

組込みソフトウェア開発における トップダウンな振る舞い・データモデリングプロセス

岡田 康治[†] 松浦 佐江子[†]

芝浦工業大学[‡]

1. はじめに

ET ロボコン[1]は規定されたコースと走行体ロボットの下で組込みソフトウェアのモデリングと制御の技術を競う大会である。本大会でも一般的なオブジェクト指向開発と同様に、クラス設計を行った後に振る舞いを設計し、そのモデルを元に実装するというプロセスを推奨している。しかし、運用環境の測定値やハードウェアのセンサ値、制御値といった相互作用がブラックボックスなデータが対象であるため、組込み開発に慣れていない者が設計したクラスは実際の観測結果と異なることが多くクラス設計や振る舞い設計の修正という手戻りが発生し易い。

我々はこの問題に対し、ソフトウェアが達成すべき全体的な目標からトップダウンに振る舞いと振る舞いに必要なデータを分析・設計したモデルを作成し、ラウンドトリップにモデルを改善した後、振る舞いとデータからクラスの分析・設計を行うモデリングプロセスを適用することで、実際の動作としては満足の行くソフトウェアの開発に成功した[2]。

本稿では、ET ロボコン 2012 大会において作成したモデルを例に、提案手法で設計したクラスの問題点を挙げ、その問題点を解消するクラス設計手法を提案する。

2. 既存のクラス設計手法の問題点

提案手法[1]で作成したクラス図を図1に示す。

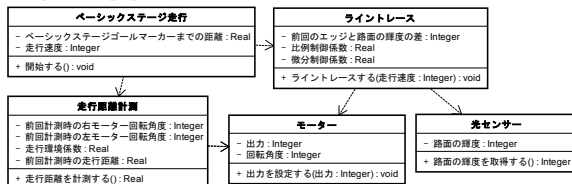


図 1 ベーシックステージ走行のクラス

この図はベーシックステージと呼ばれる ET ロボコン大会競技コース前半部分をクリアするためのクラスの一部である。ベーシックステージは白地のコース上に引かれた黒色のラインに沿って走行体を移動してゴールまでの走行タイムを競う部分であるラインエッジに沿って移動するためにライントレースを、コース後半部分のシナリオへの切り替え判断のために走行距離計測を用いている。

このクラス図の問題点は、クラス図の設計が機能分割的である点と、図から運用環境の構造を読み取れない点である。前者については依存関係に沿ってシナリオ、制御、ハードウェアと段階的に分解された機能を表すクラスしか存在しないため、データや振る舞いの変更されると依存するクラスに連鎖的に変更が発生する可能性が高い。また、データもプリミティブな定数属性として定義しているものが多く、データの変更が属性の変更に直結する点も問題である。後者は、運用環境の要素がシナリオや制御とどう関係するのか表現されていないため、設

A Top Down Modeling Process based on Behavior and Data in Development of Embedded Software

[†]Koji OKADA, [‡]Sacko MASTUURA

[‡]Shibaura Institute of Technology

計者が問題領域をどの様に解釈してこのモデルを作成したのかが他者に伝わらないという問題がある。

3. 振る舞い・データモデリングプロセス

モデリングプロセスの概要を図2に示す。

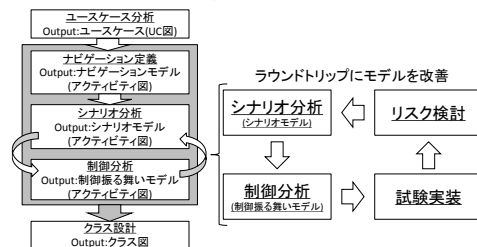


図 2 モデリングプロセス概要

本手法では、はじめにユースケース分析を行いソフトウェアが達成すべきトータルな目的に対して、どのようなユースケースが存在するかを定義する。その後、ユースケースの順序関係を表すナビゲーションモデル、各ユースケースの制御の流れを表すシナリオモデル、各制御のアルゴリズムを表す制御振る舞いモデルと段階的に振る舞いを定義する。この3つのモデルでは振る舞いに加え、それに必要なデータを分析し、必要とするアクティビティ図内でオブジェクトノードとして記述する。このデータは図中での役割に応じて表1のように分類する。

表 1 データの種類

略称	種類
生成データ	アクティビティ図内で生成・更新されるデータ
参照データ	他アクティビティ図で生成されるデータ
定数データ	理論的根拠が存在する定数
実験データ	実験によって最適値を推定した定数
HW データ	ハードウェア (HW) から取得するデータ
計算データ	他データを用いて計算するデータ

また、シナリオ分析と制御分析では、作成したモデルに従ってプログラムを実装し、動作確認を行う。実装自体はモデルでデータとアルゴリズムを定義しているため非常に容易である。動作確認の結果無視できない目的を達成できなくなるリスクが存在する場合は、振る舞い自体の変更が必要か、それともデータの値の調整で解決するかを検討し、ラウンドトリップにモデルを改善する。

4. 新たなクラス設計手法

新たな設計手法でも前章のプロセスで作成した3つのモデルに定義した振る舞いとデータを元に設計する。

新たなクラス設計手法ではシナリオの過程、例えばライントレース制御による移動の最中に、運用環境に対する作用の結果や運用環境のセンシング結果がどのように変化するかを分析し、ソフトウェア全体の振る舞いの過程における状態変化が表せるクラス設計を行う。その具体的な流れとして、2章で挙げた例を元に(1)データに基づく運用環境の構造モデリング、(2)センサ系制御による状態変化の分析、(3)駆動系制御による状態変化の分析、(4)分析結果の統合の流れで走行距離を測りながらライントレースするためのクラス設計を行う。

(1) データに基づく運用環境の構造モデリング

表 1 の定数データは予め運用環境から測定した値やその値を組み合わせて求めた値が該当する。従って、これらのデータは運用環境の要素の属性をデータとして抽出した物と言える。後のプロセスで分析する制御と運用環境を関連付けるために、先にこの運用環境をクラス化する。図 2 ではベーシックステージ走行クラスの「ベーシックステージゴールマーカまでの距離」が定数データに相当する。このデータはコースの開始地点からベーシックステージの終了地点までの距離を表している。これを運用環境の要素と属性として見ると、ある「区間」の「距離」と捉えることができるので、区間クラスを作成し、距離を属性に定義する。

(2) センサ系制御による状態変化の分析

走行距離計測制御の目的は走行体がコース上を走行した距離を計測することである。このような制御はハードウェアでセンシングした実世界の情報をソフトウェアに必要なデータに変換する制御と言える。このような制御をセンサ系制御と呼ぶ。センサ系制御の状態変化は検出対象の変化そのものである。クラス化の材料として検出対象・検出対象と運用環境の関係・検出手段を分析する。

走行距離計測制御の目的は、走行体の現在の走行距離を知ることである。よって検出対象は「現在の走行距離」である。またこの走行距離は今走行している区間の開始地点からの距離として考えているので、検出対象と運用環境の関係は、「現在の走行距離」は「区間」の要素である。検出は走行体の左右のモーターから回転角度を取得して実現している。以上より走行距離クラスを定義し、現在の走行距離を属性に定義する。さらに区間クラスと走行距離クラスを、走行距離クラスとモータークラスを関連で結ぶ。元の走行距離計測クラスの責務は検出対象のクラスが引き継ぐことが適切であるため走行距離クラスへ走行距離計測クラスの属性と操作を定義する。

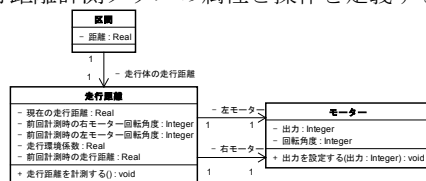


図 3 走行距離計測制御の分析結果

また、ライントレース制御はラインのエッジと走行体のズレの検出と、そのズレを最小化するための走行体の操作に分けられることから、前者の検出部分をセンサ系制御として分析する。検出対象は「ラインとのズレ」であり、このズレは「路面の輝度」と「エッジの輝度」の2つの「輝度」から求まる。検出は光センサから路面の輝度を取得し、予め光センサから取得したラインのエッジの輝度と比較することで実現している。以上より走行距離計測と同様にクラスを設計する。

(3) 駆動系制御による状態変化の分析

ハードウェアを操作することで実世界に対して作用する制御を駆動系制御と呼ぶ。ライントレース制御でのラインとのズレを最小化する制御が駆動系制御である。駆動系制御における状態変化は制御の結果の作用対象の変化である。クラス化の材料として作用対象・作用対象と運用環境の関係・作用手段を分析する。ライントレース制御が作用する対象はセンサ系制御で検出対象とした「ラインとのズレ」である。運用環境との関係についても同様に「路面の輝度」と「エッジの輝度」の2つの「輝度」から求まる。またラインとのズレへの作用手段

は元々のライントレース制御同様に走行体の左右モーターを用いる。以上の分析結果のクラス図を図 4 に示す。

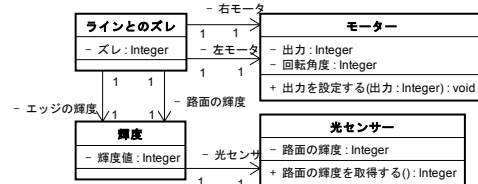


図 4 ライントレース制御の分析結果

(4) 分析結果の統合

最後に(1)から(3)の分析結果を統合する。はじめにこれらの制御がシナリオ中で何をしているかを明示したクラスを作成する。本稿の例ではライントレース走行するためにこれらの制御を利用しているため、「ライントレース走行」クラスを定義し、シナリオクラスと関連を結ぶ。そしてこのクラスに対して検出対象クラスと作用対象クラスを関連で結ぶ。統合したクラス図を図 5 に示す。

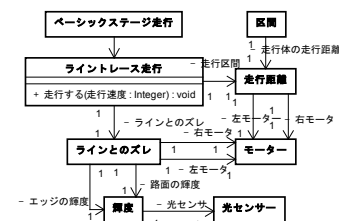


図 5 分析結果を統合したクラス図

新たなクラス図では、制御を走行距離やラインとのズレのような制御の対象に割り当てることでオブジェクトと振る舞いがセットになっており、機能分割的な設計では無くなった。また、元のベーシックステージ走行クラスが持っていた距離のデータを区間クラスとして切り出したため、距離の変更が区間クラスインスタンスの具体値の変更にも留めることができ、属性の変更に直結しなくなった。また、区間、走行距離、ラインとのズレ、輝度といった運用環境の要素をクラス化して関係を明示することで、設計者はこれらを材料にベーシックステージ走行のシナリオ・制御を分析した事が理解しやすくなった。

5. まとめと今後の課題

本稿では我々が提案する振る舞い・データモデリングプロセスのクラス設計手法では機能分割的な設計になり、設計の意図が汲み取れないという問題に対し、データから運用環境をモデリングした上でそのモデルに制御がどう関わるのかを分析する新たな設計手法を提示した。

今後は、提案モデリングプロセスに、作成したナビゲーションモデル、シナリオモデル、制御振る舞いモデル中のアクションをクラスの操作にマッピングするプロセスを追加する。本稿で提案した手法によって、これまでバラバラだったデータをオブジェクトの構成要素として構造化することが可能になった。従って、モデル中の各アクションがどのオブジェクトの状態をどう変化させるのかを分析することで、アクションを操作としてクラスへマッピングすることが可能になると期待できる。

6. 参考文献

- [1] ET ロボコン 2012 公式サイト, <http://www.etrobo.jp/2012/>, (2013/01/07 参照)
- [2] 岡田康治, 野呂惇, 松浦佐江子, “ユースケースの振る舞い分析に基づくソフトウェアモデリングの保守性改善”, 電気学会研究会資料. CT 2012(24. 26-29), pp. 1-6, 2012