Vol.2014-SE-183 No.10
2014/3/19

オブジェクトモデリング演習における学習者にとっての難所の検出手法と支援システムの提案

田中昂文[†1]　森一樹[†2]　橋浦弘明[†3]　櫨山淳雄[†1]　古宮誠一[†4]

近年，高等教育機関では，実践的なソフトウェア開発演習が行われている．演習を有効に実施するためには，学習者が直面した困難を教授者が把握し，適切に指導を行うことが有効である．現在の演習の成果確認は学習者が提出した成果物を教授者がレビューするという方法が一般的であり，演習中の学習者の困難を把握するには不十分である．そこで，本研究では学習者が成果物を作成するプロセスに着目し，学習者が演習中に難しいと感じた箇所や作業を検出する手法を提案する．今回はオブジェクトモデリング演習を対象とし，クラス図作成プロセス収集・分析用ツールを構築した．また，提案手法で定義した難所の検出のための指標の妥当性を検証する実験を行った．その結果，クラス図に対する編集イベントの発生間隔と学習者にとっての難所に関係があることが示唆された．また，編集イベント発生間隔が大きくなるときの学習者の行動パターンを確認することができた．

Proposal of a Detection Method of Difficult Places for Learners in Object Modeling Exercises and a Support Tool for the Method

TAKAFUMI TANAKA[†1]　KAZUKI MORI[†2]　HIROAKI HASHIURA[†3]
ATSUO HAZEYAMA[†1]　SEIICHI KOMIYA[†4]

In recent years, practical software development exercises have been carried out in many higher education institutions. To carry out the exercises effectively, it is important that teachers understand the difficulty of students in exercises and advise appropriately for it. Currently, a common way to check results of the exercises is that teachers review artifacts which students submitted. However, there is a problem in this way that it can't be obtained information regarding the process of students to create artifacts, therefore teachers can't fully understand difficulties of the students. We focus on students' artifacts creation process and we proposed a method for detecting difficult places of students during the exercises. We have developed a tool that collects the class diagram creation process by students during exercises and analyze it. We conducted an experiment to validate the indicators for detecting difficult places for students which were defined in the proposed method. From the result, we found the relationship between the difficult places for a participant and interval of editing events occurrence. In addition, we discovered the behavior patterns of the participant when interval of editing events occurrence increases.

1. はじめに

現代のソフトウェアは大規模化，複雑化が進み，その開発は以前よりも困難になっている．このため，現代のソフトウェア開発者は，プログラミングのスキルだけでなく要求分析や設計などの上流工程を含む包括的で高度なソフトウェア工学の知識とスキルを有することが求められている．このような背景から大学等の高等教育機関では情報技術教育の高度化のため，様々な取り組みがなされている．近年では，グループで仮想プロジェクトに取り組む形態の学習(PBL : Project Based Learning)が多数の大学等で実施され，効果が報告されている[1]．しかし，PBL を実施するためには，学習者がソフトウェア開発に関する一定の基礎スキル(開発方法論や各種成果物の作成方法など)を身につけていることが必要である．そのため，主に初学者向けの方法として，教授者による講義と学習者が実際に成果物を作成する実践的な演習(個人で行う)を組みあわせて学習を進めるという形態の授業が実施されている．

初学者向けの個人演習を効果的に実施するためには，学習者にとって理解が難しいポイントを教授者が認識し，適切な指導を行う必要がある．このためには，学習者の演習成果（成果物や演習中の活動の記録）を分析し，多くの学習者が共通して難しいと感じるポイントを見つけることが重要である．現在は，提出された成果物を教授者がレビューすることで演習成果を確認するのが一般的である．しかし，この方法では成果物作成プロセスに関する情報が得られないため，演習中に学習者が直面した難所(難しい箇所や手順)を把握することができない．そのため，提出された成果物に悪所（間違いや不適切な箇所）が存在する場合，悪所の指摘は可能だがその原因となった難所を探すといったことは不可能である．また，目立った悪所が存在しなければ「問題なし」と判断せざるを得ない．しかし，最終成果物に問題がなくても，手戻りややり直しが多く発生したポイントや教授者の想定よりも多くの時間がかかったポイントなどは学習者にとって困難であった可能性がある．このような推測は現状では不可能である．このことは効果的な

†1 東京学芸大学
Tokyo Gakugei University.
†2 芝浦工業大学大学院
Shibaura Institute of Technology
†3 日本工業大学
Nippon Institute of Technology
†4 国立情報学研究所
National Institute of Informatics

演習の実施のためには問題と言える．

一方で，現代のソフトウェア開発者は上流工程のスキルを有することが求められている．オブジェクト指向開発におけるオブジェクトモデリングなど，開発するシステムをモデル化するモデリングのスキルは上流工程で重要なスキルであり，その育成のため大学等では実践的な演習を実施するなど教育に注力している[2]．一般に，モデリングのスキル習得は経験によるところが大きく，勘所を掴めるとスムーズにモデリングできるようになると言われている[3]．しかし，勘所をつかめずに意欲を失ってしまう学習者が少なからずいるのが実情である．勘所をつかめる学習者とそうでない学習者の違いや，勘所をつかめない原因などについて客観的なデータに基づいて示すことは難しく，はっきりとはわかっていない．このことはモデリング教育の難しさの大きな原因となっている．

そこで，本研究ではオブジェクトモデリング演習を対象とし，学習者のクラス図作成プロセスから演習中の学習者の活動のデータを収集し，客観的な指標に基づいて分析することで学習者が演習中に難しいと感じたポイントを自動検出する手法とその支援ツールを提案する．

2. 関連研究

学習者の成果物作成プロセスのデータを収集，分析した研究は，プログラミングに関するものが多数存在する．

谷川ら[4]は大学の C 言語プログラミング演習において，学生のソースコード作成中の関数呼び出し順の記録から試行錯誤の過程を分析し，学生の習得項目を算出する手法を提案した．まず，教員が関数呼び出し記録用の関数(スパイ関数)，演習過程記録ツール，スパイ関数を用いて達成できる課題を用意する．スパイ関数を用いて学生が作成した課題プログラムがツール上で実行されるたびに，スパイ関数の呼び出し順序が記録される．谷川らは，課題作成中の多くの学生に共通する関数の呼び出し順序の変遷を呼び出しパターンとして導出し，別途行った習得項目確認小テストの結果と呼び出しパターンの間に有意な関連性があることを示した．これにより，教員は呼び出しパターンを用いて効率的に習得項目を把握し，効果的な指導が行えるとしている．谷川らの成果物作成過程に注目するアプローチは本研究に近いが，適用場面が異なる．また，スパイ関数を使用させるため，自由度の高い課題作成ができないことは問題と言える．

モデリング教育に関する研究として，小木曽ら[2]の研究がある．小木曽らは，大学で情報システムの設計原理の理解を目的として実施されている PBL 型のモデリング演習にティーチングアシスタントとして参加し，演習中の問題を調査した．さらに，調査結果を分析し，モデリング演習の問題の原因を，①モデリング演習の目標やモデル作成アプローチに関する説明不足・理解不足，②モデルの設計ノウハウがわからない，③UML に関する用語，使用方法がわからない，④astah community(モデル図作成ツール)の使用方法の説明不足，⑤受講生のモチベーションが高くない，⑥グループワークで協力できない，⑦ハードウェアの故障，の 7 項目に分類した．小木曽らはこれらの原因分類に対して対策案を検討し，モデリング演習，UML 技法，モデル設計ノウハウを支援する Web コンテンツを作成した．小木曽らのモデリング演習における問題の原因分類①②③は，本研究で対象とする初学者向け個人演習でも共通するものと考えられる．しかし，挙げられている問題と個々の学習者や成果物との関係については述べられていない．

3. 提案手法

本手法では，「学習者が成果物を作成する過程で難しいと感じた箇所および作業段階」を「難所」と定義する．

一般に，学習者が難所に直面した時には，手が止まる，何度も同じ箇所を書き直すなど特徴的な行動パターンが現れる．そこで，本手法が対象とするオブジェクトモデリング演習でも以下のような行動パターンが現れると仮定する．

- **作業が止まる**
  作業の進め方が分からず，手が止まってしまう状態
- **同じ箇所を何度も編集する**
  試行錯誤を重ね，同じ箇所を繰り返し編集している状態
- **手順の手戻り**
  作成手順の作業段階が順調に進まず，手戻りが何度も発生する状態
- **ある作業段階に多くの時間がかかる**
  難しい作業段階で作業時間が長くかかっている状態

本手法では，学習者のクラス図作成プロセスからこれらの行動パターンを検出することで難所を検出する．

提案手法の流れは以下のとおりである．

(1) 4 節で述べるツールを用いて学習者のクラス図作成プロセスのデータを自動収集する

(2) 収集したデータを分析し，学習者にとっての難所を客観的な指標に基づいて検出する

3.1 節で(1)について，3.2 節で(2)について詳細を述べる．

3.1 クラス図作成プロセスのデータ収集

クラス図作成プロセスのデータとして，クラス図に対する編集の履歴(以下，編集イベント)をツールで自動収集する．具体的には，以下のイベントを収集する．

- クラスの作成，削除
- クラス名の更新
- 属性の追加，更新（削除含む）
- メソッドの追加，更新（削除含む）
- 関連の作成，削除
- 関連名の追加，更新（削除含む）

● 多重度の更新（削除含む）
● ノートの作成，更新（削除含む）
● クラス，ノートの移動

3.2 難所検出のための指標

本手法では，収集した編集イベントを分析して次に挙げる 4 つの指標を計算し，特徴的な行動パターンを検出する．これにより，クラス図中の関連する箇所，および作業手順中の作業段階を学習者個人にとっての難所として検出する．

● **作業が止まる**
編集イベントの発生時間の間隔
● **同じ箇所を何度も編集する**
図上の同名のクラス，関連の編集回数
● **手順の手戻り**
学習者が入力する作業段階の手戻り回数
● **ある作業段階に多くの時間がかかる**
作業段階ごとの作業時間の合計

4. 提案手法の支援ツール

本節では，提案手法を支援するための支援ツールについて述べる．提案ツールは Web ブラウザ上でクラス図を作成できる UML エディタであり，提案手法で示した編集イベント収集などの機能を持つ．

4.1 機能要件

提案ツールに必要な機能要件について述べる．

(1) クラス図作成プロセス収集機能

提案手法の実現のため，3.1 項で示した学習者のクラス図編集イベントを自動収集し，分析可能な形式（関係データベース等）で記録できる必要がある．

(2) 作業段階記録機能

学習者が行っている，オブジェクトモデリングの手順（講義等で教授者が説明したもの）中の作業段階を記録できる必要がある．

(3) データ分析結果表示機能

収集した編集イベントから 3.2 項で述べた指標を計算し，難所を検出して結果を表示できる必要がある．

また，編集イベント発生間隔の分析結果をクリックすることでそのイベントが発生したタイミングのクラス図を再現する，編集回数が多いクラス名や関連名をクリックすることでそのクラスや関連が作成されたタイミングのクラス図を再現するなど，分析結果を有効に利用できる仕組みが必要となる．

(4) 作成プロセス再現機能

難所として検出された箇所または作業段階で学習者が行っていた編集操作を教授者が確認するため，編集イベントから，学習者のクラス図作成プロセスを再現し，閲覧する機能が必要である．

4.2 提案ツールの構成と実装

提案ツールは大学の講義等で使用することを考慮し，多数の学習者に容易に画一的なモデリング環境を提供するため，Web ブラウザから利用できる Web アプリケーションとして実現する．これにより，提案ツールが収集したデータをサーバ上で一元的に管理し分析することが容易になる．

本研究では，オープンソースプロジェクト GWTUML[5]が開発している，Web ブラウザ上で UML を作成できるオープンソースソフトウェア GWTUMLAPI および GWTUMLDrawer を拡張し，機能要件で述べた機能を実装する．

(1) GWTUML のプロダクトについて

本研究で使用する GWTUML のプロダクトは，ブラウザ上での UML 作成に必要な基本的なパーツを API 化した GWTUMLAPI と，それを用いて作成された Web アプリケーション GWTUMLDrawer の 2 つである．いずれも GWT（Google Web Toolkit）[6]を使用して開発されている．これらの関係を図 1 に示す．

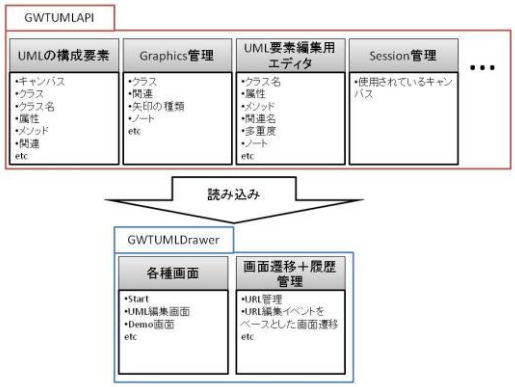

図 1　GWTUML のプロダクト
Figure 1　Products of GWTUML.

GWTUMLDrawer が有する機能を以下に示す．

① Web ブラウザ上でクラス図，オブジェクト図，シーケンス図，およびそれらを混合した図を作成することができる（図 2）

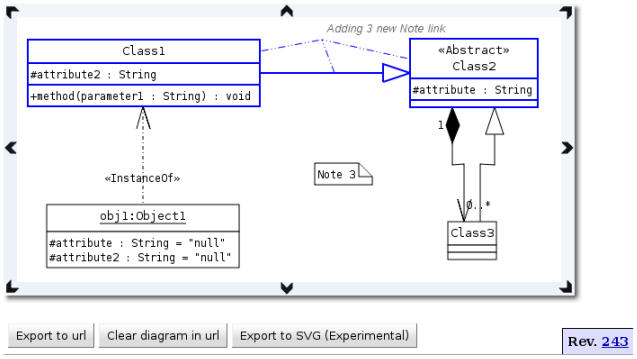

図 2　UML 作成画面
Figure 2　Screen shot of UML drawer.

② 作成したモデル図の情報を Base64 で文字列にエンコードし，URL にエクスポートする．また，UML 作成画

面で，エンコードした文字列を URL に入力することでモデル図を復元することができる

今回はクラス図を対象としているので，主にクラス図作成機能を拡張する．また，モデル図の URL エンコード機能はデータ収集で利用する．

(2) 拡張箇所について

提案手法実現のため，GWTUMLAPI および GWTUMLDrawer に以下の①~③の拡張を行った．拡張部分のイメージを図 3 に示す．

① ユーザ管理，データ蓄積用データベースの追加

MySQL を用い，ユーザ管理，およびデータ蓄積用に表 1,2 に示すテーブルを持つデータベース（以下，DB）を作成した．

表 1　ユーザ管理用テーブル
Table 1　Student table in DB.

カラム	意味
student_id	学生に一意に割り振られる ID
password	パスワード

表 2　編集イベント蓄積用テーブル
Table 2　Editing events table in DB.

カラム	意味
edit_event_id	全ての編集イベントに一意に割り振られる ID
student_id	イベントを発生させた学生の ID
edit_event	編集イベント．編集イベントの内容
canvas_url	イベント発生時のモデル図を Base64 でエンコードした文字列
edit_datetime	編集イベントが発生した日時
step	学生が入力した作業段階

② 編集イベント収集機能の追加

クラス図に対する編集が行われるたびに編集内容とクラス図のスナップショットを DB に格納する機能を実装した．

③ データ分析，結果表示機能の追加

収集した編集イベントのデータを用いて難所検出の指標を計算する機能を実装した．具体的な計算式は以下のとおりである．

● 編集イベント発生間隔

i 番目の編集イベント発生時刻を T_i とする．

i 番目と i-1 番目の編集イベントの発生間隔 Δ_i は以下の式で計算する．

$$\Delta_i = T_i - T_{i-1}$$

● ある箇所の編集回数

クラス図上の同名箇所（クラス名，関連名等）ごとの編集回数を合計する．

● 作業段階の手戻り回数

作業手順は(1)クラス定義，(2)関連定義，(3)属性定義の順で進むものとし，後の段階から前の段階に戻った回数を手戻り回数として計算する（(3)から(2)に戻った場合，(2)への手戻り 1 回と計算）．

● 作業段階の作業時間

作業段階ごとの作業時間の合計を計算する．

④ クラス図作成プロセス再現機能

収集したクラス図のスナップショットを用い，編集イベント単位でクラス図の変化を再現する機能を実装した．

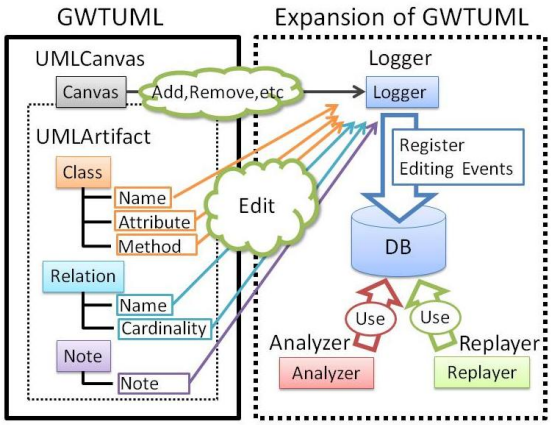

図 3　GWTUML の拡張
Figure 3　Expansion of GWTUML.

5. 実験

本節では提案ツールを用いた実験について述べる．

5.1 目的

本実験の目的は，提案ツールによってデータを収集し，3.2 項で述べた難所検出のための指標が妥当であるか評価することである．

5.2 対象

提案手法はオブジェクトモデリングの初学者を対象としている．よって，本実験の実験協力者（以下，協力者）も同様の初学者が適切である．今回は東京学芸大学情報教育専攻の 2 年生 1 名を対象に実験を行った．

5.3 方法

大学で実施されている講義と演習を組み合わせた形式の授業を再現し，演習中のクラス図作成プロセスのデータを収集した．演習終了後，協力者にインタビューを行い，難しいと感じた箇所などを調査した．以下に実験の流れを述べる．

(1) 講義

講義では，著者が OMT 法に基づいてオブジェクトモデリングの手順を説明した（1 時間 30 分程度）．OMT 法では，大きく分けて①クラス識別，②関連識別，③属性識別の作業を行う．今回の実験では①～③を作業段階と定義してデ

ータを収集した．

なお，今回は抽象度の高いモデリングを想定し，操作の識別や継承による図の洗練などの手順は省略している．

説明の後，小規模の練習問題を用意し，協力者に提案ツールを用いて演習問題に取り組んでもらいながら，ツールの操作方法について説明した．

(2) 演習

演習では著者らが課題を用意し，協力者に提案ツールを用いてクラス図を作成してもらった（1 時間程度）．課題は会社の人員管理システムの開発を題材にしており，著者の想定ではクラス数が 6，関連数が 6 程度の規模であった．

演習中は講義で用いた資料等の閲覧は自由とし，協力者からの質問はツールの操作についての質問のみ受け付けた．

(3) インタビュー

演習終了後，協力者にインタビューを行った．まずは演習終了直後に以下について調査した．

① 難しかった作業段階とその理由
② 難しかった箇所とその理由

次に，編集プロセスを再現して閲覧しながら以下について調査した．

③ 難しかった箇所の編集時の思考過程

最後に，分析結果をもとに以下について調査した．

④ 分析結果で難所と推測される箇所について，協力者は難しいと感じていたか

また，実験やツールに関しての感想と意見についても調査した．

6. 実験結果

6.1 協力者が作成したクラス図

今回の実験で協力者が作成したクラス図を図 4 に示す．

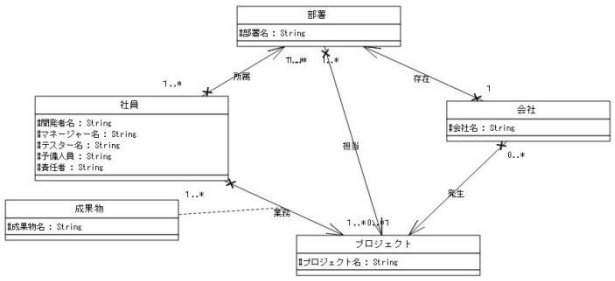

図 4 協力者が作成したクラス図
Figure 4 Class diagram made by a participant.

図 4 のクラス図の作成プロセスにおいて，122 個の編集イベントが収集された．編集イベントの例を図 5 に示す．

edit_event_id	edit_event	edit_datetime
971	DropArtifact:Class$(739,89)!Class2!!!	2014-01-16 14:40:28
972	DropArtifacts:	2014-01-16 14:40:28
973	ClassName:会社:Class2:会社	2014-01-16 14:40:42
974	DropArtifact:Class$(690,274)!Class3!!!	2014-01-16 14:40:51
975	DropArtifacts:	2014-01-16 14:40:51
976	ClassName:部署:Class3:部署	2014-01-16 14:40:58
977	DropArtifact:Class$(460,103)!Class4!!!	2014-01-16 14:41:10
978	DropArtifacts:	2014-01-16 14:41:11
979	ClassName:社員:Class4:社員	2014-01-16 14:41:18
980	DropArtifact:Class$(522,239)!Class5!!!	2014-01-16 14:41:36
981	DropArtifacts:	2014-01-16 14:41:36
982	DropArtifact:Class$(472,225)!Class5!プロジェクト!!	2014-01-16 14:41:51
983	DropArtifacts:	2014-01-16 14:41:51

図 5 編集イベントの例
Figure 5 An Example of editing events.

6.2 指標の計算結果

(1) 編集イベントの発生間隔

編集イベントの発生間隔を計算した結果を図 6 に示す．横軸は時系列に編集イベントを並べた時の番号を示しており，縦軸は編集イベント発生間隔を示している．編集イベント発生間隔の平均は約 21 秒であり，グラフ上に図示している．編集イベント番号が 1~22 のイベントはクラス識別段階，23~81 のイベントは関連識別段階，82～122 のイベントは属性識別段階の編集イベントである．ただし，編集イベント番号 90~94 において関連識別段階への手戻りが 1 度発生している．

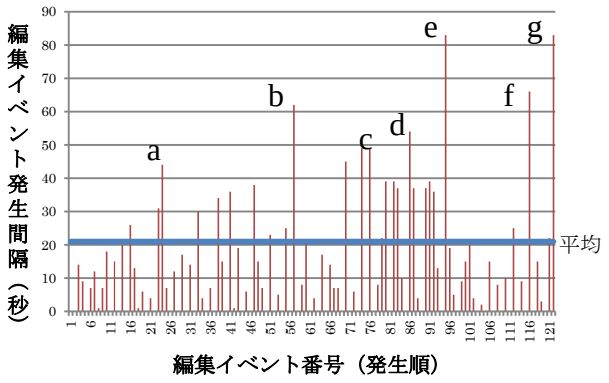

図 6 編集イベント発生間隔
Figure 6 Interval of editing events occurrence.

(2) ある箇所の編集回数

同名クラスの要素に対する編集回数を計算した．結果を表 3 に示す．

表 3 クラスごとの編集回数
Table 3 Number of editing for each class.

クラス名	編集回数
社員	8
成果物	3
会社	2
部署	2
プロジェクト	2

(3) **作業段階の手戻り回数**

作業段階ごとの手戻り回数を表 4 に示す. 手戻り回数は，後の作業段階からその作業段階に手戻りした回数で示している．例えば，関連識別の手戻り回数 1 は，後の作業段階である属性識別から関連識別への手戻り 1 回という意味である．

表 4　作業段階の手戻り回数
Table 4　Number of rework.

作業段階	手戻り回数
クラス識別	0
関連識別	1
属性識別	0

(4) **作業時間ごとの作業時間の長さ**

作業段階ごとの作業時間の合計を表 5 に示す.

表 5　作業段階ごとの作業時間
Table 5　Working time of each step.

作業段階	作業時間（秒）
クラス識別	184
関連識別	927
属性識別	509

6.3 インタビューの結果

演習終了後に行ったインタビューの結果を示す.

(1) **演習直後のインタビュー**

①　難しかった作業段階とその理由

- 属性抽出が最も難しかった
  - クラスにするべきか属性にするべきかわからなかった
  - どのクラスの属性として定義するべきか難しかった

②　難しかった箇所とその理由

- 社員クラスの属性が難しかった
  - 問題文から何を抽出するべきかわかりにくかった

(2) **編集プロセスを再現しながらのインタビュー**

③　難しかった箇所の編集時の思考過程

- クラス抽出は比較的スムーズにできた
- 関連については，問題記述に明記されていない多重度の読みとりが難しかった
- 社員クラスの属性を編集しているときは，属性となるべき語句がどれなのかよくわからなかった
- 成果物クラスの属性として責任者を抽出していたが，社員クラスを編集しているうちによくわからなくなり，成果物クラスから責任者を削除して社員の属性にしてしまった
- 関連定義はツールの操作が煩雑だったため時間がかかった

(3) **分析結果をもとにしたインタビュー**

④　分析結果で難所と推測される箇所について，協力者は難しいと感じていたか

- （作業時間，手戻り回数から難所と推測される関連識別について）関連識別も難しかったが，より難しかったのは属性定義
- （編集イベント発生間隔，編集回数から難所と推測される社員クラスについて）社員クラスの属性識別が最も難しかった
- （手戻り回数が少なかった理由について）クラス図の間違いや思考方針の誤りは他者に指摘されなければ自分では気づかない．よって，同じ箇所を何度も編集したり前の作業段階に戻ってやり直したりすることは少ない

(4) **その他のインタビュー**

- 手戻りの際に作業段階を切り替えるのを忘れやすい
- 関連識別のツールの操作が煩雑でわかりにくい
- （実際の演習で編集プロセスを収集されるとしたら，抵抗はあるか）特にない．教授者がプロセスを見てアドバイスを行うのは良いと思う

7. 議論

7.1 編集イベントの発生間隔

図 6 より，属性識別段階に入った 82 番目から 95 番目までの編集イベントには平均の約 2 倍程度の長い時間がかかっているものが多い．また，インタビュー結果から，協力者が属性識別段階を難しいと考えていたことが分かる．そこで，作業段階ごとに編集イベント発生間隔の平均を計算した．結果を表 6 に示す.

表 6　作業段階ごとの編集イベント発生間隔の平均
Table 6　Average interval of editing events occurrence by each step.

作業段階	編集イベント発生間隔の平均（秒）
クラス識別	10.9
関連識別	21.8
属性識別	24.7

表 6 より，属性識別段階の編集イベント発生間隔が最も大きいとわかる．このことから，編集イベントの発生間隔と協力者にとって難しい作業段階の間には関係があると考えられる.

さらに，協力者はインタビューで「社員クラスの属性識別が難しかった」と答えているが，社員クラスの属性を編集した 86 番目や 95 番目の編集イベントでは 86 番目で 54 秒，95 番目で 83 秒と，平均に比べて編集イベント発生間隔が大きくなっている．このことから，協力者にとって難

しい箇所と編集イベント発生間隔の間にも関係がある可能性が考えられる．これらの検討から，編集イベント発生間隔は難所検出のための指標として有効であることが示唆された．

7.2 ある箇所の編集回数

表 3 より，編集回数が最も多い箇所は社員クラスであるとわかる．また，協力者はインタビューで社員クラスの属性識別が最も難しかったと答えている．しかし，作成プロセスを再現して確認したところ，同じ箇所に対して更新を繰り返すなど試行錯誤と考えられる挙動は確認できなかった．この理由については，協力者がインタビューで「自分自身では間違いに気付かない．間違いに気付くためには，他者からの指摘(インスペクションなど)が必要」という趣旨の回答をしている．社員クラスの編集回数が多いのは，単純に属性数が多かったためと考えられる．

また，演習全体の作成プロセスを再現して試行錯誤と思われる挙動を調査した．その結果，同じ箇所に対して 1 度の更新（修正）を行った様子は散見されたものの，複数回更新するような試行錯誤と言える様子は見当たらなかった．インタビューの回答のとおり，初学者は自分で間違いに気付くことが難しいため，誤りを作りこんでもそれに気付かない傾向があると言える．

このことから，難所であっても試行錯誤によって編集回数が大きく増加するとは言えず，今回の実験からは編集回数と難所の関連性は見られなかった．

ただし，演習中にインスペクションなどで他者からの指摘を受け，クラス図を修正するようなプロセスを入れた場合は有効な指標となる可能性もあると考えられる．

7.3 作業段階の手戻り回数

表 4 より，関連識別への手戻りが 1 度発生したことがわかる．しかし，インタビュー結果によると，協力者は属性識別を難しいと認識しており，手戻り回数と協力者にとって難しい作業段階の関連性は見られなかった．

また，今回は協力者に作業段階を入力させることでデータを収集したが，この方法では協力者はたびたび作業段階の記録を忘れているようだった（関連クラスを抽出した際に手戻りが発生しているが，記録されていなかった）．この点については，編集イベントから作業段階を自動で分類してデータを収集する方が正確であると考えられる．

また，演習全体を通しても手戻り回数が 1 回と少なかった理由として，7.2 項で述べた編集回数と同様の理由が考えられる．

7.4 作業段階ごとの作業時間の長さ

表 5 より，各作業段階にかかった時間は関連識別が最も長く，次いで属性識別，最も短かったのはクラス識別であるとわかる．しかし，インタビューの結果，協力者にとって最も難しかったのは属性識別であり，分析結果と一致していない．このことから，作業段階ごとの作業時間の長さと協力者にとって難しい作業段階の関係は見られなかった．

ただし，ツール上で関連を作成する操作は，関連生成，関連名生成，多重度の編集などの手順が必要であり，属性やクラス名の編集に比べて手間がかかる．しかし，今回の実験では，この点を考慮していない．学習者にとっての難しさによる作業時間の差をより正確に検出するためには，作業段階で固有の作業量の差による作業時間の差を検出し，分析において考慮する必要がある．そのため，今後は協力者数を増やしてデータを収集し，集団に共通して現れる作業時間の差を調査し，それを考慮して結果を再考察する必要があると考えている．

7.5 編集イベント発生間隔が大きい箇所の行動

図 6 に a～g で示した編集イベントの発生間隔は他に比べて特に大きくなっている．これらの編集イベントの周辺を作成プロセス再現機能で再現し，時間がかかった理由を調査した．

a. 関連識別に入った直後のイベント

a で示した編集イベントは関連識別に入ってから 2 つ目の編集イベントであり，最初に関連名を編集したイベントであった．この時，協力者からツールの使用法（関連名の編集）について質問を受けて対応したため，他に比べて編集イベントの発生間隔が大きくなったと考えられる．

また，この直前の編集イベントは関連識別に入った直後の編集イベントであるが，これも周囲に比べて発生間隔がやや大きめ（31 秒）に空いている．このことから，協力者はクラス識別を終えたタイミングで抽出漏れが無いか等の確認を行っていたと考えられる．

b. 1 度作成した関連の多重度を更新

b で示した編集イベントは 1 度作成した関連の多重度を更新（修正）したイベントであった．この時，協力者は課題文に示された要件と，作成しているクラス図の相違に自ら気付き，どのように修正すべきか検討していたと考えられる．このように，1 度作成した箇所を更新する場合は課題文等の確認と修正方法の検討がなされるため，編集イベント発生間隔は大きくなると考えられる．

c. 関連クラスを作成した箇所

c で示したイベントは，社員-プロジェクト間の関連クラスを作成したイベントであった．著者らの経験から，関連クラスの抽出はオブジェクトモデリングで学習者が（特に初学者が）つまずきがちな箇所であり，協力者も抽出に苦労したと推測される．

d. 属性識別に入った直後

d で示した編集イベントは属性識別に入った直後のイベントであった．作業段階の境目であるため，a と同様に関連識別の抽出漏れや関連名などについて見直しを行っていたと考えられる．このことから，作業段階の境目では編集イベント発生間隔は大きくなると考えられる．

e. 間違いを作り始めたイベント

e で示した編集イベントは社員クラスの属性に間違いを作りこんだ最初のイベントであった.社員クラスの属性は,インタビューで協力者が「最も難しかった」と話しており,実際に間違いを含んでいる箇所である．e で示した編集イベントでは，それまで正しく作られていた属性を更新して間違った属性の 1 つ目を作成しており，その後続けて社員クラスに間違った属性を追加していた．作成プロセスの再現とインタビューから，協力者は e で示した編集イベントを発生させる直前に課題文を見ながら進め方を検討し，その結果間違った解釈をして作業を進めてしまったと推測される．このように，方針や解釈の誤りによって大きな間違いを作りこむ直前には編集間隔が大きくなると考えられる.

f. **必要な属性を上書きしてしまう直前**

f で示した編集イベントは，関連クラスである成果物クラスを移動し，位置を修正したイベントであった．また，直後に成果物クラスの属性を更新し,必要な属性(責任者)に新たな属性（成果物名）を上書きするイベントが発生していることがわかった．f のイベントで発生間隔が大きくなった理由は，クラスの移動よりもその後の属性の更新に関して検討していたためと考えられる.

g. **最後の見直しの後の編集**

g で示したイベントは最後の編集イベントであった．これは，全体の見直しを行って抽出漏れなどを確認した編集と考えられる.

7.6 成果物の間違い箇所の考察

間違い箇所が作られた経緯を調査した．協力者が作成したクラス図で間違いがある主な箇所は,社員クラスの属性,成果物クラスの識別方法（関連クラスとして識別されているが，本来は関連クラスではない)，会社-プロジェクト間の関連である.

社員クラスの属性については，インタビューによって難所であったことと作った経緯が明らかになっている.

成果物クラスの識別方法については，インタビューから協力者の関連クラスについての理解が不十分であったことが原因と考えられる.

会社-プロジェクト間の関連については,この関連が引かれた前後の作成プロセスを再現し，関連が引かれた経緯を調査した．調査の結果，この関連は属性識別段階から手戻りして引かれたことがわかった．このことから，協力者は属性識別を行っている途中で関連の抽出漏れがあると思い,本来は不要な関連を追加してしまったと推測できる.

8. 結論と今後の課題

8.1 結論

本研究ではオブジェクトモデリング演習における学習者にとっての難所を自動検出する手法を提案し，その支援ツールの開発を行った．さらに，提案手法で仮定した難所検出のための指標が妥当であるか検証するため，実験を行った．今回の実験結果から，編集イベント発生間隔と協力者(学習者)にとっての難所の間に関係があり，指標として有効であることが示唆された.

また，作成プロセスの再現により，編集イベント発生間隔が大きくなる時の協力者の行動に，①作業段階の境目，②1 度作成した箇所の修正，③間違いを作り始めるポイント（その後の方針を決めたポイント)，の 3 つのパターンが存在することが確認できた．さらに，成果物上の間違い箇所を作り始めた編集イベントを探し，その前後の作成プロセスを再現することで，間違いが作られた経緯を従来よりも詳細に推測することが可能になった．このように，学習者のクラス図作成プロセスから，演習指導に役立つ可能性のある情報が得られることが確認できた.

8.2 今後の課題

今後は協力者数を増やして実験を行い，さらなるデータ収集を行う．これをもとに，定量的な分析を行い，編集イベント発生間隔を用いた難所検出のための閾値を検討する.また，複数の学習者が存在することを前提にした難所検出のための分析方法および指標を検討し，その妥当性を評価する.

提案ツールについては，提出されたクラス図，分析結果，作成プロセスを効果的に組み合わせて可視化する方法を検討し，実装する．また，クラス図作成機能の UI を改善し，利用者がより自然にモデリングに取り組めるように修正する．その後，実際に大学の授業に導入して実証実験を行いたい.

謝辞

本研究は JSPS 科研費 25750090 の助成を受けたものである．また，実験に協力して頂いた学生に深く感謝する.

参考文献

1) 松浦佐江子: 実践的なソフトウェア開発によるソフトウェア工学教育，情報処理学会論文誌, Vol.48, No.8, pp.2578-2595 (2007).
2) 小木曽禎，遠山紗矢香，湯浦克彦: 学生向けモデリング演習支援システムの開発と評価，情報処理学会研究報告, Vol.2011-CE-109, No.5, pp.1-9 (2011).
3) 加藤由花，南葉幸雄: 概念データモデリングによる情報システム上流工程教育，情報処理学会論文誌, Vol.50, No.2, pp.626-636 (2009).
4) 谷川紘平，ディン ドン フォン，原田史子，島川博光: C 言語関数呼出しの記録を用いた演習過程での習得項目の把握，電子情報通信学会論文誌 D, Vol.J95-D, No.12, pp.2079-2089 (2012).
5) GWTUML, http://code.google.com/p/gwtuml/
6) GWT Project, http://www.gwtproject.org/