7J-04

組み込みシステムにおける設計制約の導出手法

西川　俊†　　松浦　佐江子‡

芝浦工業大学　システム理工学部　電子情報システム学科†‡

1 はじめに

ソフトウェア技術者が，組み込みシステムの要求分析段階において，ハードウェアに関わる部分を含んだシステム全体の要求を分析する場合，不可能なハードウェアの使用方法という設計制約を分析結果に含んでしまうことがある．これを回避するために，ハードウェアの仕様書の確認やテストプログラムを作成してハードウェアの評価実験をするが，その確認内容であるテスト項目の作成は技術者の経験則による．そのため確認漏れが発生する可能性があり，開発における手戻りの原因となる．

本研究では，要求分析段階でシステム全体の要求を実現するサブシステム間の動作の流れを表すワークフローを作成し，これを満たし，かつハードウェアの仕様書に適合するユースケースを分析することで，設計制約を明らかにし，適切なテスト項目を出力する手法を提案する．

ここではAPI(Application Programing Interface)が既知である開発済みのハードウェアを使った開発を対象とし，要求分析にはUML(Unified Modeling Language)を用いる．

2 適用事例

事例として芝浦工業大学の授業課題である荷物自動搬送システムを使用する．このシステムはLEGO MINDSTORMS NXTが3台，PCが1台で構成され，NXTが組み込みシステムである．配達の流れは，受付所(PC)にある荷物を収集担当ロボット(NXT)が受け取り，中継所(NXT)へ移動する，中継所は収集担当ロボットから荷物を受け取り，配達担当ロボット(NXT)に渡す，配達担当ロボットは受取人宅(PC)へ移動し，荷物を渡すというものである．なお，荷物はデータで仮想的に作成され，受け渡しには Bluetooth 通信を使う．コーディングには，leJOS NXJ を用いる．

この事例で想定する設計制約は，使用できるセンサが規定されていることと，Bluetooth 通信のコネクション確立要求が NXT から PC へ送れないことである．

A Derivation Method of Design Constrains in Embedded System

†Shun Nishikawa ‡Saeko Matsuura

†‡Department of Electronic Information System, Collage of System Engineering and Science, Shibaura Institute of Technology

3 提案手法

3.1 ハードウェアクラスの作成

ハードウェアクラスの作成にはクラス図を用いる．ハードウェアの API を参照して，必要となる値，関数を属性，操作に加えていく．これにより，ハードウェアの基本的な使用方法を学ぶ．また，これがハードウェアの仕様になる．事例のハードウェアクラスを図 1 に示す．

図 1. ハードウェアクラス

3.2 ワークフローの作成

システム全体の要求を実現する動作系列をアクティビティ図で表したものがワークフローである．ワークフローはパーティションによって複数のサブシステムで構成することができ，組み込みシステムはその中の 1 つとして扱う．これは，組み込みシステムと他のサブシステムとの協調動作を表すためである．事例のワークフローを図 2 に示す．事例では組み込みシステムである収集担当ロボット，中継所，配達担当ロボットがサブシステムとしてそれぞれのパーティションに割り当てる

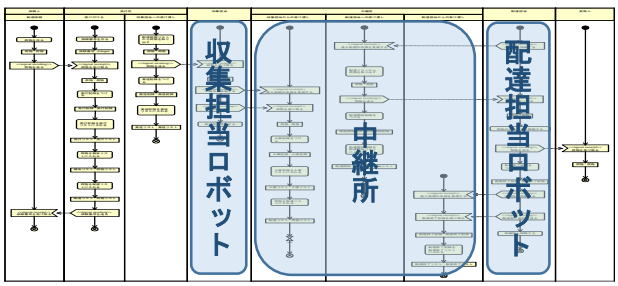

図 2. ワークフロー

3.3 ハードウェア依存アクションの抽出

本研究では，アクティビティ図のハードウェ

アを使用するアクションをハードウェア依存アクションと呼ぶ．サブシステムのユースケースはワークフローのアクションの1つ，もしくは複数のまとまりである．事例の収集担当ロボットのワークフローは，受付所から**荷物を受け取り**，**中継所へ移動し**，中継所に**荷物を渡し**て，**受付所へ移動する**ことであり，太字がユースケースとなる．このなかで，「中継所へ移動する」と「受付所へ移動する」は「ライントレースをする」というハードウェア依存アクションを含む共通のユースケースを含む．

3.4 アクションの詳細化

3.3節で作成したハードウェア依存アクションを含む共通のユースケースをアクティビティ図で表す．このとき，ハードウェアクラスで学んだ基本的な使用方法に則した動作系列にする．図3に収集担当ロボットが持つ移動するためのユースケースである，「ライントレースをする」を表したアクティビティ図を示す．ただし，この段階では入出力ピンとノートは書いていない．

3.5 アクションの分析

図3の「光量を取得する」というアクションはハードウェアクラスの「光量を取得する」という操作で実現できる．また，この操作の戻り値は，コースの明るさに依存するので，明るさの段階を「照明が全灯・照明が消灯」のように定義し，これらをノートにまとめる．「モータの出力値を設定する」という操作では引数に0から100までの整数であるという条件がある．このように，ハードウェアクラスの操作とアクションを対応させて分析する．その後，入出力ピンで操作による値の流れを表す．流れのなかの，「灰色線を通っているか判定する」というアクションでは，「光量を取得する」という操作の戻り値を利用するが，その利用方法により0から255までの整数でなければならないという条件がある．これもノートにまとめる．このように分析していき，アクションがハードウェアクラスの操作で実現できることの確認をするとともに，操作によってやり取りする値が正しいとするのに必要な条件を明らかにする．

3.6 テスト項目の出力

テスト項目はテストプログラムと，その実行手順をまとめたものである．図3では，テストプログラムの作成に，まず「モータの出力値を算出する」というアクションの関数を作成する．これで，「光量を取得する」，「モータの出力値を算出する」，「モータに出力値を設定する」という入出力ピンの流れにあるアクションと対応する関数ができた．関数の戻り値を次の関数の引数に与えることで，入出力ピンの流れを表すことができ，これがテストプログラムになる．このプログラムの実行手順は「コースの白・黒・灰色部分に光センサを当てて，光量は0から255の整数か，出力値は0から100の整数かを確認する」ということになる．また，「光量を取得する」の操作がコースの明るさに依存することが分かっているため，テストは「照明が全灯」と「照明が消灯」の環境設定をして行うことも合わせて実行手順に記す．

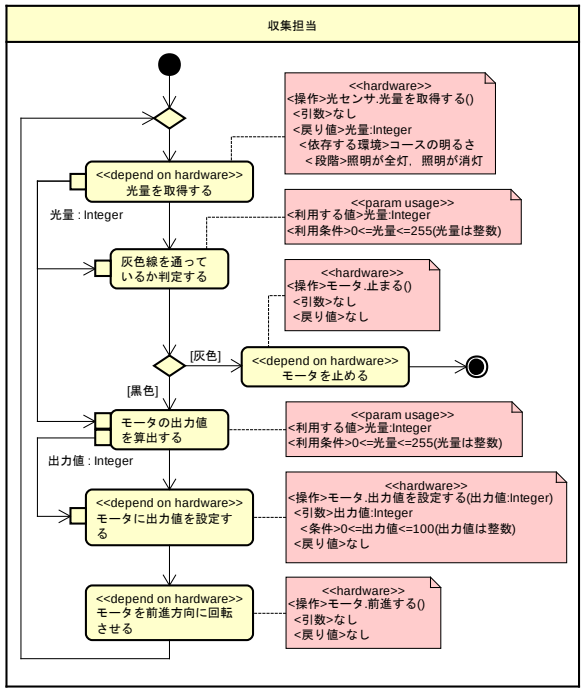

図3. ライントレースをする

4 まとめと今後の課題

適用事例において，NXTからPCへのコネクション確立要求の送ることができないことは，提案手法の3.5節の時点で確認できた．3.5節のハードウェアクラスからアクションを実現する操作を対応させる場面で，PC側がコネクション確立要求を受ける操作を持っていなかったためである．この設計制約は，配達担当ロボットから受取人宅へ荷物を渡す部分で見つかった．ワークフローで荷物を渡す方向が，配達担当ロボットから受取人宅であったため，潜在的にコネクション確立要求の送信も同じ方向だと思いこみ，その結果，NXTからPCへ，コネクション確立要求を送ってしまったことが原因である．

今後は，他のハードウェアの動作を待ってから動作を始める時間イベントアクションがあった場合のテストにも対応できるように改良を加えたい．

5 参考文献

[1]astah* , http://astah.net/