制御ソフトウェア開発のための参照モデル

非会員 池田 信之 ((株)東芝 SI技術開発センター)
正員 早瀬 健夫 ((株)東芝 SI技術開発センター)
非会員 松本 一教 ((株)東芝 SI技術開発センター)
非会員 守安 隆 ((株)東芝 SI技術開発センター)

A Reference Model for Control Software Development

Ikeda Nobuyuki, Non-member, Hayase Takeo, Member, Kazunori Matsumoto, Non-member, Takashi Moriyasu, Non-member (System Integration Technology Center, TOSHIBA Corp.)

To build and apply framework is a practical way to improve reusability of software products and to realize effective system development. A well-designed framework depends on correct estimation of the requirements for software modification in its system domain. In this paper, we propose "Device-Control-Constraint Model" as a reference model to build effective frameworks in control system domain. We design the reference model by assuming that the main requirements of system modification are physical device interface, device composition, and control algorithm. By our reference model, frameworks are composed of three sorts of objects. They are Device objects, Control objects, and Constraint objects. Each object is designed to deal with different requirements of modification. We discuss the design of Device-Control-Constraint model and show an example of framework construction in air-conditioner control domain by using our model.

キーワード：制御ソフトウェア，参照モデル，オブジェクト指向，フレームワーク

1. はじめに

ソフトウェアの再利性を高め開発を効率化する技術としてオブジェクト指向フレームワーク（以下、単にフレームワークと呼ぶ）が注目されている。フレームワークは、再利用可能なオブジェクト群とオブジェクト間の構造の両方を提供することでソフトウェア開発を容易化する⁽¹⁾⁽²⁾⁽³⁾。有効なフレームワークを構築するためには、対象ソフトウェアドメインでの再利用や変更の要求を的確に捕らえ、フレームワークの構造に対応させる必要がある。このような対応を実現するための手段が参照モデルである。

本稿では制御ソフトウェアドメインでのフレームワーク構築のための参照モデルとしてデバイス-制御-制約モデル（Device-Control-Constraint モデル，DCC モデル）を提案する。DCC モデルでは、変更要件を物理 I/F 変更，デバイス構成変更，アルゴリズム変更と想定し，これらに対処するフレームワークの基本構造を与える。

以下，2 章では参照モデルに基づいてフレームワークを開発する我々のアプローチと従来方法との比較について論ずる。3 章では制御ソフトウェアフレームワークの要件について述べる。4 章では DCC モデルの構成と機能につい

て説明する。5 章では 4 章で説明した DCC モデルに基づいてオブジェクト指向設計について記述する。6 章ではエアコン制御システムについて DCC モデルを用いて開発した事例とその効果を紹介する。7 章は考察である。

2. 従来方法との比較

制御ソフトウェアの分野においても，生産性向上の目的のため，オブジェクト指向手法を適用した開発がなされるようになってきている。OMT 法⁽⁴⁾や OOSE 法⁽⁵⁾のような典型的なオブジェクト指向開発方法論に従えば，それ以前の構造化手法に比べ，変更対処性に優れ，再利用性の高いソフトウェアアーキテクチャを構築することができる。我々自身も組込み制御ソフトウェア開発でのオブジェクト指向手法の適用において，変更容易性の向上の効果を確認している⁽⁶⁾。しかしながら，これらの方法論の対象は個々の開発に限られており，製品のシリーズ開発のように，類似した対象の開発が複数回発生する場合に，それら複数の開発を通して再利用を実現するための指針については充分でない。このような不足点は，製品のシリーズ開発が一般的である制御ソフトウェアの分野においては問題となる。

個々の開発に特定されず，汎用性を持つ再利用技術の一つ

にデザインパターン⁽²⁾⁽⁷⁾が知られている。デザインパターンは特定の目的を実現するためのオブジェクト構成方法を提供する。ただし、提供されるのは部分的な構成のみであり、システムの全体構成を設計するための方法は含まない。また、利用は設計、実装といった下流工程に限られる。

一方、アーキテクチャパターン⁽⁸⁾では、システム全体のアーキテクチャ構成をパターンとして捉えている。デザインパターンと比して、システム設計から実装までといった、より上位工程からのサポートを目指している。しかし、分野に依存しない汎用的な構成についての議論に留まっており、本論文で対象としている制御分野での検討は充分でない。さらに、下流のデザインパターンとのつながりが明確でなく、実適用には課題が多い。

また、ドメイン分析に基づき、フレームワークを構築するというアプローチも多く提案されている⁽⁹⁾⁽¹⁰⁾⁽¹¹⁾⁽¹²⁾。ただし、これらのアプローチも事務処理分野に特化されたものが多く、制御分野での事例に乏しい。

本論文では、上述した従来方法の課題に対し、DCCモデルという参照モデルを中心に据えたフレームワーク構築アプローチを提案する。DCCモデルは、制御ソフトウェア一般に必要な論理構成の抽象化モデルである。DCCモデルは、開発の上流の分析工程ではオブジェクトの抽出の指針を提供し、下流の設計工程においては分析とデザインパターンのようなオブジェクト構成設計を繋げる手立てとすることが可能である。これらの効果は試作により確認している。

3. 制御ソフトウェアフレームワークの要件

ここでは、まず、我々のアプローチの対象である制御システムを定義し、制御システムを開発するためのフレームワークに求められる幾つかの要件を明らかにする。要件はシステムが実現すべき機能の観点と、発生する変更への適応性の観点から整理することができる。

〈3・1〉 制御システムの機能要求 制御システムはシーケンス系制御システムとフィードバック系制御システムに大別できる。シーケンス系システムの目的はシステムのデバイス(センサ、アクチュエータ)を正しい順序で動作させることである。一方、フィードバック系システムの目的は、センサからのフィードバックに基づいて算出される出力値によってデバイスを動作させ、システム外部の環境を操作することである。実製品としての制御システムは、しばしば、これらの2種類の制御システムの要素を併せ持っている(図1)。例えば、エアコン制御システムはフィードバック制御システムの典型例であり、定常状態ではセンサデータからファン、コンプレッサの出力を算出するというフィードバック制御処理を行っているが、リモコンからのキーイベントによる運転開始/停止/変更時ではデバイスの起動や切替えのシーケンスを制御する。このようなことから、制御システムにおける基本的な機能要求として次の2点が挙げられる。

図1 制御システムの構成
Fig. 1. The typical structure of control systems

- 機能要求1：順序出力
デバイスに出力を送信する順序やタイミングに関する動作仕様の実現。状態を持つ複雑な動作仕様や状態を持たない単純な順序出力を含む。前者は後者を遷移に伴うアクションとして包含する。
- 機能要求2：出力値算出
デバイスへの出力値をセンサ入力値やシステムの状態を基に算出するアルゴリズムの実現。
また、実際の制御システムの開発では、理想的な制御モデルを現実的な制約の下に調整して実現する必要がある。これは、実際の制御システムには、アクチュエータの耐久性や、システムの安全性、省エネ性を制御性に優先することが必要な状況が存在するためである。従って、このような制約の実現も現実の制御システムの機能要求として捕らえることが妥当である。このような制約は、順序出力/出力値算出の両方に考えられ、それぞれに関する制約の実現を機能要求として捕らえることができる。
- 機能要求3：起動制約の実現
個々の制御動作の開始/終了に関する制約、また、同時に複数の制御動作が存在する場合の並列処理や優先処理に関する制約の実現。
- 機能要求4：出力値制約の実現
アクチュエータへの出力値の値域に関する制約の実現。

上記2つの機能要求の例をエアコン制御システムで説明する。最初の起動制約については、コンプレッサ制御において通常、運転開始ボタンが押されると起動するが、デバイスの保護のため一定期間以内に一度電源をOFFされている場合はこの期間が経過するまで起動せず待機するといった制約がある。また、出力値制約に関しては、やはりコンプレッサの制御に関して、外気温が0度以下のときには出力を一定周波数以上に上げてはならないといった制約がある。この場合、理想的な制御モデルを仮定して算出した出力値の上限を補正する必要がある。

〈3・2〉 制御システムの変更要求 制御システムの変更要因として我々が特に重要と考えるのはつぎの2つである。

図2 DCCモデル
Fig. 2. The DCC Model

- デバイス変更：センサやアクチュエータの追加、削除、置換といった変更。
- 制御方式変更：入出力順序や出力値算出方式など、制御アルゴリズムの変更。

これらは制御システムのシリーズ開発において高い頻度で発生する典型的な変更要因といえる。従って、これらの変更要因に対応可能であることを我々のフレームワークの要件とした。

4. DCCモデル

我々は前章で述べた実現要求、変更要求に対処するフレームワークの論理構成を示す参照モデルとして、DCCモデルを考案した。以下、DCCモデルの詳細について説明する。

〈4・1〉 DCCモデルの構成 DCCモデルは異なる役割を持つ3種類のオブジェクトから構成される(図2)。

〈4・1・1〉 デバイスオブジェクト デバイスオブジェクトはデバイスに関わる入出力データの窓口となる。実デバイスの入出力データ表現形式、データ入出力順序に関する差異を吸収することで対象システムドメインにおいて十分に抽象化された実デバイスに依存しない標準的な入出力インターフェースを提供する。これにより、利用側の制御オブジェクト、制約オブジェクトの再利用性を高める働きを持つ。デバイスオブジェクトと物理デバイス間には図にあるようにAPI群を設け物理デバイスとデバイスオブジェクトとの間の独立性を高めることでデバイスオブジェクト自体の再利用性を向上させることができる。

〈4・1・2〉 制御オブジェクト 制御オブジェクトは、制御システムの動作順序や出力値計算についての主要な制御アルゴリズムを実現する部品である。例えば、動作順序には、デバイスへのデータ入出力順序、通信プロトコル、中間的なものを含めた制御データの算出順序といったものが該当する。また、出力値計算には、PID制御やファジー制御といったものが該当する。それぞれに対応し、順序制御オブジェクトと値制御オブジェクトの2種類を定義する。

〈4・1・3〉 制約オブジェクト 制約オブジェクトは、制御オブジェクトの部品としての汎用性を高めるために、制

御オブジェクトの制御動作に付随する条件を分離し部品化するオブジェクトである。この条件は出力順序と出力値算出のそれぞれに対して定義される。出力順序については、動作を起動するための条件、出力値算出については、上下限値の制約がこれに相当する。典型的な例としては、前者では運転モード等により制御動作の切り替え、後者ではハードウェア性能による上限出力の制限などがある。それぞれに対応し、制御を起動するための条件を監視し、条件成立時に制御動作を起動する順序制約オブジェクトと、最終的な出力値をシステムの持つ出力制約を満たすよう調整する値制約オブジェクトを定義する。これらの制約オブジェクトは複数の制御オブジェクトの起動順序や算出する出力値を現実のシステムの制約の下に調整し、出力をデバイスオブジェクトに伝える働きを持つ。

〈4・2〉 DCCモデルの働き DCCモデルの3種のオブジェクトは対象システムの扱うデータについて、ハードウェアからのデータ入力、出力値の算出、ハードウェアへの出力送信までデータの流れる道筋を形成する。流れるデータの実体はシーケンス系システムでは入出力イベントなど、フィードバック系システムではセンサデータやデバイス出力値となる。デバイスオブジェクトは常にこの3つのオブジェクト間のデータの流れの起点および終点となる。順序制約オブジェクトは制御オブジェクトを選択的に起動する働きを持ち、起動条件を判断するためのデータをデバイスオブジェクトを介して入力する。順序制約オブジェクトによって、順序制御オブジェクト、値制御オブジェクトが起動されそれぞれに対応する制御動作を実行する。順序制御オブジェクトは出力データを一定の順序およびタイミングでデバイスオブジェクトに送信する。この際、必要に応じて出力データの算出を値制御オブジェクトに委託する。値制御オブジェクトは出力データを算出するのに必要なデータをデバイスオブジェクトを介して入力する。また、値制約オブジェクトの役割は最終的な出力値を調整することである。このために、値制約オブジェクトは制御オブジェクトからの出力を受け取り、調整を加えた後にデバイスオブジェクトへ送信する。デバイスオブジェクトは受け取った出力をデバイスAPIを介して物理デバイスに送信し、制御動作を完結させる。

〈4・3〉 モデルと実現/変更要求との対応 2.1節で示した各実現要求、2.2節で示した各変更要求はDCCモデルの枠組みにおいて効率的に対応することが可能である。実現要求である順序出力、出力値算出、起動制約、出力値制約はそれぞれ、DCCモデルの順序制御オブジェクト、値制御オブジェクト、順序制約オブジェクト、出力値制約オブジェクトに対応し、それぞれ部品化される。これにより、各観点ごとの柔軟な再利用が可能になる。また、変更要求については、まず、デバイスI/F変更についてはデバイスオブジェクトを用いた抽象化により、物理デバイス変更の影響を局所化している。次に、制御方式の変更については、制約仕様を制御仕様から分離することにより、制御仕様の

汎用性を高め、再利用性を容易にしている。また、それぞれについて順序に関するアルゴリズムと出力値の算出に関わるアルゴリズムを別個に部品化することで、対応する変更を局所化し、対処を容易にしている。

5. DCC モデルに基づくオブジェクト設計

DCC モデルに基づいてオブジェクトを設計する際の課題と手順について述べる。

〈5・1〉 デバイスオブジェクトの設計 デバイスオブジェクト設計のポイントは物理的なデバイスをデータ入出力方式の点で抽象化するインターフェースの実現である。

〈5・1・1〉 データ入出力の抽象化 データ入出力の抽象化は、入出力データ表現の標準的な表現を与えることと、物理的な制約による通信プロトコルを隠蔽することの2点により実現できる。

● 入出力データ表現の標準化

デバイスオブジェクトで取り扱われるデータの表現形式は将来の変更への適応性を考慮し、可能な限り汎用性を確保すべきである。このために物理デバイスのデータ形式とは独立に、より普遍的な表現形式を選択する。例えば、整数形式より浮動小数点形式を選択し、データ長も十分な余裕を持たせる。

● 物理的な入出力順序制約の隠蔽

データの表現形式とともにデバイスオブジェクトの汎用性を高めるためにはデータ送受信順序の抽象化が必要である。このために個々の物理デバイスの制約による入出力順序動作をより上位の動作概念でまとめ、細部をデバイスオブジェクトの外部から隠蔽する。これにより、デバイスオブジェクトの利用側では論理的かつ汎用的な制御アルゴリズムとしての入出力順序のみを実現すれば良いようになる。ここで、個々の物理デバイスの制約に依存する入力順序とは、例えば、IC にデータを送る際のレジスタ設定や割り込み発生などの一連の送信手順等をさす。

〈5・1・2〉 デバイスオブジェクトの抽出 デバイスオブジェクトは基本的には物理デバイスを写像することで得られる。しかし、デバイスをそのまま写像したオブジェクトは物理デバイスの変更からソフトウェア構成が強く影響を受けてしまうことがあり、安定なデバイスオブジェクトを得るためには、デバイス自体ではなく、デバイスを利用する側の観点で、デバイスやデバイス群の持つ機能や役割を捉えてオブジェクト抽出することが有効である。このような方法で抽出されたオブジェクトを機能デバイスオブジェクトと呼ぶことにする。例えば、プリンタの紙送り等、位置決めを行うようなシステムを考える。位置決めを実現するためには、物理デバイスとして、動作角度のフィードバックが得られるステッピングモーターを利用する方法や、位置センサと通常のモーターを用いて制御する方法がある。この場合、単純にデバイスを写像して、例えばセンサとモーターをオブジェクトを抽出してしまうと、次機種でス

テッピングモーターによる方法に変更されてしまった場合、デバイス変更に伴うソフトウェア構成の変更が必要になる。このような場合、センサとモーター、またはステッピングモーターを“位置指定モーター”として、位置の指定ができる機能を持つ機能オブジェクトとして抽出しておけば、ソフトウェア構成を変更する必要がなく、影響を最小化することができる。

〈5・2〉 制御仕様の分割 システムの制御仕様をソフトウェアの要求定義として眺めた場合、多くの観点からの仕様が入り交じっていることが多い。これらの仕様を単一の観点ごとに分割整理することでより汎用性の高い形にすることができる。特に重要な分割指針は、順序／出力値算出仕様の分離、順序制約の分離、出力制約の分離である。

〈5・2・1〉 順序／出力値算出仕様の分離 データの送受信に関する順番、出力値算出のための中間データの算出順序などの順序に関する仕様と、出力値を算出するためのデータの変換過程に関する出力値算出仕様を分離する。

〈5・2・2〉 順序制約の分離 順序制約とは制御動作を起動する要因となる条件である。例えば、“リモコンからの電源キー入力”、“運転モードが暖房時”、“温度センサの入力が20℃以上のとき”といった条件である。順序制約と起動される動作の仕様を分離することで、それぞれに対して発生する変更にも別個に対処可能とする。

〈5・2・3〉 値制約の分離 値制約とは制御出力の値そのものや出力を送信する順序やタイミングの制限に関する条件である。例えば、“モータ保護のための回転数の上限制限”、“加熱時のモーター停止”といった条件である。

〈5・3〉 制御／制約オブジェクトの役割分担と構成 制御オブジェクト、制約オブジェクトは前節の制御仕様分割と対応して構成する。この際、制約オブジェクトは順序制約、値制約として分離した仕様を実装するソフトウェア部品として、制御オブジェクトはより汎用的な順序制御、出力算出アルゴリズムを実装する部品として作成する。また、それぞれのオブジェクトは順序に関するものと出力値を算出するものの2種類がある。制御オブジェクトでは、出力値を算出し、出力を送信する順序に関する処理は順序制御オブジェクト、入力値や内部データから値を変換する処理は値制御オブジェクトの役割となる。順序制御オブジェクトは出力値を送信する際に値制御オブジェクトを利用する。制約オブジェクトでは、システム全体の状態や入力から起動すべき制御オブジェクトを判定する順序制約オブジェクトと、制御オブジェクトにより算出された制御出力値に対し、値制約に基づいて補正する値制約オブジェクトがある。これら4種のオブジェクトの構成を図3に示す。起動順序とデータの流れに従い、順序制約オブジェクト、順序制御オブジェクト、値制御オブジェクト、値制約オブジェクトの順に並ぶ。

〈5・4〉 制御オブジェクトの設計

〈5・4・1〉 順序制御オブジェクト 順序制御オブジェクトは制御シーケンスを実装する。制御シーケンスとは制御

図 3 制御／制約オブジェクトの役割分担

Fig. 3. Cooperation between control objects and constraint objects

図 4 Observer パターンの利用

Fig. 4. Applying Observer pattern

シーケンスは入出力の送受信順序や、中間的な制御データの算出順序である。順序に関する仕様はしばしば状態遷移図などにより表現されるが、この実装にはデザインパターンである State パターンなどが利用できる。また、条件分岐やループなどの少ない単純な制御シーケンスではクラスを用いず、関数として実装することもありうる。順序制御オブジェクトは並列性を持たせることで自然にタスクとして拡張することができる。

〈5・4・2〉 値制御オブジェクト 値制御オブジェクトはデータの変換を実装する。変換によって値を算出するが時系列に沿った値を入力しなければいけない場合、たとえば PID 制御のような漸加的に値を算出していくようなアルゴリズムでは内部に順序に関する仕様が実装されている。この順序動作は内部に隠蔽することが可能であるので値制御の部品とする。

〈5・5〉 制約オブジェクトの設計

〈5・5・1〉 順序制約オブジェクト 順序制約オブジェクトはシステム全体の状態と入力データから一定の条件を判定し、適当な制御オブジェクトを選択し、起動または停止させる。また、起動すべきでないときに動作を停止させる。順序制約オブジェクトはシステム全体の状態や運転モード

により起動すべき制御を選択する表構造や、キー入力により起動する制御を割付けたテーブルなどにより実現される。

〈5・5・2〉 値制約オブジェクト 値制約オブジェクトは出力値を値域に関する制約に従うように加工する。この操作はデータに対して上限、下限をとり、フィルタリングする処理で実現できる。制約のうち、複雑なものは順序制約オブジェクト、制御オブジェクト、値制約オブジェクトの連携により実現される。

〈5・6〉 デザインパターンの適用 DCC モデルの各オブジェクトの構成にはデザインパターンの利用が有効である。

〈5・6・1〉 入力データ更新の通知 (Observer パターン) デバイスオブジェクトが管理するの実デバイスの入力データは、複数の制御オブジェクト、制約オブジェクトで利用される。この入力データの更新を通知するための仕組みとして Observer パターンが利用できる。図 4 は利用の例である。この例では PIDController クラスが Sensor クラスの入力データを監視する Observer クラスである。

〈5・6・2〉 制御、制約オブジェクトの組み合わせ (Strategy, Composite パターン) 制御、制約オブジェクトに呼出しインタフェースを共通化する Strategy パターンや複数のオブジェクトを組み合わせる Composite パターンを利

図 5 Strategy, Composite パターンの利用 (1)
Fig. 5. Applying Strategy and Composite pattern (1)

図 6 Strategy, Composite パターンの利用 (2)
Fig. 6. Applying Strategy and Composite pattern (2)

用することで柔軟な部品利用が可能となる。図 5 は値制御オブジェクトに対し、これらのパターンを利用した例である。Strategy パターンを利用し、複数の値制御オブジェクトの呼出しインタフェースを Calculate() という関数に統一している。これにより、利用側での値制御オブジェクトの交換が容易となる。また、AddControl というクラスは Composite パターンを利用し、複数の値制御オブジェクトを包含し、それらの総和を出力とするような値制御オブジェクトの構築を可能としている。ChoiceControl オブジェクトはこの応用であり、関連付けられた 1 つの条件オブジェクト Condition が判定する条件により、2 つの値制御オブジェクトを切り替えて値を出力する。

図 6 は値制約オブジェクトの例である。この例では Strategy パターンにより、値制約オブジェクトのインタフェース Revise(int) という関数に統一している。また、Composite パターンにより、複数のフィルタを統合するフィルタとして CompositeFilter クラスを定義している。ChoiceFilter クラスは条件により値制約をするかどうかの切替えが可能

な値制約オブジェクトである。

〈5・6・3〉 状態遷移による順序制御の実装 (State パターン) 順序制御オブジェクトの順序仕様を実現するために State パターンが利用可能である (図 7)。

6. DCC モデルの構成事例

DCC モデルの各オブジェクトの具体例についてエアコン制御システムの事例をもとに説明する。

〈6・1〉 要求仕様

● デバイス構成：

室内・室外ユニットとリモコンにより構成される。室内ユニットには室内ファン、ルーバ、マイコン、赤外線受光器、室温センサ、熱交換器センサといったデバイスを内蔵している。室外ユニットはコンプレッサ、室外ファン、外気温センサといったデバイスを内蔵している。室内機と室外機の間は通信ケーブルで連結され、センサデータ、制御データがやり取りされる。ユーザからの入力のは赤外線リモコンにより送信され、室内機の赤外線受光器を介してデータが入力される。

図 7 State パターンの利用
Fig. 7. Applying State pattern (2)

- 制御仕様：
 - 制御動作はユーザのリモコン操作により開始される。基本的なリモコン操作は電源の ON/ OFF と冷暖房の切替えである。
 - ルーバ、ファン、コンプレッサといった出力デバイスの制御値を室温、外気温、熱交換機温、湿度などを計測する各種センサからの入力値から一定の制御アルゴリズムのもとに算出し、適切なタイミングで各デバイスに送信する。
 - 安全性や耐久性の観点から、制御出力に一定の制限がある。(例えば、熱交換器温度や、コンプレッサ回転数の上限設定)
 - コンプレッサ、ファンの出力値算出には PID 制御アルゴリズムを利用する。

〈6・2〉 デバイスオブジェクトの抽出 デバイスオブジェクトは図 8 に示すように以下のものが抽出される：リモコン、ルーバ、ファン、コンプレッサ、室温センサ、熱交換器センサ、外気温センサ、モード、ファン制御、ルーバ制御、コンプレッサ制御これらのうち、モード以降はマイコンの機能を分解した機能デバイスオブジェクトである。

〈6・3〉 制御仕様の分割 制御仕様のうち、順序に関わる仕様と値の算出関わる仕様を分離する。順序に関わる仕様には以下のようなものがある。

- 順序に関わる仕様：
 - a. リモコンによる入力を受け、運転モードを変更する。
 - b. 運転モード変化によって運転を切り替える。
 - c. 除霜発生時の処理（コンプレッサ停止→逆転→停止→正転）これらのうち、a と b の仕様は制御の起動の条件に関わるので制約仕様に分類できる。c については、除霜発生を判断する部分は起動の条件であり、実際の除霜動作は順序的な動作に関する制御仕様である。また、値に関わる仕様には以下のようなものがある。
- 値に関わる仕様：
 - d. PID 制御アルゴリズム
 - e. コンプレッサの上限値、ファンの上限値d の PID 制御に関しては、周期的に入力値を取得し、出力値を算出するといった順序的な仕様も含むが、こ

図 8 エアコンのデバイスオブジェクト抽出
Fig. 8. Device objects in air-conditioner control system

れは出力値を算出するまでの順序に関する仕様であり、出力を送信する順序とは独立と考えられるので値制御の仕様と分類する。e については d によって算出された値に対して上限値を設定し、上限を超えるときには上限値を最終的な出力とする。この上限値はコンプレッサについては外気温、ファンについては熱交換器温により決定される。

〈6・4〉 制御／制約オブジェクトの設計

〈6・4・1〉 順序制約オブジェクト 順序制約としては運転モードによる条件と除霜についての条件があり、これらの条件によって制御が切り替えられる。運転モードは機能デバイスオブジェクトであるモードオブジェクトが管理し、常に、冷房、暖房、停止のいずれかとなる。これにより、一つの順序制約オブジェクトとして、運転モードにより各モードに対応する制御を起動する“制御表”オブジェクトを設

図 9 制御表オブジェクト
Fig. 9. Control Table object

図 10 除霜順序制約オブジェクト
Fig. 10. Defrost Condition object

計した(図 9)。

除霜条件についてはこれに対応して外気温と熱交換器温が一定の条件を満たすときに制御を起動する下図のようなオブジェクトを設計した(図 10)。

〈6・4・2〉 順序制御オブジェクト 除霜時のデバイス制御シーケンスを実現するオブジェクトを設計した(図 11)。

〈6・4・3〉 値制御オブジェクト PID 制御中で出力値の算出をテーブルから参照するような場合はこのテーブルが値制御オブジェクトとなる。また、外気温、熱交換器温からコンプレッサ、ファンの上限を算出する部品もオブジェクトとして抽出できる(図 12)。

〈6・4・4〉 値制約オブジェクト ファン出力には安全性確保のための上限値がある。このような場合、ファン出力を、まず、制御性能の観点から仮出力値として算出し、この仮出力値に対して安全性の観点から上限を補正するといった処理が考えられる。このとき、仮出力の算出までを値制御オブジェクト、上限補正処理を値制約オブジェクトの役割として設計する(図 13)。出力補正は出力の上限、下限を与えることで行われるが、この閾値は一般には、デバイスからの入力値やシステム状態によって変化する。このような場合、閾値の算出は値制御オブジェクトの役割とし、出力制御オブジェクトは算出された閾値をもとに仮出力値を調整する。

一つの出力値に対して複数の制約がある場合にはそれぞれの制約ごとに値制約オブジェクトを設計し、制約調整の際にこれらを多段に組合わせた形で構成する。これにより、一つの仮出力に対して順次、複数の制約条件に対応した補正を行うことができる(図 14)。また、また制約の優先度の高い制約に対応するオブジェクトをフィルタリングの順番の後方に配置することで優先度に則した値制約の調整ができる。

図 11 除霜制御オブジェクト
Fig. 11. Defrost Control object

図 12 PID 制御オブジェクト
Fig. 12. Defrost Control object

図 13 ファン上限制御オブジェクト
Fig. 13. Fan Upper Limit Control object

〈6・5〉 評価 DCC モデルを利用して開発したエアコン制御ソフトに対し、実際の製品仕様を用いて、制御仕様を複雑化する次機種への拡張開発を行った。この結果、同等の仕様変更に対する工数比較において、従来開発(構造化手法)の約 4 倍の効率化が確認できた。

表 1はこの際に生じたソースコードの変更を、変更の種類、DCC モデルのどのカテゴリに対応するオブジェクトでの変更かを分類し、その回数を記録したものである。この表から、まず、変更の 9 割は制御、制約の各カテゴリで発生していることが分かる。このことは、制御アルゴリズムを複雑化する変更に対し、制御アルゴリズムを実現する役割を持たないデバイスオブジェクトがほぼ無修正に再利用できたことを示している。これはすなわち、不要な修正の発生による作業コストの増加を抑制できたことを意味している。また、発生した変更の種類についてみると、既存オブジェクトの修正が必要なケースは 3 割以下であることが分かる。これは、7 割以上のケースで既存オブジェクトが再利用されていることを表しており、また、その半数のケースで作業コストの小さいパラメータ変更のみで対処できていることが分かる。以上のようなことから、DCC モ

図 14 値制約オブジェクトによるフィルタリング
Fig. 14. Iterative structure of Ouput Constraint objects

表 1 仕様変更により発生したコード変更
Table 1. Occurrence of code modification due to extension of specification

DCC \ 変更	新規クラス追加	クラス修正	パラメータ変更	合計 (割合)
デバイス	1 回	3 回	0 回	4 回 (11 %)
制御	10 回	6 回	7 回	23 回 (66 %)
制約	0 回	0 回	8 回	8 回 (23 %)
合計 (割合)	11 回 (31 %)	9 回 (26 %)	15 回 (43 %)	35 回 (100 %)

デルの利用により有効な部品化が実現できたことが確認できる。
なお、今回の評価は試行開発によるもので、今後、実製品への適用結果からより厳密に評価を行う予定である。

7. 考 察

DCC モデルは、それぞれ異なった役割を持つ 3 種のオブジェクトにより制御システムを次のような異なる機能単位に分割し、再利用と変更を容易にする。

(1) デバイスの物理的／論理的制御
(2) 制御の起動と出力値に関する制約と場面ごとの制御や汎用的な制御アルゴリズム
(3) 制御順序に関する制御とデータ変換に関する制御

(1) はデバイスオブジェクトにより実デバイスのデータ入出力の抽象化により実現される。物理デバイスの変更に起因するソフトウェアの変更がデバイスオブジェクトに局所化されるため、利用側の汎用性が増し、再利用性が向上する。(2) は制御オブジェクトと制約オブジェクトを分離することで実現される。制約オブジェクトに制御の起動や出力値の制限といったシステム全体に関する制約の実現を委譲することで、制御オブジェクトはこの制約とは独立に作成することができるため、部品としての汎用性が向上する。(3) は制御オブジェクト、制約オブジェクトのそれぞれを順序制約オブジェクトと値制御オブジェクト、順序制約オブジェクトと値制約オブジェクトに分離することで実現される。それぞれ、順序と出力値に観点が絞られ、他方とは独立に作成されるため、部品の汎用性を向上させることができる。

8. おわりに

本稿では制御システム向けのフレームワークを構築するためのアーキテクチャモデルとして DCC モデルを提案し、このモデルによるフレームワークの設計事例を示した。
DCC モデルはシーケンス系制御システム、フィードバック系制御システムの両方をカバーするため、このモデルを基に両者に共通の制御システム向けフレームワークを構築することが可能である。今後は、DCC モデルの利用により、家電のような小規模な制御システムからコンビニ端末などの中規模の制御システムへ適用可能な汎用的な制御向けフレームワークの開発を検討している。
(平成 12 年 4 月 25 日受付、同 13 年 1 月 19 日再受付)

文 献

(1) Johnson, R., 中村安明, 中山裕子, 吉田一樹: パターンとフレームワーク, 共立出版, 1999.
(2) Gamma, E., Helm, R., Johnson, R., and Vlissides, j.: Design Patterns, Addison-Wesley Publishing Company, 1995 (本位田真一, 吉田和樹訳: デザインパターン, ソフトバンク, 1995).
(3) 永井保夫, 大須賀昭彦, 本位田真一: 制約パターンを用いたオブジェクト指向ソフトウェア開発, コンピュータソフトウェア Vol.15, No.1, pp.2-23, 1998.
(4) Rumbaugh, J. et al.: Object-Oriented Modeling and Design, Prentice-Hall, 1990 (羽生田栄一監訳: オブジェクト指向方法論 OMT, トッパン, 1992).
(5) Jacobson, I. et al.: Object-Oriented Software Engineering, ACM, Addison-Wesley, 1992 (西岡利博, 渡邊克宏, 梶原清彦監訳: オブジェクト指向ソフトウェア工学 OOSE, トッパン, 1995).
(6) Ikeda, N., Matsumoto, K., Kageyama, Y., Kishimoto, T., Nukushina, H., and Isshiki, M.: Applying Object-Oriented Technology to Embedded Software, TOOLS 23, pp.6-15, 1997.

-
- (7) Pree, W.: Design Patterns for Object-Oriented Software Development, Addison-Wesley, 1994.
 - (8) Buschmann, F., et al.: Pattern-Oriented Software Architecture, John Wiley and Sons, 1996.
 - (9) Fayad, M., Schmidt, D., Johnson, R.: Building Application Frameworks, John Wiley and Sons, 1999.
 - (10) Kang, C. et al.: Feature-oriented Domain Analysis Feasibility Study, (CMU/SEI-90-TR-21), Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1990.
 - (11) 名取万里, 加賀屋聡, 本位田真一: データ中心アプローチとユースケースに基づくオブジェクト指向フレームワーク構築手法, 情報処理学会論文誌, Vol.38, No.3, pp.634-pp.656, 1997
 - (12) Frakes, B., Prieto-Diaz, R.: Introduction to Domain Analysis and Design Engineering, Tutorial Text of 4th International Conference on Software Reuse, IEEE Computer Society Press, Orlando, FL, 1996.
 - (13) 桑原修二, 藤村茂, 富田昭司: MVCの拡張, MVCGとそれに基づく UIMS, Talkieの実装, コンピュータソフトウェア Vol.9, No.1 (1992), pp. 27-41.
 - (14) 池田信之, 松本一教, 斎藤悦生: オブジェクト指向によるエアコン制御シミュレータの開発, 信学会技報 KBSE96-19 (1996-09), pp. 9-16.

守 安 隆 (非会員)

1954年生。1978年京都大学卒。1982年同大学大学院博士課程修了。1983年(株)東芝入社。現在, 同社情報・社会システム社 SI 技術開発センターに所属。分散オブジェクト技術などの研究・開発に従事。IEEE, 情報処理学会各会員。

池 田 信 之 (非会員) 1967年生。1990年筑波大学情報学類卒。1992年同大学大学院理工学研究課理工学専攻修士課程終了。同年(株)東芝入社。現在, 同社情報・社会システム社 SI 技術開発センターに所属。オブジェクト指向技術の研究・開発に従事。ACM, 情報処理学会各会員

早 瀬 健 夫 (正員) 1967年生。1990年上智大学理工学部機械工学科卒。1992年同大学大学院理工学研究科機械工学専攻修士課程修了。同年(株)東芝入社。現在, 同社情報・社会システム社 SI 技術開発センターに所属。オブジェクト指向技術の研究・開発に従事。ACM, IEEE, 情報処理学会, 人工知能学会各会員。

松 本 一 教 (非会員) 1961年生。1984年九州大学理学部数学科卒。1986年同大学大学院総合理工学研究科修士課程修了。同年(株)東芝入社。現在, 同社情報・社会システム社 SI 技術開発センターに所属。博士(理学)。データマイニングなどの研究・開発に従事。ACM, IEEE, 情報処理学会, 人工知能学会各会員。