MDD のための基礎教育環境の構築

原行人 [1], 西尾圭太 [2], 増田壮志 [2], 香山瑞恵 [2], 久住憲嗣 [3], 伊東一典 [2], 橋本昌己 [2], 大谷真 [2]

本研究の目的は, Model Driven Development 技術者の育成を指向した, 基礎教育環境の構築にある. 我々は対象を抽象化する力の育成を特に意識する. 提案環境では, クラス図と状態遷移図を用いた設計課題に取り組ませる. 本環境の特徴は以下の3点である. 1)課題の難易度に応じたモデル部位の名称表現の変更機能, 2)課題に対して初期状態モデルを与える機能, 3)実行コードを生成する対象デバイスに応じた開発ツールの入替機能.

A Basic Study of an Educational Environment for Modeling and Abstraction

Yukito Hara[1], Keita Nishio[2], Takeshi Masuda[2], Mizue Kayama[2],
Kenji Hisazumi[3], Kazunori Itoh[2], Masami Hashimoto[2] and Makoto Otani[2]

The purpose of this study is to develop an educational environment for basic modeling and abstraction in software engineering. In proposed environment, learners are asked to draw class-diagram and state-machine diagram. Then, the pictured models are translated to an executable code for a target device. Learners can evaluate the conformance to specifications and validity of their models by observing the moving of the device. In this paper, we firstly discuss about the difficulty levels of modeling task, then show functions of our environment.

1. はじめに

論理的思考力の育成手段として, 従来よりアルゴリズム教育やプログラミング教育が展開されてきた. この教育は, 電子計算機での情報処理と関連付けて教授されることが多い. しかし, 今日の情報科学・技術の進展を考慮すると, 順序立てた手続きの系列とその自動実行というパラダイム以上に, より広い対象を曖昧性が少なく抽象化すること・モデル化することを思考した思考力の育成も重視されよう[1][2][3].

情報システム設計の上流工程にあたるシステム設計ではモデル化・抽象化を伴う作業が生じる. これらの作業を含むソフトウェア工学の分野は, 高等教育機関においても近年注目され始めた分野であり, 未だその教育方法論は確立されていない[4]. これらの分野の基礎ともいえる抽象化思考を教育する際に, 初学者にとって理解が困難な概念や事柄, 初学者が生じやすいエラー等を考慮した教育方法の工夫が求められる.

本研究では, 学習者の抽象化能力として, 対象の静的な性質とそれらの関係を表現するクラス図作成と, 対象が有する動的な関係を表現する状態遷移図作成とに注目をする. これらを記述する能力を向上させることを目的とし, 利用者の能力に応じた課題表現をカスタマイズ可能な教育支援ツールを開発する. このツールでは, クラス図や状態遷移図として表現されたモデルの妥当性検証機構としての MDD 機能[5]を具備するものとする.

†[1] 信州大学大学院工学系研究科, [2] 信州大学工学部, [3] 九州大学システム情報科学研究院.
[1] Graduate School of Science and Technology, Shinshu University, [2] Faculty of Engineering, Shinshu University, [3]Faculty of Information Science and Electrical Engineering, Kyusyu University.

2. 研究目的

本研究では, 抽象化能力としてのモデリング力の育成方法を探究している. そのために, まず, システム設計を意識した抽象化をビジュアルモデリングを通して学ぶことができる教育環境を構築することとした. この環境を利用することで, 学習者は, 他者との共有にも適し, 自身の思考に近い図的な表現によりシステム設計という複雑な課題解決を体験できるようになると期待する.

近年, CloudMDD のコンセプトによる開発環境の提案もある[6]. 商用の MDD ツールと合わせて, 専門家の利用を想定した開発環境が整いつつある中, 抽象化および MDD に関する初学者の教育を意識した開発環境は未だ見られない.

本稿では, まず, 高校生と大学生とを対象に行ったビジュアルモデル作成実験の結果を示し, 前提知識の

差異による抽象化力の違いに関して考察する．その結果に基づき，抽象設計課題における難易度の指標を提案する．そして，学習者の習熟レベルに応じた課題の難易度を設定可能な MDD によるシステム設計学習支援システムを提案する．

3. クラス図作成力を確認する実験

高校生と大学生を対象に，モデル設計能力を確認する実験を行った．ここでは一般的な対象物のクラス図を記述させた．被験者はシステムモデルの設計に関する知識の有無で高校生群と大学生群とに分類した．

3.1. 実験方法

被験者は工学系大学生 14 名(以下，大学生群)，高校生 20 名(以下，高校生群)である．実験時に，大学生群はシステム開発に関わるモデル設計の知識を有しており，高校生は一切有していなかった．

被験者には，自転車を例にしてクラスモデルの作成に関して 15 分間の説明を与えた．その後，人間の手を対象としたクラスモデルを記述させた。モデル記述のための時間は15分とした．この際，クラスの例として「手のひら」，「指」，「爪」の 3 つの要素が実験者より被験者全員に示された．

3.2. 実験結果

図 1 に大学生が記述したクラス図の例を，図 2 に高校生が記述したクラス図の例を示す．ここでは，これら被験者が記述したクラスモデル(以下，被験者モデル)におけるクラス数と関係数，およびクラス要素の内容と関係を，大学生群と高校生群とで比較する．

■クラス数とクラスの種類

被験者モデル中のクラス数を整理した．なお実験者によって例示した 3 つのクラスの一部または全部が被験者モデルに含まれている場合には，それらを含んだ数として整理した．高校生群では，最多 9 個，最少 3 個，平均 5.8 個であった．一方大学生群では最多 5 個，最少 3 個，平均 3.6 個であった．クラス数の上では，高校生群は大学生群の約 2 倍程度多く列挙できていた．

また，実験者が例示した 3 つのクラス以外に記述されたクラスの種類は，高校生群では 12 種類，大学生群では4種類であった．表 1 に記述されたクラスの種類をまとめた．高校生群が大学生群の 3 倍の種類をあげていた．さらに高校生群では 40%以上の被験者が大学生よりも多い種類を記述していた．

高校生群でも対象の要素をとらえること自体は比較的容易に行えるのではないかと考える．しかし，高校生群のクラスの属性に不適切な記述が多々みられた．これは，高校生群にとっては，クラスの構成要素と状態を

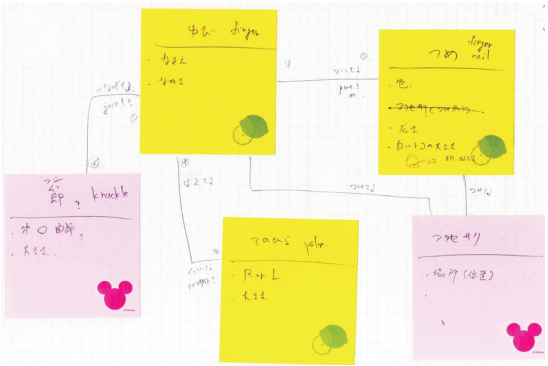

図 1　大学生が記述したクラス図の例

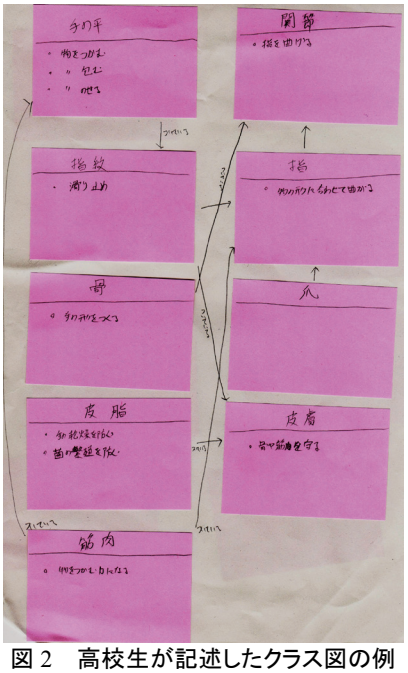

図 2　高校生が記述したクラス図の例

表 1　例示した 3 クラス以外のクラスの種類

高校生群		
クラス名	数	割合
関節	9	47.4%
骨	8	42.1%
手相・シワ	7	36.8%
指紋	7	36.8%
皮膚・皮	4	21.1%
毛	3	15.8%
筋肉	3	15.8%
手の甲	3	15.8%
血管	3	15.8%
皮脂	1	5.3%
マメ	1	5.3%
ささくれ	1	5.3%

大学生群		
クラス名	数	割合
関節	6	42.9%
毛	1	7.1%
手の甲	1	7.1%
アクセサリ	1	7.1%

分離することが難しかったのではないかと考える. 一方, 大学生群では属性の不適切な記述は見られなかった. しかし, 大学生群ではクラスの列挙数は少なかった. 実験後の内観報告で, 一般的な対象をUMLで記述することへの違和感が多分にあった等の意見が得られた.

■関係数

被験者モデルでの関係数を整理した. 高校生群は最多 12 個, 最少 2 個, 平均 6.3 個であった. 大学生群は最多 5 個, 最少 2 個, 平均 3.6 個であった.

関係数でも高校生群の方が多かった. これは, 高校生群のクラス列挙数も多いことが影響している. しかし, 多様性(m:n 関係)については, 大学生群は適切に指摘できていたが, 高校生群には不適切・不完全な記述がみられた. また要素間の関係に関する記述では高校生大学生共に不完全な記述であった.

3.3. 考察

一般対象物のクラス図作成によるモデル化は, 工学系大学生であっても満足に回答できていない. システム設計という範疇を超えたモデル作成の経験がないことが影響しているようである. また高校生は, 要素の列挙や関係の表現はできても, それらの要素の属性や要素同士の関係を適切に記述できていない.

このことから, システム設計のみならず一般対象物をも含んだモデル設計を意識した教育・訓練をしなければ, この種の能力は育成されないともいえる.

4. 状態遷移図作成力を確認する実験

次に, システム設計課題の難易度指標を検討するために, 条件の異なる複数課題に対する状態遷移図を作成させる実験を行った. 対象者は高校生と大学生である. 各被験者をプログラミングに関する知識の有無, 課題内容に関する知識の有無で分類した.

4.1. 課題と課題遂行ステップ

LEGO ロボット制御に関する 3 種の課題を準備した. 課題 1(飛脚課題)ではタッチセンサの ON/OFF を考慮することが必要である. 課題 2(飛脚型ライントレース課題)では 2 種類のセンサ(タッチセンサ, 光センサ)を制御し, 更にタッチセンサと光センサの優先順位を考慮しなければならない. 課題 3(ライントレース課題)では, 課題 1 と同様にセンサを 1 つ(光センサ)利用する. しかし, センサの測定値を考慮しなければならない. そのため, 課題の複雑さは課題 1 と課題 2 の間とした.

モデル設計の過程を次の 4 ステップとした.

ステップ 1: 課題内容の理解
ステップ 2: 解決方針の立案
ステップ 3: 紙ベースでのモデル表現
ステップ 4: MDD ツール上でモデル表現

表 2 状態遷移モデル記述実験の結果

		高校生群	大学生A	大学生B	大学生C
プログラミング知識		×	○	○	○
課題知識(ライントレース)		×	○	○	×
課題	ステップ				
1 (飛脚)	1	○			
	2				
	3				
	4				
3 (ライントレース)	1	−	○ (7分)	○ (4分)	○ (24分)
	2	−		×	×
	3	−		×	×
	4	−		×	×
2 (飛脚型ライントレース)	1	○ (20分)	○ (12分)	○ (4分)	○ (12分)
	2				×
	3	×		×	×
	4	×	×	×	×

○:完了, ×:未完了. ()には課題解決にかけた時間を示す.

4.2. 実験方法

■高校生対象の実験

被験者は高校 3 年生 20 名とし, 1 グループ任意の 5 名で構成される計 4 グループに分けた. 被験者には, モデル作成に必要な情報と記述の説明を 10 分間行った上で, 課題 1 の設計を 15 分, 課題 2 の設計を 20 分で状態遷移図を記述させた. 全員がプログラミングに関する知識を有していない.

■大学生対象の実験

被験者は工学系大学生 3 名(A,B,C)とした. 試験者は全員プログラミングに関する知識を有している. しかし, 課題 2 と 3 に関連するライントレースに関する知識を有する者 (A,B)とそうでない者(C)とがいる. 実験課題は課題 3 と課題 2 である. なお, 課題 1 に関しては予備実験でモデル設計させ, 全員が記述できることを確認している. 被験者には, モデル作成に必要な情報と記述例の説明を 10 分間行った上で時間制限を設けずに状態遷移図を記述させた.

4.3. 実験方法

被験者の設計成果を表 2 に示す. 高校生は課題 2 のステップ 2 まで完了できた. 大学生は課題 3 と課題 2 で 3 名中 2 名がステップ 2 まで完了できた.

しかし, いずれの被験者も, 被験者が考えた解決方針を, Lego という制約条件を考慮した状態遷移モデルとして表現できていない. それは, 実験でのモデル記述の方法が市販ツールに強く依存した表現であったことが考えられる. また, 課題 2 においてはリアルタイム

OS 上のマルチタスク管理を意識しなければならなかった．そのような条件での状態遷移を設計することの困難さがうかがえる．

4.4. 考察

実験結果からは，センサの ON／OFF だけを考慮する課題(課題 1)と比較して，複数のセンサの制御を伴う課題(課題 2)や ON/OFF ではなくパラメータの値に基づく制御が必要な課題(課題 3)は設計が難しいようだ．このことからセンサの数や制御方法の違いによって難易度が異なるのではないかと考えられる．

また今回，モデル記述の市販ツールに用いたため，被験者が自身の設計成果を状態遷移モデルとして表現する際にツール依存の表現方法を取らねばならず，困惑したとも考えられる．このことからイベントや振舞表現の方法を，被験者の設計スキルに応じて変更することが必要であると考えられる．

今回の実験では，あらかじめ 2 つの状態を与えた上で，それを発展させる形でのモデルを記述させた．あらかじめ与える状態を増やすことで難しい課題への解決の足場提供となる可能性もある．すなわち，課題解決の初期条件の制御も考慮すべきであろう．

これらをふまえ，以下の 3 種を課題の難易度指標として仮定した．

指標 1： 制御対象となるセンサの数や制御方法
指標 2： 課題解決の初期条件
指標 3： イベントや振舞の表現の抽象度

5. MDDのための基礎教育環境

提案する教育環境では、クラス図と状態遷移図を用いた設計課題に取り組ませる．教育環境の全体構成概要図を図 3 に示す．クラス図と状態遷移図で設計した結果はシステム内のモデルコンパイラにより対象デバイスの実行コードに変換される．実行コード生成対象デバイスは，現在，Lego Mindstorms NXT[7]としている．学習者は Lego の動きを観察することで、モデルでの設計結果を視覚的に確認しながら評価することになる．また，提案システムでは，MDD による設計環境とモデルコンパイル環境を分離した構成とする．このような構成により，開発対象デバイスの変換やモデルコンパイラの入れ替え等に柔軟に対応できる．

この環境には，4.4 で示した課題難易度指標に基づいた各種の課題が登録されている．本環境の特徴は

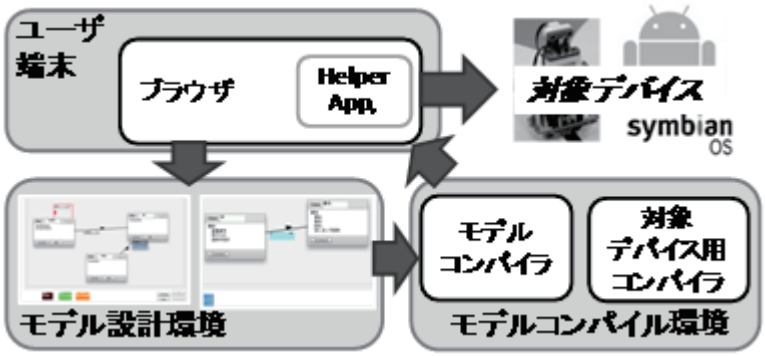

図 3　システムの全体構成概要図

図 4　イベントおよび状態表現の変更例 [左:タスク依存な日本語での表現，右:開発ツール依存な表現]

以下の 3 点である.

(1) 課題の難易度に応じたモデル部位の名称表現の変更機能.
(2) 課題に対して初期状態モデルを与える機能.
(3) 実行コードを生成する対象デバイスに応じた開発ツールの入れ替え機能.

(1)は, 4.4 で示した難易度指標に応じて, 学習者がモデルを作成する際に用いる部品(クラス／状態／イベント等)の表現を変更することである. すなわち, タスク依存な方式と, 将来開発に利用するであろうMDDツール依存な表現とを課題および課題回答者に応じて切り替える. 図 4 に同一課題に対する 2 種の表現スタイルの例を示す. タスク依存な表現では日本語もしくは平易な英単語で表す. ツール依存な表現では, 開発ツールで利用される関数やイベントが実際の表現のまま示される. (2)に関しては, 課題解決を初期モデルの改良により進めること, およびモデル設計に利用するモデル部品を指定・限定することにより実現している. これにより, 学習者の認知的負荷を軽減し, より設計行為に集中できることを期待する. (3)に関しては, モデル設計環境とは別にモデルコンパイル環境を設けることで実現している. これにより, 開発対象デバイスや教育目的に応じて, モデルコンパイラ／開発言語／実行コード生成用ライブラリ・ツールをより柔軟に入れ替えることができる. また, MDD を意識しない抽象化能力育成の際には, モデル設計環境のみを利用することで, よりライトな設計環境を提供できる.

提供環境では, ActionScript で記述された Flash ベースの Web アプリケーション上で、モデル図作成・保存・呼出が実現している. モデル図は XML 形式で管理される. 連動するコンパイル環境では, モデル設計環境より XML を受け取り, Lego MindStorms 上で実行可能なバイナリコード(実行コード)を出力する. この実行コードはユーザ端末に送信され, 提案システムを介して実機に組み込まれる.

6. おわりに

本稿では, 高校生と大学生とを対象に行ったビジュアルモデル作成実験の結果を示し, 前提知識の差異による抽象化力の違いに関して考察した上で, モデル化課題の難易度の指標を提案した. また, 抽象化力育成のための教育環境を提案した. 現在, この環境を利用した情報系学部初期教育を実施しつつ, システムの有効性および実用性に関する検証を進めている.

参考文献

[1] Jeff Krame:" Is abstraction the key to computing?", Communications of the ACM, 50(4), pp.37-42, 2007.
[2] 経済産業省:"2010 年版組込みソフトウェア産業実態調査報告書", 2000.
[3] 小倉信彦:"ロボットコンテストを利用した組込み教育の実践", 情処論,49(10), pp.3531-3540, 2008.
[4] J. Bezivin et al. : "Teaching Modeling: Why, When, What?", MODELS 2009 Workshops, pp. 55-62, 2010.
[5] スティーブ J.メラー、マーク J.バルサー: "Executable UML",翔泳社, 東京, 2003.
[6] 部谷修平他:"クラウド上のモデル駆動開発ツール、CloudMDD の開発", 2010 年度未踏 IT 人材発掘・育成事業未踏ユース採択案件概要, 2010.
[7] LEGO マインドストーム.:http://legoeducation.jp/mindstorms/(2011/09/20 accessed)