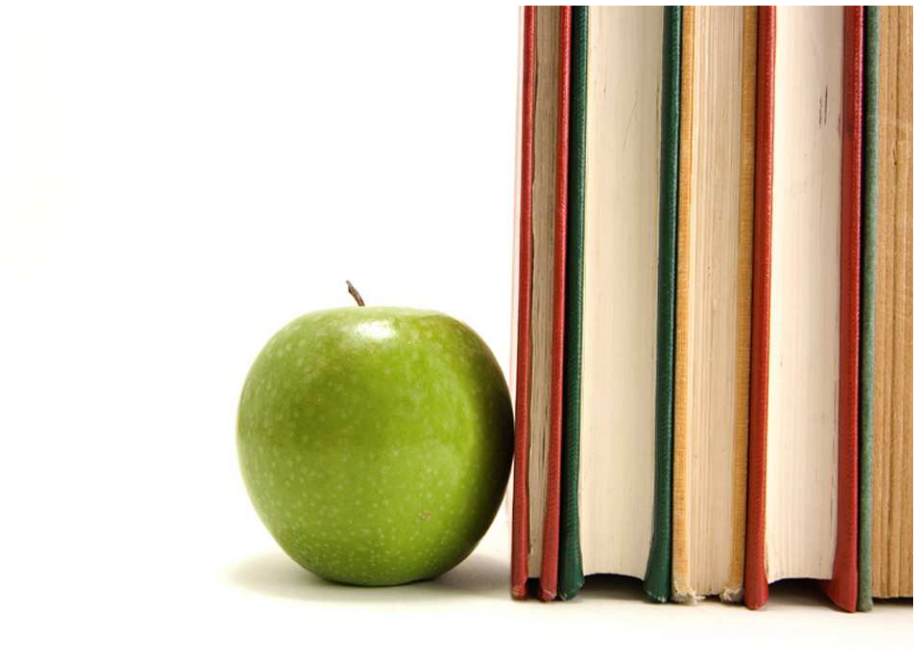
組込み分野のための
UML モデルカタログ



部品編

目標制御

本書は、UML モデルカタログに含まれる「目標制御」のモデルの詳細を記述したものです。モデリングの初心者には教科書や参考書として、モデリングのベテランの方々にはモデルのヒントとして、ぜひともお手元に置いて活用してください。

UMTP 組込み
モデリング部会

2012.4.15 更新

目　次

はじめに 2
要求仕様 2
モデル一覧 12
エンティティに着目したモデル 14
　分析モデル 14
　PIM 設計モデル 18
参考文献 26
付録：各種制御方式の詳細 27

はじめに

目標制御は、特定の制御対象・システムに対して式やテーブルといった最適化が施された形態で実装されることが多くあります。このモデルは、目標制御の実装形態はブラックボックスとし、その中身に依存しない形にしました。このモデル（部品編目標制御）では、登場するクラス数が少なく、モデルの全体と対象物が把握しやすいものとなっています。

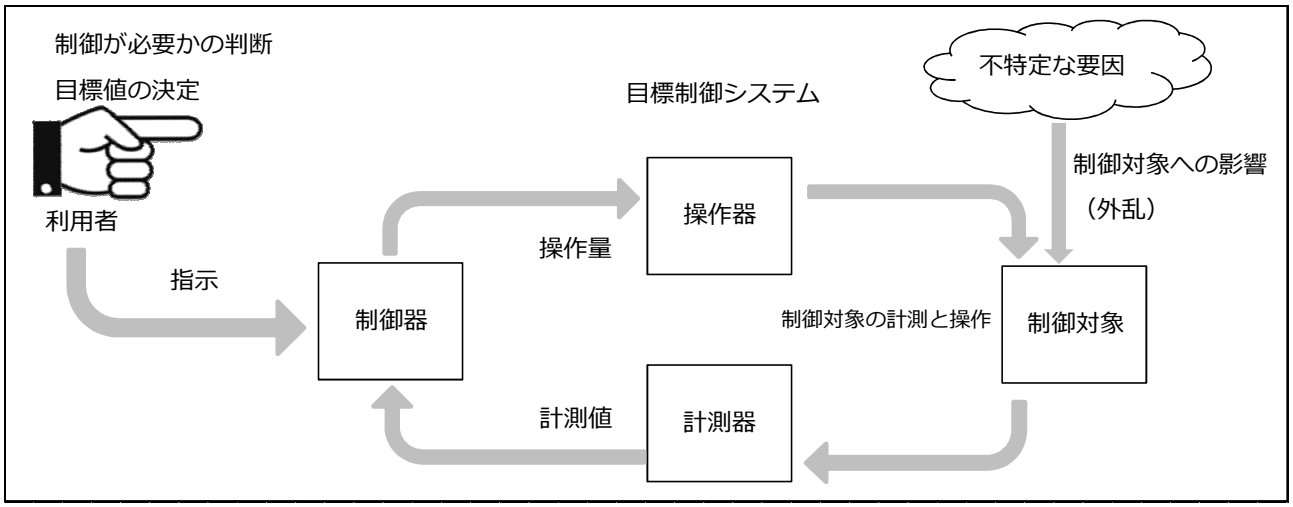
要求仕様

目標制御システムとは、制御対象の計測値が目標値となるように制御する仕組みで、エアコンの温度調節や自動車の ABS など数多くの製品に組み込まれています。

目標制御システムの基本的な構成と動作

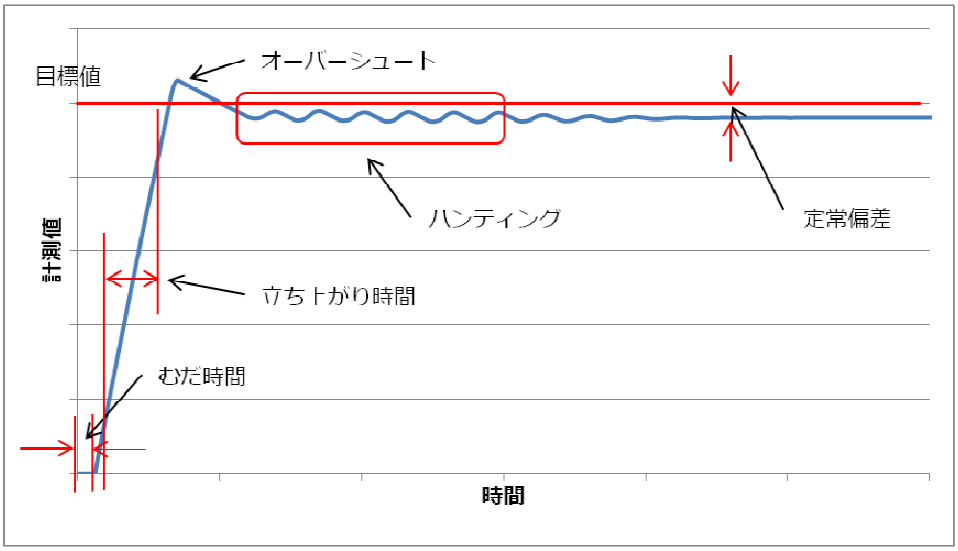
目標制御システムは計測器、操作器、制御器で構成されます。利用者は制御が必要な状況で、目標制御システムに制御の開始を指示します。目標制御システムは、制御中は一定時間周期で制御対象を計測し、それに応じた操作を行うというフィードバックを繰り返します。

制御対象は目標制御システム以外の不特定な要因からも影響をうけますが、その都度出力を修正することで、フィードバックを繰り返すことで、うまく制御を行えるようになっています。



目標制御の実例

目標制御を実行した場合に、制御対象の計測値がどのように目標値に近づいていくかを説明します。



目標制御を開始すると、操作器の操作量が制御対象に伝わるまでの時間（むだ時間）を経過した後、計測値は目標値に近づいていきます。このときの目標値に近づくまでの時間が立ち上がり時間です。計測値が目標値に達したときに、勢いがついていたりすると、オーバーシュートが発生します。その後、ハンティングをしながら、徐々に計測値は目標値付近で安定するようになります。最終的に安定したときの計測値と目標値の差が定常偏差として残ることになります。

目標制御を実現するアルゴリズムには、ON/OFF 制御、PID 制御、ファジィ制御などの制御方式があり、制御方式によって、目標値への近づき方の特徴に違いがあります。注：これらの制御方式については付録を参照してください。

用語定義

すでにいくつかの用語を使用していますが、あらためて用語の定義を行います。しかし、それだけではイメージがつきにくいと思いますから、具体的な製品（例としてエアコンと ABS）で、何に該当するかもあわせて以下に示します。

用語	用語の意味	エアコンの場合	ABS の場合
目標制御システム	制御対象を制御するための仕組み	−	−
制御対象	目標制御を行う対象	室温	車輪速度
目標値	目標とする制御対象の値	指定された室温の値	指定された車輪速度の値
計測値	計測された制御対象の値	現在の室温の値	現在の車輪速度の値
外乱	制御対象に加えられる予測できない影響	室外との熱のやり取り、等	路面との摩擦、等
制御器	目標制御を司る部品	−	−
入力器	利用者からの指定内容や指示を受け付ける部品	入力パネル	ブレーキペダル
計測器	制御対象を計測する部品	室温センサ	車輪速度センサ
操作器	何からのアクチュエータを駆動させることで、制御対象を操作する部品	コンプレッサー	油圧サーボ
操作量	操作器がもつ指定可能な量で、結果的に制御対象を操作する量	コンプレッサー出力	油圧サーボ出力
制御方式	目標制御を実現する方式、アルゴリズム	−	−

本カタログでの目標制御システムの前提条件

本カタログで検討する目標制御システムは、特定の製品ではなく、どの製品でも共通となる抽象的な目標制御システムとして検討しています。具体的な製品は、検討したモデルを継承・拡張して開発するイメージとなります。

しかし、製品ごとにさまざまな目標制御の実現方法があり、純粋に共通部分のみを扱うことは困難であるため、本カタログで検討する目標制御システムには、以下の前提条件を設定します。

1. 検討する制御方式について

   閉ループ制御を対象とし、デフォルトの制御方式をＰＩＤ制御として検討します。

2. 一定周期で動作させる仕組みについて

一定周期で動作させるためにタイマーを使用します。タイマーはシステムが起動したタイミングで開始します。

3. 目標制御の終了条件について

   制御対象を目標値に留めておきたい場合があるため、利用者が目標制御の終了を指示するまで、目標値に到達しても制御を続けます。

4. 目標値やその他の制御項目の変更タイミングについて

   任意のタイミングで目標値やその他の制御項目を変更できますが、変更された値が実際に使用されるタイミングは、次回の制御時となります。

ユースケース

本カタログモデルが実現する範囲のユースケースを示します。

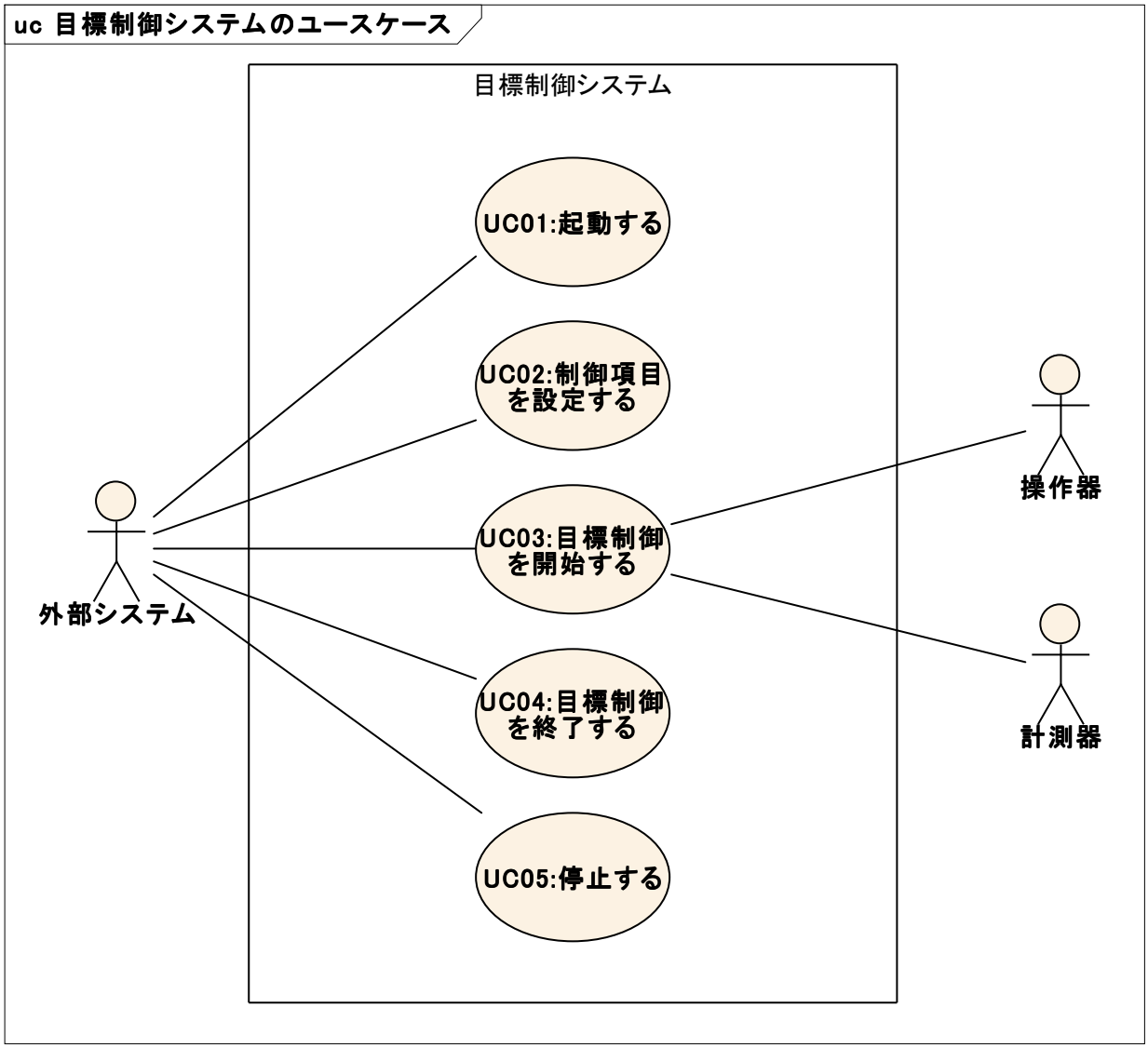


図1

補足：アクター「外部システム」とは、最終的な製品（エアコン等）に含まれるプログラム（目標制御システムを使用するプログラム）を指します。

ユースケース一覧

No.	ユースケース名	概要／備考
UC01	起動する	目標制御システムを起動する。
UC02	制御項目を設定する	目標値やその他の制御項目を変更する。 操作量の計算途中にも制御項目を変更できます。
UC03	目標制御を開始する	制御対象の計測値が目標値となるように制御する。 一定時間周期で制御対象の計測と操作を繰り返すことで、 制御対象は徐々に目標値に近づいていく。
UC04	目標制御を終了する	制御対象の操作を止める。
UC05	停止する	目標制御システムを停止する。

ユースケースの実行順とシステムの処理

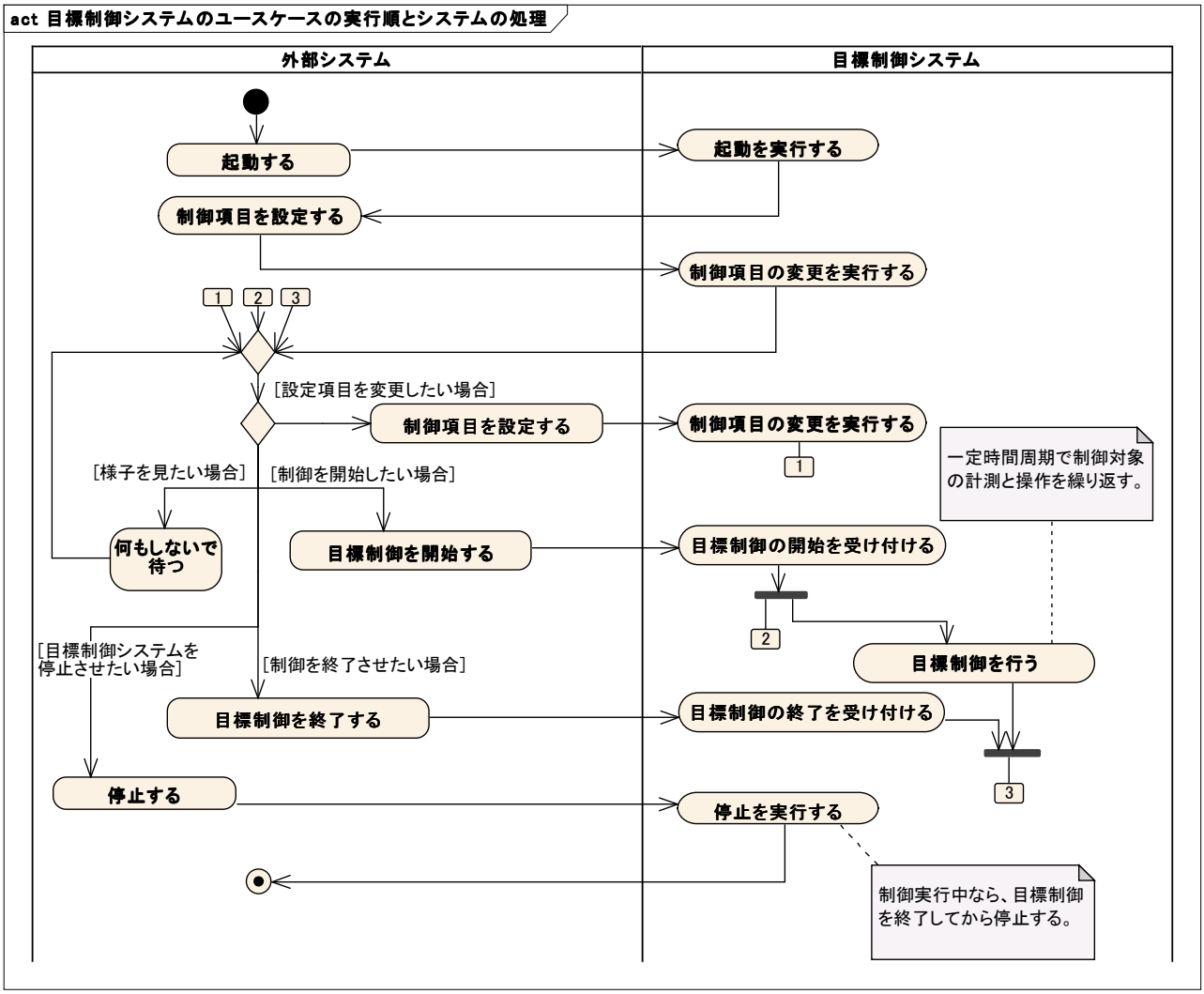


図2

ユースケース記述

<UC01：起動する>

■概要

目標制御システムを起動する。

■アクター

外部システム

■事前条件

・目標制御システムが停止している。

■事後条件

・目標制御システムが起動している。

■メインフロー

1. アクターは、システムの起動を要求する。
2. システムは、起動する。
3. システムは、アクターに起動したことを通知する。
4. UC を終了する。

■代替フロー

なし

■課題やＴ.B.D 項目

なし

■備考

なし

<UC02：制御項目を設定する>

■概要

制御項目を設定する。

■アクター

外部システム

■事前条件

・目標制御システムが起動している。

■事後条件

・制御項目が設定されている。

■メインフロー

1. アクターは、制御項目の設定を要求する。
2. システムは、制御項目を設定する。
3. システムは、アクターに制御項目の値を設定したことを通知する。
4. UC を終了する。

■代替フロー

なし

■課題やＴ.B.D 項目

なし

■備考

操作量の計算途中にも制御項目の値を変更できますが、変更した値を計算に反映させるタイミングを計算前に限定することで、計算途中の値が書き換わることを防ぎます。※<UC03：目標制御を開始する>の備考を参照。

<UC03：目標制御を開始する>

■概要

制御対象の計測値が目標値となるように制御する。

■アクター

外部システム

■事前条件

・制御項目が設定されている。
・目標制御が行われていない。

■事後条件

・目標制御が行われている。

■メインフロー

1. アクターは、目標制御の開始を要求する。
2. システムは、制御の周期処理を開始する。
3. システムは、アクターに目標制御を開始したことを通知する。
4. UC を終了する。

■代替フロー

なし

■課題やＴ.B.D 項目

なし

■備考

目標制御が開始されると、システムは以下を行い、制御対象を目標値に近づけていく。

1. システムは、<UC02：制御項目を設定する>で設定された制御項目の値を取得する。
2. システムは、計測器を使用し、制御対象を計測する。
3. システムは、計測値と目標値を比較し、制御対象が目標値に近づくような操作量を算出する。
4. システムは、操作量に従って、操作器を使用し、制御対象を操作する。
5. 一定時間経過後、1 に戻る。

※この処理はアクターが操作を別の操作を行っているときも、それに平行して行う。この処理は<UC04：目標制御を終了する>が行われるまで続ける。

<UC04：目標制御を終了する>

■概要

目標制御を終了する。

■アクター

外部システム

■事前条件

・目標制御が行われている。

■事後条件

・目標制御が行われてない。

■メインフロー

1. アクターは、目標制御の終了を要求する。
2. システムは、制御の周期処理を終了する。
3. システムは、操作器の操作を止める。
4. システムは、アクターに目標制御を終了したことを通知する。
5. UC を終了する。

■代替フロー

なし

■課題やＴ.B.D 項目

なし

■備考

なし

<UC05：停止する>

■概要

目標制御システムを停止する。

■アクター

外部システム

■事前条件

・目標制御システムが起動している。
・目標制御が行われていない。

■事後条件

・目標制御システムが停止している。

■メインフロー

1. アクターは、システムの停止を要求する。
2. システムは、停止する。
3. システムは、アクターに停止したことを通知する。
4. UC を終了する。

■代替フロー

なし

■課題やＴ.B.D 項目

なし

■備考

なし

モデル一覧

モデル名	概要	ポイント
機能に着目したモデル	なし	
エンティティに着目したモデル	目標制御システムを構成する部品をエンティティとして捕らえたモデルです。	部品を中心としたシンプルなモデルです。
状態に着目したモデル	なし	
メタファを使ったモデル	なし	

エンティティに着目したモデル

モデリングのコンセプト

静的な構造として目標制御を行うのに必要な構成部品に着目しました。

また、要求仕様に登場する ON/OFF 制御、PID 制御、ファジィ制御といった複数の制御方式に対応できるように考慮しました。

分析モデル

１．制御対象および計測、操作について

実際の制御対象（エアコンであれば室温で、ABS なら車輪速度です）の今の状態の保持やそれに関する情報を保持するクラスとして「制御対象クラス」を定義しました。制御対象クラスは属性として目標値と計測値を持ちます。なお、「計測値」属性は実際には操作であり、操作を呼ぶと計測器クラスの「計測する」操作を呼び出します。

制御対象には一般的に、制御対象特有の上限値・下限値、あるいは計測器が持つセンサからくる上限値・下限値（計測限界）が存在します。そのため、制御対象クラスは範囲クラスを保持する形となります。

２．制御方式について

制御器が制御を行う際、どのように制御するか？（制御対象への出力をどうするか？）（例えば計測値が目標値に近づいてきたから、出力を弱めるといったこと）を考える必要があります。これを制御方式クラスが行います。制御方式には ON/OFF 制御、PID 制御、ファジィ制御といろいろな方法がありますから、制御方式クラスは抽象クラスとなります（ストラテジーパターン）。制御方式クラスが操作量を算出する際には、元となるデータ（制御パラメータ）が必要となりますので、制御方式クラスは制御パラメータクラスを保持することになります。

制御方式クラスでは操作量算出時に必要な内部データ（PID 制御であれば例えば、偏差（目標値と現在値の差）の時間積分値）が必要ですが、これは実際の制御方式に依存するため、ここでは抽出していません。制御パラメータが持つデータについても同様で、PID 制御であれば比例係数、積分係数、微分係数の 3 つを持つことになりますが、これも制御方式に依存するものですのでここでは抽出していません。注：PID 制御の詳細については付録を参照してください。

３．定期的な制御について

制御は定期的に行う必要があるためタイマーが必要となりますから、それを定義しました。分析の動的モデルではタイマーが目標制御システムで定義される操作を直接呼び出す形としますが、本来は、一般にタイマーは目標制御部品とは別の部品であり、タイマーが目標制御システムで定義している操作を直接呼び出すことはできません。これについては PIM で検討します。

４．その他

制御対象を実際に制御する場合、その良し悪し、具体的には立ち上がり時間、オーバーシュート量、ハンティング量、定常偏差の量などを、考える必要があります。しかし、これについては制御方式クラスのサブクラスが考えることであるため、クラス構成を抽出する際には考慮対象外としました。

静的モデル

目標制御システムのクラス構造

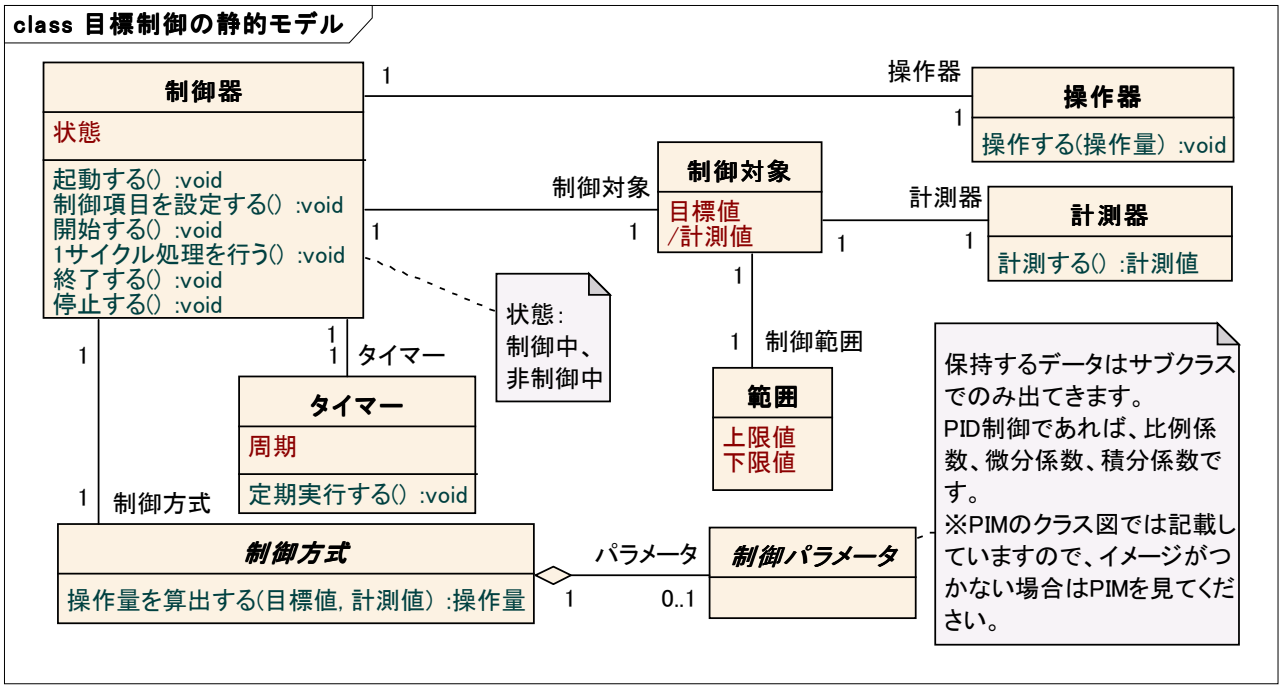


図3

補足：基本的には、属性の値を取得、設定する操作は省略してあります。制御器クラスの操作についてはユースケースにあるものは記載しています。

- 制御器クラス：
  外部システムが実際に使用するクラスとなります。分析のクラス図では記載を省略していますが、このクラスは制御項目（目標値、制御パラメータ、リスナー、制御範囲）の設定と取得を行う操作と、制御の開始や終了を行う操作を持ちます。また、タイマーを起点としての制御（計測、算出、操作という一連の動作）も行います。
- 制御対象クラス：
  制御対象に関するデータを持つクラスで、目標値と計測値を属性として持ちます。
  すでに述べましたが、派生属性「計測値」は実際には操作であり、これを呼ぶと、計測器の「計測する」操作が呼び出されます。
- 計測器クラス：
  実際の制御対象に対して計測を行うクラスです。実際にはドライバ等になります。
- 操作器クラス：
  実際の制御対象に対して操作を行うクラスです。実際にはドライバ等になります。
- 範囲クラス：
  一般に制御対象には制御できる範囲があります。この範囲は制御対象の特性（水なら 0～100℃）や計測器の特性（計測できる範囲）によります。その範囲外になった場合には一般に故障であり、それを検出するためにあります。
- タイマークラス：
  定期的な制御を行う際の起点となるクラスです。
- 制御方式クラス：
  目標値、計測値、および、制御パラメータから操作量を算出するクラスです。

◆ 制御パラメータクラス：
制御方式が使用する制御パラメータを保持します。ただし、実際のデータは制御方式に依存するため、このクラス自身はデータを持ちません。
※これだけだとクラス存在意味が分かり難いのですが、例えば、制御方式クラスの操作として「制御パラメータを設定する」というのが必要になったときに、引数の型としてこのクラスが必要となります。このように書くと、うまく設計するために出てきたクラスのように思えますが、PID 制御の比例係数、積分係数、微分係数のように目標制御の一般的な概念として、制御パラメータというのがありますので、抽出しています。

動的モデル

目標制御システムの状態遷移

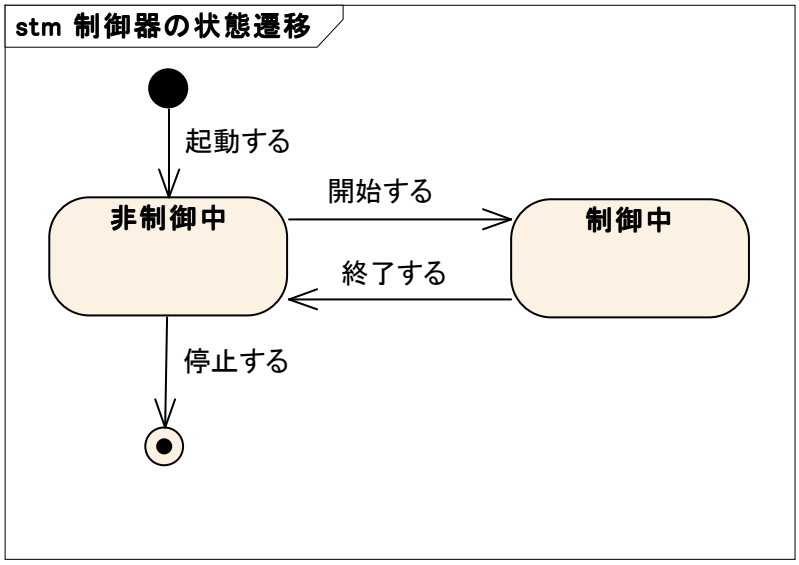


図4

「開始する」の相互作用

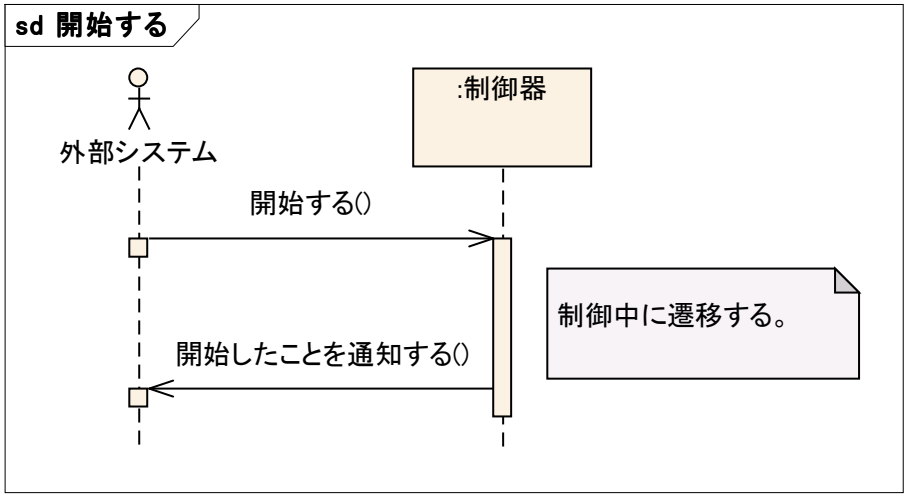


図5

「制御中」の相互作用

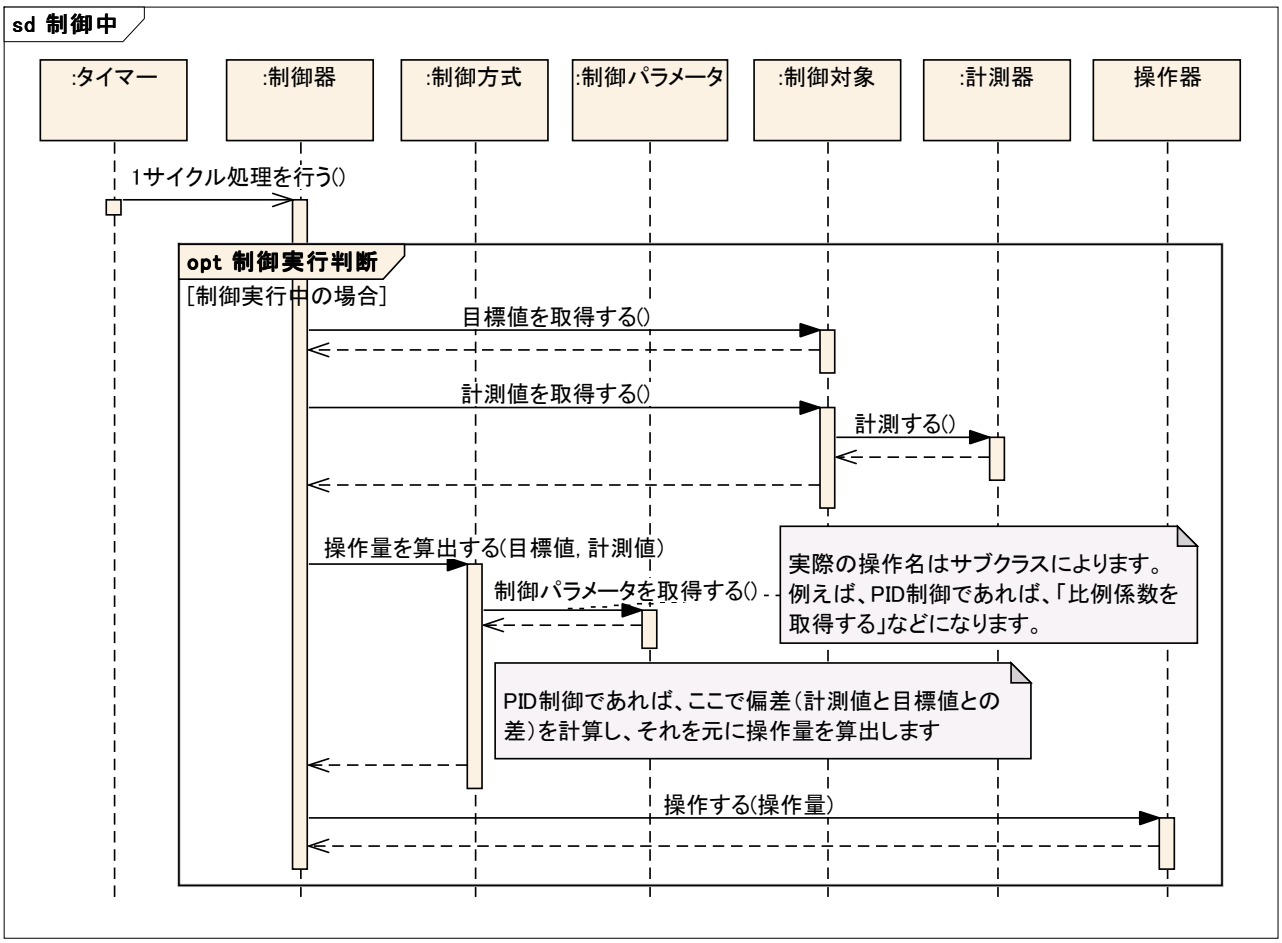


図6

補足：制御対象の派生属性「計測値」の取得は「計測値を取得する」操作として記載しています。

「終了する」の相互作用

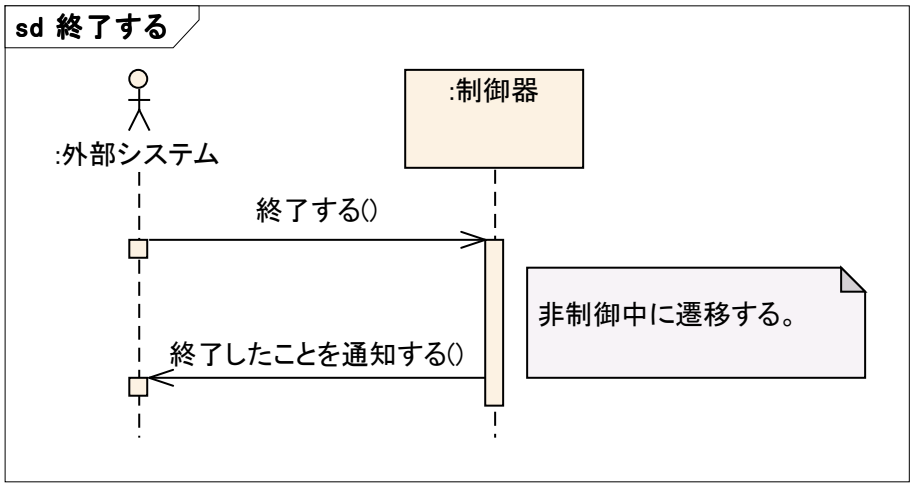


図7

PIM 設計モデル

分析との違いについて説明します。

１．制御対象および計測、操作について

分析では計測値が範囲を超えた場合にどうするかを考慮していませんでしたが、ここでは考慮します。方法としては、計測値をチェックし、範囲外になったらリスナーを使用して通知するようにしました（オブザーバパターン）。これにより外部システムは制御を中止したり、他のこと（管理者に通知を行うなど）を行ったりといったができます。

２．制御方式について

制御方式クラスと制御パラメータクラスは分析では抽象クラスのみ記載しましたが、制御するには当然、実際のクラスが必要となりますから、ON/OFF 制御、PID 制御、ファジィ制御の 3 つについてサブクラスも記載してあります。注：これらの制御方式については付録を参照してください。

３．定期的な制御について

分析ではタイマーが目標制御システムの持つ操作を直接使用していましたが、本来このようなことはできません。ここでは、タイマータスクという抽象クラスを定義し、制御器クラスがそのサブクラスとなることで、タイマーが間接的に目標制御システムの持つ操作を使用する形としました（オブザーバパターン）

分析ではスレッドについて何も触れていませんでしたが、制御はそれを利用する外部システムとは別のスレッドで行うこととしました（※スレッドのスケジューリングポリシーは OS に依存するので、それには依存しない作りにしました）。目標値、制御パラメータ、リスナー、制御範囲の変更は制御を行っている間でも可能とするため（複数のスレッドから使用されるため）、ミューテックスあるいはアトミック操作による排他を意識しました。

スレッドのスケジューリングポリシーや優先度によっては制御が定期的に行われない（若干ずれる）ため、計測を行った後で現在時刻を取得し、計測時刻とすることで操作量を算出する際に時間を補正できるようにしました。

静的モデル

目標制御のクラス構造

1 つのクラス図では収まらないため、制御方式に依存しない部分と、制御方式に依存する部分でクラス図を分けました。

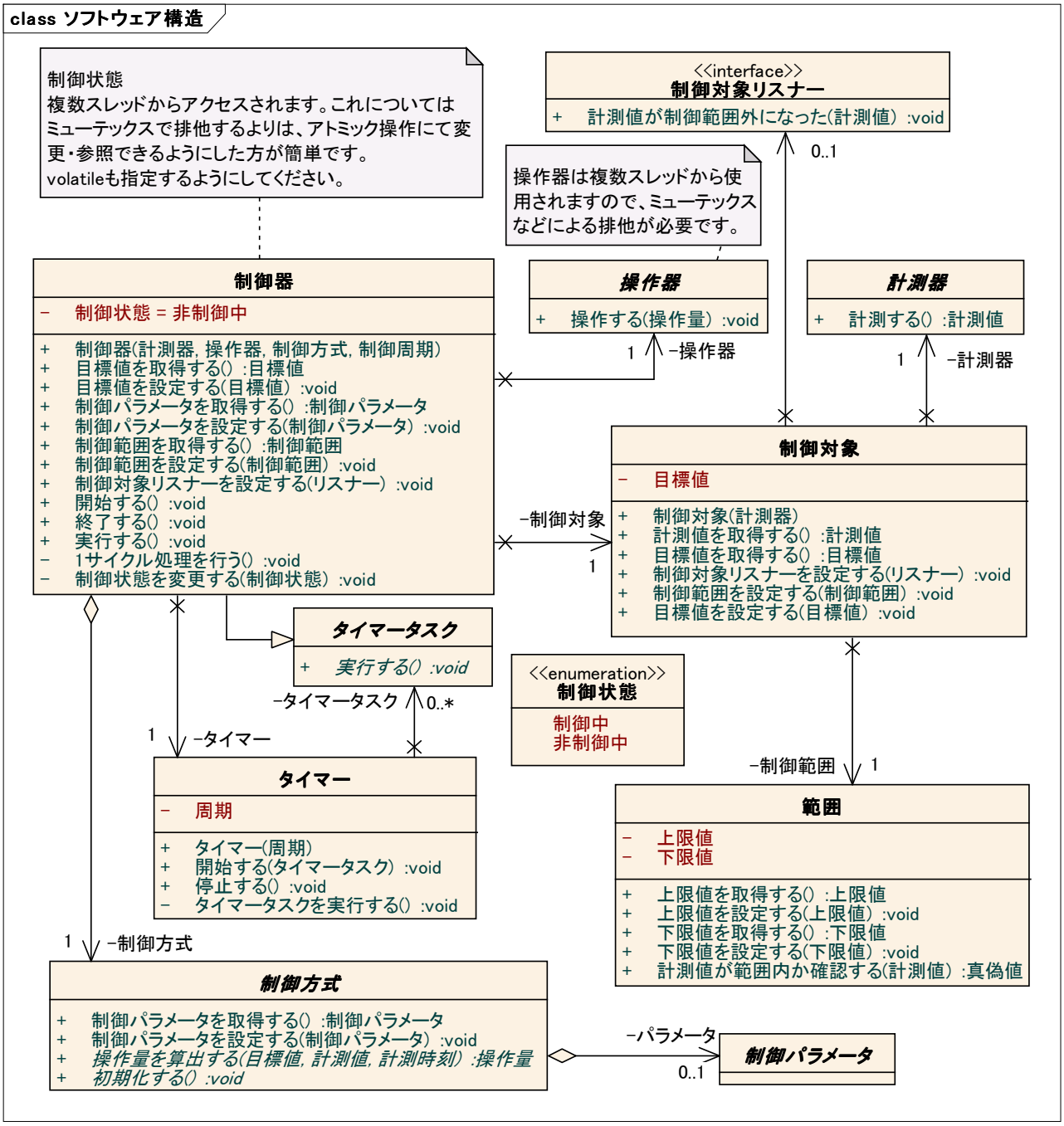


図8

補足：コンストラクタは引数が必要なもののみ記載しています。

以下、分析との違いについて説明します。

◆ 制御器クラス：
「実行する」操作を追加しています。この操作は「タイマータスク」の「実行する」操作をオーバライドしたものであり、外部システムが使用する操作ではありません。

◆ 制御対象リスナー：
制御対象が異常を検知したというイベントを外部システムが分かるようにするためのインタフェースです。外部システムはこのインタフェースのサブクラスを定義し、それを制御器クラスの「制御対象リスナーを設定する」操作を使用することで、イベントを受け取ることができます。

◆ タイマー、タイマータスク：
タイマーは一般に別の部品となりますので、タイマーが制御器を直接持つことはできません。そのため、制御器をタイマータスクのサブクラスとし、それをタイマーに設定することで、制御器クラスの操作を定期的に実行できるようにしています。

◆ 制御方式：
「初期化する」操作を追加しています。制御中→非制御中→制御中→・・・と、制御の開始と終了を繰り返す場合、制御を再実行するときに内部データを初期化できるようにするためです。

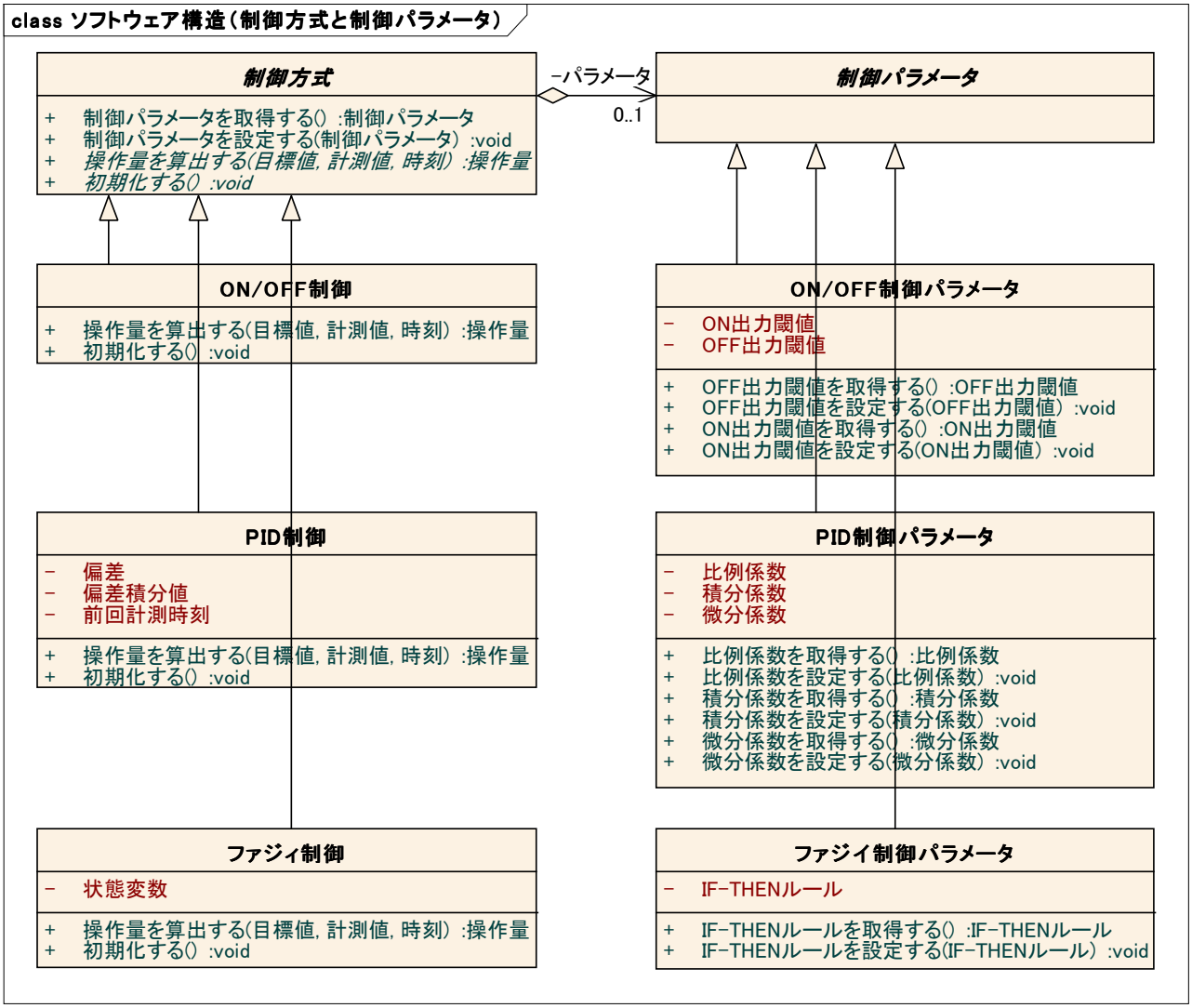


図9

◆ 制御方式の各サブクラス、制御パラメータの各サブクラス：
ここでは例として ON/OFF 制御、PID 制御、ファジィ制御の 3 つについて、必要となる属性と操作を定義しました。なお、動的モデルでは PID 制御を使用する場合で記載しています。注：これらの制御方式については付録を参照してください。

動的モデル

「起動する」の相互作用

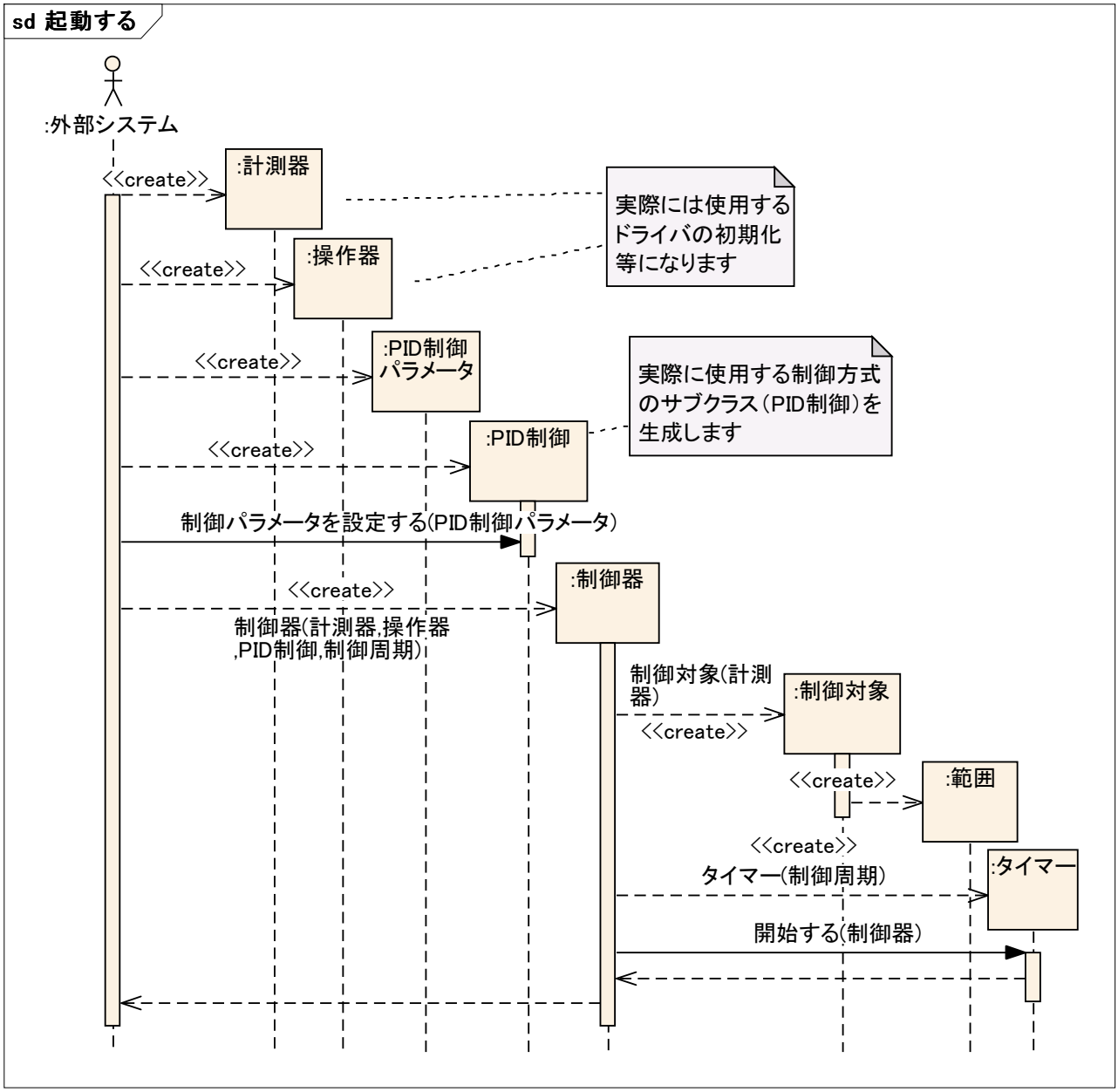


図10

補足：制御器はもちろんですが、計測器、操作器、制御方式、制御パラメータの４つのクラスのサブクラスについてもオブジェクトの生成は外部システムに行ってもらいます。こうすることにより、目標制御が実際に使用するサブクラスに依存することなく、制御を行うことができます。

「開始する」の相互作用

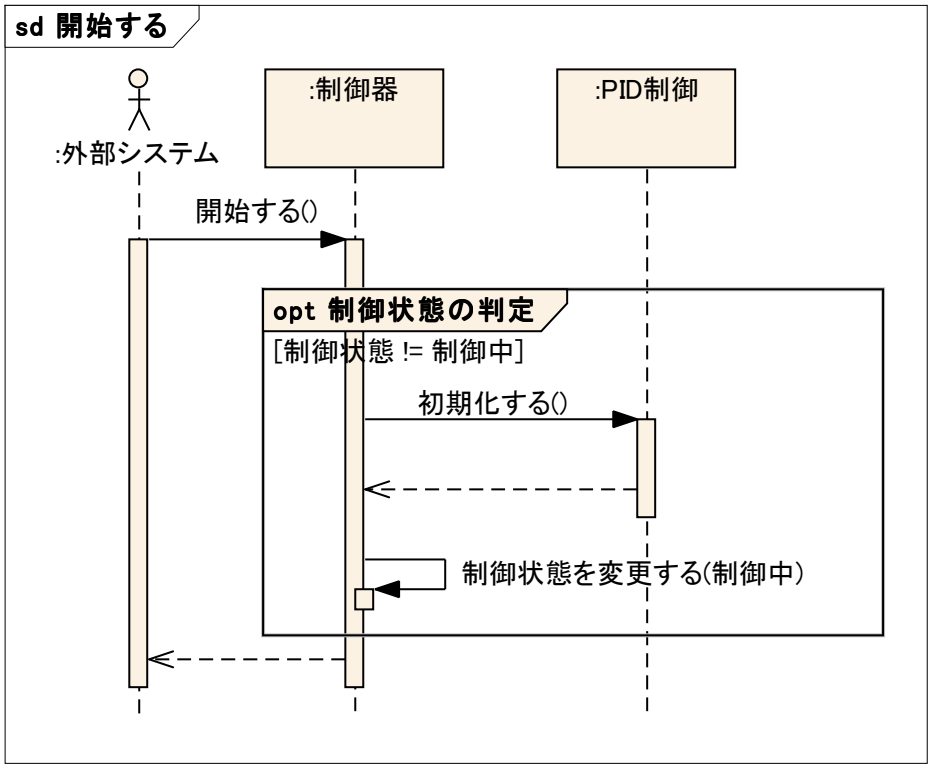


図11

補足：初期化を行うのも分析モデルからとの違いになります。

「制御中」の相互作用

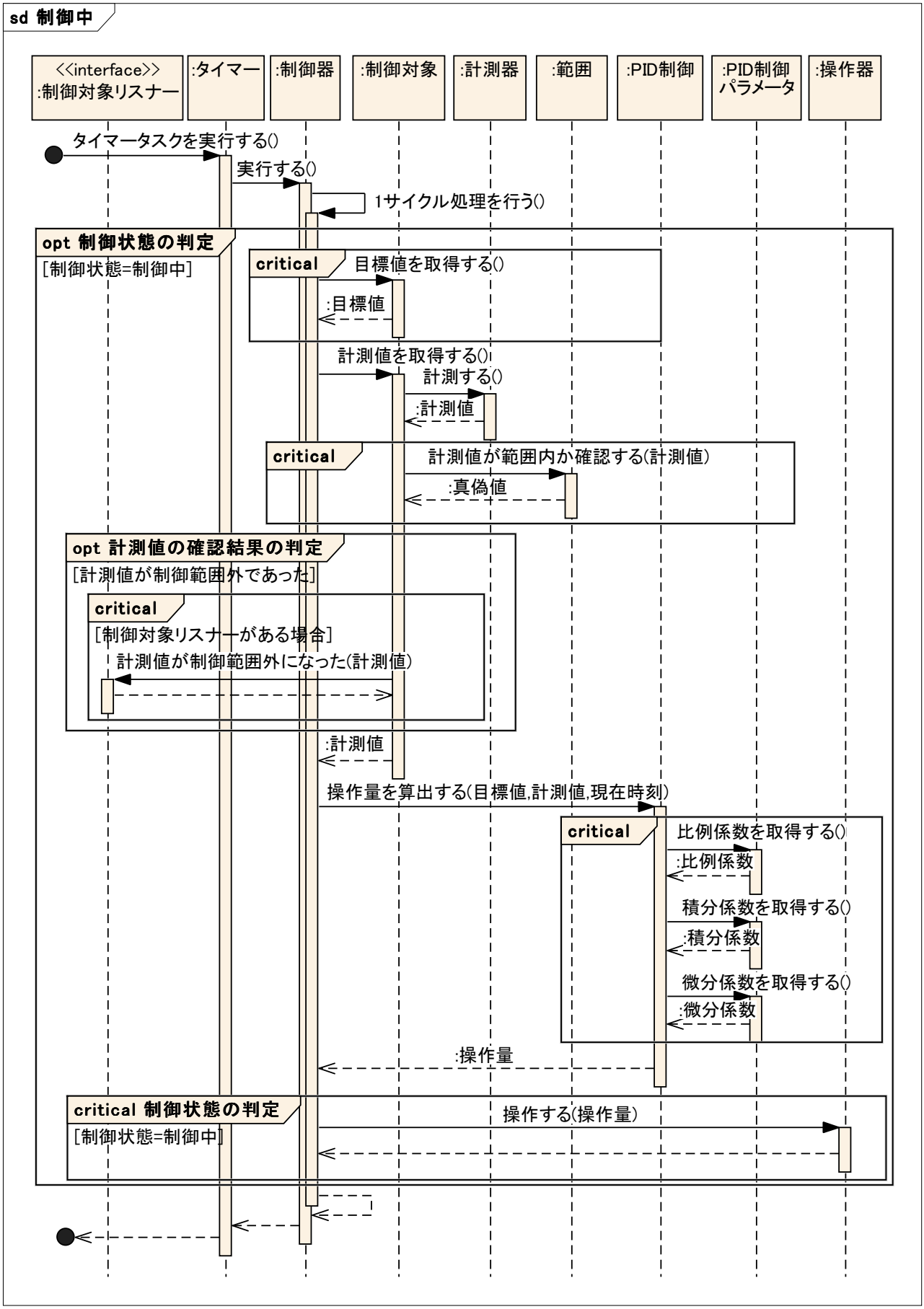


図12

計測してから、操作するまでの間に、目標制御の終了が要求された場合やリスナーの操作の中で終了要求された場合のために「操作する()」を実行する直前に再度、制御状態を確認します。

制御器クラスが変更する操作を持っているデータ（目標値など）を使用する場合にはミューテックス等による排他が必要ですので、critical で囲っています。

「終了する」の相互作用

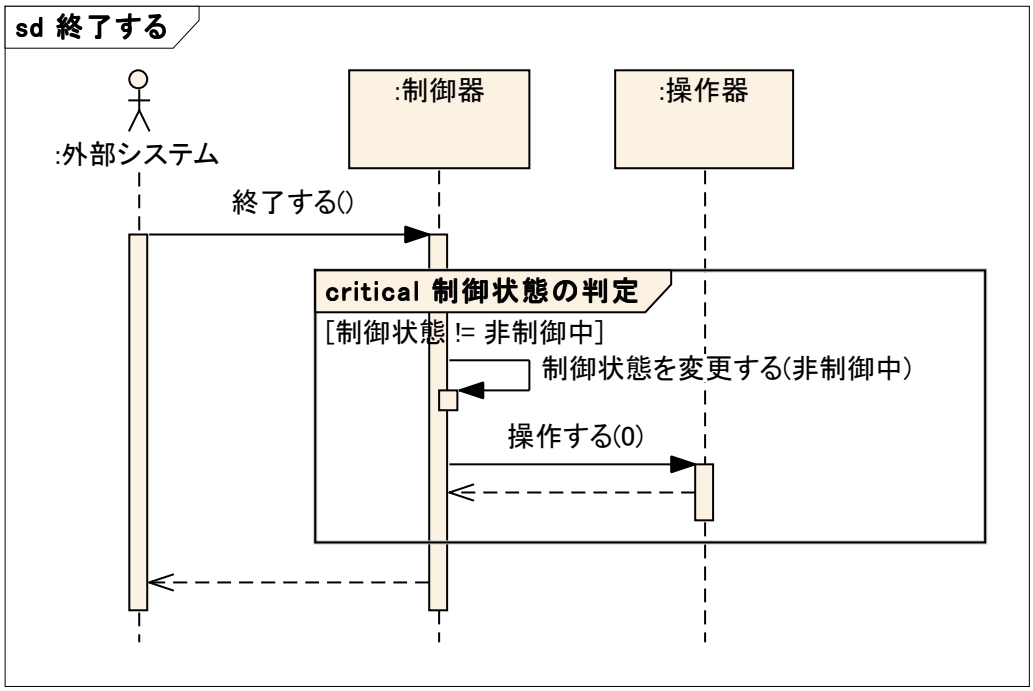


図13

目標制御を終了する際には、操作量を 0 とし、操作を止めます。

「目標値を変更する」の相互作用

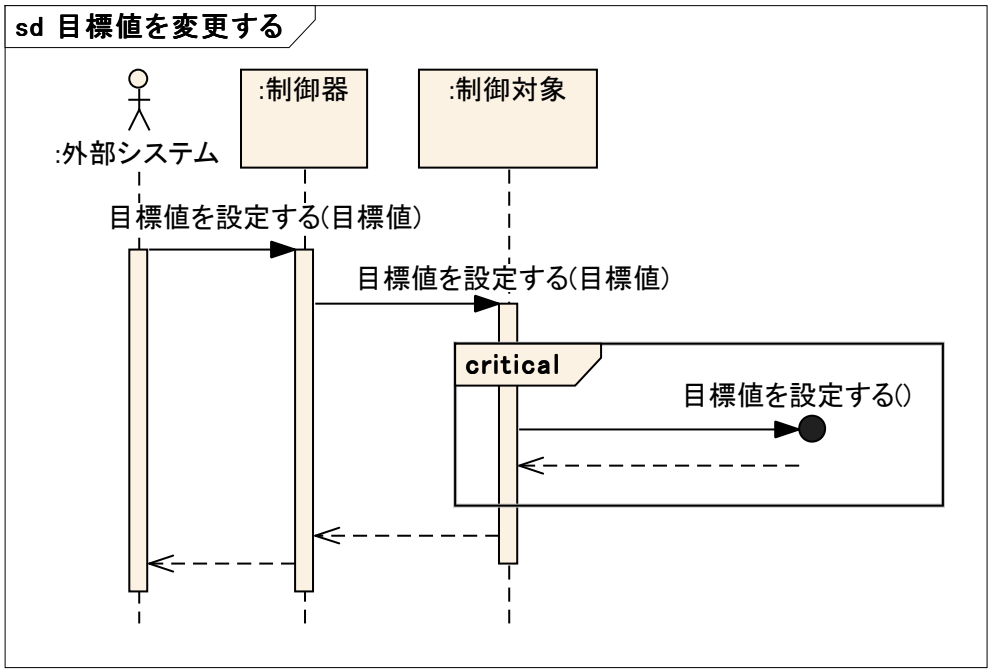


図14

「停止する」の相互作用

停止する場合は、起動とは逆の順でタイマーやオブジェクトを停止、破棄を行います。

参考文献

◆『ＰＩＤ制御の基礎と応用』山本重彦・加藤尚武著、朝倉書店、1997

◆『基礎システム制御工学』土谷武士・江上正著、北森出版、2001

付録：各種制御方式の詳細

ON/OFF 制御

算出される操作量が 0%か 100%のいずれかとなるような制御です。操作量を 100%とする計測値（ON 出力閾値）と 0%とする計測値（OFF 出力閾値）を設け、操作量を切り替えて制御していくといった方法がとられます。

この方式では必ずハンティングを起こすというデメリットがあります。しかし、例えば温度管理の場合で、制御対象の熱容量が大きい場合にはハンティングの振幅は小さくなりますし、他の制御方式と比べ安価でコストもかからず、ハード制御のみで可能というメリットもあるため、非常によく使われる制御方式です。身近にある例としてはコタツの温度管理がそうです。

PID 制御

最も使用されている制御方式です。P、I、D はそれぞれ、Proportional（比例）、Integral（積分）、Differential（微分）の略であり、偏差（目標値と現在値の差）、偏差の時間積分、偏差の時間微分のそれぞれに定数を乗じたものを足し合わせたものを操作量とする方式です。このときの定数をそれぞれ、比例係数、積分係数、微分係数と言います。

この方法では比例係数、積分係数、微分係数の 3 つの値が適切であれば、ON/OFF 制御のデメリットにあるようなハンティングはほとんど起こしません。ただし、実際の制御というのは難しいことが多く（例えば、むだ時間が長い、制御対象の特性や外乱の大きさが制御中に変化するなど）、PID 単独での使用というよりも他の制御方式と組み合わせて使用するといった方法がより一般的です。

ファジィ制御

ファジィ推論演算を行って，操作量を決定する制御です。今まで人の手で行っていた制御を自動化したいときに使用される制御方式です。

この制御方式のデメリットとしては操作量の算出に時間がかかることがあげられます。しかし、人の手で行っていた制御を自動化できるというメリットがあり、よく使用される制御方式です。

この制御方式も PID 制御と同じように他の制御方式と組み合わせて使用されることも多いです。

組込み分野のための UML モデルカタログ
「目標制御」

初版発行 ２０１２年（平成２４年）３月３０日
発行者　UMTP, Japan
編　著　組込みモデリング部会
印刷

UMTP, Japan
東京都渋谷区代々木 1 丁目 22 番 1 号
http://www.umtp-japan.org/