モデリング教育支援のための ステートマシンモデル評価手法の提案

小形 真平[1,a)]　香山 瑞恵[1,b)]

概要：本稿では，モデリング教育支援を目的として，学習者の作成した UML(Unified Modeling Language) ステートマシン図を中心としたモデル（ステートマシンモデル）の質を端的に示す方法を提案する．提案手法では，教育向けステートマシンモデル評価メトリクスを新たに提案し，そのメトリクス値算出やさらなる教育支援のために先行研究のステートマシンモデルシミュレータ・テストツールを拡張する．提案手法の有効性評価の結果，学習者全体でモデルの質が拡張モデルシミュレータの導入後に向上したことが，提案メトリクスの値から容易かつ客観的に確認できた．このことから提案手法が有効である見込みを得た．

キーワード：モデリング，ステートマシン図，教育支援，テスト，シミュレータ，メトリクス，UML

A Method to Evaluate State Machine Models for Supporting Modeling Education

Ogata Shinpei[1,a)]　Kayama Mizue[1,b)]

Abstract: In this paper, we propose a method to comprehend the quality of state machine models focusing on UML state machine diagrams created by learners for the purpose of modeling education support. In the proposed method, we propose a new metric set to evaluate these types of state machine models, and upgrade our previous state machine model simulator / test tool to calculate the metric values and support further learning. As the result of an evaluation of the proposed method, the values of the proposed metrics were useful to easily and objectively confirm that the quality of all the learners' models improved after using the upgraded model simulator. Therefore, we believe that the proposed method is effective.

Keywords: modeling, state machine diagram, education support, test, simulator, metrics, UML

1. はじめに

MDD(Model-Driven Development) 手法や IoT(Internet of Things) 開発手法では，中核的な仕様記法の一つとして UML(Unified Modeling Language) ステートマシン図があり，そのモデリング技能はますます重要になっている．従来では，モデルからプログラムを自動生成する MDD 技術や，モデルに基づいてシステム開発を行う MBD(Model-Based Development) 技術を用いた教育がなされてきた [1], [2] が，そのような研究は少ない．我々もステートマシンモデリングの学習支援を目的に，モデルシミュレータである SMart-Learning(SML)[3] や，これを利用したモデルのテストツール SML4Instructor(SML4I)[4] を実現してきた．

学習・教育過程では，学習者の振り返りや教授者のフィードバック提供を行うために，学習者が作成したモデルの質の変化を把握することは重要である．本稿でのモデルの質とは，与えられた文法を正しく用いた，要求を満たす，規模や複雑度が適度なモデルとなっているかどうかを指す．JIS X 25010:2013 に照らすと，この質は機能完全性を中心とした機能適合性や，解析性や試験性を中心とした保守性に係わる．そして，その質が高められるよう学習者のモデリング技能を向上することが教育目的となる．しかしながら，ステートマシンモデルの質を端的に示す有効な方法が未確立であり，その把握に時間がかかる問題がある．

そこで本研究では，ステートマシンモデルの質を容易かつ客観的に把握するための教育向け評価メトリクスを新たに提案し，さらに文法チェックのための SML の拡張と，メ

[1] 信州大学
Shinshu University
[a)] ogata@cs.shinshu-u.ac.jp
[b)] kayama@cs.shinshu-u.ac.jp

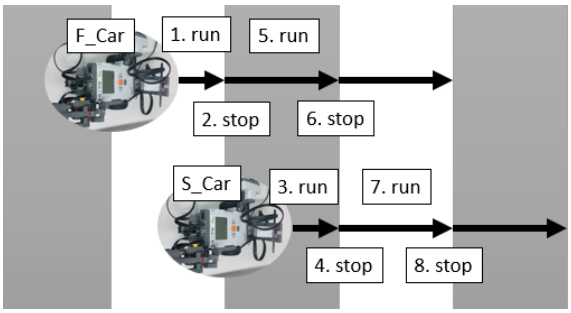

図 1 モデリング課題における協調動作ロボットの動作イメージ

トリクス値算出のための SML4I の拡張を行った．その後，信州大学の授業において，学習者に課したモデリング課題の解答例と答案などに基づいて提案メトリクスの値を算出して，SML の教育的効果を評価した．評価の結果，拡張 SML の導入により学習者全体でモデルの質が向上していたことが提案メトリクスの値から容易かつ客観的に確認できた．このことから提案手法が有効である見込みを得た．

2. 準備

2.1 ステートマシンモデル

UML ステートマシン図は，離散的な状態とその遷移で振舞いを表す [5]．状態には，その状態で実行する do アクティビティなどの属性があり，遷移には，遷移のきっかけとなるトリガや，遷移条件となるガード，遷移時の振舞いとなるエフェクトなどの属性がある．状態の遷移は，システム内外で生じるイベントで実行される．本研究では，複数のステートマシン図を中心として一つのシステムを表すモデルをステートマシンモデルと呼ぶ．さらに，本研究のステートマシンモデルは DSL (Domain Specific Language)，通信，スクリプトという特徴をもつ．

図 1 は，協調動作を行うロボット二機の動作イメージであり，これをモデリングする課題が本稿での説明事例となる．二機（F_Car と S_Car）は，F_Car から直進を始めてコースの色（白または灰）が変わる度に停止して S_Car に停止を通知し，通知を受け取った S_Car が直進を始める，というサイクルを交互に繰り返す．図 2 と図 3 はそれぞれ，F_Car と S_Car の振舞いを表したステートマシン図例である．二機とも同様な振舞いを持つが，停止と前進を交互に行うために開始疑似状態（黒丸）直後の振舞いが異なる．

2.1.1 Domain Specific Language

DSL は，コード片に紐づけたドメイン特化語を用いて設計記述を作成し，そこから実行可能なプログラム等の後続成果物を自動生成する手法である．たとえば，DSL は Sirius[6] といったモデリングツールで採用され，ステートマシン図にも用いられる [7], [8]．したがって，DSL を学ぶことは，MDD 手法を実用することに役立つ．

本研究では，DSL を do アクティビティとトリガに導入している．たとえば do アクティビティには DSL の語群 {run, stop} から一つを与える．先行研究のツール [9] は，このように作成したモデルをプログラムに変換できる．

2.1.2 通信

通信は，異なる複数の遷移を同期させて実行するメカニズムであり，SPIN[10] や UPPAAL[11] などのモデル検査技術にも導入されている．一般に，ステートマシン図はクラスやサブシステム等のシステム部品（群）ごとに描かれ，図間の連係によりシステム全体の振舞いを表す．したがって，通信を学ぶことは，モデル検査技術の利用や，多数のデバイスが相互作用する IoT システムの設計に役立つ．

本研究では，UPPAAL[11] 同様にメッセージの送受信という形で通信をトリガやエフェクトに導入している．メッセージ送信はエフェクト中にメッセージ名と!記号を併記する（例：FCarStopGray!）ことで表される．メッセージ受信はトリガ中にメッセージ名と?記号を併記する（例：FCarStopGray?）ことで表される．メッセージの送受信は，同名のメッセージ間で行われる．

2.1.3 スクリプト

モデルが実行可能であることは，学習者が試行錯誤的にモデルの理解を深めるために重要である．しかし，本来の UML ステートマシン図記法では，必ずしも実行可能な記述とはならない．そこで，本研究ではスクリプトをガードやエフェクトに導入している．スクリプトの例として，ガードには count==1 など，エフェクトには count++; などが記述される．本スクリプトは，JavaScript 構文規則に従う．

2.2 ステートマシンモデルの教育支援環境

2.2.1 SMart-Learning：モデルシミュレータ

SMart-Learning (SML)[3] は，ステートマシンモデル上でその振舞いを確認するためのシミュレータである．SML は学習者による自モデルの確認や，教授者による答案モデルの評価を支援する．本シミュレータはこれまで，ロッカスイッチ式電灯，縄跳び，レジ打ち，押しボタン式信号機といった，本稿の説明事例とは異なる複数例題・課題に適用できた実績がある．SML はモデリングツール astah*[12] の GUI プラグインとして Java で開発されている．

SML では，入力としてシミュレートするステートマシン図の種類と数やイベント列を与え，出力として GUI 上で，現在状態や最近の遷移を赤く強調表示し，また現在状態での変数の値を表示する．スクリプトの変数識別には，パーサジェネレータ ANTLR[13] から生成された ECMAScript(JavaScript) のパーサを使用している．なお，ECMAScript の構文規則ファイルは ANTLR 4 grammar repository[14] に公開されたものを使用している．スクリプトの実行には Java のスクリプトエンジンを使用している．

2.2.2 SML4Instructor：モデルテストツール

SML4Instructor(SML4I)[4] は，SML のシミュレーションログを基に作成されたテストケースと，複数のステート

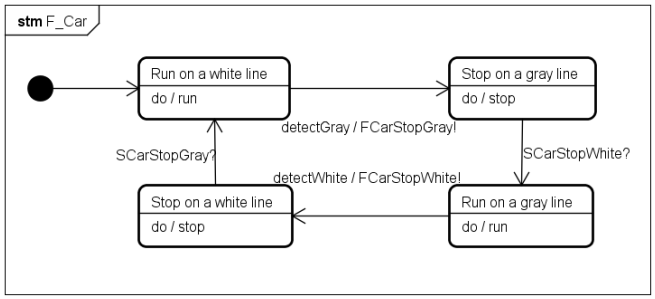

図 2 ロボット F_Car の解答例

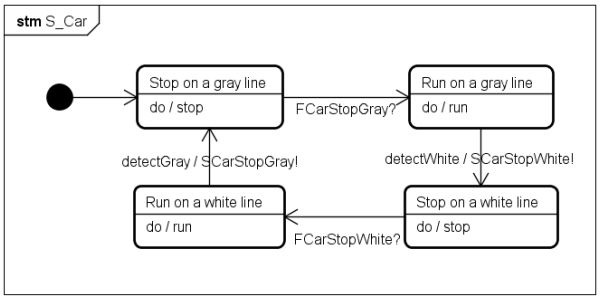

図 3 ロボット S_Car の解答例

マシンモデルを入力にとり，モデルごとのテスト結果を出力する．テストケースは，実行条件やテスト入力，期待される結果の組をもつ [15]．SML4I では，シミュレートするステートマシン図の種類と数が実行条件に対応し，イベント列がテスト入力に対応する．また，シミュレーション後に SML から得られる状態遷移のログは，一定の抽象化を施すことで期待される結果に対応する．抽象化は，教授者などがスクリプトやメッセージ名など，記述者個人に依存する表現部分を手動で削除するなどして実施される．

SML4I はまず，与えられたテストケースの実行条件からシミュレートする図の種類と数を決定してシミュレーションを開始する．つぎに，テスト入力から順にイベントを取り出して生じさせる．最後に，シミュレーション実行後に得られた状態遷移のログが，期待される結果から識別した状態遷移の列を満たすかどうか調べる．期待される結果を満たせば成功とし，満たさなければ失敗とする．これらの処理を入力されたモデルの数分繰り返す．テストの成否判定方法の詳細説明は，紙面の都合により文献 [4] に任せる．

3. 提案手法

3.1 ステートマシンモデルの評価メトリクス

評価メトリクスの目的は，ステートマシンモデルの質を端的に表し，学習者の振り返りや，教授者による評価を支援することである．提案メトリクスでモデルの低質な部分を示せば，学習者が自モデルを見直す動機となる．また，全ての答案のメトリクス値をグラフ化して俯瞰すれば，教授者は学習者全体の理解の傾向や，重点的にフィードバックすべき箇所を容易に把握できることが期待される．本研究では著者らのこれまでの教育経験に基づき，表 1 に示すように文法正当性，要求充足性，モデル合理性の 3 つの観点で分類した 9 つのメトリクスを提案する．

文法正当性では，ステートマシンモデルが MDD 指向に拡張されていることを踏まえ，DSL や通信，スクリプトの各種記法を正しく用いているかどうかを計る．この “正しさ” は，要求を満たすかどうかではなく，DSL の用語を用いている，評価可能な記述である，実行可能な記述である，およびメッセージの送受信が対応づくかどうかを指す．

表 1 提案メトリクスの概要

分類	No.	メトリクス名
文法正当性	1	正当な do アクティビティの割合
	2	正当なトリガの割合
	3	正当なガードの割合
	4	正当なエフェクトの割合
	5	正当なメッセージの割合
要求充足性	6	テストの成功率
モデル合理性	7	規模の差異
	8	複雑度の差異
	9	カバレッジ

リスト 1 は，ステートマシン図記法と DSL・通信・スクリプトの対応関係を示す文法規則 (EBNF 形式) である．評価可能な記述，実行可能な記述はそれぞれ，ANTLR が生成したパーサで Expression Sequence, Block Context としてエラーなく解析できれば妥当と判定する．なお，Block Context として解析するスクリプトは解析前に {} で囲む．

文法正当性メトリクスにおける最も良い値は 1.00 であり，最も悪い値は 0.00 となる．なお，UML ステートマシン図本来の文法チェック方法 [16] は既提案であることから，提案手法では UML ステートマシン図の文法は正しく書かれた MDD 指向なステートマシンモデルに焦点を当てる．メトリクスに用いる数値は，ガードやエフェクト等の UML 記法要素単位で計測する．たとえば，エフェクト内のスクリプトが複数の文で構成されても 1 とカウントする．

要求充足性では，記述課題の要求を満たしているかどうかを計る．この計測では，既提案のテスト手法 [4] を応用し，与えたテストケースに対するテストの成功率を計る．最も良い値は 1.00 であり，最も悪い値は 0.00 となる．

モデル合理性では，モデル規模などのプロパティに関する解答例と答案の差異や，テスト実行時に通過した点（状態や疑似状態の総称）や遷移の数を計る．これらの値は答案の是非を直接的に示さないが，答案が冗長，不完全，または複雑であることを疑える．たとえば，解答例との規模の差が大きい答案には，冗長な記述や記述不足を疑える．

```
1 <トリガ>            = <DSL>|<通信(メッセージ受信)>|""
2 <doアクティビティ> = <DSL>
3 <ガード>            = "",[<スクリプト(評価可能な記述>]
4 <エフェクト>        = "",[<通信(メッセージ送信)>],[スクリプト(実行可能な記述)]
```

リスト 1　ステートマシン図記法と DSL・通信・スクリプトの対応関係

3.1.1　メトリクス 1：正当な do アクティビティの割合

正当な do アクティビティの割合とは，全ての do アクティビティの中で DSL の用語で表されるものの割合を指し，式 (1) で表わされる．

$$M_{\mathrm{do}} = D_{\mathrm{con}}/D_{\mathrm{all}} \tag{1}$$

ここで，D_{all} は do アクティビティの総数を表す．D_{con} は DSL に適合する記述を持った do アクティビティの数を表す．

3.1.2　メトリクス 2：正当なトリガの割合

正当なトリガの割合は，全てのトリガの中で DSL の用語またはメッセージ受信のどちらか一方で表されるものの割合を指し，式 (2) で表わされる．

$$M_{\mathrm{trigger}} = T_{\mathrm{con}}/T_{\mathrm{all}} \tag{2}$$

ここで，T_{all} はトリガの総数を表す．T_{con} は DSL に適合するか，メッセージ受信を表す記述を持ったトリガの数を表す．

3.1.3　メトリクス 3：正当なガードの割合

正当なガードの割合は，全てのガードの中で評価可能な記述で表されるものの割合を指し，式 (3) で表わされる．

$$M_{\mathrm{guard}} = G_{\mathrm{con}}/G_{\mathrm{all}} \tag{3}$$

ここで，G_{all} はガードの総数を表す．G_{con} は評価可能な記述を持ったガードの数を表す．

3.1.4　メトリクス 4：正当なエフェクトの割合

正当なエフェクトの割合は，全てのエフェクトの中で実行可能な記述とメッセージ送信の両方または一方で表されるものの割合を指し，式 (4) で表わされる．

$$M_{\mathrm{effect}} = E_{\mathrm{con}}/E_{\mathrm{all}} \tag{4}$$

ここで，E_{all} はエフェクトの総数を表す．E_{con} は実行可能であるか，メッセージ送信を表す記述を持ったエフェクトの数を表す．

3.1.5　メトリクス 5：正当なメッセージの割合

正当なメッセージの割合は，通信において，メッセージの総数に対して送受信が対応づいているものの割合を指し，式 (5) で表わされる．なお，同名のメッセージで送信 (!) と受信 (?) がそれぞれ 1 つ以上存在すれば対応づいているとみなす．

$$M_{\mathrm{message}} = A_{\mathrm{co}}/A_{\mathrm{all}} \tag{5}$$

ここで，A_{all} はメッセージの総数を表す．A_{co} は送受信が対応づいているメッセージの数を表す．

3.1.6　メトリクス 6：テストの成功率

要求充足性は，与えて結果が得られる全てのテストケースに対してテストが成功したものの割合を指し，式 (6) で表わされる．

$$M_{\mathrm{satisfy}} = R_{\mathrm{pass}}/R_{\mathrm{all}} \tag{6}$$

ここで，R_{all} はモデルに与えて結果が得られたテストケースの総数を表す．R_{pass} はモデルに与えて結果が得られたテストケースの内，成功したものの数を表す．

3.1.7　メトリクス 7：規模の差異

規模の差異は，教授者の解答例等の基準となる解答と答案の点と遷移の合計数の差異を指し，式 (7) で表わされる．

$$M_{\mathrm{size}} = |S_{\mathrm{sub}} - S_{\mathrm{ref}}| \tag{7}$$

ここで，S_{ref} は基準となる解答の点と遷移の合計数を表す．S_{sub} は答案の点と遷移の合計数を表す．この値は冗長な記述や記述不足があるモデルを探し，また解答例と異なる戦略を見つけるヒントとなる．

3.1.8　メトリクス 8：複雑度の差異

複雑度の差異は，教授者の解答例等の基準となる解答と答案とのサイクロマチック複雑度 [17] の差異を指す．サイクロマチック複雑度 v は，式 (8) で与えられる [17]．

$$v = e - n + 2p \tag{8}$$

e はエッジ総数を指し，n は点の総数を指し，p は連結成分となる．ステートマシンモデルは複数のステートマシン図で構成されうるため，p は図の数となる．

サイクロマチック複雑度を踏まえて複雑度の差異は，式 (9) で表わされる．

$$M_{\mathrm{cycle}} = |C_{\mathrm{sub}} - C_{\mathrm{ref}}| \tag{9}$$

ここで，C_{ref} は基準となる解答のサイクロマチック複雑度を表す．C_{sub} は答案のサイクロマチック複雑度を表す．複雑すぎる答案は再考すべきであるが，この値はその複雑な答案を探すヒントとなる．

3.1.9　メトリクス 9：カバレッジ

カバレッジは，全ての点と遷移に対して，全てのテスト実行後に一度でも通過したものの割合を指し，式 (10) で表わされる．

$$M_{\mathrm{cover}} = V_{\mathrm{transit}}/V_{\mathrm{all}} \tag{10}$$

```
1  [do]
2  run
3  stop
4  [trigger]
5
6  ((?!\p{Punct}).)+\?
7  detectGray
8  detectWhite
9  [guard]
10
11 <guard_script>
12 [effect]
13
14 ((?!\p{Punct}).)+!
15 ((?!\p{Punct}).)+!<effect_script>
16 <effect_script>
```

リスト 2　文法ファイル

ここで，V_{all} は，全ての点と遷移の数を表す．V_{transit} は，全ての点と遷移の数のうち，全てのテスト実行後に一度でも通過した点と遷移の数を表す．

M_{cover} の最高値は理想的には 1.00 であるが，そもそもテストケースセットが点と遷移を網羅するに不十分である場合は 1.00 になりえない．一方，最低値は 0.00 である．答案に対するこの値が解答例に近い値かどうかを調べることで，振舞いの構成が類似しているかどうかを知るヒントとなる．答案が解答例よりも小さい値を取る場合，冗長な記述を疑え，大きい値を取る場合，記述不足を疑える．

3.1.10　SML の拡張：文法チェッカ

提案メトリクスに基づき，SML に文法チェッカを導入した．この文法チェッカは，表 1 の No.1–4 のメトリクスにおける不正な状態や遷移を GUI 上で赤く強調表示する．

DSL は課題ごとに変わる可能性が高く，また通信やスクリプトの記述は今後拡張される可能性がある．そこで文法チェッカの汎用性を高めるため，リスト 2 に示す文法ファイルを SML 外部に定義し，SML がその文法ファイルを利用してモデルをチェックする方式を採用した．リスト 2 中の [do] や [trigger]，[guard]，[effect] は，それぞれ do アクティビティ，トリガ，ガード，エフェクトといった記述箇所を示しており，その直下に正当とみなす表現等を行単位で記述する．この文法ファイルの具体的なデータの記法は以下の 3 つに分類される．

(1) 文字列：リスト 2 では，do アクティビティには run か stop の文字列しか認められず，空白を含め他の表現は違反となる．なお，空白は空白行の追加で許される．
(2) 正規表現：リスト 2 では，任意のメッセージ名と?を併記するメッセージ受信を ((?!\p{Punct}).)+\?で表す．
(3) キーワード：<guard_script> は評価可能な記述を表し，<effect_script> は実行可能な記述を表す．

3.1.11　SML4I の拡張：メトリクス値算出機能

先述の文法ファイル，テストケース，答案モデル（群），解答例モデルを入力とし，各種メトリクス値を自動算出するように SML4I を拡張した．SML4I は SML が持つ情報にアクセスできる．メトリクス 1–4 は，文法チェッカの機能を応用して文法ファイルに基づき計算される．メトリクス 5 は，メッセージ送受信の記法に基づき対応しているかどうかから計算される．メトリクス 6 は，モデルをテストした結果に基づき計算される．メトリクス 7–8 は，必要なプロパティをモデルから取得・計算し，解答例モデルとの比較により計算される．メトリクス 9 は，テスト時に SML が記録した通過要素を基に計算される．結果は図 4 のように表 (CSV) 形式で出力される．1 列目は答案ファイル名，2 列目は図の名前空間を指す．名前空間は課題の識別などに用いる．3 列目以降が提案メトリクスの算出値を表す．

4.　評価

4.1　Research Questions

我々は提案手法の有効性を評価するために，以下のように 3 つの Research Questions(RQs) を設けた．

RQ.1　提案メトリクスの値は答案の質を表すか？
RQ.2　提案メトリクスの値からモデル全体の質の傾向が分かるか？
RQ.3　拡張モデルシミュレータは教育支援に有効か？

4.2　概要

本評価で用いる答案を作成した学習者は，MDD を学ぶ科目を受講する電子情報系の大学 2 年次相当の者である．前提知識として，Python プログラミングの基礎知識を有するが，UML モデリングの知識は持たない．本評価以前に教授者は，先述した MDD 指向なステートマシンモデルの記述方法を教授し，複数の記述課題を課している．

本評価は，最終的に RQ.3 に答えるために，従来のエディタのみを用いたモデルの質と拡張モデルシミュレータ導入後のモデルの質を比較評価することが趣旨である．授業実施上の制約から同一課題（図 1）を長期的に取り組む中でモデルの質を評価するアプローチを採った．なお，本評価では教授者（授業実施者）と評価者（答案の自動・手動評価の実行者）は異なるものであった．

4.3　手順

評価の手順を次に示す．

Step 1　教授者は拡張前モデルシミュレータを学習者に提供し，動かしながら説明を行った．
Step 2　Phase I: 学習者はモデルシミュレータを用いずに従来のエディタのみで授業期間内一週間程度かけて答案を作成し提出した．
Step 3　教育的配慮から教授者は答案に基づき，文法的な

file	namespac	Mdo	Mtrigger	Mguard	Meffect	Mmessag	Msatisfy	Msize	Mcycle	Mcover
aqs_036.a	課題 7	0.00	0.83	1.00	0.67	1.00	0.00	8.00	0.00	0.50

図 4　メトリクス値の自動算出の結果例

表 2　自動テストと手動評価の結果比較

Phase	手動合格	手動不合格	自動合格	自動不合格	Recall	Precision	Accuracy
Phase I	5	58	4	59	0.80	1.00	0.98
Phase II	28	35	27	36	0.96	1.00	0.98
Phase III	41	22	38	25	0.93	1.00	0.95

誤りとその修正方法をおよそ 90 分で講義した．

Step 4　Phase II: 学習者は拡張前シミュレータ（文法チェッカ無し）を用いて授業期間内一週間程度かけて答案を変更し提出した．

Step 5　Phase III: 学習者は拡張後シミュレータ（文法チェッカ有り）を用いて授業期間内一週間程度かけて答案を変更し提出した．

Step 6　評価者は全ての答案から有効回答（答案）を選定した．有効回答は，全ての Phase で提出があり，Phase I でシミュレータを用いた形跡がないものとした．

Step 7　評価者は有効回答全てに対して，メトリクス値を算出した．なお，テストケースは，最も基本的な振舞いを確認するものを一つだけ用意した．

Step 8　RQ.1 や RQ.2 に答えるべく，各答案に対する SML4I での自動テスト結果と手動での評価結果を比較した．自動テストではメトリクス 1–6 全てが最も良い値なら合格，そうでなければ不合格とした．手動評価では，評価者が目視等により DSL，通信，スクリプトを正しく用い，かつ要求を満すと判断したモデルを合格とした．手動評価で不合格となったモデルについては評価者が気づいた範囲で不合格の要因を記録した．

4.4　結果

有効回答は 63 人分 189 個のモデルであった．全てのメトリクス値を，箱ひげ図として図 5 から図 13 に順に示す．I-NoTool ラベルは Phase I の答案であることを示し，続いて II-Sim と III-Sim+Chk はそれぞれ Phase II，Phase III の答案であることを示す．提案メトリクスに基づくと，箱ひげ図は全体的に，エディタのみよりも拡張シミュレータを利用した方が質の良いモデルとなっていることがわかる．

自動テスト結果と手動評価結果の比較結果は表 2 となった．“手動” は手動評価を表し，“自動” は自動テストを表す．さらには，自動テストを予測評価とし，手動評価を実評価としたときの精度も算出した．Recall, Precision, Accuracy はそれぞれ式 (11)，式 (12)，式 (13) で求められる．

表 2 から非常に高い Accuracy でテスト結果を得ることができており，メトリクス 1–6 を総合した値は信頼できるものであった．また，メトリクス 1–6 の各値について，手動評価での要因記録とメトリクス値の低下箇所を手動で比較したところ，ほぼ整合する結果となっていた．

$$Recall = \frac{TP}{TP + FN} \tag{11}$$

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{13}$$

さらにメトリクス 1–6 の値から，手動評価で記録が漏れた要因に気づくことができた．また，メトリクス 7–9 の値は，特徴的な答案を見つけるヒントになっていた．

一方で，Recall が 1.00 とならない原因は次のものであった．第一に，テスト開始時の変数の初期値がデフォルト値（たとえば 0）以外となる場合，現 SML4I では未対応であった．第二に，文法ファイル中の不適切な正規表現により，メッセージ名に括弧が登場した場合に誤判定が生じていた．第一点目は答案モデルの提出方法の指定で解消でき，第二点目は文法ファイルの修正で解消できるなどいずれも軽微な変更で対応可能であった．

4.4.1　RQ.1：提案メトリクスの値は答案の質を表すか？

表 2 の高い Accuracy から，文法正当性・要求充足性の観点では提案メトリクスは答案の質を表すと言える．このことは，図 5 から図 13 のデータが信頼できる根拠となる．

また，一般にモデルでは実装や実行環境の詳細部分等を抽象的に扱うため，教授者・評価者間での想定に差異が生じ，これによって教授者と評価者で評価結果が異なるなどの可能性は否定できない．本評価では，答案の自動・手動評価の実行者は同一であったため，2 つの評価の間で適否を与える方針に差があったとは考えにくい．

一方で，メトリクス値は端的な評価結果を表すため，たとえば文法は正しいが，要求を満たさない（テストに失敗する）答案があっても，どこが誤っていたかをメトリクス値から判断することはできない．そのため，教授者がフィードバックを与えるべき箇所を漏れなく効率的に発見するために，質的な評価支援や，戦略的なテストケースの生成方法とテスト結果の解釈方法を今後実現する必要がある．

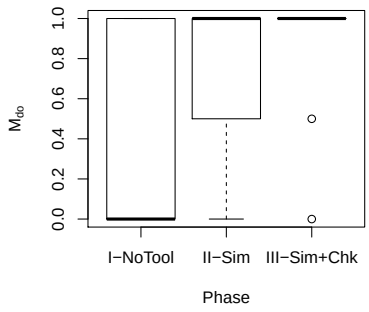

図 5 メトリクス 1：M_{do} 値

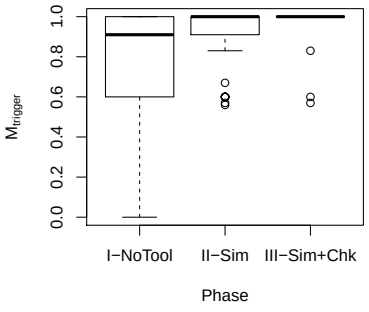

図 6 メトリクス 2：M_{trigger} 値

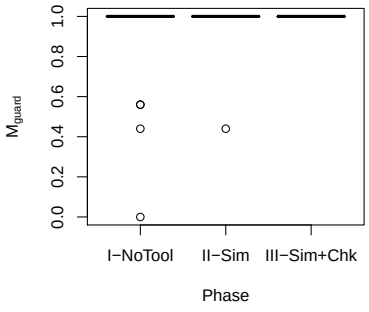

図 7 メトリクス 3：M_{guard} 値

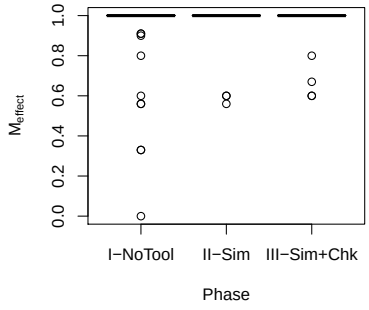

図 8 メトリクス 4：M_{effect} 値

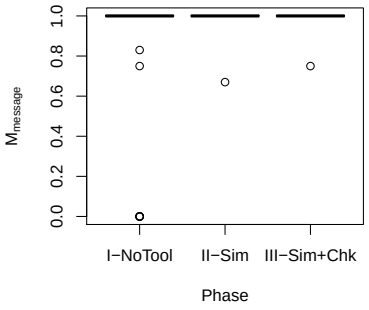

図 9 メトリクス 5：M_{message} 値

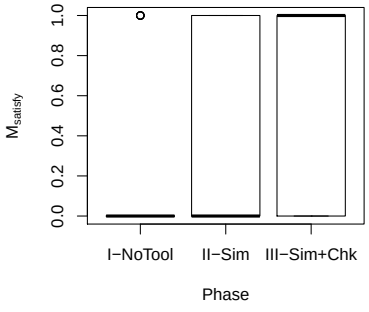

図 10 メトリクス 6：M_{satisfy} 値

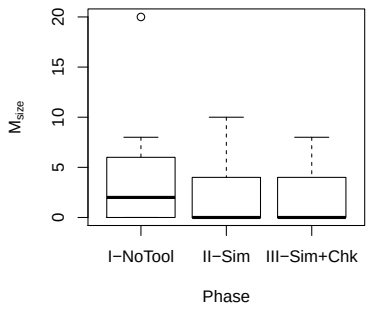

図 11 メトリクス 7：M_{size} 値

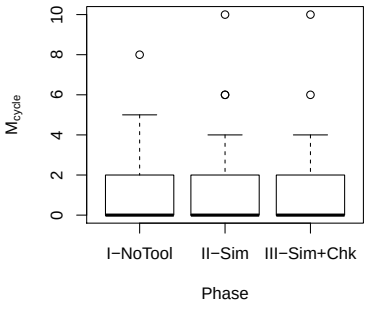

図 12 メトリクス 8：M_{cycle} 値

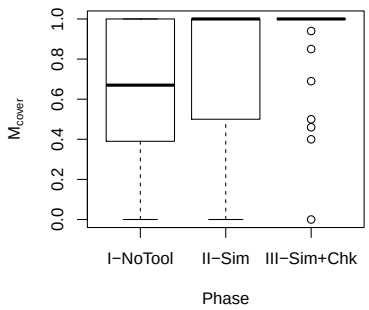

図 13 メトリクス 9：M_{cover} 値

4.4.2 RQ.2：提案メトリクスの値からモデル全体の質の傾向が分かるか？

RQ.1 の回答により，図 5 から図 13 は信頼できることが示唆された．そして，Phase が進むにつれて，表 1 の 1–6 の値は最も良い値（1.00）に近づく結果となっている．このことから，提案メトリクスの値からモデル全体の質の傾向が分かると考えられる．

興味深いことは，メトリクス 7–9 の値は，Phase が進むにつれて少しではあるが，合理化が図られていることである．特にカバレッジを表す図 13 は，大きな改善傾向を示しており，学習者が DSL の用語を適切に用いるようモデルを改善できたことが示唆される．

4.4.3 RQ.3：拡張モデルシミュレータは教育支援に有効か？

表 2 の合否の結果に基づいて合格者が増加していることから，拡張モデルシミュレータは本稿で掲げた教育目的での教育支援に有効であったと言える．一方では，学習者がイベント列を自由に想定してシミュレータに与えることで気づかなかったであろう誤りも存在した．この改善には，教授者と学習者間，または学習者間で簡単にテストケースを共有してテストできる仕組みの実現が必要と考えられるが，今後の課題とする．

自動テスト結果と手動評価結果の比較を踏まえて，教授者にとってはメトリクス値からどの答案が DSL を不適に

用いているのか，特徴的な答案がどれなのかなどが把握しやすくなったと言える．しかし，本評価を通して，合理的なテストケース群の準備や，テストケースの適切な抽象化は，教授者の能力に依存する状況が見受けられたため，モデルからテストパスを自動生成する MBT(Model-Based Testing) 技術の応用するなどして改善を図りたい．

5. 関連研究

メトリクスについて，Genero ら [18] は do アクティビティなどの各種要素数とサイクロマチック数を併せたメトリクスを提案している．また，ステートマシン図のメトリクスには様々なカバレッジがある [19]．本研究の提案メトリクスはサイクロマチック複雑度など共通項もあるが，DSL や通信，スクリプトに着目した文法正当性のメトリクスは新規提案となる．そして，ステートマシンモデルへのテストによる要求充足性や，教育コンテキストであるからこそ着目した規模や複雑度の差異のメトリクスは我々の知る限り従来では扱われていない．カバレッジ [19] については，ステートマシンモデルをより深く分析するため，MBT 技術と併せて連係方法を今後検討する．

6. まとめ

本稿では，モデリング教育支援を目的として，学習者の作成したステートマシンモデルの質を端的に示す方法を新規に提案した．本稿での高い質のモデルとは，与えられた文法を正しく用いた，要求を満たす，規模や複雑度が適度なモデルを指す．そして，その質が高められるよう学習者のモデリング技能を向上することが教育目的であった．

有効性評価の結果，提案メトリクスは学習者の答案の質を端的に表せ，そこから学習者全体の質の傾向が俯瞰できる見込みを得た．さらには，メトリクスによって，従前に比べてどこに欠陥が含まれていそうかを容易かつ客観的に把握でき，また特徴的な答案を識別するヒントにもなることがわかった．拡張モデルシミュレータは，評価結果に基づき教育支援として有効であったことが示された．

今後の課題としては，答案モデルの詳細な質的評価を経て，メトリクス値をどのように質的な評価ひいてはフィードバック提供に繋げるかを検討する．そして，学習社と教授者の双方の支援のためにも MBT 技術などの導入によって，モデルテストのさらなる自動化を進める．

謝辞 本研究は JSPS 科研費 JP16H03074 の助成を受けたものです．

参考文献

[1] Akayama, S., Kuboaki, S., Hisazumi, K., Futagami, T. and Kitasuka, T.: Development of a Modeling Education Program for Novices Using Model-driven Development, *Proceedings of the Workshop on Embedded and Cyber-Physical Systems Education*, WESE '12, pp. 4:1–4:8 (2013).

[2] Arya, K., Coelho, B. and Pandya, S.: A Model Based Design Approach to System Building Using the e-Yantra Educational Robot, *SIGBED Rev.*, Vol. 14, No. 1, pp. 37–43 (2017).

[3] Ogata, S., Kayama, M. and Okano, K.: SMart-Learning: State Machine Simulators for Developing Thinking Skills, *2017 IEEE 17th International Conference on Advanced Learning Technologies (ICALT)*, pp. 81–83 (2017).

[4] Ogata, S., Kayama, M. and Okano, K.: Approach to Testing Many State Machine Models in Education, *Proceedings of the 11th International Conference on Computer Supported Education*, CSEDU '19, pp. 481–488 (2019).

[5] Object Management Group: Unified Modeling Language 2.5.1, Retrieved from https://www.omg.org/spec/UML/2.5.1/PDF (2017).

[6] The Eclipse Foundation: Sirius, https://www.eclipse.org/sirius/ (accessed 2019-7-26).

[7] Vasudevan, N. and Tratt, L.: Comparative Study of DSL Tools, *Electronic Notes in Theoretical Computer Science*, Vol. 264, No. 5, pp. 103–121 (2011).

[8] Heinze, T., Jerzak, Z., Martin, A., Yazdanov, L. and Fetzer, C.: Fault-tolerant complex event processing using customizable state machine-based operators, *Proceedings of the 15th International Conference on Extending Database Technology*, EDBT '12, pp. 590–593 (2012).

[9] 香山瑞恵，小形真平，永井　孝：モデル駆動開発方法論に基づく UML プログラミング教育環境，教育システム情報学会誌，Vol. 36, No. 2, pp. 118–130 (2019).

[10] Holzmann, G. J.: The Model Checker SPIN, *IEEE Trans. Softw. Eng.*, Vol. 23, No. 5, pp. 279–295 (1997).

[11] David, A., Larsen, K. G., Legay, A., Mikučionis, M. and Poulsen, D. B.: Uppaal SMC tutorial, *International Journal on Software Tools for Technology Transfer*, Vol. 17, No. 4, pp. 397–415 (2015).

[12] Change Vision: Astah, http://astah.net/ (accessed 2019-7-26).

[13] Parr, T.: *The Definitive ANTLR 4 Reference*, Pragmatic Bookshelf (2013).

[14] Antlr Project: ANTLR 4 grammar repository, https://github.com/antlr/grammars-v4 (accessed 2019-7-26).

[15] IEEE: IEEE Standard for Software and System Test Documentation, *IEEE Std 829-2008*, pp. 1–150 (2008).

[16] 但馬将貴，香山瑞恵，小形真平，橋本昌巳：UML に基づく概念モデリングにおける状態遷移図に対するモデル記法チェック機能の効果，信学技報，Vol. 116, No. 418, pp. 7–12 (2017).

[17] McCabe, T. J.: A Complexity Measure, *IEEE Transactions on Software Engineering*, Vol. SE-2, No. 4, pp. 308–320 (1976).

[18] Genero, M., Miranda, D. and Piattini, M.: Defining Metrics for UML Statechart Diagrams in a Methodological Way, *Conceptual Modeling for Novel Application Domains* (Jeusfeld, M. A. and Pastor, Ó., eds.), Berlin, Heidelberg, Springer Berlin Heidelberg, pp. 118–128 (2003).

[19] Salman, Y. D., Hashim, N. L., Rejab, M. M., Romli, R. and Mohd, H.: Coverage criteria for test case generation using UML state chart diagram, *AIP Conference Proceedings*, Vol. 1891, No. 1, pp. 020125–1–020125–6 (2017).