経験知習得のための思考の協調的追体験学習支援システムの開発 -デザインパターンを対象にして-

代表研究者	小尻智子	関西大学システム理工学部　准教授
共同研究者	瀬田和久	大阪府立大学大学院理学系研究科　教授
共同研究者	林佑樹	大阪府立大学現代システム科学域　助教

1 はじめに

デザインパターンは，オブジェクト指向パラダイムでのソフトウェア設計ノウハウを集めたカタログである[1]．よく出会う様々な問題に対して先人らが試行錯誤して得た経験を一般化したものであり，オブジェクト指向設計における経験知とみることができる[2]．先人による創意・工夫のプロセスを経て形式知化されたデザインパターンの学習を通じて，その設計方法の意義や適用条件をより深く理解することは，オブジェクト指向設計における設計意図の習得と捉えられ，パターンを直接使用しない場合であっても優れた設計を行えるようになると考えられている[1]．この意味でデザインパターンはオブジェクト指向の初学者にとってもよい教材であると考えられる．しかし，そこに埋め込まれた設計意図を理解しないまま単に記憶していくのでは，新しい問題に対して自ら創意工夫して設計できるようになるとは考えにくい．

デザインパターンのような経験知が生み出される過程を体験することは,そこでなされた意志決定基準(以後，設計意図と呼ぶ）の理解を促すと考えられる．しかし，成果物のみが示されその生成過程が明記されていることは多くはない．経験知が形成されるまでには様々な生成物を比較し，利点・欠点を検討する過程を経て何らかの基準に基づく選択がなされているが，その基準である設計意図が暗黙になっているために，学習者自身がその過程を推察して意図を理解できるかどうかは学習者の能力に依存する．

デザインパターンの学習支援では，問題を与え，これに対処する設計を学習者が作成することからスタートし，修正を加えていくことでデザインパターンを導出させようとする研究がある[3, 4]．しかし，デザインパターンの設計意図を修得しようとする学習者にデザインパターンを導出させることは必ずしも容易ではない．本研究では，正解としてのデザインパターンを出発点として，機能を維持しつつ望ましくない設計（代替設計)に変更させることで,デザインパターンの設計意図を読み取るような学びを促すことを目的とする．このことで，経験知の形成過程の一部を体験する学びが学習者に課せられることとなり，デザインパターンを教材とした設計意図の学びを促すこととなる．

このような目的のもと，申請者はこれまで，学習者が他の設計を試行錯誤するための土台として，デザインパターンの用いられたクラス図を用いないクラス図（代替設計）に変更するという学習手法を提案し，適切な代替設計を作成するための支援システムを構築してきた[5]．しかし，代替設計を作成させるだけではデザインパターンの利点を明示的に理解させることはできない．そこで，対象とする目的に拡張や変更を加えたシナリオを学習者間で相互に提示し合うことのできる環境と，シナリオを満足するようにデザインパターンを用いた設計と代替設計を変更できる環境を構築してきた[6]．2016 年度には，様々なシナリオを学習者間で相互に生成しあうことにより，多様な観点からの代替設計を生成・議論できるような協調学習環境を導入する．多様な観点からデザインパターンを導出するためには，多様なシナリオが生成されることが望ましい．構築した協調学習支援システムは，シナリオを導出するための発想支援の機能を含む．

デザインパターンはソフトウェア設計に関する経験知であるが，このような経験知はデザインパターンだけではない．良いプログラムの構造はまた経験知とみることができる．2016 年度は，デザインパターンを対象とした協調的追体験学習をデザインパターン以外のプログラミング言語，具体的にはＣ言語にも適用することにより，本提案手法の一般性も示す．

2 協調学習環境

2.1 目的

本研究課題では，デザインパターンを題材として，経験的に獲得された良いプログラム構造の利点を理解

させるため，デザインパターンを用いたプログラム（正解）をいない代替設計に変更させる学習方法を提案している．また，プログラム作成のために与えられた状況を拡張・変更するシナリオを学習者自らが設定し，そのシナリオをもとに正解と代替設計を拡張させてデザインパターンの拡張性を理解させる学習手法を提案している．2015 年度の前期は，学習者の作成したシナリオを他の学習者と共有することのできる学習環境と，各シナリオを用いて変更した正解・代替設計を学習者間で議論できる協調学習環境を構築した．

シナリオとは，プログラム作成のための問題として用意された状況を，新しい状況に変更させるためのストーリである．小島らは，問題は解と状況を表す属性で構成されていると述べている[7]．したがって，シナリオは作成するプログラムが異なる問題と，プログラムは同じだが状況の属性が異なる問題の 2 種類に大別することができ，多様なシナリオを構成するためには解だけではなく状況を表す属性も変更することがのぞましい．状況を変更させるには，様々な属性を想定する発想が必要となる．そこで，状況を表す属性の発想を支援するための発想支援機能を有した協調学習環境を構築した．

2.2　学習支援システム

構築した発想支援機能の枠組みを図 1 に示す．本システムでは学習者が生成した状況を表す語に対し，関連する語を提示することで発想を促す機能（発想支援機能）を有する．連想的なアイディア生成過程では，元の語と発想された語の間には何らかの関連が存在する．元の語に適用できる関連はあらかじめ決っているわけではなく，また人によって思いつく関連は異なる．このように連想的に語を導出していく過程では，関連から導き出せる語だけでなくどのような「関連」を思いつくことができるかが創出される語の質を左右する．発想支援機能では，発想に行き詰まっている状態に対し，現在導出している語に適用可能な「関連」を提示することで，学習者自身で新しい語を創出することを支援する．既出の発想過程を見ることで学習者は発想方法を推測できる可能性が高いため，自身あるいは他者の既出の発想過程を提示することで発想方法への気付きを促進する．

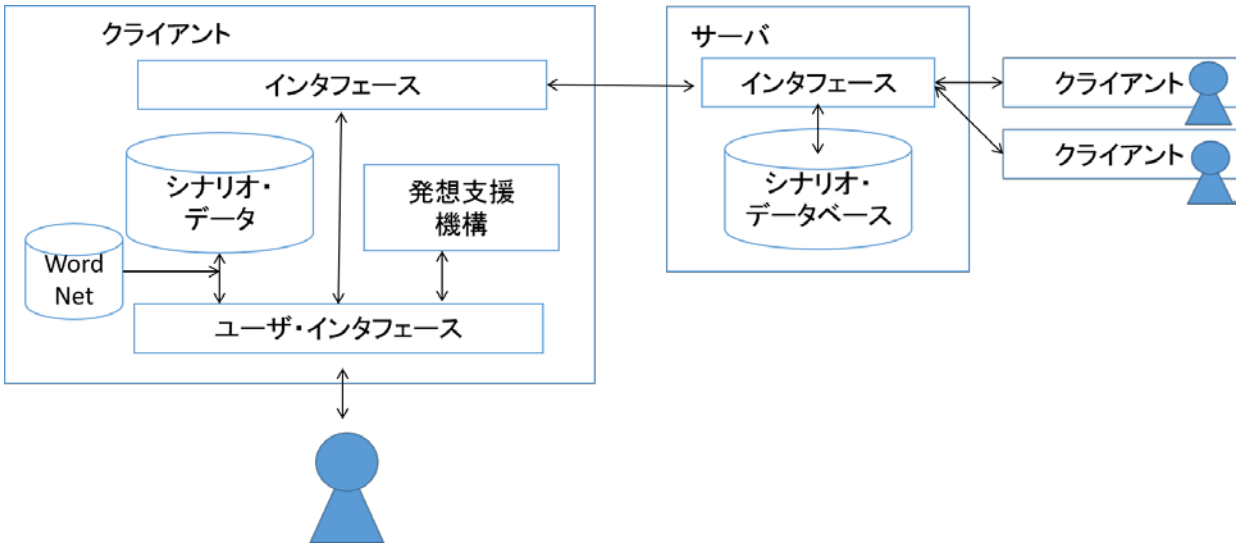

図 1　システム概要（協調シナリオ提示システム）

関連を提示するためには，個々の語に対してメタなタグを付与する必要がある．本研究課題では，WordNet を用いて語に自動的にタグを付与する．また，WordNet の階層関係を利用して類似の概念から構成される語を特定し，自身が行き詰っている語に対し，他者が同じ概念から語を導出していた場合に，導出された語をヒントとして提示することとする．

図 2 に構築した発想支援のためのインタフェースを示す．語をノードで表現し，ある語から導出された語をリンクで接続する概念マップ形式で表現する．本システムは OS X10．10．1 上で動作する統合開発環境 Xcode6．1．1 を用いて Objective-C の iOS アプリケーションとして開発している．WordNet の語彙データは SQLite 形式で情報通信研究機構より提供されている．これを Objective-C で簡易に扱えるようにしたライブラリ WordNetJPN[8]を使用した．図 3 はヒントの提示例を示す．語を表すノードをクリックすると，過去に自身もしくは他者が同じ語から語を導出していた場合に，導出された語が提示されるようになっている．

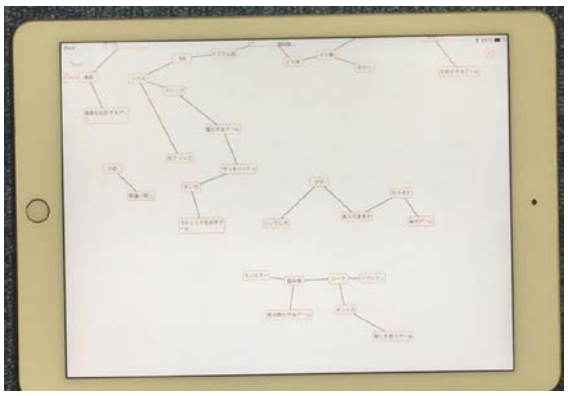

図 2　発想支援機能のシステム・インタフェース

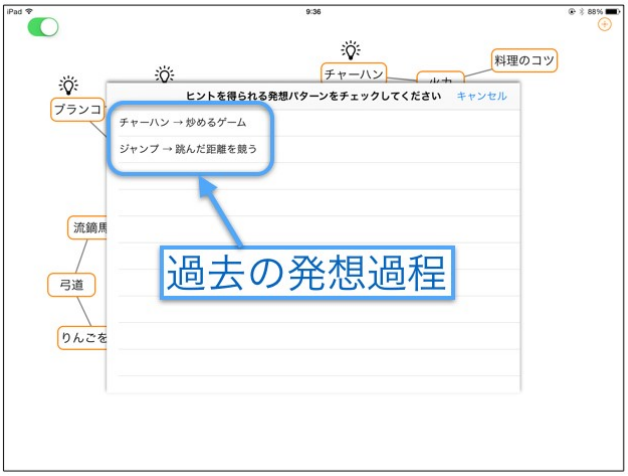

図 3　ヒントの提示例

2.3　評価実験

発想支援機能にのみ焦点をあてた評価実験を行った．被験者は本学の学生 8 名である．実験では，発想過程の提示の有無による語生成数の変化を調査した．発想過程を提示する機能を停止した状態で，本システムを用いて連想的に発想しながら概念マップを構築してもらった.発想してもらう時間は制限時間を 30 分とした．次に，発想過程を提示する機能を有効にし，そこから可能な限り新しい語を生成してもらった．発想過程をある程度提示するためにあらかじめ著者がシステムを使用して同じテーマで語を生成した．最後に本システムのユーザビリティを評価するためにアンケート調査に回答してもらった．発想過程の提示の有無による語の生成数の変化を表 1 に示す．被験者全員が提示された発想過程から新しい語を生成できことがわかった．

表 1 発想支援機能による語の数の変化

被験者	ヒント提示前の語の数	ヒント提示後に追加されたの語の数
A	53	4
B	43	5
C	38	3
D	44	4
E	28	2
F	34	6
G	24	5
H	18	3

次にシステムのユーザビリティを評価するための事後アンケートの結果を述べる．「ノードの移動」は全員が「使いにくかった」と答えた．ノードを移動するにはノードを少しの間ホールドした後，ホールドしたま

まスワイプする必要がある．一方，画面をスクロールする操作は画面をスワイプする操作であるので，ホールドする間隔が短ければスワイプが画面のスクロールと間違えられてしまう．ノードに指が触れるのを検知した時点でスクロールを不可能にしてノードの移動のみ可能にすると，この問題は解決することができる．また，ノードの作成を使いにくかったと答えた被験者が１名いた．これは本システムが新規ノードを作成する際にサーバと通信するという仕様になっており，この被験者の実験時にネットワークの接続が安定していなかったためだと考えられる．上記以外の操作については全員が「使いやすかった」と答えた．よって，軽微な修正は必要であるが，ある程度使用可能なインタフェースであることがわかった．

3　C 言語への拡張

3.1　目的

本章では，経験知学習を他のプログラミング言語に適用する例として，C 言語の文法学習を対象とする．デザインパターンの利点は主に再利用性，拡張性，柔軟性であるが，C 言語の文法の理解を対象とした場合，文法の利点そのものが経験知となる．例えば，for はプログラムの流れを繰り返すことができるし，構造体は複数の異なるデータを一つに扱うことができる．このような文法そのものの意義を C 言語の文法学習では経験知として扱うこととする．

C 言語の文法は，C 言語のアルゴリズムに変化をもたらす．したがって，プログラムの利点を意識できるようにするためには，アルゴリズムが意識できるようになることがのぞましい．フローチャート（流れ図）はプログラムの内容を視覚化して表現したものであり，その構造の違いがわかりやすい．したがって，本研究ではフローチャートを変形させることによって C 言語の文法の利点を学習者に気付かせることとする．小島らが，作問学習において，与えられた問題から解法を改変することの困難さを述べているように[8]，解法を表しているフローチャートを変形することは，学校教育でも行われておらず，困難である．また，たとえフローチャートが修正できたとしても，正しく変形できていなければ文法の利点の理解にはつながらない．本研究では，学習者の作成したフローチャートを判定し，正しいフローチャートが書けるよう助言を生成するシステムを構築する．

C 言語のプログラムの要素には，データ構造と，データを用いて表現されるアルゴリズムがある．データ構造は整数や文字の基本型に加え，配列や構造体など，人が定義する型が存在する．ユーザが定義する型は基本型に対して，複数の値をまとめて扱えるなどの利点が存在する．一方，アルゴリズムは逐次制御に加えて，繰り返しをする for 文や条件分岐をする if 文などが存在する．これらを使用することで，プログラムの構造が簡単化し，制御がわかりやすくなるという利点がある．本研究では，まずデータ構造として配列と構造体，制御として for と関数を扱う．表 2 にこれらの文法の利点を示す．

表 2　C 言語の文法と利点の例

文法	主な利点
For	プログラムの流れを繰り返し，決めた回数分だけ特定の動作を行わせることが可能
関数	プログラムの流れの一部を関数化させることで，１つのブラックボックスとして扱うことが可能
配列	一つの変数によって，他の変数を保存する領域を複数管理させることが可能
構造体	同じ構造を持った変数を手軽に沢山作ることが可能

3.2　学習支援システム

図 4 に構築するシステムの概要を示す．実線は制御の流れ，点線はデータ参照を表している．本システムは，文法を使わないフローチャートと，正誤判定をするための正解データを有する問題データと，学習者の作成したフローチャートの誤りを修正するためのアドバイス知識を有する．また，学習者の作成したフローチャートの正誤を判定するための正誤判定機能を持つ．学習者が学習したい文法を選択すると，用いられていないプログラムのフローチャートが提示される．学習者が，提示されたフローチャートを文法を用いたプログラムのフローチャートに変形して入力すると，正誤判定機能が問題データを参照し，作成されたフローチャートが文法を用いており，かつ等価な動作をするか判定する．適切なプログラムではないと判定された

場合，誤っている箇所についてアドバイス知識を用いて，アドバイスを提示する．

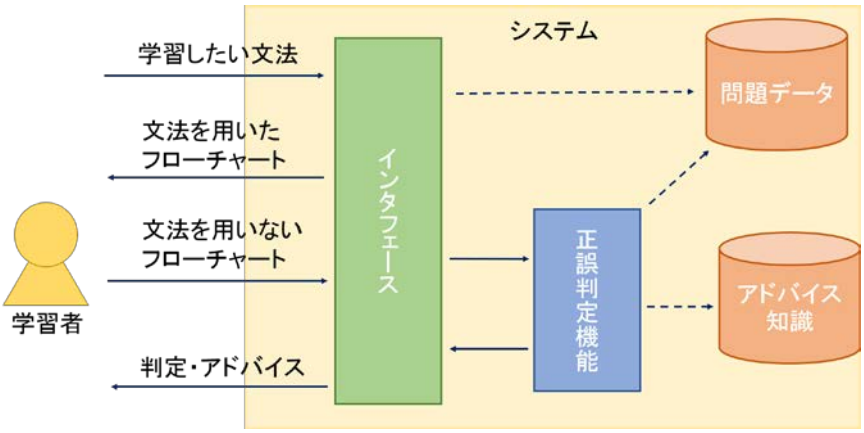

図 4　C 言語を対象とした学習支援システムの概要

図 5 に，配列を学習させたい場合を例とした学習方法を載せる．図 5(a)の配列を使用したフローチャートを学習者に提示し，学習者がそれを見ながら，今度は配列を使用しないフローチャートに直す．例えば，図 5(b)のように配列の要素 1 つずつを 1 つの変数で扱うようなフローチャートを作ると，配列を用いないとステップ数が増えることを実感できる．

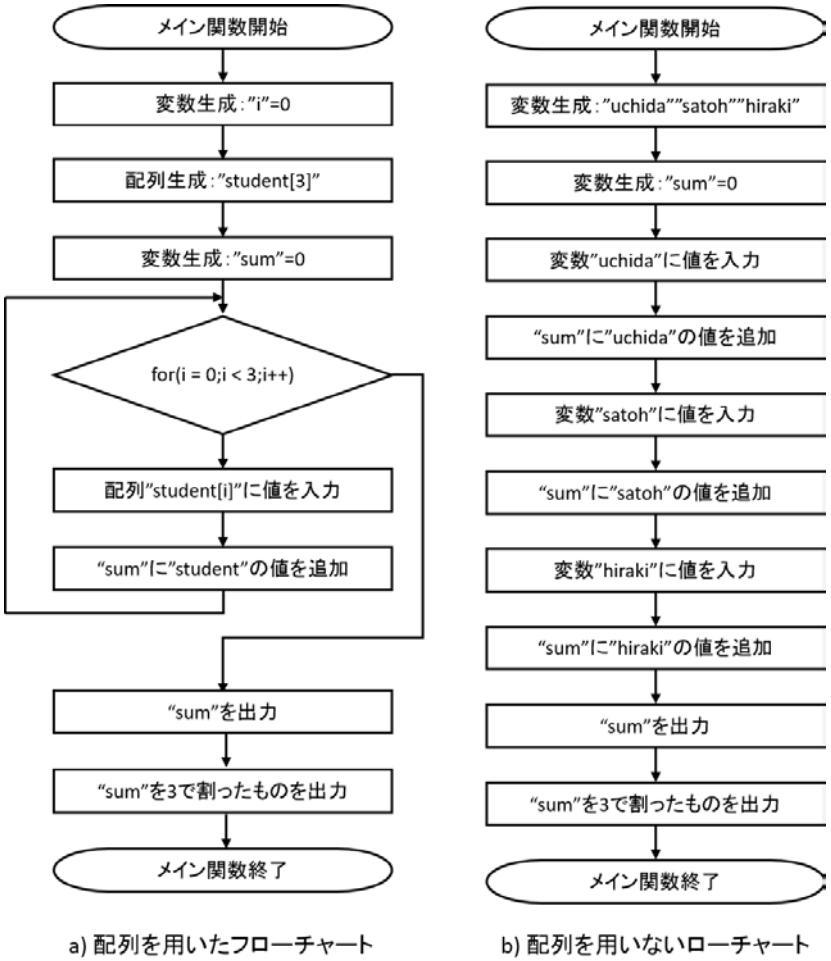

図 5 配列に関するフローチャートの例

本研究では，フローチャートの正誤を判定するために必要な処理の種類をタグとして付与し，学習者の作成したフローチャートの正誤をタグで判定する．表 3 に，用意したタグとその説明を載せる．また，図 6(a)のプログラムをタグで表現したものを図 6(b)に示す．

一方，フローチャートの変形において，たとえ文法を用いないフローチャートが描かれていても，必ずしも全ての学習者が同じフローチャートを描くとは限らない．そこで，文法を使用しないプログラムが成り立

つのに最低限必要なフローチャートの変形を定義し，これらがすべてなされていたら正解とする．表 4 に，配列と関数に必要なフローチャートの操作例を示す．

表 3　システムで用意したタグ

メイン関数開始，サブ関数の定義，メイン関数終了，サブ関数の定義終了，変数を返す，if 文，for 文，int 変数生成，char 変数生成，ポインタ変数生成，int 配列生成，char 配列生成，double 配列生成，変数に代入，配列に代入，ポインタ変数に代入，四則演算，剰余計算，変数に入力，配列に入力，サブ関数に入力，変数の値の出力，構造体のメンバの出力，サブ関数の宣言，サブ関数を呼び出す，構造体の型枠の宣言，構造体の宣言，ループ，分岐結合

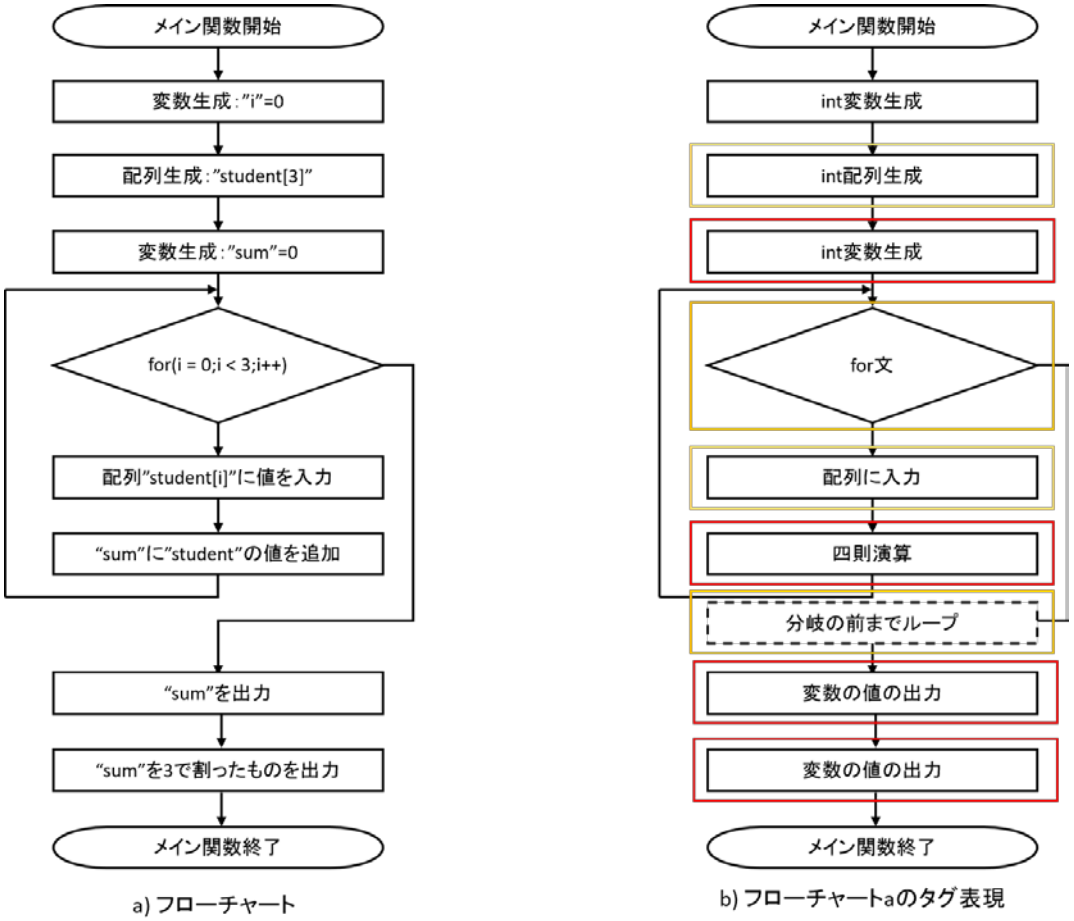

図 6　タグ付与の例

表 4　文法に必要な操作例

文法	操作
配列	削除：int 配列生成，配列に入力配列に入力 追加：int 変数生成，変数に代入
関数	削除：サブ関数の定義，サブ関数を呼び出す

学習者のフローチャートが正解でないと判定された場合，システムは学習者が正解のフローチャートを導出できるようなアドバイスを生成する．アドバイスは，単に足りない要素を指摘するだけでなく，その要素の必要性を考えさせるようなものが望ましい．文法を活用する理由は，文法ごとに異なる．文法の利点は文法ごとに異なるため，そのアドバイスもその文法ごとに合わせたものを用意しなければならない．例えば，配列は複数のある種類の変数を格納できるという利点があるため，特定のデータをばらばらにする方法を考えさせることを促す．関数は Main 関数内の行動をサブ関数に分離させることで，どの部分でどの処理が行われているかをはっきりさせ，同じ処理を何度も書く手間を省くことができるという利点がある．よってサブ関数を Main 関数内に埋め込む方法を考えさせることを促す．

プロトタイプ・システムを C#で実装した．図 6 にインタフェースを示す．「読み込みボタン」を押し，自分の学習したい文法を選択すると，問題フローチャート表示エリアに問題のフローチャートが表示される．学習者は，フローチャート作成ボタンを用いることで，フローチャート描写エリアに文法を用いないフローチャートを描いていく．学習者が「解答判定ボタン」を押すとシステムはフローチャートの正誤を判定する．不正解だった場合には図 8 のようなアドバイスを表示し，正解だった場合には「正解です」という文字を表示する．

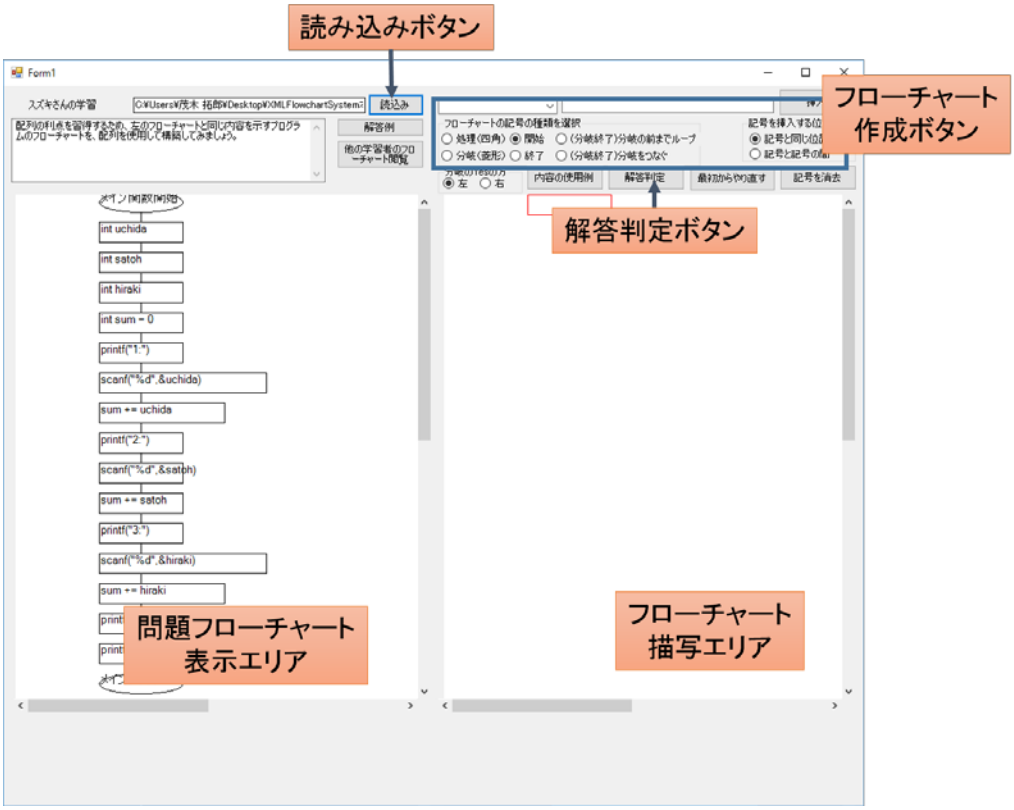

図 7　プロトタイプ・システム　インタフェース

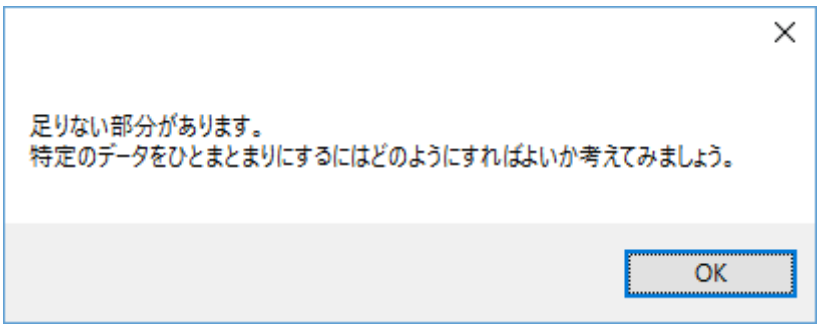

図 8　不正解のときのメッセージ

3.3　評価実験

本研究で構築した C 言語文法理解のためのフローチャート変形学習システムの有効性ならびにシステムの課題についての検証することを目的に評価実験を実施した．被験者は C 言語のプログラミング経験のある大学生 7 名（A～G）である．

初めに，C 言語の文法の理解度を確認するために事前テストを実施した．事前テストとしては学習者にプログラムを与え，そのプログラムをより命令数の少ないものに変形させた．配列を扱うものと，switch 文を扱うものの 2 種類のプログラムを用意した．次に，学習者にシステムを利用してもらった．まず全ての学習者に配列の問題を解かせてから，関数や構造体の問題も解きたい場合には解いてもらうようにした．その結

果，今回は学習者 A のみが追加で関数と構造体の問題を解いた．システムの操作後，文法の理解度の変化を確認するため，事後テストを実施した．事後テストでは，事前テストと同様に与えたプログラムを変形してもらった．配列を扱うものと switch 文を扱うものとなっているが，事前テストより難易度の高い問題となっている．なお，今回この 2 種類を用意した理由は，配列の問題をシステムで学習させることで，同じ文法の配列の問題，異なる文法の問題の理解度に与える影響を確かめるためである．後に，事後アンケートに回答してもらい，本システムに関する意見や感想，提案に関する事項を記述してもらった．

表 5 に事前・事後テストの正答をまとめたものを載せる．配列の問題では，7 人中 3 人は事前問題では解くことができなかったが，事後問題では解くことができるようになっていた．また，switch 文の問題では，簡単な方の事前問題を解けている被験者でも，事前問題に比べ難しい事後問題の場合は不正解になっているパターンが多かった．この結果より，配列では事前テストでは上手く配列を使えなかった 3 人が，システムによる学習を行った後は配列を正しく使用できるようになっていたと判断できる．なお，学習者 A は配列問題の事前問題は正解だったものの事後問題では不正解だった．学習者 A は，事後問題においては配列に関しては正解だったものの，ポインタ文も使用しており，その変形が誤っていた．したがって，学習対象となっていた配列は理解できていたといえる．

次に表 6 に事後アンケートの結果を載せる．質問 1 より，7 人中 6 人がシステムによって文法を正しく使用したプログラムのフローチャート構築することができたと回答した．質問 2 より，そのうち 3 人が今回の学習によって利点を理解できたと答えた．質問 3 よりアドバイスが適切であったと回答した．その 3 人はアドバイスを受けることでテストでも正しく配列を使用することができていた．アドバイスが適切でないと回答した被験者のうち，用意されたアドバイスが，ユーザの視点にたったものでないというものがあった．質問 4 より，4 人がタグを思ったように選択することができなかったと回答した．その理由となる意見として，「変数に入力」や「配列に入力」などの「入力」がどういう意味なのかわかりづらかったというものがあった．また，「ループ」と「分岐結合」の意味の違いが判らず間違って描いていた被験者もいた．今回，記号の一つ一つにタグをつけるという手法を取ったが，一つではなく複数のタグが付けられる可能性があるのではないかという意見があった．具体的には，「int i = 0」の場合は，「int 変数生成 (int i)」と「変数に代入 (i = 0)」の二種類のタグをつけることができる，というものである．このことから，2 種類以上のタグを付けられるようにシステムを改良する必要がある．

システムの操作性を評価する．質問 5 では，7 人中 4 人が「操作しづらかった」と答えた．具体的な意見としては，元に戻す機能があればもっと使いやすくなると思った」というものがあった．現在のシステムでも，描画を行った記号の消去は可能だが，マウスでその記号を選択して消去しなければならないため手間がかかり，間違えて違う記号を消してしまうことがあった．削除機能を構築する必要がある．

表 5　事前・事後テストの解答

	事前テスト		事後テスト	
被験者	配列	switch	配列	switch
A	○	○	×	×
B	×	○	○	×
C	○	○	○	○
D	×	×	○	×
E	○	×	○	○
F	×	○	○	×
G	○	○	○	×

表 6　事後アンケート結果

質問項目	選択肢	回答数
1. フローチャートの作成をこなすことができましたか.	できた できなかった	6 名 1 名
2. 文法の利点に気付くことができましたか. (質問 1 で「できた」と回答した人のみ)	できた まぁまぁできた あまりできなかった できなかった	0 名 3 名 1 名 2 名
3. システムによって伝えられたアドバイ	できた	1 名

スは適切であると感じましたか.	まぁまぁできた あまりできなかった できなかった	3名 1名 2名
4. フローチャートの要素を選択する際, 自身の思ったような要素を選択することができましたか.	できた まぁまぁできた あまりできなかった できなかった	1名 2名 3名 1名
5. システムは操作しやすかったと感じましたか.	操作しやすかった まぁまぁ操作しやすかった 操作しづらかった	0名 3名 4名

4. まとめと今後の課題

本研究課題では，デザインパターンを対象としたプログラミング学習において，他者とシナリオを交換できる機能，および新しいシナリオを発想できるようになるための支援機能を構築した．発想支援機能を対象とした実験は行っているが，シナリオを交換することに関する評価実験は実施できていない．今後は，すべての機能を含んだ評価実験をする必要がある．

また，本研究課題では提案した経験知学習の手法をC言語の文法学習に適用するとともに，その支援システムを開発した．評価実験の結果，本システムはデザインパターンだけでなく，C 言語の文法のように，他の経験知の学習にも有効であることが明らかになった，しかし，システムを用いて学習していない文法に関しては，優位な変化はみられなかった．このことは，システムを用いることで経験知の理解は進むが，学習手法そのものの習得はなされていないことを示唆している.システムにより,学習手法自体を習得されると，システムで学習していない経験知も学習者自身で理解できるようになる．今後は，学習手法を教授するような手法も考えていきたい．また，今後は建築手法や運動などの身体動作など，提案した学習方法を他の形式の経験知に適用するための手法を明らかにするとともに，他の経験知を支援するための技術を提案していきたい.

参考文献

1. アラン・シャロウエィ，ジェームズ・R・トロット著，村上雅章訳：「デザインパターンとともに学ぶ オブジェクト指向のこころ」，ピアソン・エデュケーション（2005）
2. マイケル・ポランニー：「暗黙知の次元」，筑摩書房（2003）
3. Stephen WEISS, "Teaching Design Patterns by Stealth", Proc. of SIGCSE 2005, pp.492-494（2005）
4. Neishia PILLAY, "Teaching Design Patterns", Proc. of SACLA（2010）
5. 大江洋希，小尻智子，瀬田和久：「別解作成に基づいたデザインパターン学習における学習者の別解特定機構の構築」，情報処理学会第76回全国大会講演論文集，pp.4-551－4-552（2014）
6. 小尻智子，大江洋希，瀬田和久：「代替設計との拡張性の比較に基づいたデザインパターン設計意図理解支援システムとその評価」，電子情報通信学会技術研究報告，Vol. 114, No.53, pp.31-36（2014）
7. 小島一晃，三輪和久，松居辰則: 産出課題としての作問学習支援のための実験的検討，教育システム情報学会誌，Vol. 27, No.4, pp. 302-315（2010）
8. 大森智史：Objective-C API. http://cocoaapi.hatenablog.com/entry/20090401/p1,（参照日2015.01.24）