Vol.2016-SE-192 No.15
Vol.2016-EMB-41 No.15
2016/6/3

モデル駆動開発における D-Case を用いたモデル構築プロセスの適用

菊池雄太郎[†1]　力武克彰[†1]

組込みシステム開発の手法として注目されているモデル駆動開発(MDD:Model Driven Development)では，モデルを作成するための足掛かりとなる手法が求められている．一方，システムに求められる要求とその実現方法を構造的に記述する手法として D-Case が知られており，要求からアーキテクチャの設計を効果的に行うことが出来ると期待される．本研究では，D-Case を用いたシステム要求分析を 2 段階に分けて行うことで，システム要求から UML モデルの構築をシームレスに行う開発プロセスを提案する．さらに，実際にそのプロセスを適用して組込みシステムの開発を行うことにより，そのプロセスの評価と D-Case を MDD に適用する際の実践例の提供を行う．本稿では，その具体的な手法を提示し，その手法を開発で実践することで事例提供，評価を行った．

キーワード：組込みシステム開発，モデル駆動開発，D-Case

The applying the model building process with D-Case in the Model Driven Development

YUTARO KIKUCHI[†1]　YOSHIAKI RIKITAKE[†1]

Abstract: In this paper we would like to propose an efficient system development process in which one can seamlessly connect a requirements analysis to a software architecture design. In the proposed process, one proceed a system requirements analysis in two stages by using D-Case. Through such a stepwise analysis, one can derive internal specifications of the system and can obtain components of the system architecture models. We also discuss the benefits of the proposed process by applying it to actual development of a line tracking robot system.

Keywords: Development of Embedded system, Model Driven Development, D-Case

1. はじめに

組込みシステムの開発手法の一つとして，モデル駆動開発（Model Driven Development）[1] が知られている．モデル駆動開発とは，主に Unified Model Language（UML 記法）などに代表されるモデル記述言語などによって決められたモデルを利用した開発である．MDD の利用により，現在開発しているソフトウェアの全体を把握することが容易になる．さらに，MDD はチームでソフトウェアを開発することにあたってもメリットがある．ソフトウェアの構造をモデルにして表すことは，開発の方針やメンバー間の意思の共有が容易になることが期待できる．モデルの種類は多数存在し，状況に合わせた使用が可能である．

実際の開発で使うモデルには，モデル自身が優れている必要がある．優れたモデルには様々な要因が絡んでくるが，本稿では保守性に優れたモデルを考える．保守性とはソフトウェアの品質特性の国際規格である ISO/IEC 9126 において定められたソフトウェアの品質[2] の 1 つで，ソフトウェアの変更，維持のしやすさの指標となる．この特性を満たすモデルを生み出すには，訓練と経験が求められ，モデルの描き方を学んだだけでは作成できるものではない．この結果から，モデリング技術を持った設計者が少なくなり組込みシステム分野においてのモデリング技術は低い傾向にあることが挙げられている[3] ．

以上の課題から，モデリング技術を学んだだけでは MDD を効率的に実施することは難しい．そこで，MDD を行うための足掛かりとなる手法が必要である．

その問題を解決するため，本研究では要求からアーキテクチャ設計を行うための足掛かりとして，D-Case[4] と呼ばれる手法を活用する．D-Case とは開発ソフトウェアの要求とその実現方法を，顧客と開発者間で合意を形成するための手法である．D-Case を用いることにより，要求からシステムに必要とされる機能または仕様を抽出することが可能となる．土樋ら[5]は実際に D-Case を用いて ET ロボコンで競うためのシステムを開発した．それを通して，D-Case を使ったゴール共有の有用性が示されている．本研究では，D-Case を活用して UML を使ったモデルを抽出し，実際にシステムを開発して D-Case によるモデル抽出の有用性を示す．

本研究では，UML モデルの作成を D-Case を使って行った MDD を実践した．本稿におけるモデルとは，UML のク

†1 独立行政法人 国立高等専門学校機構　仙台高等専門学校
National Institute of Technology, Sendai College

Vol.2016-SE-192 No.15
Vol.2016-EMB-41 No.15
2016/6/3

ラス図によるアーキテクチャ設計と UML のシーケンス図によるシステムの振る舞いの設計のことを指す．D-Case を用いてシステムを開発する中で，どのようなモデルが作成されるのかを実践し開発事例を提供する．本稿では，実際に開発するシステムとして自律 2 輪走行ライントレースロボットを実装した．開発を通して，個人で開発する場合，D-Case が開発効率にどのように作用するかを考察しその結果を記述した．

2. モデル駆動開発

モデル駆動開発とは 2001 年に Object Management Group が発表したソフトウェア設計手法である[1] ．UML などのモデル記述言語を用いてモデルを作成し，それを設計の中心として開発を進めていく．

MDD を適用することのメリットとしては，モデルを使うことでソフトウェアの構造を視覚的に把握することが見込める．さらにモデルを開発チーム内で共有することにより,開発の方針やメンバー間との意思の伝達も容易になる．また，あらかじめモデルを利用してシステムのシミュレーションを行うことができ，効率的な開発が実現できるといった点が挙げられる．

MDD の具体的なプロセスは複数存在する.本研究では，V 字プロセスを参考とした MDD プロセスを考えた．

(1) 要求分析・分析モデルの作成

開発するシステムの要求の分析を行う．そして分析の結果をモデルで表し,仕様の視覚化を図る.具体例としては，ユーザや外部機器などをはじめとするアクターとシステムとの関連をユースケース図に直し，ケースごとのシナリオを設定したり，開発対象の振る舞いを設計し，ステートマシン図として示したりするなどと言った方法が挙げられる．

(2) アーキテクチャ設計

前段階で作成した分析モデルを基に，システムのアーキテクチャを表すモデルを作成する．具体例としては，クラス図やオブジェクト図などを用いてシステムの論理構造を記述するなどがある．本研究ではクラス図によってシステムの論理構造を表現する．

(3) 実装

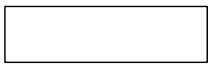
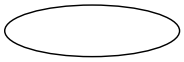
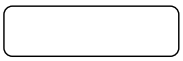
前段階で作成したモデルを基にプログラムを作成する．本研究では，ツールでアーキテクチャ設計モデルからメソッドの内容やコンストラクタを記述していないスケルトンコードを生成し，コンストラクタやメソッドの内容を実装していく．

MDD を実施する上での課題として，モデルを導出する過程が定まっていないという点がある．要求分析からシステムのアーキテクチャ設計を行うには，設計者に対して訓練などが要求される．同時に，MDD に関するワークショップや勉強会なども開催されている[6] ことから，モデルの記法を学んだだけで優れたモデルを作成することは困難である．そのことから，アーキテクチャ設計を表すモデルの足掛かりとなる手法が求められる．同時に，モデルを記述する過程が決められていないことから，モデルを記述しても，第三者の視点からなぜそのようなモデルになったのかを認識することが困難であるという点も存在する．これらの問題から，モデルを導出する過程を明確に示す手法が必要である．

3. D-Case

D-Case とはシステムに対する要求とその実現についての顧客と開発者間の合意を構造的に記述する手法であり，その手法に基づいて作られた記述である．本研究では，MDD で用いるモデルの作成を支援するツールとして扱った．具体的には，システムの要求を基に，それに必要な命題を構造的に記述し，最終的に要求を満たすことが出来る機能を抽出する．本稿で使用する D-Case の表記を次の表 3.1 に示す．

表 3.1 D-Case で用いられる表記

名称	表記	説明
ゴールノード		システムに対して議論すべき命題
ストラテジノード		ゴールノードを分解・詳細化する際の観点
エビデンスノード		ゴールノードが満たされることを証明できる証拠
コンテキストノード		ゴール・ストラテジノードに関する補足情報

D-Case は表 3.1 で示したノードを組み合わせて，要求とその実現方法を構造的に表現することが出来る．

開発者はシステムに対する要求をゴールとして設定する．中でも最も重要なゴールを「トップゴール」と称し, D-Case の頂点として記述する．更に，開発者はトップゴールをストラテジノードに記述した観点にそって,「サブゴール」に分解する．分解とはトップゴールの内容を具体化・詳細化するということである．最終的に分解したゴールノードはエビデンスノードとつながり，ゴールが満たされたということを保証できる証拠を示す．また，記述したゴールノードまたはストラテジノードにはコンテキストノードを必要に応じて接続する．これによって，ゴールノードを分解する際の前提条件や補足情報を第三者から見ても理解可能にする．

次の図 3.1 に実際の D-Case の使用例を記述する．

Vol.2016-SE-192 No.15
Vol.2016-EMB-41 No.15
2016/6/3

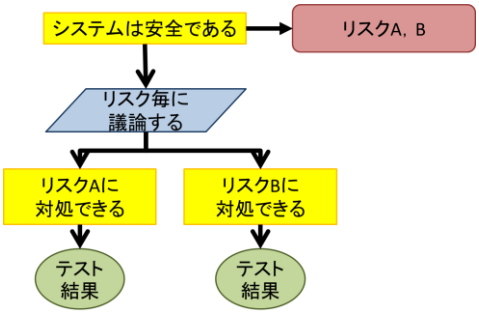


図 3.1D-Case の使用例

図 3.1 の使用例では，とあるシステムが安全であることを保証するということを目的に作られている.「システムは安全である」という命題をトップゴールとして設定し，コンテキストノードに記述されているリスクを基に議論している. その結果, トップゴールは「リスク A に対処できる」,「リスク B に対処できる」という 2 つのゴールに分解される．最後の 2 つのサブゴールが満たされたということを保証するためにそれぞれのテスト結果をエビデンスとして D-Case に記述している．この D-Case によって，システムは安全であるということを構造的に示すことが出来る．

4. モデル駆動開発における D-Case を用いたモデル構築プロセスの提案

本研究では，MDD での開発の過程に，D-Case による要求分析手法を適用し，組込みシステムを開発する．本章ではどのようなプロセス化を述べる. 具体的な方法としては，実際に開発するシステムを定め，そのシステムに求められるべきゴールを定める. そのゴールを D-Case を用いて具体化もしくは詳細化を繰り返し，D-Case から UML モデルに変換する．

4.1 開発プロセスの概要

本研究で実施した開発プロセスの概要を示す．本研究では要求項目を D-Case を用いて具体化・詳細化することで，アーキテクチャ設計を行う．このプロセスは以下の 6 つの段階で構成される．

(1) トップゴールの設定

この段階では，開発システムに対する最も重要な要求を検討し，トップゴールとして設定する．ここで設定する要求は後に作成する機能 D-Case のトップゴールとなる．

(2) 機能 D-Case の記述

前段階で設定した要求をトップゴールとし，D-Case を作成する．この段階で作成する D-Case を本稿では「機能 D-Case」と呼称する．機能 D-Case とは，本研究で示すプロセスにおいてトップゴールから，対象システムに必要となる具体的な機能を抽出することを目的とした D-Case である．

機能 D-Case を作成するにあたって, トップゴールを様々な観点で分解する必要がある．トップゴールの分解例として，ゴールを実現できなくなる要因を基に分解したり，機能 D-Case を作成する途中で抽出した問題を解決する手法を基に分解したりするなどと言った分解方法が挙げられる. 分解する観点は分解する前に検討し，ストラテジノードに観点内容を記述しておく．それに加えて，分解の際に必要となる前提条件や補足情報などをコンテキストノードに記述する．ストラテジノードとコンテキストノードを記述することで, 第三者の視点から見ても D-Case の内容を理解しやすくなる.

トップゴールを基にゴールを分解し，最終的に具体的な機能を抽出できたと判断した場合，各機能が正常に動作できているのかを判定する証拠を，エビデンスノードとして各末端のゴールノードに関連付ける．具体的には，各機能の単体テストの結果などがエビデンスとして設定される. 今後の工程で，エビデンスノードに記述された内容の事柄を獲得できるよう，モデリング・実装を行う.

(3) 内部仕様 D-Case の記述

前段階で記述した機能 D-Case で記述した機能から, アーキテクチャを抽出する必要がある．そのため，アーキテクチャを抽出することを目的とした新たな D-Case を記述する．この段階で新たに作成する D-Case を，本稿では「内部仕様 D-Case」と呼称する.

設計者は, 前段階で作成した機能 D-Case の末端のゴールノードごとに，内部仕様 D-Case を記述する．機能 D-Case の末端のゴールを内部仕様 D-Case のトップゴールとして転用し，トップゴールに記してある機能の振る舞いをシナリオとしてコンテキストノードに記述する．同時に，開発において必要な非機能要求もコンテキストノードに記述する．そしてトップゴールをシナリオが実現されるように具体的ゴールに分解していく．トップゴールを開発システムのモジュールの粒度までに分解したら，再び末端のゴールノードにエビデンスを関連付ける. 内部仕様 D-Case の場合, エビデンスノード内容は，モジュールを実現できるアーキテクチャ要素の作成が主である. この内部仕様 D-Case を作成することで, 機能 D-Case で得た機能に必要なアーキテクチャ要素を抽出することが出来る.

(4) UML 図によるモデリング

前段階で抽出した各アーキテクチャ要素をシーケンス図とクラス図に変換する.

内部仕様 D-Case を作成することでシステムに必要な要素は抽出することが出来たが，抽出したアーキテクチャが具体的にどのように関連するのかを明らかにしていない. そこで内部仕様 D-Case で示したシナリオを実現するよう，シーケンス図を記述し，各要素の関連の様子を定義することで設計をスムーズに行う．抽出した各アーキテクチャをライフラインとし，各ライフライン間のメッセージを記述していく．この過程を通して，各クラス間の操作を定義することができる.

Vol.2016-SE-192 No.15
Vol.2016-EMB-41 No.15
2016/6/3

作成したシーケンス図によって，具体的なアーキテクチャ同士の関連を定義できたのち，それを基にクラス図を記述していく.クラス図には内部仕様 D-Case によって抽出したアーキテクチャ要素，シーケンス図によって定義されたメッセージから各クラスが持つ操作の 2 つを記述する．各クラスが持つ属性も必要に応じて記述する．この過程によって，ソースコードを記述するためのモデルが完成する．

(5) 実装・各エビデンスの獲得

前段階で対象システムのクラス図を記述した後，それを基にコーディングを行う．

コーディングを行うにあたって,機能 D-Case 内で記述したエビデンスを実現できるようにコーディングを行う．例えば，エビデンスの内容が検証結果であった場合，開発者はコーディングを行った後，エビデンスとして記述された検証実験を行い,結果を獲得する.コーディングを通して，エビデンスを獲得することで，システムの要求通りの動作を保証する．

(6) D-Case およびモデルの洗練

モデリングや実装の段階などで，足りない機能や実現が困難なエビデンスが発生した場合，これまで記述した D-Case やモデルを修正する必要がある．そのため，D-Case を修正した都度，(2)から(5)までの手順に従い，ソースコードを変更していく．この過程を繰り返すことで，修正の意図や修正の具体的な内容などが，D-Case やモデルとなって残され，開発の効率化を実現できる．

5. 提案プロセスの適用事例

前節で示したプロセスを評価するため，実際にシステムを開発する.同時に D-Case を用いた組込みシステム開発の事例提供も行う．

5.1 開発システム

実際に開発する装置として，本研究では，図 5.1 に示したコースを走行する自律 2 輪走行ライントレースロボット(以降走行体と呼称)を開発した．

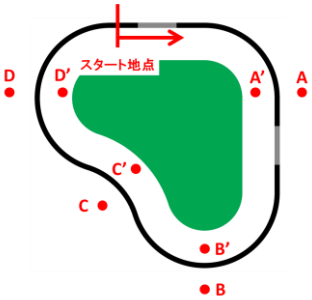


図 5.1　走行体が走るコース

走行体は図 5.1 に示してあるスタート地点から走行を始め，コースを 1 周する．コースを 1 周したらゴールしたとみなし，スタート地点からゴール地点までのタイムを計測する．更にコース上にはラインを部分的に灰色にした「灰色ゾーン」と呼ばれる場所が存在する．光センサでライントレースする場合，取得する値が変化するため，走行体の動作が安定しないことがある．このルールの上，更に 3 つの条件を追加する．

- 計測した時間は 40 秒以下であること
- 図 5.2 の AA´間，BB´間，CC´間，DD´間を通過すること
- 転倒して走行不能な状態にならないこと

これらの条件を満たすライントレースシステムを開発する．

これから開発する走行体のハードウェア構成を示す．次の図 5.2 に実際の走行体の画像を示す．

図 5.2 自律 2 輪走行ライントレースロボット[7]

この走行体のハードウェアは LEGO MINDSTORMS EV3[8] (以下 EV3)を用いて実装してある.EV3 とは LEGO 社が開発している教育用ロボットキットであり，マイクロプロセッサが組み込まれたブロックに対してプログラミングをすることで，各外部インターフェースの制御が可能である．走行体の外部インターフェースは EV3 付属のサーボモータ，光センサ，ジャイロセンサから構成されている．サーボモータは左右の両輪についている．光センサは走行体の下部についており，反射光を測定する．ジャイロセンサは走行体の角速度を計測し，2 輪倒立制御のフィードバックとなる．また，サーボモータは以降に示すライブラリによって PWM 制御が可能である．

開発環境について, EV3 を Java で制御するためのファームウェアとして leJOS EV3[9] を用いた. leJOS EV3 とは，外部インターフェースを動作させるためのプロセッサを有しているインテリジェントブロック上で動作する Java バーチャルマシンである．Java の基本的な仕様に加え，EV3 の各種デバイスを制御するための API を実現している．

本稿において，2 輪走行を実現するため，配布されてある 2 輪振り子倒立ライブラリ[10] を用いた．そのライブラリ上では，2 輪走行をする中で速度値と旋回量という 2 つのパラメータによってサーボモータを制御している．速度値は正の時，前進し，負の場合後退する．速度値の絶対値が大きいほど，走行体は早く動き，速度値が 0 の時，走行体は静止する．速度値の定義域は-100 から 100 までとしている．また，旋回量が正の時，走行体は左方向に回り，負の時は右方向に回る．この速度値と旋回量を制御すること

Vol.2016-SE-192 No.15
Vol.2016-EMB-41 No.15
2016/6/3

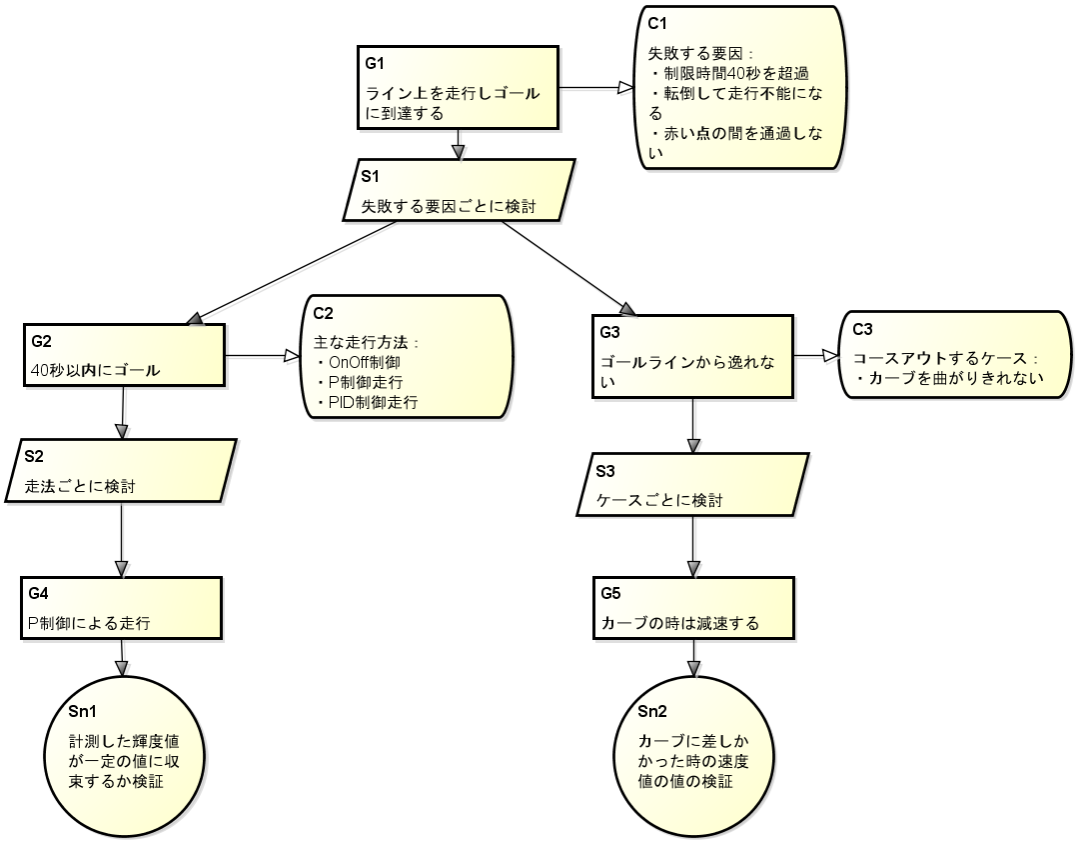


図 5.3 初めに得られた機能 D-Case の一部

でライントレースを可能にする．旋回量の定義域は-100 から 100 までとしている．また本開発では，速度値と旋回量を制御するため，輝度値を用いたフィードバック制御を行った．輝度値とは光センサから得られる明るさの値のことである．API 内では単位は決められておらず 0 から 1 に正規化されている．

5.2 提案プロセスの適用

本項目では提案したプロセスを実際に適用し，適用した結果を述べる．4.1 節にて示したプロセスに沿って述べる．

(1) トップゴールの設定

この段階では，開発システムに求める要求を検討し，設定した．本研究では「ライン上を走行しゴールに到達できる」という要求を設定した．この段階で設定した要求を後の段階で機能 D-Case のトップゴールとして記述する．

(2) 機能 D-Case の作成

前段階で設定した「ライン上を走行し，ゴールに到達できる」という要求をトップゴールとし，機能 D-Case を作成した．この過程で，初めに得られた機能 D-Case の一部を図 5.3 に示す．図 5.3 では，作成した D-Case 内の 40 秒以内で走行する機能と，コースラインから逸れない機能について示してあり，各ノードに ID 番号を割り当ててある．

前段階で設定した「ライン上を走行し，ゴールに到達できる」を図 5.3 の G1 のゴールノードに記述する．その後，トップゴールが満たせなくなる要因として「制限時間 40 秒を超過」，「転倒して走行不能になる」，「赤い点の間を通過しない」の 3 点を C1 のコンテキストノード内に記述した．

トップゴールとコンテキストノードを記述した後，失敗する要因を基に検討したことを S1 のノードに記述し，それらの要因に対する反例またはそれに準ずる命題をサブゴールとして設定した．例えば，G2 のゴールノード「40 秒以内にゴールする」は，「制限時間 40 秒を超過」という失敗する要因を基に設定したゴールである．さらに G2 を実現する手法を C2 にリストアップし，それらの手法から 1 つを選び，最終的に G4 「P 制御による走行」というゴールノードを抽出した．このゴールノード G4 の機能が正常に動作するかを判定する証拠として「計測した輝度値が一定の値に収束するか検証」という内容を Sn1 のエビデンスノードに記述し G4 と関連付けた．後の段階で，ここで記述したエビデンスノードを獲得できるよう，モデリングと実装を行う．

(3) 内部仕様 D-Case の記述

前段階で記述した機能 D-Case から内部仕様 D-Case を記述する．作成した内部仕様 D-Case の一部を図 5.4 に示す．

図 5.4 は「P 制御で走行できる」というゴールノードをトップゴール G4 にして，そのゴールに必要なアーキテクチャを抽出している．

C4 のコンテキストノードには，トップゴールが実行される流れを表すシナリオと，実装する上で必要な非機能要求を記述した．記述した内容は輝度値を測定・取得し，目標

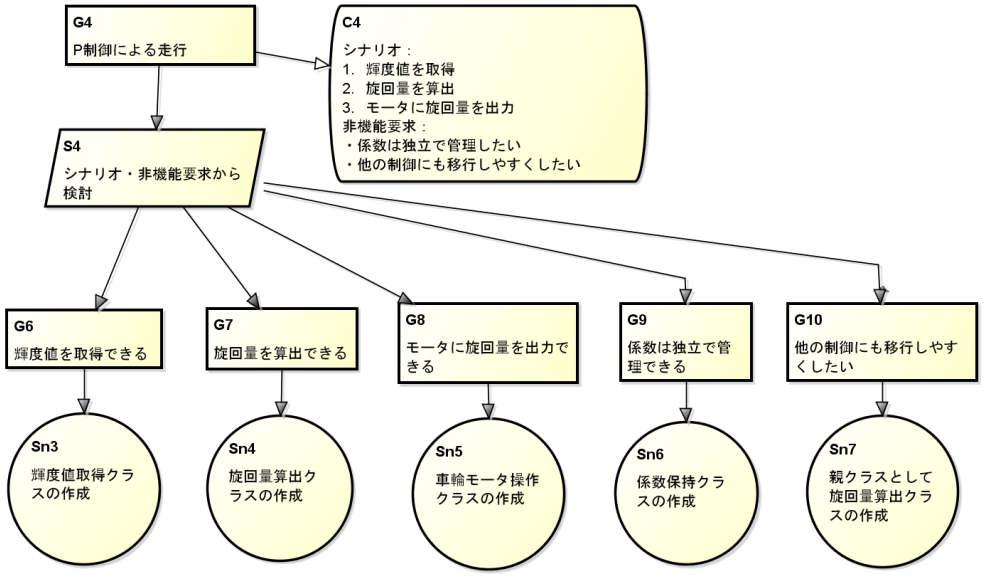


図 5.4 「P 制御による走行」に関する内部仕様 D-Case

値と取得した輝度値から走行体の旋回量を算出．最後に算出した旋回量を二輪倒立走行ライブラリに入力するというシナリオである．また，C7 の非機能要求としては「P 制御に使う係数は独立で管理したい」，「P 制御以外の制御方法にも移行しやすくしたい」という 2 点を記述した．

その後 C4 の内容からトップゴールを具体化した．例として，G6，G7，G8，G9 のノードはシナリオと非機能要求が達成できる命題を設定した．それらのエビデンスノードとして，各ゴールノードの責務を担うクラスを作成という内容を記述した．一方，G10 のノードは他のゴールノードと同じ観点で具体化されたが，エビデンスノードに関しては,「親クラスとして旋回量算出クラスを作成」と記述した．

この工程によって，各機能に必要なアーキテクチャを抽出することができた．

(4) UML 図によるモデリング

前段階で抽出したアーキテクチャの要素を UML モデルに変換する．実際に作成したシーケンス図とクラス図を図 5.5 に示す．図の左側がシーケンス図で，右側がクラス図である．

初めに，図 5.4 の Sn3，Sn4，Sn5，Sn6 のエビデンスノードから図 5.5 に示すライフラインを記述した．続いて図 5.4 の Sn7 のノードに記述されているシナリオを実現できるよう，各ライフライン間のメッセージを記述した．車輪モータ操作クラスはライントレースをするというメッセージを受けた後，旋回量を算出するというメッセージを P 旋回量算出クラスに送る．P 旋回量算出クラスは輝度値取得器クラスから輝度値を，係数保持器からは予め設定した係数を取得する．取得した値から旋回量を算出し，車輪モータ操作クラスにそれを返す．最後に車輪モータクラスはモータに旋回量を入力し，モータを駆動させる．このシーケンス図によってシナリオを満たすクラス間の関連を記述した．

シーケンス図を記述した後，ライフラインとメッセージを基にクラス図を記述した．また，クラスの操作を定義する際，各操作が返す値の型も定義した．シーケンス図に従い，輝度値取得クラス，旋回量算出クラスにはシーケンス図に記述したとおりの操作を記述した．係数保持器については P 制御用の係数を変数として保管する必要があったため，P 係数という属性を追加した．また，図 5.4 の Sn7 のノードを満たすため，親クラスとして旋回量算出クラスを定義し，P 旋回量算出クラスに継承させた．最後に車輪モータ操作クラスに関しては，シーケンス図と同様の操作を記述したのち，属性として旋回量と速度値を追加した．2 輪倒立走行には旋回量と速度値の 2 つのパラメータが必要であるためである．

(5) 実装・各エビデンスの取得

対象システムのクラス図を記述した後，コーディングを行った．

コーディングの後, 機能 D-Case で示したエビデンスノードを参照して，各機能の検証を行った．例えば，図 5.3 の Sn1 に記してある「計測した輝度値が一定の値に収束するか検証」を実行するため，走行体をコースで走らせた際，輝度値を取得しその時間ごとの変化を示したグラフを得た．その結果，輝度値が収束した様子が確認できたため，Sn1 に関連付けられた機能「P 制御による走行」は達成できたと言える．このように, 機能 D-Case で示したエビデンスを達成できるよう，実装を行った．

(6) D-Case およびモデルの洗練

モデリングや実装の段階などで，足りない機能があったため D-Case，モデルを修正した．図 5.6 に修正した機能 D-Case を示す．

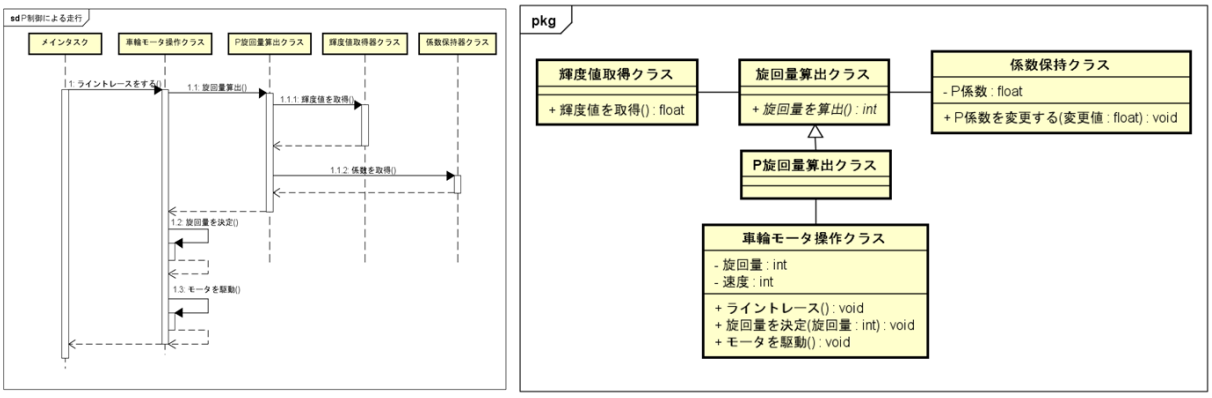


図 5.5 作成したシーケンス図とクラス図

図 4.3 の機能 D-Case では G5 の「カーブ上にいるときは減速する」というゴールノードが存在したが，図 5.6 の機能 D-Case ではそれが削除してあり，代わりに別のゴールノードが記述されている．G5 のゴールノードは，カーブを走行する時，走行体がコースラインを外れる可能性を考えて設定したゴールである．しかし，「P 制御で走行する」機能を実装したところ，カーブ上でもコースラインを外れずに走行できることが判明した．その代わり，コース上に存在する灰色ゾーン上を走行する時，動作が不安定になり，コースラインを外れることがあった．それと同時に，環境光の違いで光センサが得られる輝度値が一定しないため同様に動作が不安定になるケースが存在することも判明した．これらの事実から，機能 D-Case に G11「灰色ゾーンでもコースに沿って走行できる」と G12「環境光の変化に対応できる」というゴールノードを設定し，内部仕様 D-Case やモデルを作成し直すことで，機能を達成できるように改善した．

5.3 プロセス適用の結果

示したプロセスを実践した結果，図 5.5 のモデルが得られた．また，そのモデルを基にコーディングを行い，その結果図 5.6 の D-Case で示したエビデンスをすべて獲得することが出来た．最後に走行体をコースで走らせた結果，コースから逸れたり転倒したりすることもなく，24.3 秒でゴールすることが出来た．したがって，提案プロセスで要求を満たしたラインとレースシステムを開発できた．

6. 考察

6.1 D-Case を適用することによる効果

本研究では，システムに対する要求を D-Case を用いて分析し，それを基に MDD を行った．以下に適用したことで得られた効果を示す．

- モデリングの根拠・ストーリーの明示

  D-Case を用いることでトップゴールを満たすことを念頭に置き，トップゴールに求められる機能を検討した．更に，検討した機能にそれぞれ内部仕様 D-Case を作成したため，モデリングに根拠が生まれ，少ない労力でモデルを作成することが出来た．加えて，モデルを作成する過程・根拠が図として表現されていることから第三者に対してモデルを見せるようなことがあっても，スムーズなモデルの説明を実現できると考えられる．

- モデルの修正の容易化

  開発途中で修正する必要が発生した場合，機能 D-Case や内部仕様 D-Case を通してコードを修正することが出来た．

  例えば，初期の D-Case では走行自体のみを考えた設計になっていたが，開発を進めていく過程で，走行体のテイルに関する動作について D-Case を記述していないことに気が付いた．しかし，素早く D-Case を使って必要な機能を抽出することが出来たため，少ない時間でモデルを修正することが出来た．このように，D-Case を用いれば，手早いコードの修正が可能である．

6.2 D-Case を適用したことによる課題

- 機能 D-Case のゴールノードはどの程度まで分解するべきか

  本研究ではモデルを作成する足掛かりとして，機能 D-Case を作成した．機能 D-Case の作成では，トップゴールを基に，システムに求められる具体的な機能を抽出する．この工程において，機能はどの程度まで具体的に分解するべきなのか不明瞭である．

  例えば，図 5.6 に示した機能 D-Case に「40 秒以内にゴールする」というゴールノードがある．その機能 D-Case では 40 秒でコースを走りきるためのライントレースの走行方法を列挙し走行方法毎に検討し，最終的に「P 制御による走行」という機能 D-Case の末端のゴールノードが得られた．しかし，「40 秒以内にゴールする」のノードを代わりに末端のゴールノードにすることも可能である．チームで開発する場合，この「機能 D-Case のゴールノードはどの程度まで分解するべきか」という問題は，チームメンバー間での意識の違いが生まれ，開発の障害となる可能性がある．

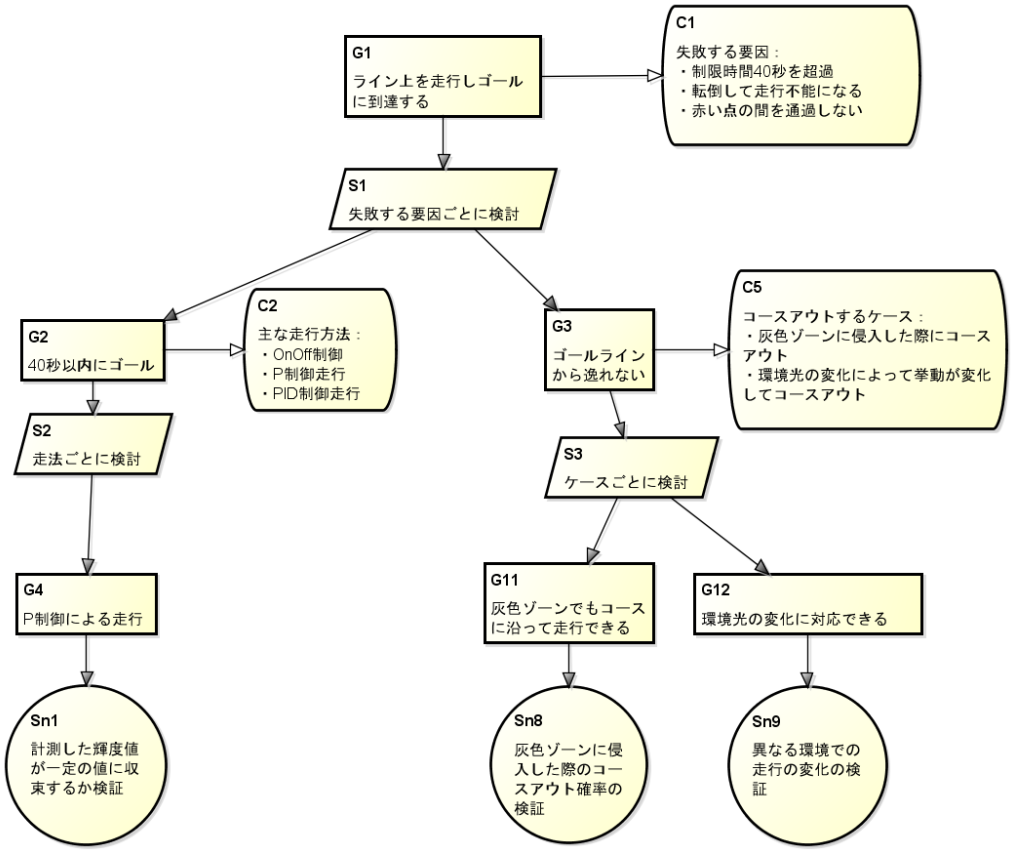


図 5.6 最終的に得られた機能 D-Case

7. おわりに

本研究ではモデル駆動開発における D-Case の有用性を示すため，D-Case を活用するプロセスを提示し，そのプロセスの適用事例を示した．実際にライントレースシステムを設計・開発することで，D-Case を用いた MDD の実践例を提供した．これによって，D-Case を用いた MDD の支援となる．

今後の展望としては，このプロセスで作成した UML モデルの評価などがあげられる．

参考文献

[1] OMG/MDA http://www.omg.org/mda/，(2016/5/11)
[2] ISO/IEC 9126-1:「Software engineering - Product Quality - Part 1」，Quality model, 2011
[3] 情報処理推進機構:「組込みシステムの先端的モデルベース開発実態調査」調査報告書，(2012)
[4] 所眞理雄：「DEOS~変化し続けるシステムのためのディペンダビリティ工学~」，近代科学社(2014)
[5] 土樋裕希：「D-Case を用いたゴール共有による開発プロセスの適用 ~ET ロボコンでの思考と成果~」，先進的な設計・適法事例報告書(2015)
[6] 野田夏子:「モデル駆動で開発しよう –実適用における課題と先進技術」，ソフトウェアエンジニアリングシンポジウム(2014)
[7] ET ロボコン EV3 環境構築ガイド：https://github.com/ETrobocon/etroboEV3/wiki，(2016.5.10)
[8] HOME –LEGO MINDSTORM: http://www.lego.com/ja-jp/mindstorms，(2016.5.7)
[9] LeJOS, Java for LegoMindstorms，http://www.lejos.org/，(2016.5.7)
[10] 2 輪型倒立振り子ロボットバランス制御プログラム：https://github.com/ETrobocon/etroboEV3/blob/master/SampleCode/EV3way_leJOS_sample/ev3sample/src/jp/etrobo/ev3/balancer/Balancer.java，(2016.5.9)