6N-01

ソースコードからの UML モデル自動生成による ソフトウェア概略の理解支援

滝本　竜海†　　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

ソフトウェアの変更の際にソフトウェアの理解が必要であるが、大規模になるほどソースコードを読み解くことは困難である。ソースコードをリバースして作成できる UML モデルとして、クラス図やシーケンス図がある。クラス図では、クラスの構造的な情報の可視化が可能であるが、振舞いフローを見ることはできない。シーケンス図では、クラス間の振舞いフローを見ることができるが、ソースコードと同等な情報量となるため、適切な抽象化が必要になる。ソフトウェアの理解には、抽象度の高いレベルでの理解から、抽象度のレベルを下げていくことで、ソフトウェアの仕組みが理解しやすくなる。本研究では、ソフトウェアの理解のため、抽象度の高い状態の理解からの段階的な理解の支援を目的とする。

2　ソフトウェア理解支援の方法

ソフトウェアとは、ユーザの要求を実現するための機能の塊である。オブジェクト指向での機能は、メソッドと呼ばれる振舞いのフローによって定義される。メソッドの振舞いもまた、振舞いフローで定義される。機能を知るには振舞いフローを表現し、メソッドを知るには、抽象度（メソッドのネストの深さ）を変更した振舞いフローを表現する必要がある。ソフトウェアの理解には、ユーザが触れられる「機能の種類」、「可変的な抽象度での振舞いフロー」を知ることが理解の手助けとなる。本研究でのソフトウェア概略とは、「機能の種類」と「可変的な抽象度の振舞いフロー」を可視化した図とする。「機能の種類」を知るための図として、機能をまとめる UML モデルのユースケース図を使用する。「可変的な抽象度の振舞いフロー」を知るための図として、メソッドの振舞いを表現するシーケンス図があるが、フロチャートに似通っていて、慣れ親しんでいるアクティビティ図で可視化をする。

3　UML モデルによる概略の表現

3.1　ツールの流れ

本研究では、java ソースコードを入力とし、ユースケース図とアクティビティ図を出力するツールを開発した。ツールでは ASTParser[1]を用いて、ソースコードからメソッドの流れとインスタンス生成のデータを抽出し、そのデータから PlantUML[2]の構文を生成し、図を生成する。

3.2　ユースケース図の作成

ユースケース図には、ユースケースとシステム名を記述する。ユースケースは main 文の中のメソッド名と event などの使用者が呼び出すメソッド中のメソッド名を用いて生成する。システム名は指定したディレクトリ、またはファイル名とする。アクターの特定には、どの利用者がどの機能を利用するかを判定するには、権限設定の箇所を解析する必要があるため、今回はアクターの生成は行わない。

3.3　アクティビティ図の作成

アクティビティ図には、メソッドの呼び出し関係にループ、分岐、インスタンス生成を加えた抽象度の変更可能な処理フローを生成する。

アクションノードには、メソッド呼び出しステートメント「object.method();」の object のクラス名と method の名前を表示する。API は object 部分が null になるため、メソッド名のみを表示する。オブジェクトノードには、インスタンス生成のステートメント「A a = new A();」の A のクラス名を表示する。パーティションには、ノードを呼び出したクラスの完全修飾名、デジションマージノードにループまたは分岐の条件式を表示する。フローに登場するすべてのノードには、その出現順序と階層関係がわかるように、ユニークに識別可能な ID を付与する。これをメソッドを指定して抽象度を変更する際に、同一名のメソッドが存在した場合にも対応するために用いる。

3.4　抽象度の変更

アクティビティ図では、デフォルトで出力される図はユースケースとなるメソッドの body となる部分の API も入れた振舞いフローである。引数や返値、アクセス修飾子はメソッドの振舞いを知るためには必ずしも必要ではない。そのため、図 1 に示す ComboBox で ID を指定することで情報を閲覧できるようにした。

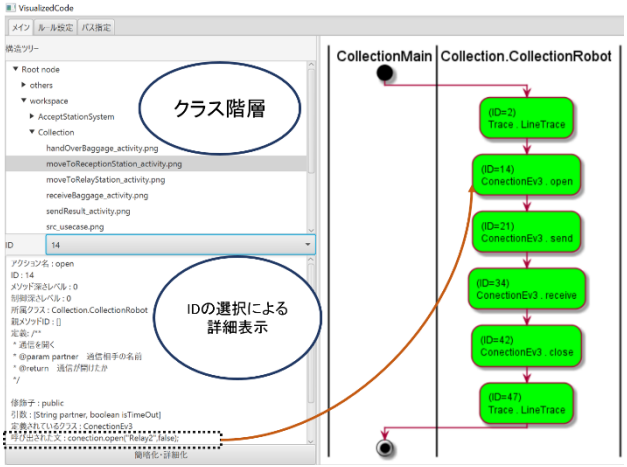

図 1 理解支援ツールの GUI

抽象度を変更することで、メソッドの詳細を表示した、各使用者にとって適切な抽象度のモデル図に変更できる。抽象度変更の方法はつぎの二種類の方法がある。

- メソッド指定による抽象度の変更

指定されたメソッドの定義となる振舞いフローを表示する。特定のメソッドの振舞いが名前からだけでは理解できないときに、メソッドの振舞いを表示できる。

Support for Understanding Software Online by Automatic UML model Generation from Source Code
†Tatsumi Takimoto ‡Saeko Matsuura
†‡Department of Electronic Information System, Collage of System Engineering and Science, Shibaura Institute of

● new 指定による抽象度の変更

newによるインスタンス生成を表示する。データのインスタンス生成のポイントを表示できる。

4 使用事例：荷物自動搬送システム

荷物自動搬送システムとは 3 台の EV3 と 1 台の PC で、図 2 のような環境において荷物を自動搬送するサービスを実現するシステムである。受付所で荷物を受け取り、収集担当ロボットが中継所へと配送し、配達担当ロボットが中継所から受取人宅へと荷物を送り届ける。荷物自動搬送システムは Java で開発され、コード行数は、5837 行である。今回、事例として注目する収集担当ロボットの動作は、受付所から荷物を受け取り、中継所へ荷物を渡した後に受付所へと戻ってくる。

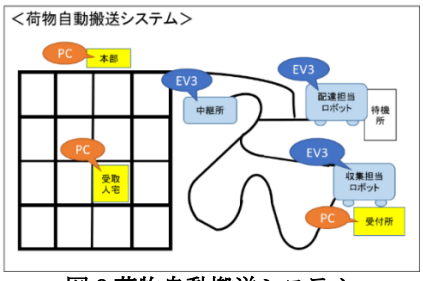

図 2 荷物自動搬送システム

収集担当ロボットのソースコードから生成されたユースケース図は、図 3 のようになる。

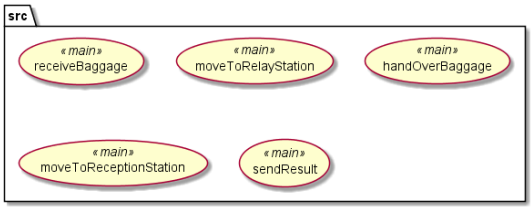

図 3 収集担当ロボットのユースケース図

このユースケース図から収集担当ロボットは、機能として、荷物を受け取る、中継所へ移動する、荷物を手渡す、受付所へ移動する、結果を送信するという機能を持つことが読み取れる。また、<<main>>というのは main 文で呼ばれた処理というのが分かるように記述されている。

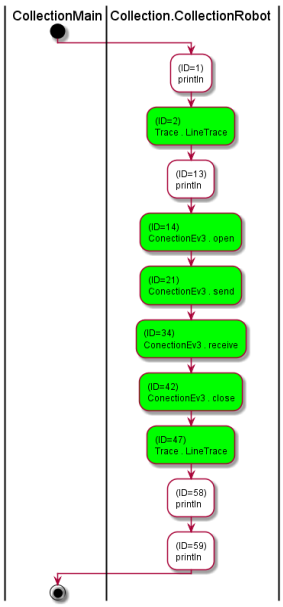

図 4 moveToReceptionStation のアクティビティ図

「中継所へ移動する」というメソッドのアクティビティ図を図 4 に示す。アクティビティ図を見ると、中継所から受付所へと移動するだけではないことが分かる。Trace クラスの LineTrace メソッドで道を辿る。ConectionEV3 の open、send、receive、close が通信の開始、送信、受信、終了をしている。移動をする途中で通信を行っており、収集担当ロボットと配達担当ロボットが衝突をするのを防止するために、通信が必要となったことが解った。このように、作成された図をユースケース図からアクティビティ図へと辿っていくことで、ソフトウェア概略の理解を支援する。また、「中継所へ移動する」という機能はこれらのメソッドによって定義され、その定義が Collection の CollectionRobot クラスにあると分かる。

5 可変ルールの事例

図 4 の open メソッドを指定して詳細化をしたアクティビティ図がいかに示す図 5 である。

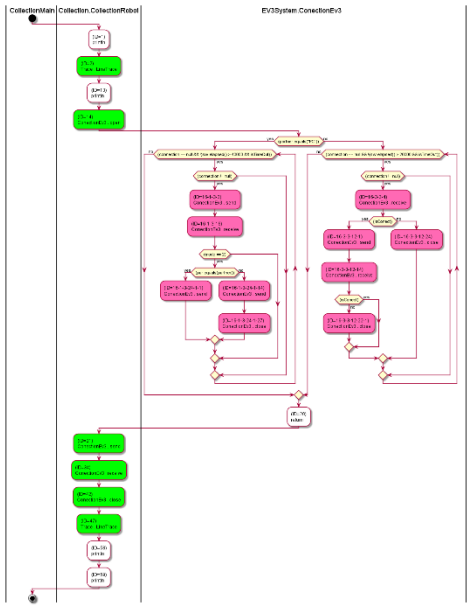

図 5 詳細化したアクティビティ図

open メソッドを指定して抽象度を変更することで、open メソッドの詳細化ができる。白色のノードから呼ばれるメソッドは存在せず、緑やピンクのノードは、そのメソッドの中に表示できる情報が残っていることを示している。表示できる情報が残っている場合の色として 5 色ある。その 5 色を順番に使いまわすことで、どんな深さになっても対応できるようになっている。また、色を変化させることによって、メソッドの入れ子関係が分かりやすくなる。

6 考察とまとめ

本研究では、ルールによる抽象度の変更によって、機能の理解を可能にした。また、図 4 にあるようにデフォルトでは API が表示されるが、理解支援ツール上で非表示設定することで非表示にできる。しかし、適切なルールを設定するためには、何度かルールを変更する必要があり、図の生成時間やデータの抽出時間も短くない。今後の課題は推奨設定の自動化による設定の簡略化と、図の生成時間とデータの抽出時間の短縮である。

参考文献

[1]ASTParser, https://projects.eclipse.org/projects/eclipse.jdt (2019/1/9 参照)
[2]PlantUML, http://plantuml.com/ (2019/1/9 参照)