用いてアドバイスを対象箇所に適用することができるフレームワークである．代表的なフレームワークとしては，JBossAOP，JAC，AspectWerkz などがある．

- **Proxy を利用してアドバイスを適用するフレームワーク**

Proxy を利用したインタセプタによって，メソッド呼び出しなどが行われた時に，アスペクトの持つ振る舞いを挿入する方法によって, プログラム実行時にアドバイスを適用することができる．代表的なフレームワークとしては，dynaop，Nanning，Spring などがある．本研究では，Spring を用いて動的なアスペクト指向技術を利用する．

3.3 Spring

本研究で用いる Spring では，Proxy を利用したインタセプタによって，メソッド呼出しなどのタイミングでアドバイスを適用させることで, プログラム実行時にアドバイスを適用することができる．Spring において，ジョインポイントモデルの機能を実現するための様々な記法が用意されている．ここでは，AspectJ というアスペクト指向言語で使用されている記法の一部が利用できる AspectJ アノテーションという記法を利用した Spring のアスペクトの設定方法の中で, 代表的な機能に絞り説明する．

3.3.1　ポイントカットの設定

アドバイスを適用したいジョインポイントを設定するための表記方法である．

- execution

呼出し先の「メソッド」「コンストラクタ」を指定する．

- within

呼出し元の「クラス」を指定する．指定されたクラスで宣言されたメソッド呼び出しのみが対象となる．

- this

呼び出し元の「クラス」を指定する．親クラスで宣言されたメソッドも対象となる．

- target

呼出し先の「クラス」を指定する．

- args

呼出し先「メソッド」の引数の型を指定する．

3.3.2　アドバイスの設定

指定したポイントカットの箇所において, 適用したい処理内容を設定するための表記方法である.

- @Aspect

アスペクトとなるクラスを指定するアノテーションである.

設定例：

```
@Aspect
public class ExAspect{
//各アドバイスの記述
}
```

- @Around

ジョインポイントの前後で実行される Around アドバイスを指定するアノテーションである.

設定例：

```
@Around(“execution(* exMethod())”)
public void aroundAdvice(ProcedingJoinPoint pjp) throws Throwable{
//メソッド呼出し前 Around アドバイス処理内容
pjp.proceed();
//メソッド呼出し後 Around アドバイス処理内容
}
```

- @Before

ジョインポイントの前に実行される Before アドバイスを指定するアノテーションである.

設定例：

```
@Before(“execution(* exMethod())”)
public void beforeAdvice(){
//Before アドバイス処理内容
}
```

- @After

ジョインポイントの後に実行される After アドバイスを指定するアノテーションである.

設定例：

```
@After(“execution(* exMethod())”)
public void afterAdvice(){
//After アドバイス処理内容
}
```

3.3.3 アドバイザーの設定

Spring では，ポイントカットとアドバイスをカプセル化したユニットをアドバイザーと呼ぶ．アドバイスとポイントカットを組み合わせたものであり，3.1 節で説明を行った“アスペクト”とほぼ同義の表現である．本論文では，表現の統一のため，以降の章においてアスペクトという表現で統一する．

3.3.4 動的なアドバイス適用

本研究では，動的なアスペクト指向技術のフレームワークである Spring を用いる．ここでは，提案手法の適用方法を説明する前に，Spring の機能について具体例を用いてアスペクトの設定方法とアドバイスの適用方法についての説明を行う．

今回使用する Spring などの動的なアスペクト指向技術のフレームワークの多くは，アスペクトを定義したファイルとそのアスペクトの設定ファイルの 2 種類で定義される．アスペクトの定義ファイルには，アスペクトの処理内容であるアドバイスを通常の Java 言語のプログラムとして定義することができる．また，アスペクトの設定ファイルについては，XML ファイルとして定義される．

3.3.5 アスペクトの定義ファイル

アスペクトの定義ファイルは通常の Java の class ファイルと同様にして作成することができる．3.3 節のはじめでも説明を行ったが，Spring には，様々な表記方法でアスペクトやその設定ファイルである XML ファイルを記述することができる．今回は，その中で AspectJ アノテーションを用いた表記方法を用いて説明を行う．

まず，アスペクトの宣言方法について説明を行う．AspectJ アノテーションを用いた表記方法では，下記のようになる．通常の Java での class の宣言の上に「@Aspect」というアノテーションを付けておことで，これがアスペクトの記述であるということを宣言する．

設定例：

```
@Aspect
public class ExAspect{
//各アドバイスの記述
}
```

次に，アドバイスとジョインポイントについて説明する．AspectJ アノテーションを用いた表記方法では下記のようになる．

設定例：

```
@After(“execution(* exMethod())”)
public void afterAdvice(){
//After アドバイス処理内容
}
```

この例では，ジョインポイントは「@After(“execution(* exMethod())”)」という箇所に定義されている．この例では，execution というメソッドの呼出しをジョインポイントとして宣言しており，execution(* exMethod()) という箇所で* exMethod という名前のメソッド呼出し後の箇所にアドバイスを適用するという定義内容になっている．メソッド呼出しの事前やその前後に設定したい場合には，「@After」の代わりに「@Before」や「@Around」などを用いることで，アドバイスの適用のタイミングを変更することができる．アドバイスとして記述する処理内容については，通常の Java プログラムと同じように記述する．

これらの宣言と定義をまとめると下記のようになる.

設定例：
ファイル名：AopExAspect.java

```
@Aspect
public class AopExAspect{
・・・
@After(“execution(* exMethod())”)
public void afterAdvice(){
//After アドバイス処理内容
}
・・・
}
```

3.3.6 アスペクトの設定ファイル

アスペクトの設定ファイルに記述される内容は様々であるが, その中で本研究と関連する内容について説明を行う．アスペクトの設定ファイルは，XML ファイル形式で記述する．ファイルの記述内容例の一部を下記に示す.

記述例：
ApplicationContext.xml

```
・・・
<aop:aspectj-autoproxy/>　－　①
<bean id=”aopExAspect”　class=”homenet.AopExAspect”/>　－　②
<bean id=”aopExBean”　class=”homenet.AopExBeanImpl”/>　－　③
・・・
```

①の箇所において，AspectJ アノテーションを利用した表記方法を利用することを宣言する．②の箇所において，アスペクトを定義したクラスである「AopExAspect」のクラスと，「aopExAspect」の bean id が関連付けられる．③においても同様であり，アスペクトを適用するクラスである「AopExBeanImpl」のクラスと「aopExBean」の bean id と関連付けて管理する．そして，これらの bean id を用いて Spring の機能を利用する.

3.3.7 アスペクトの適用例

3.3.5 節と 3.3.6 節で説明を行ったアスペクトの定義ファイルとアスペクトの設定ファイルを用いて Spring の機能を利用する．下記に例を示す.

```
public class Example{
        public static void main(String args[]){
ApplicationContext context =
new ClassPathXmlApplicationContext("/homenet/ApplicationContext.xml");
AopExBean ExBean = (AopExBean)context.getBean("aopExBean");
・・・
}
}
```

3.3.6 節で紹介した ApplicationContext.xml の中の③で定義した aopExBean の bean id を用いて Spring が管理する DI コンテナからオブジェクトを取得する．その際，②で定義された aopExAspect という bean id で定義されている AopExAspect のクラス（3.3.5 節の設定例を参照）の中には，AspectJ アノテーションの表記方法の「@aspect」というアノテーションが記述されており，このクラスはアスペクトとして扱われる．ジョインポイントで定義した条件に該当する箇所にさしかかると，定義しておいたアドバイスが指定したタイミングで適用される．

第４章　提案方式

本研究では，機器や協調動作を個別に扱うのではなく，機器の持つ特徴をとらえた「観点」を利用して協調動作を定義し，動的なアスペクト指向技術を用いて実行時にその定義に基づいた制御を行うことで，具体的な機器構成にとらわれない柔軟な協調動作を行う方式を実現する．

はじめに，本研究で協調動作の一例として取り上げるホームネットワークシステムの概要説明と分析を行う．

次に，本研究で扱う「観点」の定義と協調動作の実現方式について述べる．

4.1　観点

本研究で用いる「観点」とは，協調動作の実現において，機器間又は協調動作間にまたがる特徴のことを指す．具体的な例を挙げると，明るさに関わる機器には「Light」，音に関わる機器には「Sound」というように，それぞれの特徴に基づいて観点を定義する．そして，観点を利用した定義は，共通の観点を持つ機器全てに適用することができるため，具体的な動作環境や対象機器のバリエーションに依存せずに協調動作を定義することができる．

本研究では，観点を「基本観点」と「補助観点」の２つに分類した．１つ目の「基本観点」とは，機器の出力特性や機器本来が提供すべき基本機能に着目した観点であり，協調動作の実現においては，様々な機器に横断する特徴として定義・利用する観点となる．２つ目の「補助観点」とは，協調動作の実現において，協調動作の条件判断の定義要素に着目した観点であり，協調動作の実現においては，様々な協調動作に横断する特徴として定義・利用する観点となる．これら２つの観点に分類し，組み合わせて協調動作を定義することで，表現力や柔軟性が向上する．

本研究では，各家庭内に存在する家電機器を連携させて協調動作を行うホームネットワークシステムを例として取りあげる．ホームネットワークシステムにおける協調動作の一例としては，各家庭内のＴＶやＤＶＤプレーヤ，スピーカー，エアコン，照明などを連携させて，映画館のような雰囲気で映画を見ることができるようなサービスが実現されているが，これらを例に観点について説明する．

4.1.1 基本観点

基本観点とは,機器の出力特性や機器本来が提供すべき基本機能などの特徴に着目して定義した観点のことである．協調動作の実現においては，様々な機器に横断する特徴として定義・利用する観点となる．基本観点が，機器に横断する様子を図 4.1 に示す.

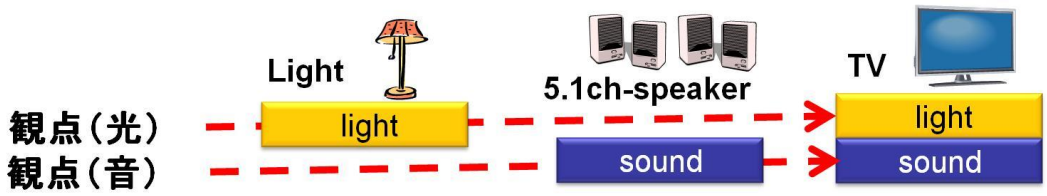

図 4.1：基本観点

本研究では，以下のような基本観点を定義した.

- Light

機器の出力や基本機能の特徴として「光」に関する特徴を持つ機器に対して定義する観点

- Sound

機器の出力や基本機能の特徴として「音」に関する特徴を持つ機器に対して定義する観点

- Temperature

機器の出力や基本機能の特徴として「熱」に関する特徴を持つ機器に対して定義する観点

- Ventilation

機器の出力や基本機能の特徴として「換気」に関する特徴を持つ機器に対して定義する観点

4.1.2 補助観点

補助観点とは，協調動作の実現において，協調動作の条件判断の定義要素に着目した観点であり，協調動作の実現においては，様々な協調動作に横断する特徴として定義・利用する観点となる．補助観点が協調動作に横断する様子を示したのが図 4.2 である．

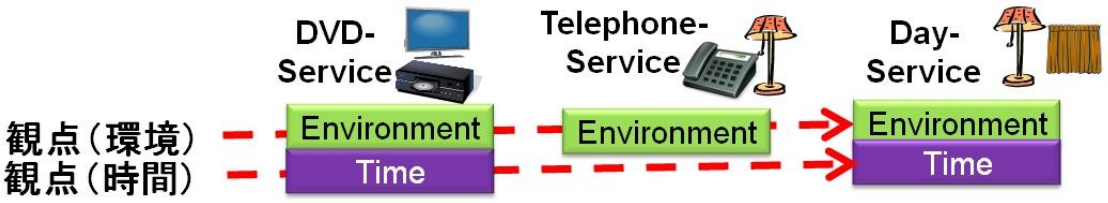

図 4.2：補助観点

本研究では，以下のような補助観点を定義した．

- Time　「時間」
  時間に関係する定義要素を持つ協調動作に対して定義する観点
- Environment　「環境」
  動作環境に関係する定義要素を持つ協調動作に対して定義する観点
- Relation　「関係」
  周辺機器や他の協調動作に関係する定義要素を持つ協調動作に対して定義する観点
- Exception　「例外」
  例外処理に関係する定義要素を持つ協調動作に対して定義する観点

4.2 観点を用いた制御指示

4.1 節で紹介したＤＶＤサービスを例題として，協調動作の分析と機器間に横断する特徴について説明する．製品のメーカーやグレード，動作環境など機器個別の事情に合わせて協調動作内容を定義すると以下のような例になる．

DVD シアターサービス｛
//協調動作設定
タイマー機能
動作環境情報の取得機能
優先度処理機能
例外時の処理機能
//各機器への制御指示
照明への制御指示
テレビへの制御指示
DVD プレーヤへの制御指示
エアコンへの制御指示
スピーカーへの制御指示
・・・
｝

このように，機器個別の指定や制御指示では，様々な機器のバリエーション，メーカー，制御方法，動作環境の違い等を考慮して各機器に対しての適切な制御指示を記述しなければならない．

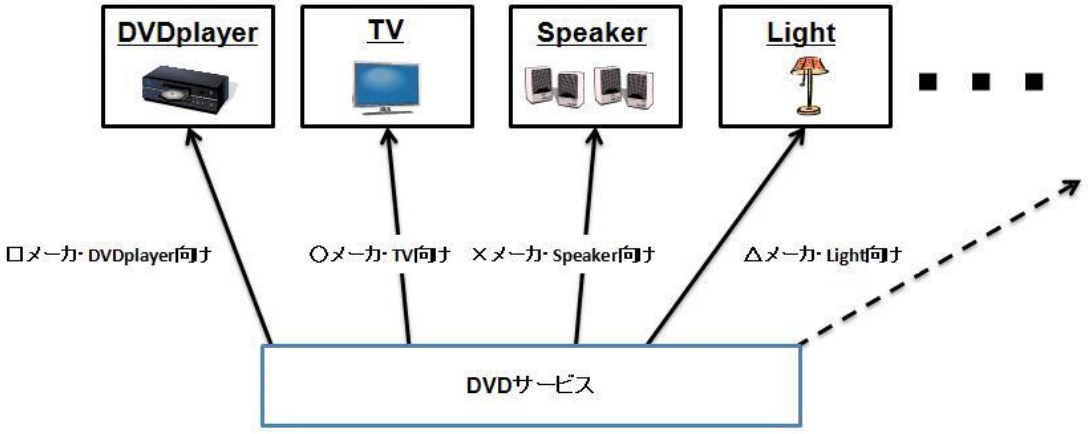

図 4.3：個別指示のイメージ図

一方，これらの協調動作をとらえる視点を変えて機器ごとではなく，本来機器が提供できる機能や協調動作の定義に横断する特徴に着目すると以下のようになる．

DVD サービス｛
//協調動作の定義間に横断する特徴
時間に関係する処理
動作環境に関係する処理
周辺機器や他の協調動作に関係する処理
例外処理

//機器が本来提供できる機能に着目
光に関わる機器への制御指示
音に関わる機器への制御指示
熱に関わる機器への制御指示
・・・
｝

このように機器本来が提供できる機能や対象に着目した場合，具体的な対象機器や動作環境に依存することなく協調動作を定義することができると考えられる．観点による制御指示のイメージを図 4.4 に示す．

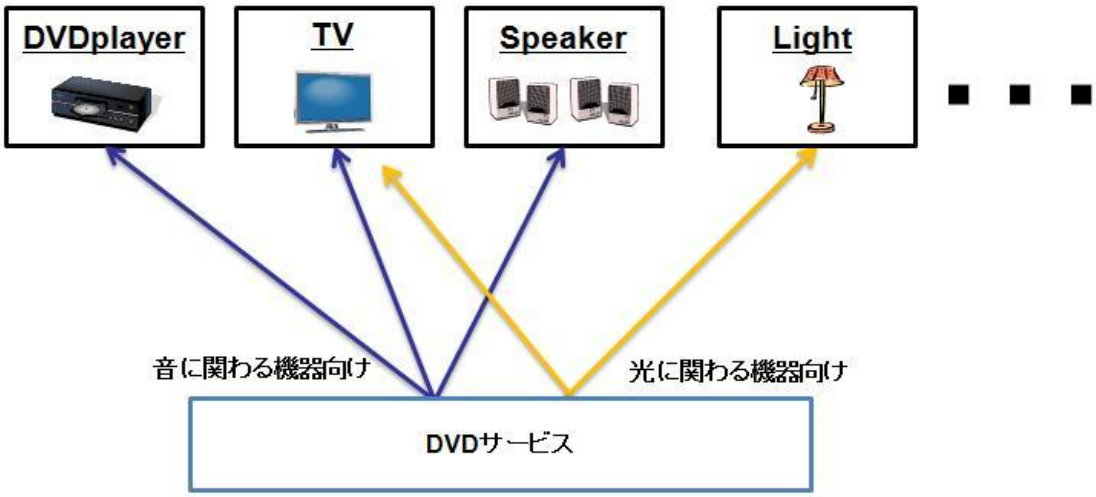

図 4.4：観点指示のイメージ図

上記の観点を用いた制御指示の実現方法について説明する.

まず,各機器の持つ基本機能や出力などに特徴に基づいて,機器設計者が基本観点を定義する.それを示したのが図 4.5 である.

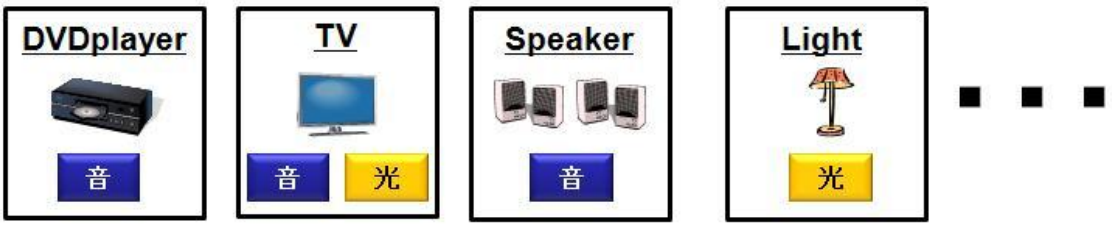

図 4.5：観点定義

そして,基本観点の情報に加えて,機器の設計者は各基本観点に関連した制御情報を定義する.具体的な記述方法については以降の章で説明する．その様子を図 4.6 に示す.

図 4.6：基本観点と制御情報

続いて，協調動作の内容を定義する．既存の協調動作と異なり，具体的な対象機器などを指定せずに観点ごとの協調動作内容を定義する.具体的な記述方法については以降の章で説明を行う.その様子を図 4.7 に示す.

図 4.7：協調動作内容

そして最後に，アドバイスの定義である．アドバイスは，協調動作の観点ごとの指示内容と各機器の制御情報を仲介する役割を持つ．協調動作内容に定義された観点ごとの指示内容を基に，対象とする観点を持つ機器に対して制御指示を行う．各機器に定義された基本観点と制御情報を基に指示を行うため，全ての機器に対してアドバイス適用することで，対象の観点を持つ機器をまとめて扱うことができる．また，各機器の制御情報を基に指示を行うため，制御方式やグレード，スペック，メーカーなどが異なっている場合でも，必要な制御情報が定義されていれば一括して扱うことができる．

以上の内容をふまえて，観点を用いた制御指示の全体像をまとめた図を示す．

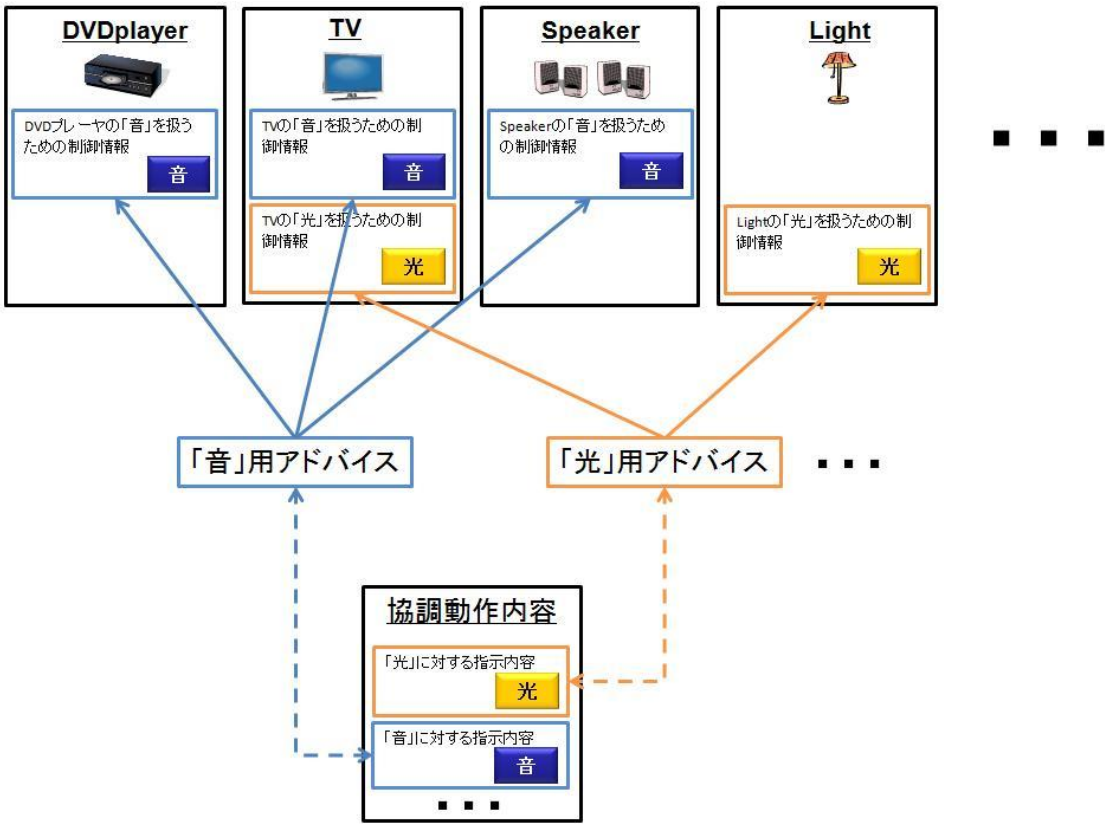

図 4.8：観点を用いた制御指示

ここでは，上記で説明した観点指示による DVD サービスの制御指示例を説明する．

DVD サービス｛
//協調動作の定義間に横断する特徴
時間に関係する処理　→　「Time アドバイス」
動作環境に関係する処理　→　「Environment アドバイス」
周辺機器や他の協調動作に関係する処理　→　「Relation アドバイス」
例外処理　→　「Exception アドバイス」

//機器が本来提供できる機能に着目
光に関わる機器への制御指示　→　「Light アドバイス」
音に関わる機器への制御指示　→　「Sound アドバイス」
熱に関わる機器への制御指示　→　「Temperature アドバイス」
・・・
｝

これらのアドバイスを基本観点と補助観点に関わるアドバイスごとに分けると下記のようになる．

補助観点に関わるアドバイス
Time アドバイス（補助観点「Time」に関わるアドバイス）
Environment アドバイス（補助観点「Environment」に関わるアドバイス）
Relation アドバイス（補助観点「Relation」に関わるアドバイス）
Exception アドバイス（補助観点「Exception」に関わるアドバイス）

基本観点に関わるアドバイス
Light アドバイス（基本観点「Light」に関わるアドバイス）
Sound アドバイス（基本観点「Sound」に関わるアドバイス）
Temperature アドバイス（基本観点「Temperature」に関わるアドバイス）

ここでは，基本観点に関わる Light アドバイスと補助観点に関わる Time アドバイスをとりあげて，それぞれの利用方法を説明する．

まず，Light アドバイスについて説明を行う．Light アドバイスは，基本観点「Light」と関連するアドバイスである．そのため，対象とする機器に基本観点「Light」が定義されていれば適用対象とするが，定義されていなければ適用対象とはしないという特徴を持たせる．それに加えて，各機器が「Light」の基本観点に対して持つ制御情報をまとめておく．これらの基本観点と制御情報をまとめたファイルを「機器設定ファイル」として定義する．機器設定ファイルの詳細については，4.4.1 節で述べる．

この条件下で全ての機器に対して Light アドバイスを適用すると，基本観点「Light」を持つ機器だけを適用対象とし，その機器の制御情報を基に各機器は自身の機能でできる範囲で対応する処理を行わせることができる．そのため全体として，「Light」の基本観点を持つ機器は光の出力を抑えるように働く．こうして機器構成などにとらわれず，光の出力を抑える協調動作を行わせることができる．他の基本観点「Sound」や「Temperature」などのアドバイスついても同様である．

Light アドバイスの例を図 4.9 に示す．

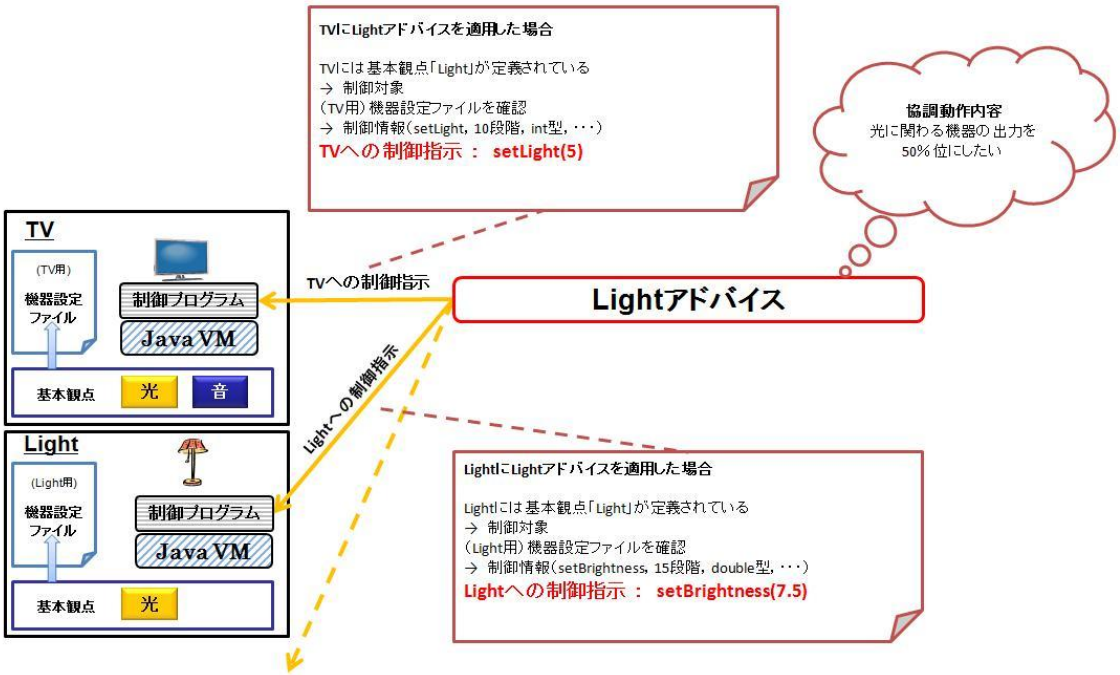

図 4.9：Light アドバイスの例

続いて，Time アドバイスについて説明する．Time アドバイスは，補助観点「Time」と関連したアドバイスである．そのため，対象とする協調動作に補助観点「Time」が定義されていれば適用対象とするが，定義されていなければ適用対象とはしないという特徴を持たせる．それに加えて，協調動作が補助観点「Time」と関わるのであれば，協調動作の指示内容をまとめておく．これらの情報をまとめたファイルを「協調動作設定ファイル」として定義する．協調動作設定ファイルの詳細については，4.4.2 節で述べる．

この条件下で全ての協調動作に対して Time アドバイスを適用すると，補助観点「Time」を持つ協調動作だけを適応対象とし，その協調動作の指示内容に基づいてアドバイスの適用が行われる．他の補助観点「Environment」や「Relation」などのアドバイスついても協調動作を対象とする処理である．

Time アドバイスの例を図 4.10 に示す．

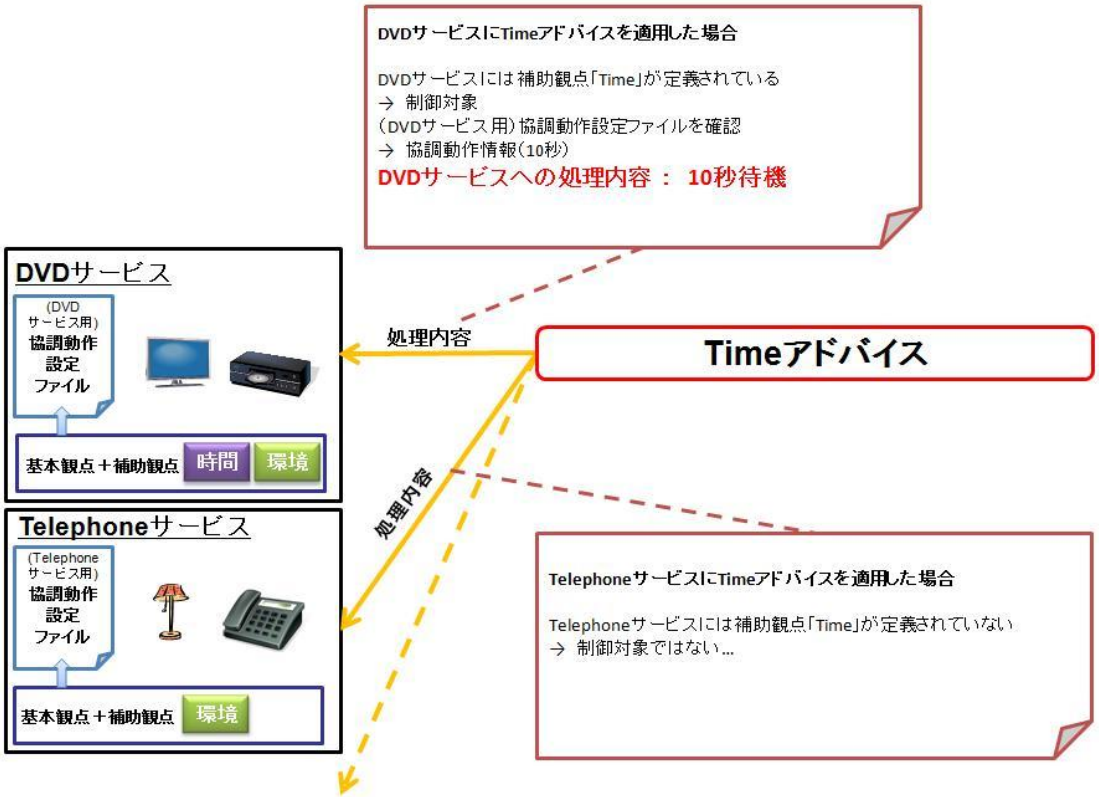

図 4.10：Time アドバイスの例

このようにして，基本観点と補助観点を用いて，それに対するアドバイスを適用することで，対象機器や動作環境などに依存することなく，協調動作を定義することができる．

4.3　協調動作の実現

4.2 節では，観点を用いた定義により，対象機器や動作環境などに依存することなく協調動作を定義できることを説明した．しかし，目的とする協調動作を行わせるためには，協調動作ごとに定義された観点に合わせて，必要なアドバイスを組み合わせる仕組みが必要である．そのための仕組みとして，本手法では Weaving Section というアドバイス適用箇所であるメソッドの集合を持つ構造を提案する．この Weaving Section を機器側の API またはホームサーバ側の API として定義しておくことで，メソッド呼出しに対するアドバイス適用を利用して，実行時に必要なアドバイスを組み合わせて協調動作を実現することができる．

この章では，Weaving Section のメソッドの一覧を示した後，各メソッドの役割と利用方法について述べる．

4.3.1　メソッド一覧

本研究で定義したメソッド例の一覧を以下に示す．

Weaving Section のメソッド一覧

- Collaboration( )
- Time( )
- Environment( )
- Relation( )
- Exception( )

各メソッドとも，メソッド呼出しをポイントカットにしたアドバイス適用に利用するためのメソッドである．そのため，各メソッド自体には特別な処理は何も記述されていない．各メソッドの役割については，次節以降で説明する．

4.3.2　各メソッドの役割

この節では，各メソッドの役割について個々に説明する．

Collaboration( )

このメソッドは，各基本観点に関するアドバイスを適用するためのメソッドである．協調動作

の実行時点の協調動作内容に基づいて組み合わされたアドバイスを適用することで,各協調動作の制御指示を実現する.

Time( )

このメソッドは,「Time」の補助観点を持つ協調動作に対してアドバイスを適用するためのメソッドである.協調動作の内容として実行タイミングの調整などが必要な場合,対象の協調動作設定ファイルに「Time」の補助観点を設定しておき,対応するアドバイスを定義しておくことで,時間を考慮した協調動作を実現することができる.

Environment( )

このメソッドは,「Environment」の補助観点を持つ協調動作に対して,アドバイスを適用するためのメソッドである.協調動作の内容として,動作環境の情報を考慮する必要がある場合,対象の協調動作設定ファイルに「Environment」の補助観点を設定しておき,対応するアドバイスを定義しておくことで,動作環境を考慮した協調動作を実現することができる.

Relation( )

このメソッドは,「Relation」の補助観点を持つ協調動作に対して,アドバイスを適用するためのメソッドである.協調動作の内容として,周辺機器や他の協調動作の情報を考慮する必要がある場合,対象の協調動作設定ファイルに「Relation」の補助観点を設定しておき,対応するアドバイスを定義しておくことで,周囲との関係を考慮した協調動作を実現することができる.

Exception( )

このメソッドは,「Exception」の補助観点を持つ協調動作に対して,アドバイスを適用するためのメソッドである.協調動作の内容として,協調動作実行中に例外が発生した場合に対応処理を考慮する必要がある場合,対象の協調動作設定ファイルに「Exception」の補助観点を設定しておき,対応するアドバイスを定義しておくことで,例外処理を考慮した協調動作を実現することができる.

4.3.3　メソッドの呼出し順序

Weaving Section の各メソッドの呼出し順序について述べる．本手法では，メソッド呼出しに対するアドバイスの適用により協調動作を実現する．そのため，メソッド呼出しの順序が重要となる．例えば，Collaboration( )に適用されるアドバイスにより制御指示を実行した後に，Time( )に適用されるアドバイスにより協調動作の実行タイミングの調整などを行っても，目的とする協調動作を実現することはできない．

適用の順序については下記のようになる．

通常時

1. Time( )の呼出し

補助観点「Time」と関連付けたアドバイスにより，協調動作の実行タイミングの調整など，時間に関わる処理を実行時に追加する．

2. Environment( )の呼出し

補助観点「Environment」と関連したアドバイスにより，実行時点の外部環境の情報を元に協調動作内容の調整を行う処理を追加する．

3. Relation( )の呼出し

補助観点「Relation」と関連したアドバイスにより，実行時点の周辺機器や他の協調動作の情報を元に協調動作内容の調整を行う処理を追加する．

4. Collaboration( )の呼出し

協調動作設定ファイルに定義された基本観点と関連したアドバイスと，協調動作設定ファイルの指示内容を基に制御指示を行う．全ての機器を対象に Collaboration( )のメソッド呼出しを行い，協調動作の対象とする観点を持つ機器に対して制御指示を行い，対象とする観点を持たない機器に対しては特別な指示は行われない．

例外時

1. Exception( )の呼出し
補助観点「Exception」と関連したアドバイスにより，例外時の対応処理を行う処理を追加する．

これらの手順を表した図を下記に示す．

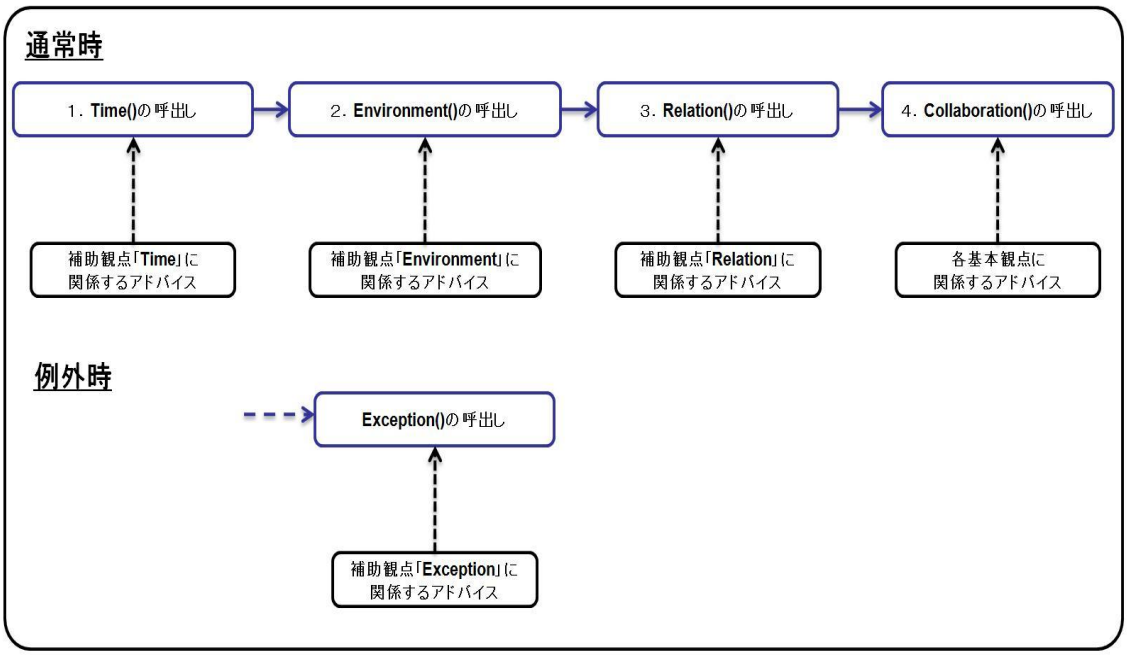

図 4.11：メソッドの呼出し順序

4.3.4　事前定義

提案手法による協調動作の実現のために，各機器の基本観点と制御情報をまとめた「機器設定ファイル」と，協調動作に関わる観点と指示内容をまとめた「協調動作設定ファイル」を事前に定義しておく．詳細については 4.4 節で述べる．

4.3.5　開始要求からアスペクトの組み換えまで

ユーザからの開始要求や機器構成の変化などをきっかけとして開始要求を受け取る. 開始要求を受け取ったら, 各協調動作の協調動作設定ファイルに定義された観点に基づき, 適用すべきアスペクトの組み合わせを変更する．具体的には，3.3.6 節で紹介したアスペクトの設定ファイル(XML ファイル)の内容を書き換えることで，Weaving Section の各メソッドに適用されるアスペクトの組み合わせを実行時に変更することができる. この段階で観点の情報を基にアスペクトの組み合わせを変更することで，この後の Weaving Section のメソッド呼出しに適用されるアドバイスが変更され，様々な協調動作を実現することができる.

4.3.6　Weaving Section のメソッド呼出し

観点の情報を基に適用すべきアスペクトの組み合わせの変更が完了したら, 次にアドバイス適用のためのメソッド呼出しを行う. Weaving Section のメソッド呼出しの順番については，4.3.3 節で紹介した通りである.

はじめに，Time( )のメソッド呼出しを実行することで，補助観点「Time」と関連付けたアドバイスを実行することができる．Time( )に適用するアドバイスにより，協調動作の実行タイミングを調節することができる. アスペクトの組み換えの段階で対応するアスペクトを設定していなければ，特別な処理は何も行われないため，即時実行ということになる.

続いて，Environment( )のメソッド呼出しを実行することで，補助観点「Environment」と関連付けたアドバイスを実行することができる．Environment( )に適用するアドバイスにより, 時間帯や気温, 天気など動作環境の情報を考慮した協調動作を行わせることができる. アスペクトの組み換えの段階で対応するアスペクトを設定していなければ, 動作環境の情報は考慮しない協調動作となる.

次に，Relation( )のメソッド呼出しを実行することで，補助観点「Relation」と関連付けたアドバイスを実行することができる. Relation( )に適用するアドバイスにより, 周囲の機器環境や, 他の実行中の協調動作等との関係を考慮した協調動作を行わせることができる. 他と同様，対応するアドバイスが設定されていない場合には，特別な処理は何も行われない.

最後に，Collaboration( )のメソッド呼出しを接続機器に対してブロードキャストとなる形で行う．各機器は基本観点に基づく制御指示を，自身の持つ「機器設定ファイル」に定義された基本観点と制御情報に照らし合わせて制御処理を行う.

また例外発生時には，Exception( )のメソッド呼出しを行い，対応したアドバイスを適用することで，接続機器に対して例外時の対応処理も行わせることができる.

4.4　観点情報の定義

本研究では,「基本観点」と「補助観点」, 第 3 章で紹介した動的なアスペクト指向技術を利用して, 機器個別に制御指示を記述するのではなく, 観点を用いた制御指示により協調動作を実現する方式を提案する. そのために, 機器が持つ基本観点と制御情報に関する情報をまとめた機器設定ファイルと, 協調動作の観点と指示内容をまとめた協調動作設定ファイルを事前に定義する.

4.4.1 機器設定ファイル

機器設定ファイルとは,「各機器が持つ基本観点とその機器の制御情報をまとめたファイル」であり, 機器ごとに定義・保持しているものとする.

本研究では, 以下のような項目を定義した.

- 基本観点

機器の持つ基本観点を定義する.

- 制御用メソッド名

各機器が持つ基本観点を扱うための制御メソッドの名称を定義. 1 つの観点に対して複数のメソッドを定義することも想定する.

- 制御幅

各制御用メソッドの制御幅を定義する.

- パラメータの型

各制御用メソッドのパラメータの型を定義する.

- ネットワーク・接続情報

各機器のネットワーク・接続情報を記述する.

これらの項目を含めた機器設定ファイルの BNF の例を以下に示す.

機器設定ファイルの BNF

propertiesfile ::= CommonView
ControlMethod
ControlRange
ParameterType
ServiceURL
ServiceInterface

commonView ::= ‘commonView =’ [view ‘,’]*
view ::= ‘*Sound*’ | ’*Light*’ | ‘*Temperature*’ |’Ventilation’

controlMethod ::= ‘controlMethod =’ [methodName [‘-’ methodName]* ‘,’]*
methodName ::= *STRING*

controlRange ::= ‘controlRange =’ [range [‘-’ range]* ‘,’]*
range ::= *REAL NUMBER*

parameterType ::= ‘parameterType =’ [type [‘-’ type]* ‘,’]*
type ::= ‘*int*’ | ‘*double*’

serviceURL ::= ‘*serviceURL* = ’ ‘rmi://’ <URL | IP> ‘:’ Port ‘/’ ServiceName
URL ::= *STRING*
IP ::= *STRING*
Port ::= *INTEGER NUMBER*
ServiceName ::= *STRING*
serviceInterface ::= ‘*serviceInterface* = ’ Interface
Interface ::= *STRING*

4.4.2 協調動作設定ファイル

協調動作設定ファイルとは，「協調動作に関する観点情報を，基本観点と補助観点を組み合わせて表現したファイル」であり，協調動作ごとに定義されているものとする．

本研究では，以下のような項目を定義した．

- サービス名

協調動作の名称を定義する．

- 優先度

協調動作間の優先度を定義する．

- 対象とする基本観点

協調動作の対象とする基本観点を定義する．

- 基本観点に適用するアスペクト名

各基本観点に適用するアスペクトの名称を定義する．

- 指示内容

各基本観点に対する指示内容を記述する．

- 対象とする補助観点

協調動作の対象とする補助観点を定義する．

- 補助観点に適用するアスペクト名

各補助観点に適用するアスペクトの名称を定義する．

- 指示内容 2

各補助観点に対する指示内容を記述する．

これらの項目を含めた協調動作設定ファイルの BNF の例を以下に示す．

協調動作設定ファイルの BNF

propertiesfile ::= serviceName
priority
targetCommonView
commonAspect
instruction
targetAuxiliaryView
auxirlialyAspect
instruction2

serviceName ::= ‘serviceName =’ [service ‘,’]*
service ::= ‘DVD_Service’ | ‘Telephone_Service’ | ’Day_Service’ | ’Cooking_Service’|
’Relax_Service’

priority ::= ‘priority =’ INTEGER NUMBER

targetCommonView ::= ‘targetCommonView =’ [commonView ‘,’]*
commonView ::= ‘Sound’ | ‘Light’ | ‘Temperature’ |’Ventilation’

commonAspect ::= ‘commonAspect =’ [cAspect ‘,’]*
cAspect ::= ‘LightAspect’ | ‘SoundAspect’ |’TemperatureAspect’ | ‘VentilationAspect’

instruction ::= ‘instruction =’ [REAL NUMBER ‘,’]*

targetAuxiliaryView ::= ‘targetAuxiliaryView =’ [auxiliaryView ‘,’] *
auxiliaryView ::= ‘Time’ | ‘Environment’ |’Relation’ | ‘Exception’

auxiliaryAspect ::= ‘auxiliaryAspect =’ [aAspect ‘,’]*
aAspect ::= ‘TimeAspect’ | ‘EnvironmentAspect’ | ‘RelationAspect’ | ‘ExceptionAspect’

instruction2 ::= ‘instruction2 =’ [REAL NUMBER ‘,’]* | [STRING ‘,’]*

第 5 章　プロトタイプシステムの実装

本研究では，ホームネットワークシステムを取り上げ，ホームネットワークシステムにおける DVD サービスを例題として提案方式の適用を行う．まず，動作環境を示し，協調動作一覧，機器構成について説明を行う．

5.1　動作環境

今回の例題の適用においては実際のネットワーク環境や実際の機器を用いずに，1 台の PC 上でのシミュレーションを行った．具体的には，1 台のホスト上にて個別のＶM上で各機器プログラムを動作させ，ホームサーバプログラムと各機器プログラム間のメッセージ通信には Java RMI を用いて制御指示を行った．また，Spring の実行環境とアスペクトについては，ホームサーバプログラムに配置している．下記に動作環境を示す．

- 各機器の実行環境として Java VM の動作環境が整っている．
- 各機器の制御コードは Java 言語で定義されている．
- 各機器とホームサーバの間はネットワークで接続されている．
- 各機器とホームサーバは RMI 等を用いてメッセージ通信を行い，またそれらのメッセージ通信における API を備えており，外部からの制御が可能である．
- ホームサーバ側には，Spring の環境が整っている．また，Spring で用いるアスペクトについては，ホームサーバ側に定義されているものとする．

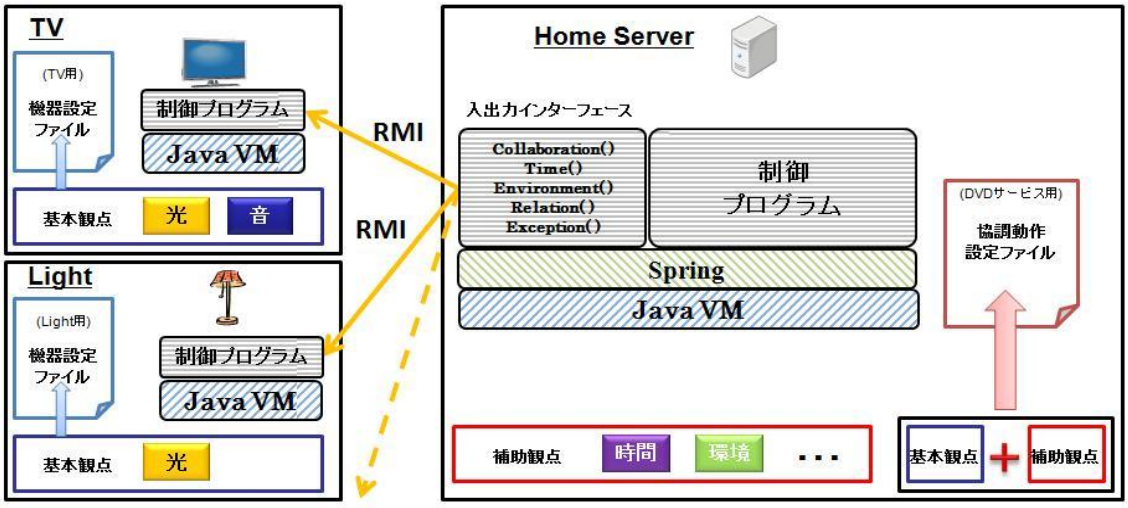

図 5.1：観点指示のための動作環境

5.2 協調動作一覧

今回取り上げるホームネットワークの協調動作として以下のサービスを想定する．

- DVD サービス

TV，DVD プレーヤ，スピーカー，照明，エアコンを連携して，映画館の雰囲気で視聴できるサービス．ユーザがサービスを起動すると，照明が暗くなり，オーディオ機器以外の音量を落とし，部屋を設定した温度に調節する．

- Telephone サービス

電話がかかってきた際に，照明やスピーカー等と連携して，最適な通話環境を整えるサービス．電話が着信状態になると，照明が明るくなり，TV，スピーカー等の音量を落とす．

- Cooking サービス

換気扇，オーブン，給湯器などを連携して炊事準備を行うサービス．ユーザがサービスを起動すると，給湯器，換気扇，オーブンの電源を点け，すぐに使える状態に準備する．

- Relax サービス

ＤＶＤプレーヤ，スピーカー，照明，エアコン，窓，カーテンを連携して，ユーザがリラックスできる空間を演出するサービス．ユーザがサービスを起動すると音楽が再生され，空調を適温に，照明を適度な明るさに調節する．

- Day サービス

朝になったら，特定の場所の照明が消灯し，カーテンが開く．
夕方になったら，特定の照明が点灯し，カーテンが閉じる．

これらの協調動作の補助観点をまとめたものが図 5.2 である.

	Time	Environment	Relation	Exception
DVDサービス	○	○	○	○
Telephoneサービス		○		○
Cookingサービス	○			○
Relaxサービス		○	○	○
Dayサービス	○	○		○

図 5.2：補助観点例

今回はこれらの協調動作の中で, DVD サービスを例として提案方式を適用した例を紹介する.

5.3　機器構成

今回取り上げるホームネットワークの協調動作に関わる機器として以下の機器を想定する．また，これらの機器以外に集中管理を行うホームサーバが存在すると仮定する．

1. テレビ
2. DVD プレーヤ
3. 電話
4. スピーカー
5. エアコン
6. 照明機器
7. カーテン
8. 窓
9. オーブン
10. 換気扇
11. 給湯器

これらの機器の基本観点をまとめたものが図 5.3 である．

	Light	Sound	Temperature	Ventilation
テレビ	○	○		
DVDプレーヤ		○		
電話機				
スピーカー		○		
エアコン		○	○	
照明	○			
カーテン	○			
窓				○
オーブン			○	
換気扇				○
給湯器			○	

図 5.3：基本観点例

これらの機器の中で今回の DVD サービスの例題では，テレビ，DVD プレーヤ，電話機，スピーカー，エアコン，照明の 6 機器を用いる．これらの機器プログラムの機能を図 5.4 に示す．

機器名称	制御メソッド	引数の型	制御幅	制御用変数名	基本観点
Television	setVolume	int	0－10	volume	Light Sound
	getVolume				
	setBrightness	int	0－10	brightness	
	getBrightness				
	ON	int	0(OFF),1(ON)	power	
	OFF				
DVDPlayer	setSound	double	0.0－12.0	sound	Sound
	getSound				
	ON	int	0(OFF),1(ON)	power	
	OFF				
Telephone	setVolume	int	0－10	volume	Sound
	getVolume				
	ON	int	0(OFF),1(ON)	power	
	OFF				
Airconditioner	setTemperature	int	20－35	temperature	Temperature Sound
	getTemperature				
	setWindVolume	int	0－10	windVolume	
	getWindVolume				
	setWindSpeed	int	0－10	windSpeed	
	getWindSpeed				
	ON	int	0(OFF),1(ON)	power	
	OFF				
Light	setLight	double	0.0－7.0	light	Light
	getLight				
	ON	int	0(OFF),1(ON)	power	
	OFF				
Speaker	setVolume	int	0－20	volume	Sound
	getVolume				
	ON	int	0(OFF),1(ON)	power	
	OFF				

図 5.4：機器プログラム機能一覧

第 6 章　例題への適用

6.1　適用の流れ

ここでは事前定義から，設定された観点に基づいてアスペクトを組み合わせ，Weaving Section のメソッド呼出しによるアドバイス適用までの流れについて説明する．

6.1.1　機器設定ファイルの定義

機器設定ファイルについては，5.2 節で紹介した機能と基本観点に基づき，機器ごとに作成した．機器設定ファイルの記法についての詳細は，4.2.1 節で紹介した機器設定ファイルの BNF を参照．ここでは，テレビの機器設定ファイルの一例を紹介する．

TV.properties

commonView = Light,Sound
controlMethod = setBrightness,setVolume
controlRange = 10,10
parameterType = int,int
serviceUrl = rmi://localhost:1099/TVInputOutput
serviceInterface = tv.TV_InputOutput

6.1.2　協調動作設定ファイルの定義

本例題においては，協調動作のメインとなる機器（TV やスピーカー等）については，予め観点による制御指示対象から除外して，その周囲の機器への適用を試す．この条件をふまえて設定した協調動作の観点による指示内容の一例を紹介する．

基本観点：Light　20 (%)
基本観点：Sound　20 (%)
基本観点：Temperature　25 (℃)
補助観点：Time　10（秒：開始までの待機時間）
補助観点：Exception　OFF（例外時対応処理：電源を落とす）

この指示内容をまとめた協調動作設定ファイルの一例を示す. 協調動作設定ファイルの記法についての詳細は，4.2.2 節で紹介した協調動作設定ファイルの BNF を参照.

DVD_Service.propeties

serviceName=DVD_Service
priority=2
targetCommonView=Light,Sound,Temperature
cAspect=LightAspect,SoundAspect,TemperatureAspect
instruction=20,20,25
targetAuxiliaryView=Time,Exception
aAspect=TimeAspect,ExceptionAspect
instruction2=10,OFF

6.1.3　アドバイスの定義

今回の協調動作内容を考慮して，Light アドバイス，Sound アドバイス，Temperature アドバイス，Time アドバイス，Exception アドバイスの 5 つを定義した．

Light アドバイス

機器設定ファイル内の基本観点「Light」と関連付けられた制御情報と協調動作設定ファイルの指示内容を基に各機器に合わせた制御指示を実現できるように制御指示を記述する．

Sound アドバイス

機器設定ファイル内の基本観点「Sound」と関連付けられた制御情報と協調動作設定ファイルの指示内容を基に各機器に合わせた制御指示を実現できるように制御指示を記述する．

Temperature アドバイス

機器設定ファイル内の基本観点「Temperature」と関連付けられた制御情報と協調動作設定ファイルの指示内容を基に各機器に合わせた制御指示を実現できるように制御指示を記述する．

Time アドバイス

協調動作設定ファイル内の補助観点「Time」と関連付けられた指示内容を基に，協調動作の実行タイミングを調整する制御指示を記述する．今回の例では，現在の時刻取得とカウント機能を定義したアドバイスを準備した．これにより，一定時間後または指定時間より協調動作を開始させる．

Exception アドバイス

協調動作設定ファイル内の補助観点「Exception」と関連付けられた指示内容を基に，例外時に行う処理内容を記述する．今回の例では，各機器の電源操作に関する処理を記述してある．

6.1.4　開始要求からアスペクトの組み換えまで

今回の例題においては，協調動作の開始要求受付の GUI を作成した．実際の環境では，直接 DVD サービスの要求または DVD の挿入，再生ボタンが押された時のイベント等となると思われるが，その代わりとして準備した．作成した GUI の画面を図 6.1 に示す．

図 6.1：開始要求受付用 GUI

「DVD サービスの開始」「Telephone サービスの開始」のそれぞれのボタンが押されることにより各協調動作の開始要求とする．今回は DVD サービスの開始要求が行われた場合の流れについて説明する．

「DVD サービスの開始」ボタンが押され，開始要求を受け付けると，DVD サービス用の協調動作設定ファイルに設定された基本観点と補助観点に基づき，適用するアスペクトを組み合わせる．本例題では，協調動作設定ファイルには，基本観点として「Light」「Sound」「Temperature」，補助観点として「Time」「Exception」が定義されている．これらの観点に対するアドバイスが適用されように 3.3.6 節で紹介したアスペクトの設定ファイルの内容を，定義されている観点情報を基に書き換えることで，適用するアドバイスを組み替えることができる．そして，この後 Weaving Section のメソッド呼出しを行うことによって，協調動作ごとに必要なアドバイスが適用されることになる．

6.1.5　WeavingSection のメソッド呼出し

4.5.3 節で紹介した Weaving Section の呼出し順序に従い各メソッドの呼出しを行う．

はじめに，Time( )のメソッド呼出しを行う．今回の例題では補助観点「Time」が設定されており，Time アドバイスの定義も行った．Time アドバイスの処理内容として，非常に単純な例ではあるが，「現在時刻を取得し，一定時間待機する」という内容を定義した．そのため，ここでは 10 秒待機するという処理が行われる．

続いて，Environment( )のメソッド呼出しを行う．今回の例題では補助観点「Environment」と対応するアドバイスを設定していないため，特別な処理は追加されない．

次に，Relation( )のメソッド呼出しであるが，今回の例題では補助観点「Relation」と対応するアドバイスを設定していないため，特別な処理は追加されない．

最後に，Collaboration( )のメソッド呼出しをブロードキャストになるように行う．今回は，各機器の GUI からホームサーバ側へ機器登録できる機能を実装してある．今回は登録された機器リストを元にブロードキャストを行っている．

図 6.2 に機器 GUI の例として TV プログラムの GUI を示す．

図 6.2：TV プログラムの GUI

機器 GUI の操作一覧

- 接続

Home Server 側に機器登録要求を行う.
実際の機器であれば，機器が接続された状況の代わりとする.

- 解除

Home Server 側に機器解除要求を行う.
実際の機器であれば，機器が取り外された状況の代わりとする.

- 更新

GUI 画面の更新・再描画を行う.

- ON

機器の電源を ON にする.
前回終了時の値をファイルから読み出し，現在の各設定値を更新する.

- OFF

機器の電源を OFF にする.
現在の設定値をファイルに書き出し，全ての設定値を 0 に更新して電源 OFF 画面に変更する.
電源 ON と OFF の時の画面の違いを図 6.3 に示す.

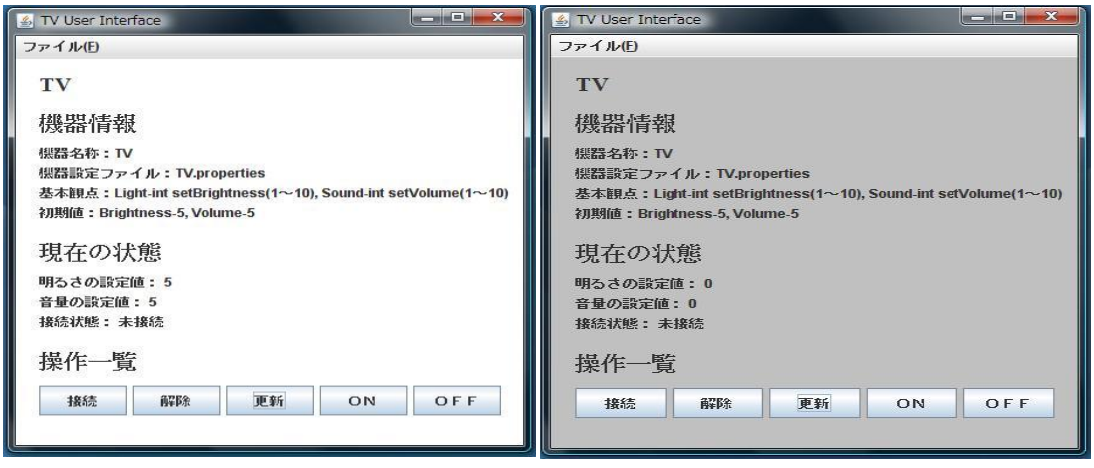

図 6.3：電源 ON 状態の画面（左），電源 OFF 状態の画面（右）

6.2 適用結果

DVD サービス実行前の状態（初期状態）が図 6.4 である.

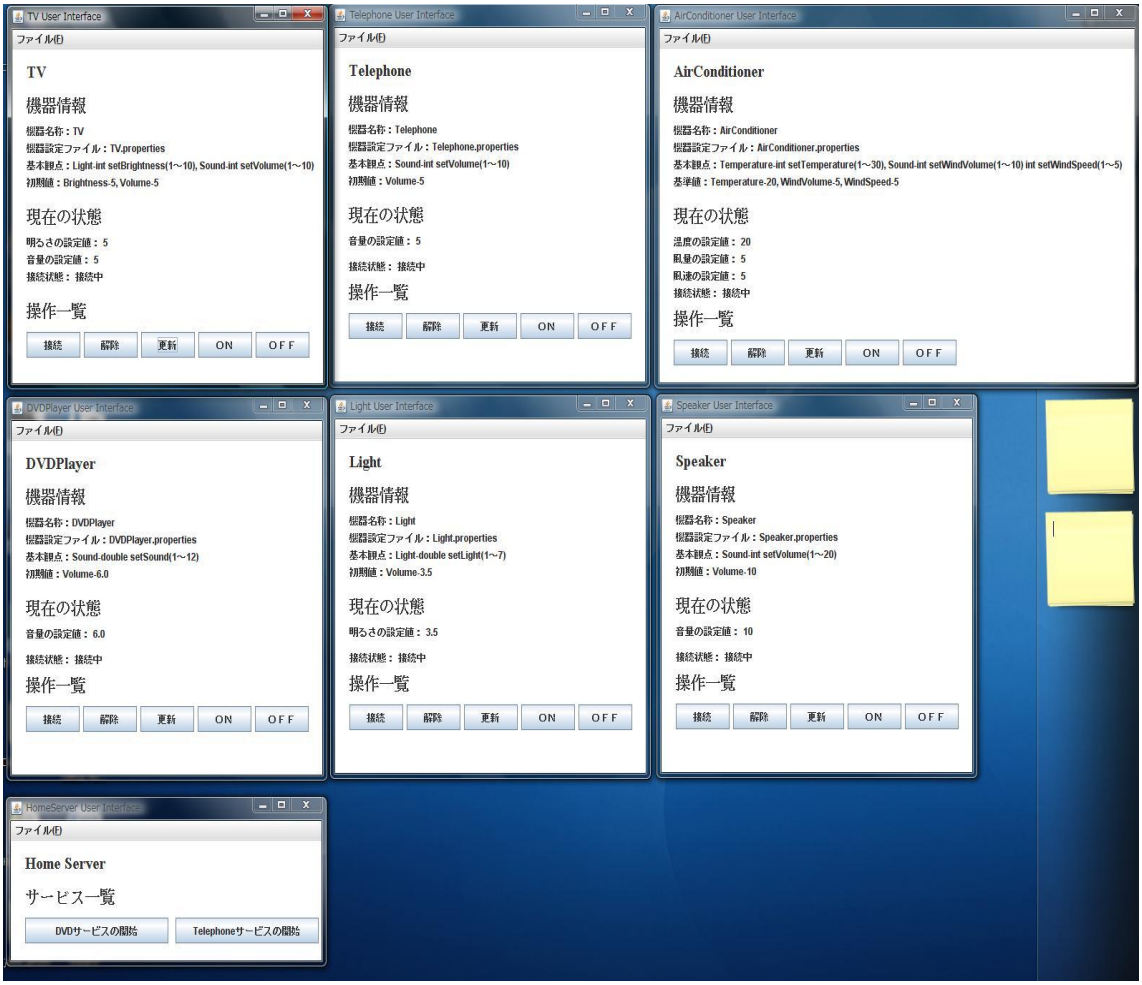

図 6.4：DVD サービス実行前の状態

DVD サービス実行前の各機器の状態

TV（明るさの設定値：5 ， 音量の設定値：5）
DVDPlayer（音量の設定値：6.0）
Telephone（音量の設定値：5）
Airconditioner（温度の設定値：20 ， 風量の設定値：5 ， 風速の設定値：5）
Light（明るさの設定値 3.5）
Speaker（音量の設定値：10）

DVD サービス実行後の状態が図 6.5 である.

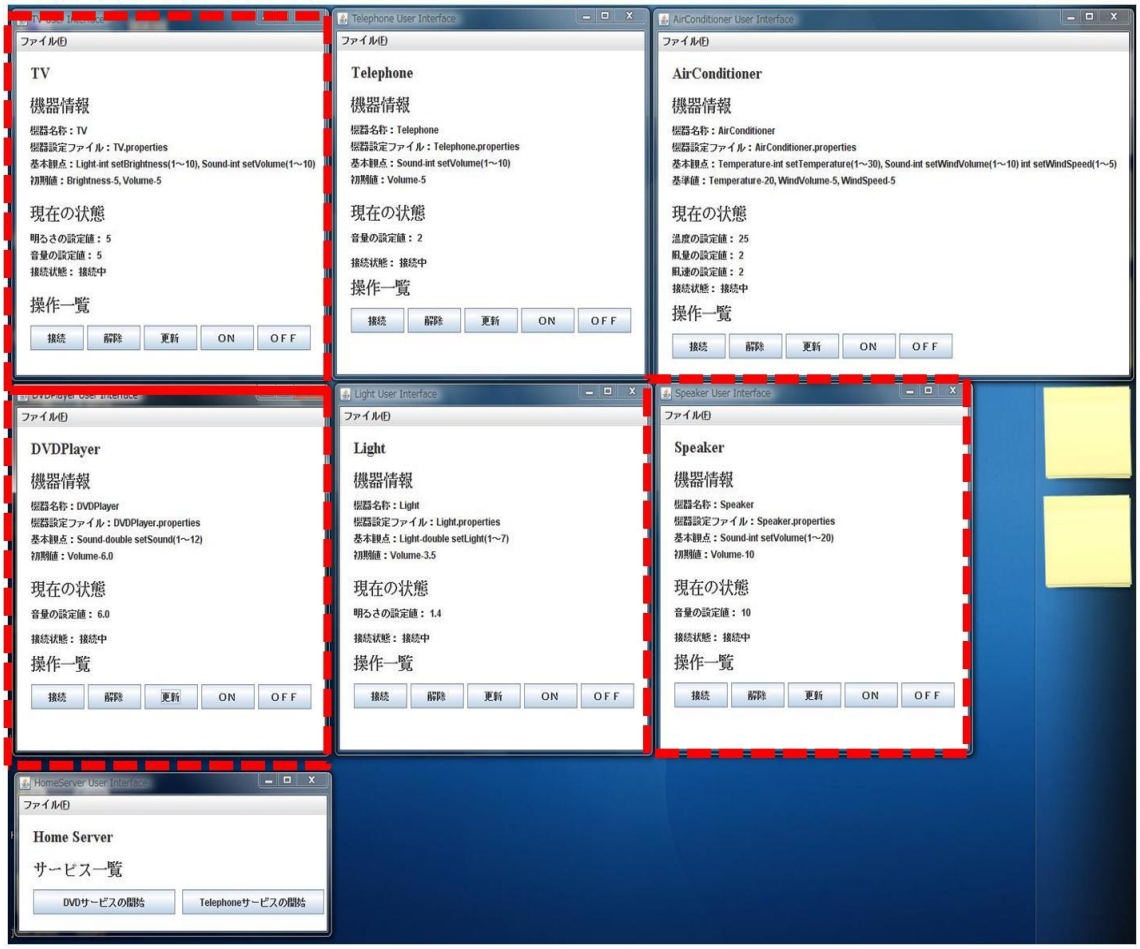

図 6.5：DVD サービス実行後

点線の枠で囲まれている機器プログラムは, 今回の協調動作のメインとなる機器であり予め対象機器から除外した. それ以外の機器に関して, 以下のように観点を用いた制御指示により設定値を変更できることを確認した.

DVD サービス実行後の各機器の状態

TV（明るさの設定値：5　，音量の設定値：5）
DVDPlayer（音量の設定値：6.0）
Telephone（音量の設定値：5→2）
Airconditioner（温度の設定値：20→25　，風量の設定値：5→2　，風速の設定値：5→2）
Light（明るさの設定値 3.5→1.4）
Speaker（音量の設定値：10）

また，Exception()を実行した場合，今回の例題では Exception アドバイスの内容として「全ての機器の電源を OFF にする」という処理内容が記述してある．実行した結果は図 6.6 のようになる．

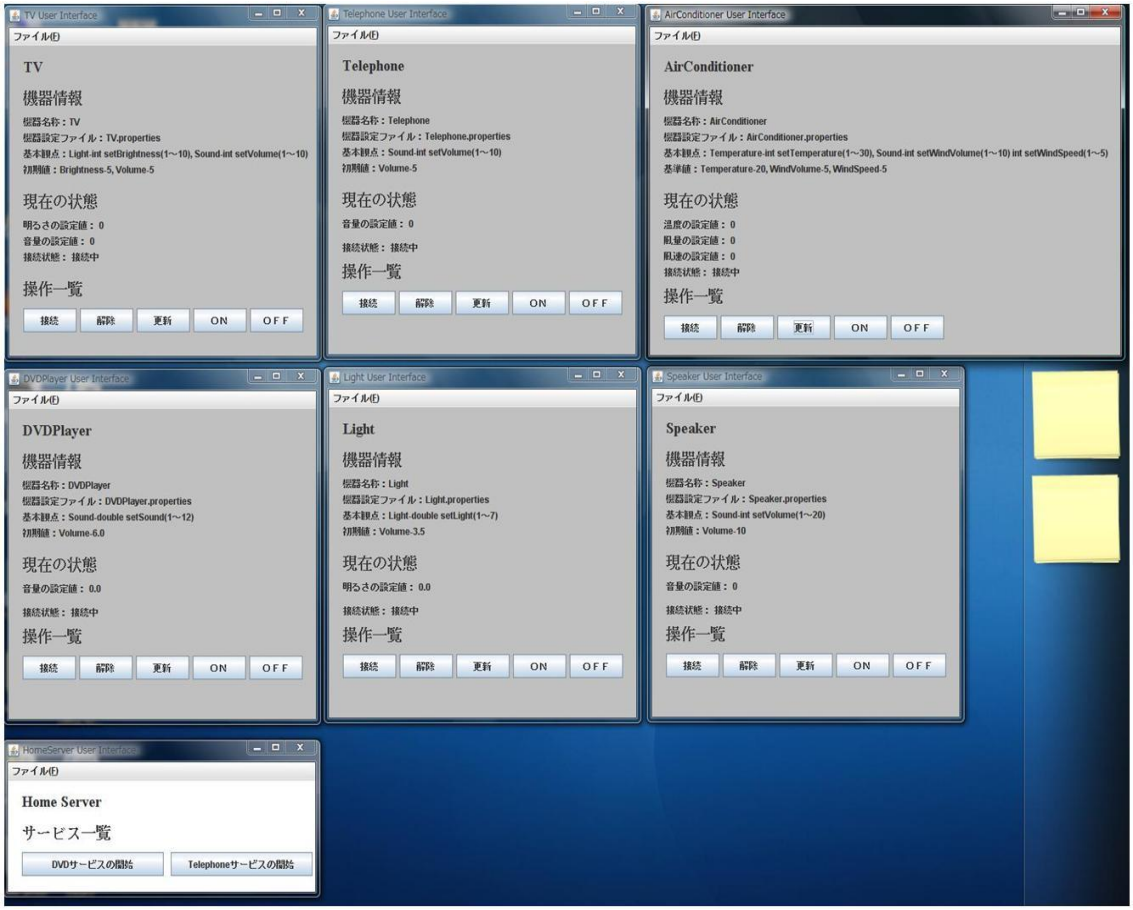

図 6.6：Exception アドバイスの適用画面

例外時には，Exception()のメソッド呼び出しを行うことで，全ての機器の電源を OFF にするという例外処理を適用することができた．

6.3 評価

6.3.1　評価 1　制御コードの定義箇所（実装範囲）

実装範囲において，本提案手法による制御コードの必要な箇所を計算したものが以下である．実装範囲については，機器数 6，協調動作数 2 の場合であり，詳細については本章の例題の説明を参照．

	制御指示のみ		条件設定含む	
（制御指示）(条件設定)	定義箇所数	相対値	定義箇所数	相対値
個別指示 ＋ 個別指示	11	100.0	23	100.0
基本観点 ＋ 個別指示	3	27.2	15	65.2
個別指示 ＋ 補助観点	11	100.0	13	56.5
基本観点 ＋ 補助観点	3	27.2	5	21.7

図 6.7：評価結果 1

この図中で用いている「制御指示のみ」という条件は，協調動作の開始時に対象機器に対して行う制御指示のみを計算したものである．また「条件設定含む」という条件は，協調動作の開始時の対象機器への制御指示を含め，終了時の処理や例外処理などその他の定義要素を全て含めた場合である．

この実装範囲における評価 1 の結果から，具体的な数字に関しては本研究の想定条件によって増減してしまうため大きな意味はないが，個別指示に比べて観点を用いた制御指示を行うことでより少ない定義で実現することができることが分かる．

続いて評価 2 で想定範囲も含め，協調動作数と定義箇所の増減の関係についてその傾向を確認する．

6.3.2　評価 2　制御コードの定義箇所（想定範囲）

本提案手法による制御コードの必要な定義箇所を計算したものが以下である．

	協調動作数:1		協調動作数:2		協調動作数:3		協調動作数:4		協調動作数:5	
(制御指示) (条件設定)	定義 箇所数	相対値	定義 箇所数	相対値	定義 箇所数	相対値	定義 箇所数	相対値	定義 箇所数	相対値
個別指示+ 個別指示	16	100.0	32.8	100.0	49.2	100.0	64.4	100.0	82.0	100.0
基本観点+ 個別指示	9.6	60.0	17.4	53.0	24.4	49.6	31.2	48.4	38.0	46.3
個別指示+ 補助観点	12.6	78.8	23.0	70.1	32.8	66.7	42.4	65.8	52.0	63.4
基本観点+ 補助観点	6	37.5	7.6	23.2	8.0	16.3	8.0	12.4	8.0	9.8

図 6.8：評価結果 2

この想定範囲における評価 2 の結果から，個別指示における協調動作の実現では，協調動作数が増加するに従って定義箇所が大きく増加し続ける傾向があるのに対して，観点指示では，指示に用いる観点が様々な機器や協調動作で共通して使用できるため，ある一定数で落ち着く傾向があることが分かる．

6.3.3　評価 3　観点指示による定義量

観点指示による協調動作の実現に必要な定義量について計算する．

	協調動作数:1	協調動作数:2	協調動作数:3	協調動作数:4	協調動作数:5
協調動作設定ファイル	1	2	3	4	5
機器設定ファイル	4.8	7.4	8.9	10.0	11.0
基本観点数	3.0	3.8	4.0	4.0	4.0
補助観点数	3.0	3.8	4.0	4.0	4.0

図 6.9：評価結果 3

今回の想定範囲における評価 3 の結果から，具体的な数字に関しては本研究の想定条件によって増減してしまうため大きな意味はないが，観点を用いた制御指示を行う際に定義すべき協調動作設定ファイルや基本観点数，補助観点数は，協調動作数に対して大きく増加する傾向はないことが読み取れる．

6.4 考察

評価 1 から評価 3 までの評価項目は，今回実装した範囲や想定した範囲内での評価となっている．しかし，実際においては，様々な動作環境や対象機器のバリエーションが考えられる．ここでは，家庭ごとに動作環境が異なる状況，つまり各家庭によって機器構成が異なり，製造元や制御方式も異なるような状況において考察を行う．

以下のような条件を想定して考察を行う．

1. 個別指示の場合の定義箇所数は，協調動作数と比例関係にある．
2. 個別指示の場合の手儀箇所数は，機器数と比例関係にある．
3. 観点指示の場合の事前条件として，機器設定ファイルが設定されている．

個別指示により実現した場合の見積もり

必要な定義箇所の概算数　＝
（1 機器・1 協調動作あたりの平均定義箇所数）×機器数×協調動作数×動作環境数

個別指示で定義を行った場合，機器の数や協調動作の定義数が増加すれば，それに合わせて定義箇所も大きく増加する．また，今回の想定のように各家庭において動作環境が異なる場合，その環境に合わせた定義がそれぞれ必要となる．

観点指示により実現した場合の見積もり

必要な定義箇所の概算数　＝（1 協調動作あたりの平均観点数）×協調動作数

観点指示で定義を行った場合，同じ観点を持つ機器に対しては共通で利用できるため，機器数による影響は受けない．また，動作環境が異なる場合においても，各機器が持つ機器設定ファイルを基に制御指示を行うため，問題とならない．定義箇所数が依存するのは，必要となる観点数（またはアスペクト数）だけである．

これまでの章の内容を含めて,個別指示と観点指示による協調動作の定義について以下のような特徴が考えられる.

- 協調動作の事前に定義すべき項目に関しては，観点指示による実現方法では，個別指示の場合に比べて観点の定義，機器設定ファイル，協調動作設定ファイルの作成などの追加の定義要素が発生する.

- 協調動作内容の検討に関しては，個別指示による実現では具体的な動作環境や機器構成を含めて検討する必要があるが，観点指示では個別の具体的な動作環境や機器構成については必要最低限の範囲で考慮すれば良い.

- 協調動作のための定義量においては，個別指示では具体的な動作環境や機器構成に合わせて定義する必要があるため，機器数や協調動作数，動作環境によって大きく増加する．しかし，観点指示では同じ観点を持つ機器に対して共通の定義を利用することができるため定義量の大幅な増加は起こり難い.

- 実現できる協調動作内容の粒度に関しては，個別指示で行った場合，具体的な動作環境や機器構成を考慮して詳細に定義することができるため，よりきめ細かい協調動作を定義することができる傾向がある．観点指示では，同じ観点を持つ機器に対して共通の定義で扱うことができるが，その半面，機器個別に特化した指示を与えることが難しい.

- 動作環境適応の面から見ると，個別指示では具体的な動作環境や対象機器に合わせて定義する必要があるが，観点指示においてはその必要がなく，他の動作環境においても同一の定義内容で協調動作を実現することができる.

これらの内容から，個別指示と観点指示の得失をまとめると図 6.10 のような関係になると考えられる.

指示方法	事前定義量	協調動作検討	協調動作定義	協調動作粒度	柔軟性
個別指示	○	△	×	○	×
観点指示	△	○	○	△	○

図 6.10：個別指示と観点指示の得失

第 7 章　まとめ

7.1 まとめ

本研究では，具体的な動作環境や協調動作内容に依存せずに，動作環境や対象機器のバリエーションにより柔軟に対応できる協調方式を提案することを目的とした．

本研究では，機器の持つ特徴をとらえた「観点」利用して協調動作を定義し，動的なアスペクト指向技術を用いて，実行時にその定義に基づいた制御を行うことで，具体的な機器構成や動作環境にとらわれない柔軟な協調動作を行う方式を提案した．具体的には，「観点」を用いた協調動作の実現方式を提案し，Spring を用いてプロトタイプを実装して，例題への適用と評価を行った. そして, 横断的な関心事が存在する協調動作の定義において, 本手法の有効性を確認した.

7.2 今後の課題

7.2.1　適用範囲

本研究において，機器や協調動作についての横断的関心事を協調動作の観点としてとらえ，観点を用いた指示による協調動作の実現及び確認を行った．しかし，実際に観点を用いて協調動作を実現する際には，観点と実現できる協調動作の粒度等をふまえた上で，最適な関係を求める必要がある．個別指示による協調動作は，非常に粒度の細かいサービスを実現できるが，定義量が多くなる傾向がある．観点指示による協調動作は，様々な動作環境や対象機器により少ない定義量で対応することができるが，個別指示に比べて細かな粒度の協調動作は扱い難い．この辺りのトレードオフの関係を踏まえて, 最適な使用方法や適用すべき協調動作などについて議論する必要があると考えられる.

7.2.2　動作環境

本研究において，実際に作成した例題では Java VM 上で Spring を用いて，家庭内の家電機器を想定したプログラムに対して観点による協調動作の実現を行った．しかし，CORBA 等の共通ミドルウェア上に構築することで，Java 言語にとらわれず，様々な言語で構成された動作環境においても，観点指示による協調動作が実現できるのではないかと考えられる．観点指示による制御方式のメリットを生かすため, より適した動作環境についても議論の余地があると思われる.

第 8 章　終わりに

本論文を執筆するに当たって多大なるご指導を賜りました，岸知二客員教授，Defago Xavier 准教授，青木利晃准教授に感謝申し上げます．また，本研究を進めるに当たって，様々な議論に応じてくださった，岸研究室，Defago 研究室，青木研究室の皆様に感謝いたします．

参考文献

[1] Spring website http://www.springframework.org/.

[2] ECHONET CONSORTIUM website http://www.echonet.gr.jp/.

[3] 井垣 宏, 中村 匡秀, 玉田 春昭, 松本 健一, サービス指向アーキテクチャを用いたネットワーク家電連携サービスの開発, 情報処理学会論文誌, Vol46 NO.2 P314-P326, 2005

[4] 井垣 宏, 串戸 洋平, 石井 健一, 中村 匡秀, 松本 健一, 家電連携機器サービスにおけるサービス競合の検出, 電子情報通信学会 信学技報, P69-P74, 2004

[5] 沢田 篤史, 多鹿 陽介, 山崎 達也, 美濃 導彦, ゆかりコア : ネットワーク家電のための分散協調型サービス基盤構築, 電子情報通信学会 信学技報 SS2004-9 P19-P24

[6] Robert E. Filman, Tzilla Elrad, Siobhan Clarke, Mehmet Aksit, Aspect-Oriented Software Development, Addison-Wesley Professional, 2004.

[7] 千葉 滋, アスペクト指向入門, 技術評論社, 2005.

[8] 長谷川 裕一, 伊藤 清人, 岩永 寿来, 大野 渉, Spring 入門, 技術評論社, 2005.

[9] 長谷川 裕一, 伊藤 清人, 岩永 寿来, 大野 渉, Spring2.0 入門, 技術評論社, 2007.