LEGO MindStorms EV3 を用いて工学的にプログラムを開発する方法の提案

2014SE059 増田 大輝　2014SE089 柴田 心太郎　2014SE114 矢澤 拓海

2016 年 12 月 7 日

1　研究背景と目的

「ソフトウェア開発」の分野において、解決すべき問題に対してどのような種類の問題であるかを分析し、どのような案が最善であるかを考え、探る能力は大変重要なものとなっている。我々の日常ではパソコンを始め、扇風機、ゲーム機、テレビなどを開発する過程においてもこれらの要求文書や、DFD などの制作においてこのような能力が必要とされる。しかし, 高校までの教育課程においてそれが育つような環境であったとはいいがたい。学習のほとんどが「教師から、もしくは教科書から与えられた問題を解く。」という形である。このような問題は最初から最善解を習っていて、それに倣って問題を解いていくという形であった。そのためほぼ全ての問題において、唯一の解が存在し、それ以外が間違いという１か０かの問題をこなしてきた。そのため大学の講義でのこのような自主的な開発という分野のものを苦手とする学生は少なくない。

こういった問題を解決するにはどういった方法があるだろうか。私たちが考えたところ、問題解決能力を高めるための授業方法の改善、学習課題の改善および、学習環境の改善を図っていくことが最良であると結論づけた。以上のことより、今回ははじめてそのような技術を学ぶ、第二学年秋学期「ソフトウェア工学実習１，２」において, 基本となる開発プロセスを提案することにした。理由は、EV3 という、一見おもちゃのような機械から学ぶことによって生徒の興味を引きつつこの様な能力を養うことができたらよいと考えたからである。

2　講義内容

2.1　今年度の講義の内容と改善

今年度の講義内容 [1] を確認すると (図 [i]) のような構成となっている。主な構成として、第 1 回から第 4 回までは EV3 を動かすにあたってのプログラムに使うブロックの種類を動画や写真を用いた説明とそのブロックを使った簡単な課題が設けてあり、その課題を取り組むといった構成となっている。第 5 回以降では、第 1 回から第 4 回で得たブロックの知識を使い、それをもとに工学的なプログラムの開発及び課題に取り組むといった構成である。

ここで第 5 回以降の課題において今年度の生徒の提出したユースケース図やイベントフローなどがとりわけ不十分であったことが判明した。これらの UML は、誰がつくっても同じ結果になるように体系的に設計することが理想である。しかし、今年度の生徒のレポート課題を参照すると、多くの生徒が UML の書き方を分かっていないように感じた。私たちが感じた感想は以下の通りである。

- アクタや状態などの概念を理解しておらず、間違った枠に入れているように感じる。
- 教授がホワイトボードに描いた解答例のみを参考にしている生徒が多く、これらの図が何を示しているか分かっていないまま描いている。
- 要求分析を感覚でおこなっている生徒もみられ、不必要な要求を記述していることもあった。

これらを改善するにあたって、私たちのの新しく提案するカリキュラムは, 始めに行う EV3 のブロッ

クの解説は一年次の理工学概論 [SE] でも行っている内容なので、その授業の時間を切り詰めた。そして、切り詰めた分で UML の書き方についての解説実践する時間を設けたカリキュラムを今回は提案したいと思う。(図 [ii])。

第 1 回: スタートアップ
第 2 回: EV3 を動かしてみる
第 3 回: モーター、センサー
第 4 回: フローブロック
第 5 回: 工学的なソフトウェア開発
第 6 回: ライントレース、表示、スイッチ、データ操作、(アンケート)
第 7 回: 応用問題

図 [i] 今年度ソフトウェア工学実習カリキュラム

第 1 回: スタートアップ、EV3 を動かす (モーター)
第 2 回: モーター復習、センサー
第 3 回: センサー復習、フローブロック
第 4 回: フローブロック復習、工学的思考、UML の解説
第 5 回: UML の復習と具体的な実践
第 6 回: ライントレース、表示、スイッチ、データ操作、(アンケート)
第 7 回: トヨタの設計 (応用問題)

図 [ii] 提案する新カリキュラム

2.2 新しく設けた講義の内容案

図 [ii] のように第 4 回, 第 5 回において、この 2 コマで UML の解説 実践を行う形を実施するにあたり、まず第 4 回については工学的思考の概念や UML の解説のみの時間に特化させた方が良いと考えた。そうすることにより、基本となる考え方や体系を理解させるためである。 以下に第 4 回の流れを記述する。

まず、今回利用するユースケースなどの一番大きな枠組みである UML について紹介する。UML はオブジェクト指向、つまり「名前」や「登録番号」などのデータと、「前へ進む」、「記録する」などの処理の集まりに基づいてソフトウェアモデルを記述する言語である。そして、ソフトウェアに対する要件、ソフトウェアの構造／振る舞い、実装設計などに利用でき、その目的によって 13 個の図に分かれており、各図の用途は、開発方法論によって決められる。UML では、どのような方法論でも使えるように 設計要素を用意しているので、それぞれの現場に合わせて利用することが可能である。開発プロセスの工程によって用いる UML は異なる。

まず、システムの振る舞いから見ていくことにする。システムの振る舞いというのは、システムが外から見てどんなふうに動作し、反応するかということである。このシステムの振る舞いを図で表すために「ユースケース図」を使用する。
ユースケース図の目的は以下のとおりである

- システムの外部と内部との境界をはっきりさせること。
- 利用者や、外部の自称がどのようにシステムを利用するか示したものであること。

次にユースケース図の作成方法を記述する。
ユースケース図はアクターとユースケースという 2 つのアイテムを使って、モデルを作成する。

アクター

- システムのユーザが果たす役割を表す。
- システムと活発に情報交換をしたり、システムから受動的に情報を受け取ったりする。
- 人間、ハードウェア、外部システムがアクターになりえる。

ユースケース

- ユースケースは、アクターとシステムとの間の対話をモデル化する。
- システムのユーザがシステムを利用して遂行する単位業務のひとつを抽象化したもの。
- ユースケースは「○○する」のように、はっきりと述語を記述する。

例として、今回の要求は「EV3 が前進し、障害物があった場合音を鳴らして一時停止し、障害物がなくなったら前進を再開する。 5 メートル進んだら停止する。」とすると、役割を持っていると思われるアクターとしては、システムから情報を受けとるため、「EV3」が挙げられる。では,「EV3」の立場からシステムの機能を発見し、アクターとユースケー

スの関わりについて考えていく。今回の要求から動詞を取り出すと、「前進する」「音を鳴らす」「一時停止する」「停止する」の４つである。しかし、この場合、一時停止するために「障害物を検知する」というユースケースが必要になる。そのため、図にすると以下の通りになる。(図 [iii])

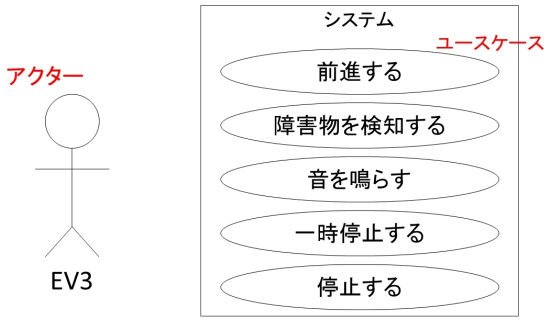

図 [iii] ユースケース

補足だが、ユースケースとアクターとの境界線を明示するために、ユースケース群 (ユースケースの集まり) を大きな長方形で囲むと図が見やすくなる。そして長方形の内側の上部には、システムの名前を明記しておくとよい。

次にユースケースを用いてイベントフローを作成する。イベントフローには、ユースケースがいつ、どのように開始・終了するか、アクターとユースケースとの相互作用について文書で記述することである。正常な動作をするときのことを基本系列であらわし、例外については、代替系列として記述する。アクターには棒人間を使う。仮にアクターが人でなかったとしてもかまわない。イベントフローを記述するときは、「○○が××する」のように主語と述語をはっきりと記述し、ユースケースの順番ごとに番号をつける。今回のユースケース図からイベントフローを記述すると以下の通りになる。

基本系列

1　EV3 が前進する。
2　EV3 が障害物がないことを検知する。
3　EV3 が障害物がないことを検知する。
4　EV3 が前進する。
5　EV3 が停止する。

代替系列

2.1　EV3 が障害物を検知する。
2.2　EV3 が一時停止する。
2.3　EV3 が音を鳴らす。

代替系列の場合、ほとんどが基本形列の間に入るので、「○.1」のようにその区間の場合分けであることをはっきりと明示したほうがよい。

最後に「状態遷移図」について記述する。状態遷移図とは、一つの「もの」の状態の変化に注目した図のことであり、普遍的なオブジェクトの状態の変化を表現する。この図は大きく分けて、「状態」「イベント」「アクション」「状態遷移」の４種類で構成されている。「状態」とは、そのオブジェクトが存在し得るひとつの状況、条件を示し、時間とともに常に変化するものである。特別な状態として「開始状態」と「終了状態」がある。(終了状態はサイクルし続けるシステムの場合不要となることがある)「イベント」とは、ある時点において起きる出来事で、遷移するきっかけとなるものである。イベントとアクションの境界は「 / 」で表す。「アクション」とは、遷移に付随する操作で、「起動する」「停止する」のように動作するものである。「状態遷移」とは、あるイベントが満たされたり、ある条件が達成されたりした場合に、他の状態へ移行することを示したものである。
これらのことから状態遷移図を記述すると以下の通りになる。(図 [iv])

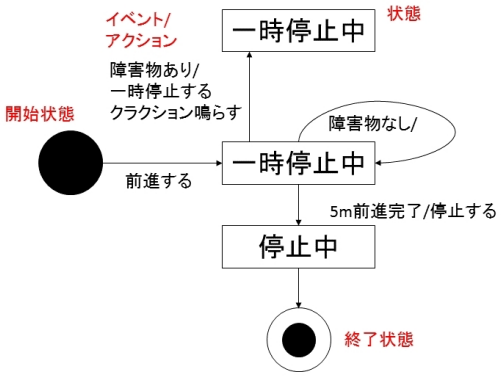

図 [iv] 状態遷移図

今回は作成しなかったが、状態遷移図を製作する場合、シーケンス図で示されたメッセージがイベントになる場合が多いので、シーケンス図を記述してから状態遷移図を製作するとわかりやすい。

以上のことを第 4 回で行い、その基本的な工学的思考を用いて実際にプログラムは書かずにモデルのみを講義時間中にミニレポートとして作成、提出させる。そうすることにより、ユースケースを書くことの大切さを理解するとともに今後プログラムを実装するにあたり、そのプログラムと図とをに対応付ける力を付けさせるためである。

第 5 回では学生の解答を元に解説を行う。これは学生の理解度を把握し、理解の薄いところを重点的に補足することができる。以下に第 5 回の流れを記述する。学生の解答の中から模範的な解答を選び、他の学生ができていない部分を解説する。模範解答として選出する際の基準は以下の通りである。

- 図全般において UML の通りに記述しているか
- ユースケース図においてアクタとユースケースの区別ができているか
- 障害物の検知など必要なユースケースを抽出できているか
- イベントフローにおいて例外処理を代替系列として記述できているか
- 状態遷移図においてイベントと状態を対応付けられているか
- 他の図との矛盾がないか

これらの基準から選んだ解答を元に、上の基準についてどこをどのように満たしているか、他の学生がどのように間違えているかを解説し復習させる。これにより、理解できてない部分の把握と理解している部分の自信を与えることが可能である。

復習の後に、プログラムとの対応のさせ方を説明する。状態遷移図における状態に対応するプログラムを作成し、イベントによってプログラムを分岐、切り替えることにより要求に対応したプログラムが完成する。思い通りに動かない場合はプログラムだけでなく図を見直すべきであることを強調する。以上のことを講義し、学生に図の見直しとプログラムの実装をさせる。プログラムは学生自身が作成したイベントフロー図、状態遷移図に対応する形で作成させ、レポートとして提出させる。これにより工学的なプログラムの作成法を理解することを期待できる。

3 今後の発展性

図 [ii] において第 4 回と第 5 回に UML について解説、実践することにより理解度が深まり第 6,7 回において理解した上で課題に取り組めるのではないか, また、この講義が後の 3 年次の講義を学ぶ上での基盤となるだろうと考える。

4 参考文献

[1]: 2016 年度 ソフトウェア工学実習 講義資料
http://tetsuo.jp/lecture/sw_jisyu/2016/
[2]: ユースケース図の作成
https://www.ogis-ri.co.jp/otc/hiroba/UMLTutorial/analysis/do_work/dowork1_1.html
[3]: 中島 進：豊かな人間性と創造力を養うものづくり教育に関する研究 (第六報)
-問題解決能力を養う技術科カリキュラムの改善と教材開発-