オブジェクト指向技術を取り入れたモデルベース開発の提案

2005MT024　平野　将貴　　2005MT105　杉浦　由季

指導教員　青山　幹雄

1. はじめに

近年の組込みシステム開発では，増大する開発コストを軽減する手段としてモデルベース開発が注目されている．しかし現在のモデルベース開発では，具体的な開発プロセスやモデル作成方針は示されていない．本研究では，現在広く行われているモデルベース開発方法の問題点をケーススタディを用いて抽出し，問題点を踏まえた提案とケーススタディに基づいた提案の評価をする．

2. モデルベース開発の問題点

一般的に行われるモデルベース開発のケーススタディを行い，問題点を抽出する．

2.1　ケーススタディで用いるシステムの要求仕様

例として二輪車の走行制御を行うシステムを用いた．二輪車は一つの車輪に一つのモータが接続されていて，モータの回転数を制御して「直進」「右折」「左折」をする．

制御命令は外部から入力し，任意の速度で走行できる．また角度と回転半径を入力すると，任意の半径で右折と左折ができる．

2.2　システム設計

設計では要求仕様をもとに制御構造を検証し，抽象度の高い制御モデルを作成する．そして制御モデルを機能に基づき詳細化する．

ケーススタディの設計プロセスを以下に示す．

(1) 要求仕様を基に制御システムを機能分割し，インタフェースを記述して抽象度の高い制御モデルを作成する．
(2) 分割した機能ごとに実現方法を検討し，制御モデルを詳細化する．
(3) 制御モデルの詳細化を繰り返し，MATLAB/Simulink でシミュレーション可能なブロック線図にする．

詳細化の流れとブロック線図を図 1 に示す．

2.3　シミュレーション

作成したブロック線図を用いて，直進を想定したシミュレーションを行った．左右の車輪回転数をブロック線図のスコープで測定し，想定した制御が行えているか確認をした．

シミュレーションにより，発車時に車輪回転数がオーバーシュートすることが判明した．原因を分析したところ，モータ制御のロジックに問題があることが分かった．

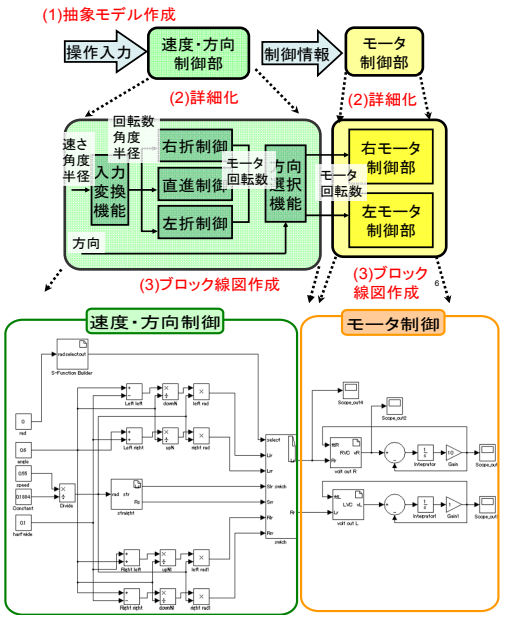

図 1　設計モデルの詳細化とブロック線図

2.4　考察と問題点

オーバーシュートの原因を特定するために，モジュールごとに制御構造を確認する必要があり，手戻りが発生した．また修正を加える際に関連を持つモジュールの詳細を確認し，必要な場合は変更を加えた．さらに一つの問題を修正すると新たな問題が発生した．

以上のケーススタディより次の問題点を抽出した．

(1) 理論的な制御と，ハードウェアの制御の間にはギャップがある．
(2) 機能に基づく詳細化ではモジュール間の結合が高くなりやすいため，修正を加えにくくなる．
(3) モジュールの粒度が大きいと問題の原因箇所の特定が難しい．

3. アプローチ

2 章で挙げた問題点を解決するためのアプローチを次に挙げる．

(1) 制御理論とハードウェアの間を階層化する．
(2) データを視点として詳細化をする．
(3) ブロック線図を階層的に作成し，モジュール単体でシミュレーションを行う．

4. 提案するモデルベース開発の設計方法

オブジェクト指向技術を取り入れた設計を提案する.

4.1 提案する設計プロセス

提案する設計プロセスには, 制御システムからオブジェクトを抽出して制御モデルを作成する工程と, 制御モデルに基づいてブロック線図を作成する工程がある. 前者を制御モデル作成, 後者をブロック線図作成と呼ぶ(図 2).

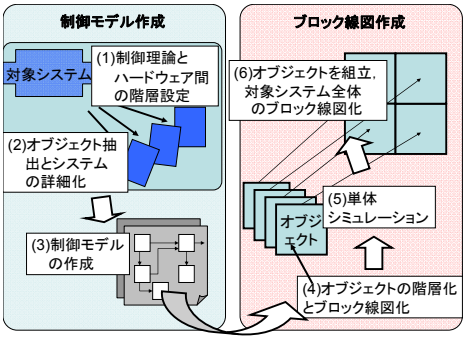

図 2 提案する設計プロセス

4.2 制御モデル作成

制御モデル作成は以下の三つのプロセスを行う.

(1) 制御システム上で現れるデータに基づいて理論層からハードウェア層までを複数の階層に分割する.

(2) 分割した階層ごとに制御目標値やシステムの状態などのデータを視点としてオブジェクトを抽出する[5].

(3) オブジェクト間の関係やメッセージ連絡を表現するために抽出したオブジェクトを用いて制御モデルを作成する.

4.3 ブロック線図作成

作成した制御モデルに基づいてブロック線図を実装する. 始めに制御モデルに基づいてオブジェクトの階層構造を抽出し, 最下位層のオブジェクトからブロック線図化する. ブロック線図化したオブジェクトの振舞いをシミュレーションで確認後, 結合させることで上位層のオブジェクトを作成する. 最終的にシステム全体をブロック線図化する(図 3).

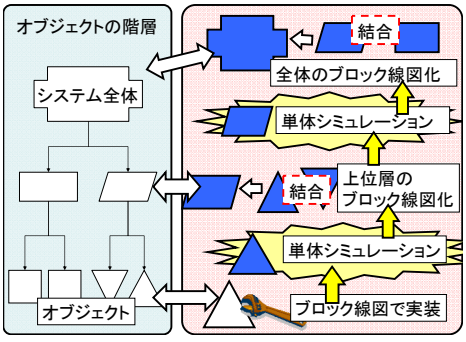

図 3 ブロック線図作成

5. ケーススタディ

前章で述べた設計プロセスを検証するために新たなケーススタディを行った.

5.1 要求仕様

実験車両は, 2 章で用いた二輪車の仕様を変更し, 電源を入れるとライン上を直進し, ラインを外れそうになるとセンサでそれを認識して軌道修正する. また走行中にストップラインを検出すると停止する[3].

5.2 制御モデル作成

(1) 始めに, 制御システムを二輪車の制御構造に特化しない制御理論層, 制御構造に特化した層, 特定のハードウェアに特化したハードウェア層の三層に階層化する.

(2) 制御理論層として「方向」と「速度」オブジェクトを抽出した. 二輪車の場合, 方向と速度はセンサと車輪で制御するので, 「方向」と「速度」オブジェクトを元に「センサ」, 「回転差」, 「回転数」オブジェクトを抽出. さらにモータで車輪の回転差と回転数を制御するので, 詳細なオブジェクトとして「モータ制御」を抽出し, 「センサ」と「モータ制御」をハードウェア層に配置する. そして「モータ制御」オブジェクトを詳細化し, 「コントローラ」と「電圧制御」オブジェクトを抽出した.

(3) 抽出したオブジェクトを基に制御モデル作成する. 図 4 は二輪車制御システムのクラス図である.

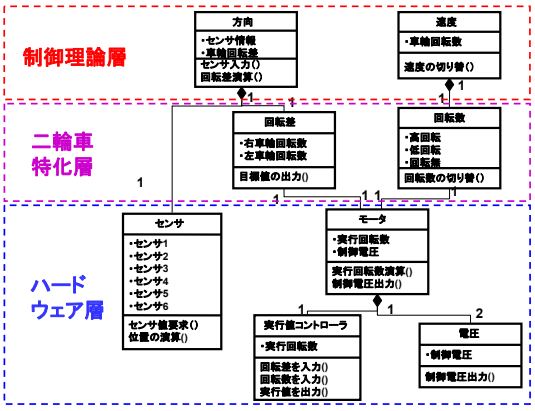

図 4 作成したクラス図

5.3 ブロック線図作成

図 4 のクラス図に基づいてオブジェクトの階層構造を抽出した. コンポジションに着目して「センサ」, 「回転差」, 「回転数」, 「コントローラ」, 「電圧」を最下位層, 「速度」, 「方向」, 「モータ」を中間層, システム全体を最上位層とする.

ブロック線図化は最下位層オブジェクトから行い, オブジェクトごとにシミュレーションで振舞いの確認をした.

次に最下位層のオブジェクト同士を結合して, 中間層のオブジェクトをブロック線図化した. 「センサ」と「回転差」を結合して「方向」を, 「コントローラ」と「電圧」を結合して「モ

ータ」のブロック線図を作成した.「回転数」は単体で「速度」となるので,「速度」のブロック線図は作成されたと見なした. オブジェクトごとにシミュレーションを行い, 振舞いの確認を行った.

最後に「方向」,「速度」,「モータ」を結合して二輪車制御システム全体のブロック線図を作成した. システム全体のシミュレーションを行い, 適正な振舞いができていると判断できた. 図 5 に二輪車制御システム全体のブロック線図を示す.

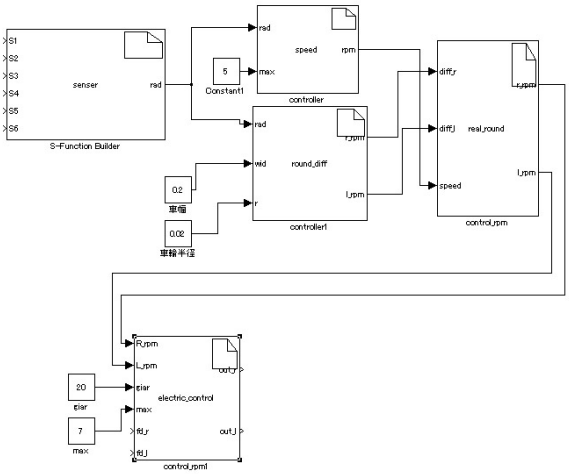

図 5 二輪車制御システムのブロック線図

5.4 シミュレーション方法

オブジェクトのブロック線図に対して, 入力値を変化させ出力応答を確認することで制御の正しさを検証した. シミュレーションは複数回行って例外処理など複数の場合を想定する. ケーススタディのシミュレーション時間は全てのオブジェクトに対して 10 秒である.

例として, 二輪車制御システム全体のシミュレーション結果を示す. 図 6 のグラフは直進中にストップラインを検出した場合のシミュレーション結果である. グラフより, t=7から1秒ほどで停止していることが分かり, システムは想定した制御を行うと判断した.

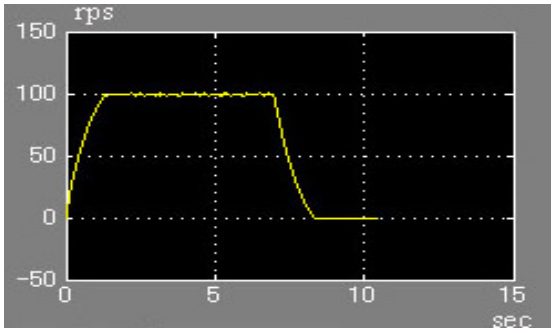

図 6 シミュレーション結果グラフ

6. 評価

5 章のケーススタディに基づいて提案を評価する.

6.1 ブロック線図作成とシミュレーションの分析

表 1 にケーススタディのブロック線図作成数とシミュレーション回数, 修正箇所数をまとめた. 表 1 から, 階層の増加に伴い作成が必要なブロック線図とシミュレーション回数が増加すると予想できる. しかし修正数に着目すると上位の層ほど修正が少ないことが分かる. 実際に修正を加えた箇所として, 最下位層のオブジェクトでは, 制御ロジックの変更に伴うブロック線図全体の作り直しや, コンパイルエラーによるコード修正など, 目標とする機能を実現させるための修正が多いのに対し, 中間層のオブジェクトでは例外処理の追加など, 制御品質をさらに向上させるための修正が多い. 上位層へ進むほどシステム品質を向上できると考えられる.

表 1 ブロック線図作成数とシミュレーション回数

階層	個数	オブジェクト	シミュレーション		
			回数	修正	修正内容
最上位	1	二輪車	5	0	-
中間	3	方向	4	1	処理追加
		速度	-	-	-
		モータ	4	3	処理追加
	小計		8	4	
最下位	5	センサ	3	1	コード修正
		回転差	3	1	ロジック変更
		回転数	3	1	コード修正
		コントローラ	1	1	ロジック変更
		電圧	5	2	ロジック変更
	小計		15	6	
合計			28	10	

6.2 オブジェクト間の結合度評価

図 7 で示すモジュール結合度の基準を用いて評価を行った. 評価基準から全てのオブジェクトはデータ結合であると分析し, オブジェクト間の結合度は低いと評価した.

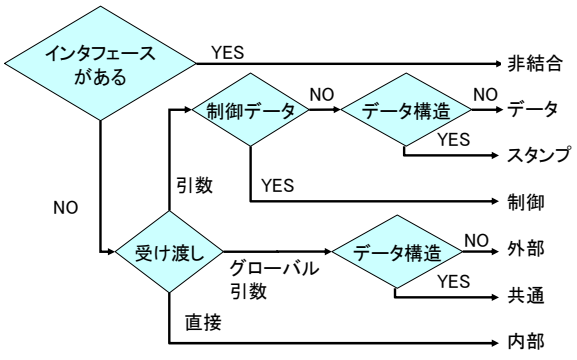

図 7 モジュール結合度の評価基準

6.3 評価のまとめ

オブジェクトを階層化することで，ブロック線図作成数とシミュレーション回数が増加する可能性は高まるが，修正しやすさの点でシステム品質の向上に貢献する．また，データに基づくオブジェクト抽出方法は，オブジェクトの独立性を保ちやすい利点がある．

なお，評価は本研究のケーススタディに基づく評価であり，今後の課題として他の性質を持つシステムに対しても同様の効果が得られるか検証する必要がある．

7. 関連研究

モデルベース開発方法について様々な研究が行われている中，オブジェクト指向技術に着目した研究として，文献[5]の「オブジェクト指向組込みシステムのモデルベース開発法」が挙げられる．この研究ではデータに着目したオブジェクト抽出を提案している．この提案は本研究と共通している点である．

しかし文献[5]の研究では，オブジェクト抽出の前段階でブロック線図化を行っていて，抽出したオブジェクトからブロック線図作成を行う本研究とは異なる．

オブジェクト抽出の前段階でブロック線図を作成した場合，システム全体の振舞いはシミュレーションで検証されているので，オブジェクト抽出後すぐに実装工程へ移ることができる．しかし，制御システム全体をブロック線図化するには，システム開発の高い技術と経験が要求されると考えられる．本研究では，システム全体の制御構造分析を容易にするために，ブロック線図作成前に段階的なオブジェクト抽出を行う．

8. 考察

ケーススタディに基づく評価では，モジュールの独立性を向上させやすい点で，データに着目したオブジェクト抽出が効果的であることを示すことができた．しかし着目するデータの説明として，制御的に意味のあるデータと記述しているので，開発者間で認識の違いを生じさせないような具体的な定義が必要である．また独立性の高いオブジェクト抽出ができたとしても，ブロック線図作成段階で制御データなどを発生させてしまうと，独立性が失われてしまうため注意する必要がある．

設計段階を終了し実装へ移る際には，オブジェクトごとにブロック線図を作成しているので，制御モデルから新たにオブジェクトを抽出しなくてもブロック線図を元に部分的な自動コード生成が可能と考えられる．オブジェクト間はデータ結合なので，コーディングはオブジェクト間の呼び出しに注意すれば，オブジェクト内部の処理について考慮する必要は少ないと考えられる．

提案に基づいたケーススタディは，2 章のケーススタディと対象システムの仕様を変えて行ったが，比較の点では同様の仕様を用いた方が効果的だったといえる．

またケーススタディのシミュレーションは，入力に対する出力結果の確認までであったが，プログラミング言語で実装を行う前に，コンピュータ上の仮想空間でライントレース走行のシミュレーションを行う必要がある．

9. 今後の課題

制御的に意味のあるデータと，オブジェクトの階層設定基準について，具体的な定義をする必要がある．また本研究の評価はケーススタディに基づくものであるので，他のシステムに対する提案の妥当性を検証する必要がある．

さらに，提案した設計方法から実装段階へアプローチする方法を具体化する必要がある，発展として実装段階まで含んだモデルベース開発方法を提案したい．

10. まとめ

従来のモデルベース開発プロセスに基づいて，ケーススタディを行ったところ，モジュールの独立性や修正箇所が広範囲になりやすい問題を発見し，オブジェクト指向開発技術を取り入れたモデルベース開発の提案を行った．提案の妥当性を検証するために二輪車制御システムを例にケーススタディを行った．ブロック線図作成数やオブジェクト間の結合度を分析し，データに基づいたオブジェクト抽出とオブジェクトの階層化において，提案が妥当であると評価した．評価はケーススタディに基づいたものであるので，今後の課題として他のシステムに対する妥当性の検証が必要である．

参考文献

[1] 本田 昭, 長崎 仁典, 図解とシミュレーションで学ぶサーボ技術入門, 日刊工業新聞社, 2004.

[2] 市村 尚規, 小山内 秀輔, オブジェクト指向を用いた自動車用組込みソフトウェアの安全化設計, 南山大学 2005 年度卒業論文, 2006.

[3] 近藤 広樹, 野澤 昌史, オブジェクト指向を用いた自動車用組込みソフトウェアの安全化設計, 南山大学 2004 年度卒業論文, 2005.

[4] 大川 善邦, MATRAB による組込みプログラム入門, CQ 出版, 2005.

[5] 吉村 健太郎, 宮崎 泰三, 横山 考典, オブジェクト指向組込みシステムのモデルベース開発法, 情報処理学会論文誌, Vol. 46, No. 6, Jun. 2005, pp. 1436-1446.