JSiSE Research Report
vol.31,no.7(2017-3)

プログラミングにおける構造的理解のための部品の段階的拡張手法の提案とそのシステムの開発

古池 謙人[*1], 東本 崇仁[*1]
*1 東京工芸大学工学部

Proposal of Granularity Expand Method in Parts for Structural Understanding on Programming and Development Leaning Support System

Kento KOIKE [*1], Takahito TOMOTO[*1]
*1 Faculty of Engineering, Tokyo Polytechnic University

本研究では，プログラミング学習において知識の拡充だけでなく，拡充した知識の関係性を理解する構造的理解の重要性を主張した. この構造的理解を深めることで, コードの有意な塊ごとに部品化を行い，部品の再利用性を高めるといった構造的な設計が行えると考えた．よって，構造的理解の支援を行うために,「部品の段階的拡張手法」の提案を行った.「部品の段階的拡張手法」では，プログラムにおける一行ずつの処理を組み合わせて部品の構築を行い，さらに構築した部品に対して新たな処理や既存の部品の追加･修正を行うことで部品の構造を変化させ, 段階的に拡張していくことが有用であると考えた. 加えて，提案手法を用いたシステムの開発を行った．本システムの評価実験では，プログラムの構造的理解に有用であることが示唆され，アンケート結果より，学習者が本システムを肯定的に受け入れていることが分かった.

キーワード: 部品の段階的拡張手法，段階的な思考，構造的理解，学習支援システム

1. はじめに

一般的にプログラミング学習では，学習者が書籍やウェブ上に掲載されている解説を読んだ後，サンプルコードを実行し，その後に自由に調整した結果から学習者自身がそのサンプルコードの意味や仕組みについて考えることで，動作の理解や経験を得ることができる．また，こういったプロセスは，プログラミングのみならず多くの学習においても有意義である. しかし，現実には最初にサンプルコードを読む段階までで学習が止まってしまう場合や，サンプルコードの切り貼りによってプログラムを作成するだけにとどまることがある．サンプルコードの切り貼りだけでプログラムを作成する場合には，学習者自身がサンプルコードの意味や仕組みについて思考する機会が少なく，理解が不足する．同様にプログラミング教育の場でも，教師がサンプルコードの提示を行い，学習者が提示されたコードを入力して動かすという行程を課題形式で行うことが多い．教師による課題形式で学習者が学習を行う場合には，提示されたサンプルコードについての説明を教師が一方的に行うものの，学習者は単純に課題をこなすためだけに学習を行うため，学習者自身が能動的にサンプルコードの意味や仕組みについて考える機会が不足する場合が多く存在する．結果として，学習者はこれまでに学習した if 文や for 文などの基礎構文や，アルゴリズムなどに対してコードの各行に対する具体的な動作，コードにおける入力と出力の関係がわからないなど十分な理解ができないことで，知識としての習得が不十分となる．独学や教育の場では，知識

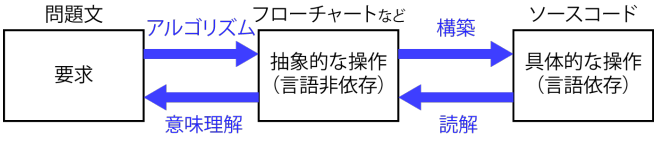

図 1　金森ら[7]によるプログラミングプロセス

の習得が十分に支援されない場合が多いため，学習者が十分な知識を獲得するための支援は多く行われており，効果も検証されている[1]–[3]．また，アルゴリズムの理解に焦点を当てた研究[4]–[6]も多く行われている．

しかし，個々の知識やアルゴリズムの理解が十分な学習者でも，実際の要求からおおよその流れを加味した手続き的な設計を行うことはできるものの，コードの有意な塊ごとに部品化を行い，部品の再利用性を高めるといった構造的な設計は苦手とする場合が多い．加えて，学習者にとってみれば，コードの単一行は理解できていても，複数行になると理解が困難になることがある．なぜなら，学習者が複数行のコードを有意な塊ごとに部品として認識することができておらず，個々の部品の関係性や特徴を理解することが十分に行えていないためである．また，コードを有意な塊ごとに部品化を行い，部品の再利用性を高めるといった構造的な設計を行う能力は，システムエンジニア職や，近年主流となっている Java や C#などオブジェクト指向プログラミングには重要である．しかし，一般的にシステムエンジニア職が行う機能設計では，大きな目的を細かい目的に細分化することで行い，詳細なコードまで知らなくても設計が行えると考えられている．

著者らはこれまでのプログラミング学習における経験から，色々なコードの有意な塊を部品として習得し，より大きな部品を構築するというプロセスを繰り返しており，習得した部品の中から必要な部品を選んでより大きな部品を構築するというプロセスは，個々の部品の関係性や特徴を理解していないと行うことができない．機能設計も同様に個々の部品の関係性や特徴を理解することで，システムのようなより大きな部品の設計を構造的に行うことができるのではないかと著者らは考える．

著者らは，拡充した知識からコードの有意な塊ごとに部品として認識を行い，個々の部品の関係性や特徴を理解し，より大きな部品の構築を構造的に行えるように知識を整理することを「構造的理解」と位置付け，この構造的理解が要求から構造的な設計を行うために重要であると考えた．

そこで本研究では，プログラミングにおける構造的理解のための「部品の段階的拡張手法」と，提案手法を用いることによる学習支援システムの開発，その評価を行った．

2.　関連研究

知識の拡充によってコードの有意な塊を習得するだけでなく，習得した塊のそれぞれの関係性や特徴を正しく理解することの重要性が金森ら[7]や Arai et al.[8]によって主張されている．

構造的理解を行うためのアプローチとして，他者のソースコードを読み，その構造を理解することや，他者のソースコードを真似て自分で作ってみるといった手段の後に，自らソースコードに対して自由な調整を行うことでより個々の部品への理解を深められるのではないかと著者らは考える．学習者が知識によって各行を理解することはできるものの，複数行になった際に理解が困難になることから，学習者がコードの各行についての関係性を正しく理解することを目的とした研究が行われている[9], [10]．渡辺らの研究[10]では，金森らが提案するプログラミングプロセス（図 1）において，プログラムの読解と意味理解を行うことが学習において有意義であるとし，その手段として段階的抽象化プロセスを提案している．段階的抽象化プロセスは，新開ら[11]の目的からプログラムを徐々に細分化していくというアルゴリズム学習の支援である段階的詳細化プロセスを参考に提案されており，段階的詳細化の有用性は木村らの研究[12]で示唆されている．段階的抽象化プロセスの手順は，与えられたコードの中で一連の操作だと考えられる部分をまとめ，その意味を考えることを繰り返すことで，“段階的に読む”学習を支援するプロセスである．例えば，(1)c に a を代入，(2)a に b を代入，(3)b に c を代入という 3 行程からなるプログラムでも，行ごとに見れば単なる代入の繰り返しであるが，3 行程をまとめて意味を考えることで，スワップという新しい概念を発見できるとしている．著者らは，この渡辺らの段階的抽象化によって，コードの各行の関係性を正しく理解することができ，有意な

塊としての認識が行えると考える．

しかし，新開らによる段階的詳細プロセスや，渡辺らによる段階的抽象化プロセスでは，プログラムを構成している部品について着目しているものの，部品同士を組み合わせてより大きな部品を作るといった，部品の発展については扱っていない．そこで著者らは，コードの有意な塊ごとに部品として認識するのと同時に，その部品を段階的に発展させ組み合わせることによって，どうすれば部品を再利用できるかなどを考える“段階的に作る”学習に焦点を当てることで，より塊ごとの構造的理解につながるのではないかと考えた．

3. 提案手法

著者らは，プログラミング学習において，知識の拡充だけでなく，知識の関係性を正しく理解する構造的理解が重要だと考え，この構造的理解を支援するための手法を提案する．提案手法は，渡辺らの段階的抽象化プロセスを用いて，まず，学習者がプログラムにおける 1 行ずつのコードから有意な塊ごとに部品の構築を行う．さらに構築した部品に対して新たな処理や既存の部品の追加をしたり，すでに構築された部品における部分的な修正を行ったりして部品を変化させる．こうした部品の変化を繰り返すことで部品の構造を段階的に拡張していく（図 2）．そうすることで，学習者に部品ごとに対する関係性の理解を促し，学習することができる．また，部品を段階的に拡張する行為そのものによって，自身で大きな部品を作る訓練にも繋がり，実際に設計を行う際に役立つと考えられる．この過程を，著者らは「部品の段階的拡張手法」と定義した．

通常，熟達者はプログラミングの際に他人のプログラムを部品ごとに構造的に読む行為と，自ら部品ごとに再利用性などを考慮しながら構造的にプログラムを設計する行為を日常的に行っている．熟達者は他人のプログラムの構造を理解し自らの知識に部品として加えることで，加わった部品を用いた新たな設計を行うことが可能となる．

段階的に読む学習の有用性については金森ら[7]によって主張されているが，加えて，従来の学習行程である作る学習で段階的にプログラムを発展できれば，設計を行う能力だけでなく，アルゴリズムの構造的な理解に繋がり，プログラミング学習としても有用であると著者らは考える．

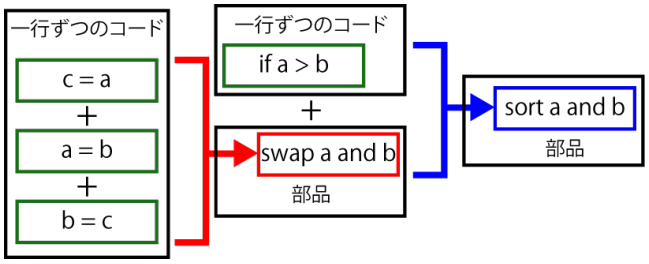

図 2　提案する「部品の段階的拡張手法」

4. 提案システム

本研究では，上述の「部品の段階的拡張手法」を用いた学習支援システム（以下，本システム）の提案を行う．本システムでは，各部品をブロックと位置づけ，従来のプログラミングにおける一行ずつの処理をスタンダードブロックとし，スタンダードブロックに対して新たな処理や既存の部品を追加して拡張した部品をアドバンスブロックと定義した．「部品の段階的拡張手法」の活用として，構築すべきアドバンスブロックを目標としてあらかじめ設定し，ブロックの組み合わせや発想が容易なものから順に習得を目指すことで，学習者におけるプログラムの構造的理解を目的とした，プログラミング学習の支援を図る．

4.1 ブロック

本項では，スタンダードブロックとアドバンスブロックからなる 2 種類のブロックの特徴を説明する．

4.1.1 スタンダードブロック

スタンダードブロックは，従来のプログラミングにおける各行に相当するものとし，「代入」や，「条件」などの単一な処理を行うものと定義した．

4.1.2 アドバンスブロック

アドバンスブロックは，従来のプログラミングにおける関数やクラスに相当するものとし，スタンダードブロックの組み合わせによる複合的な処理，すなわちアルゴリズム化された一意な部品と定義した．組み合わせの手段は三種類あり，まず，(1) スタンダードブロックのみの組み合わせからなるアルゴリズムの定義，

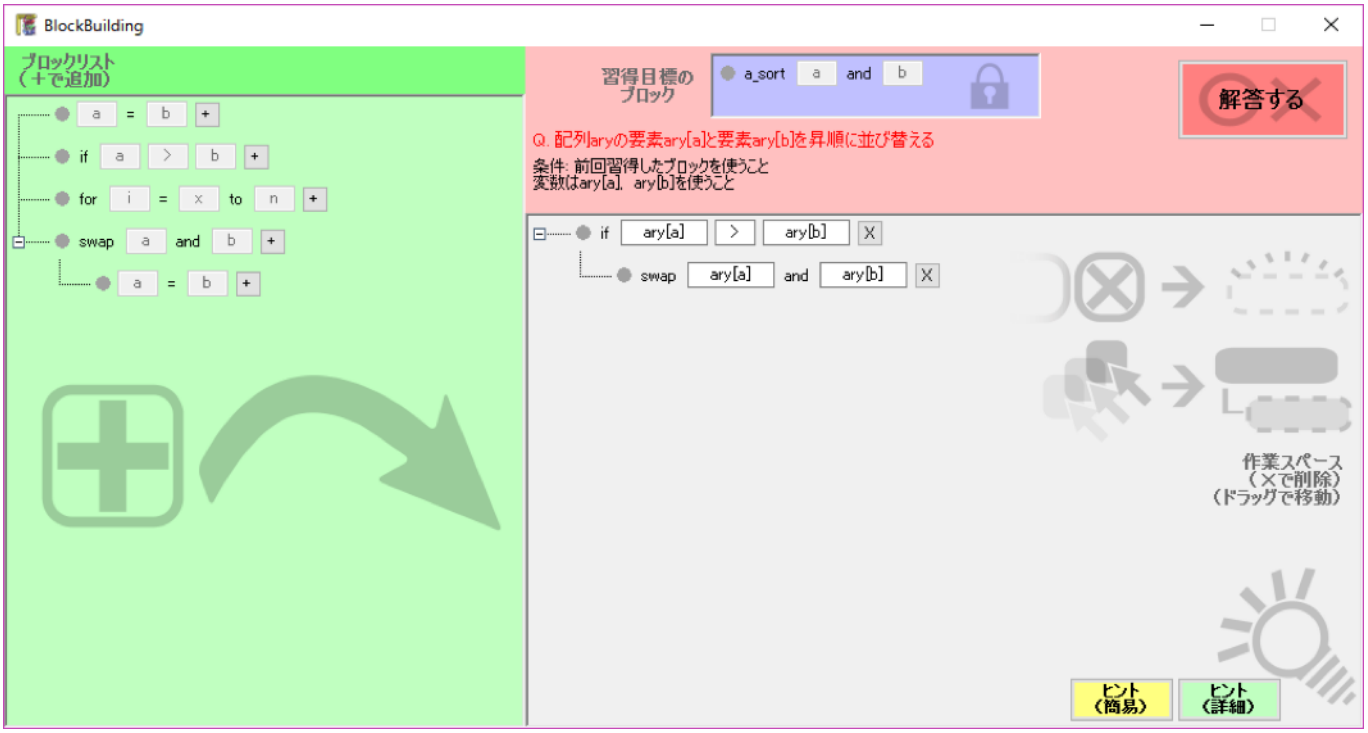

図 3　本システムのインタフェイス

次に，(2) スタンダードブロックとアドバンスブロックの組み合わせからなるアルゴリズムの拡張，加えて，(3) アドバンスブロック同士の組み合わせからなるアルゴリズムの統合からなる．本システムでは，この三種類の組み立て方法を提供し，学習者はその中から選択，構築することで設計能力の向上を目指す．

4.2　インタフェイスと学習方法

本システムのインタフェイスを図 3 に示す．本システムのインタフェイスは，習得目標であるアドバンスブロックを習得するための習得画面のみからなる．まず，左側には，学習者が構築に利用できるブロックがブロックリストとして提示される．提示されるブロックは，用意されたスタンダードブロックの全てと，今までに学習したアドバンスブロックである．スタンダードブロックはそのまま一行で表示されるが，アドバンスブロックは階層構造を持って表示され，子ノードには包含するブロックが表示される．これらのブロックはカーソルを合わせることでそのブロックの動作が表示されるようになっている．右部ではそれらのブロックを用いた構築領域が提示され，ブロックリストから追加したブロックはこの構築領域に表示される．中央上部には習得するアドバンスブロックが提示され，完成した場合には，右上部の完成ボタンを押すことで正解であれば次の習得画面に移行し，不正解である場

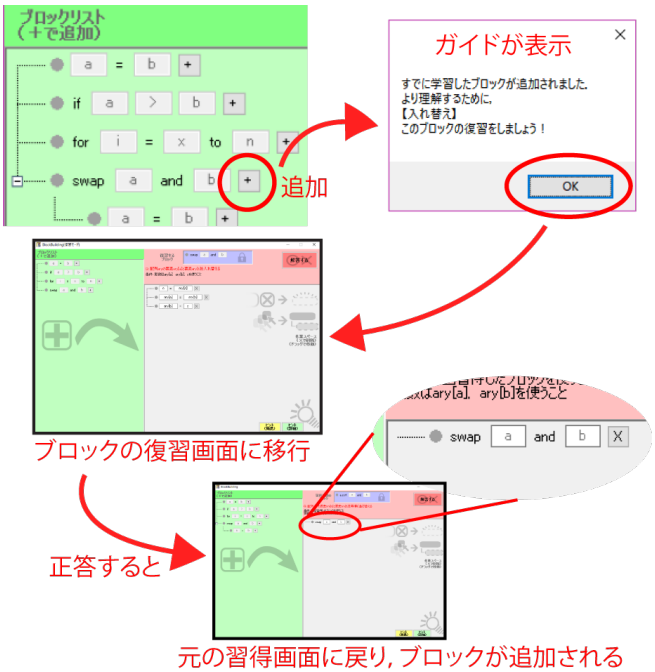

図 4　復習画面への移行

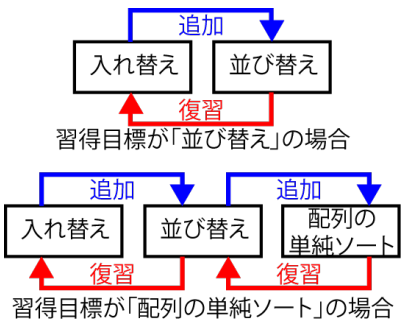

図 5　復習の移行図

合には不正解と表示される．また，右下部の簡易ヒントボタンか詳細ヒントボタンを押すことでフィードバックが提示される．

また，本システムではブロックの復習を導入している（図 4,5）．具体的には，ブロックリストからアドバンスブロックの追加が行われた際に，追加されたアドバンスブロックの習得画面へ移行する．そこで，既に学習したアドバンスブロックを同様に構築することで，学習者の理解を深められると考える．追加されたアドバンスブロックの習得が終わった際には，元の習得画面に戻る．元の習得画面で一度そのブロックの復習を行った場合は，同一のアドバンスブロックの習得画面の間は再度復習画面が出ないようになっている．

4.3 段階的な発展

本システムでは，段階的な発展を行うために問題を系列的に出題し，必ず前後の問題が関連するように設定している．例えば，スワップの習得（図 6）を行った後に，そのスワップを用いて 2 変数のソートを習得する（図 7）．また，その後には 2 変数のソートを用いて，単純ソートの習得を行う（図 8）．このように，本システムでは，問題を段階的に発展させている．そうすることで，学習者への構造的理解を促すことを志向した．

4.4 フィードバック

フィードバックの提示は，上述の簡易ヒントボタン，詳細ヒントボタンにて行われる．両ボタンともに，正答のブロック構成と学習者のブロック構成の間で比較を行い，その差異をエラーとして提示する．エラーの優先順位は上から，ブロックの数，階層の高さ，ブロックの種類，ブロック内における値の順番となっている．簡易ヒントボタンでは，最も上流にあたるエラーに対して「～が間違っています」というフィードバックを行う．例えば，ブロックの数と階層の高さが合っており，ブロックの種類が違った場合には「上から n 番目のブロックの種類が違います」というフィードバックを行う．詳細ヒントボタンでは，最も上流にあたるエラーに対して，その正答を一つフィードバックとして提示を行う．例えば，ブロックの値が間違っていた場合は「上から n 番目のブロックの m 個目の値は x です」といったフィードバックを行う．

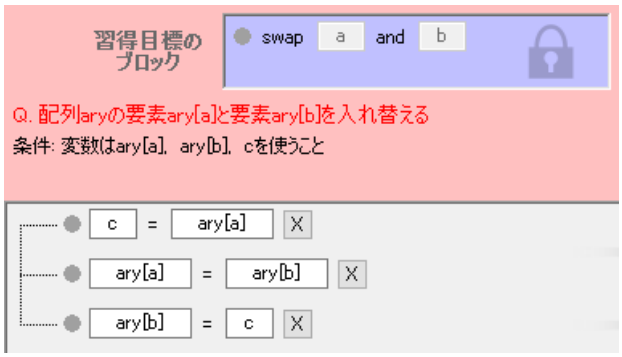

図 6　スワップの習得画面（一部）

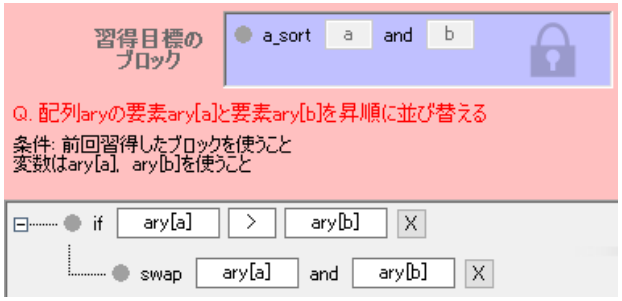

図 7　2 変数ソートの習得画面（一部）

図 8　単純ソートの習得画面（一部）

本システムでは簡易ヒントボタンと詳細ヒントボタンによって，学習者自身が自律的に修正できるフィードバックを提供しており，このようなフィードバックを学習者が通常の紙媒体による学習で得るためには，間違えた箇所を自ら正答例との差異を見比べることで確認しなければならず，誤答の度に正答例との差異を確認することは，学習者が能動的に学習を行う際に妨げとなる．また，学習者は紙媒体の学習において正答例と自分の解答を比較する際に，そもそも誤答に気づくことすら行えない場合がある．本システムでは，学習者自身の要求するレベルでフィードバックを得られ，その場で修正を行い正答に至るまでフィードバックを得ながらアプローチを繰り返すことができ，このようなシステムは通常の学習と比べて学習者の意欲を妨げることは少ない．

5. 評価実験

システム利用を通して，学習者の設計能力の向上に寄与したかを検討するために，評価実験を行った．

5.1 実験概要

被験者はプログラミングの講義を受講したことのある大学生 17 名である．事前に，基礎的なプログラミング能力や設計能力を測るための系列的に発展が可能な問題 7 問を記載した 15 分間の事前テストを行った．また，事前テストではなるべく各問で関数化など構造化を行い，他の問題で再利用するよう指示を与えた．これらの評価は，手順が正しいかを評価した素点と，関数化を行い他の問題で再利用できているかを評価した構造化点の二種類の評価を行った．その結果から，素点，構造化点がなるべく均等になるように，システムを利用して学習する実験群（9 名）と，紙媒体で同じ問題を学習する統制群（8 名）に振り分けた．本実験では，まず被験者は 30 分間システム又は紙媒体の学習教材を用いて学習を行った．なお，実験群はあらかじめシステム利用方法について 2 分間程度の事前説明を受けた．その後，事前テストと同一の内容，評価で 15 分間事後テストを行った．事後テストの終了後には，学習方法や教材についての 4 段階評価を用いたアンケートを実施した．

統制群の教材は，出題範囲が学べる一般的な教科書教材の約 2 ページ分を教材として用いた．実験群，統制群ともに教材で答えを学ぶことが可能であったが，統制群は事後テストまで自己の誤りに対するフィードバックを得ることはできなかった．

5.2 テスト結果

表 1 に素点評価における事前・事後テスト，得点の伸びの平均と標準偏差の値を示す．表 1 から，事前テストでは統制群より点数の低かった実験群が事後テストにおいて統制群より点数が高くなっており，実験群の得点の伸びが統制群より大きいことがわかる．

表 2 に素点評価における事前・事後テストの結果に分散分析を行った結果を示す．表 2 から，個人内では 0.1%で有意な差が見られた．

表 3 に構造化点評価における事前・事後テスト，得点の伸びの平均と標準偏差の値を示す．表 3 から，事前テストでは統制群より実験群の点数が高いものの，事後テストにおいて実験群の平均値が統制群より高く，得点の伸びが統制群より大きいことがわかる．

表 4 に構造化点評価における事前・事後テストの結果に分散分析を行った結果を示す．表 4 から，個人間では 5%で，個人内では 0.1%で有意な差が見られた．また，交互作用に 1%で有意な差が見られた．そこで，単純主効果の表を表 5 に示す．表 5 から，個人間の事後テストの結果に 0.1%で有意な差が見られた．よって，事前・事後テストの結果から，実験群において構造化について有意な結果が得られたため，システムがプログラムの構造的理解について有用であるといえる．また，個人内の実験群に 0.1%で有意な差が見られた．よって，表 3 における得点の伸びより，実験群のほうが統制群より点数が向上したといえる．

表 1　素点評価におけるテスト結果

	事前テスト		事後テスト		得点の伸び	
	平均値	S.D.	平均値	S.D.	平均値	S.D.
実験群	1.44	1.51	4.67	1.80	3.22	1.48
統制群	1.50	2.07	3.88	2.95	2.38	1.85

表 2　素点評価における分散分析表

変動要因	SS	df	MS	F	p
個人間	1.15	1	1.15	0.15	0.7035
誤差	114.38	15	7.63		
個人内	66.34	1	66.34	48.04	0.0000
交互作用	1.52	1	1.52	1.10	0.3107
反復誤差	20.72	15	1.38		

表 3　構造化点評価におけるテスト結果

	事前テスト		事後テスト		得点の伸び	
	平均値	S.D.	平均値	S.D.	平均値	S.D.
実験群	0.33	1.00	3.22	2.05	2.89	1.96
統制群	0.25	0.71	0.63	0.92	0.38	0.74

表 4　構造化点評価における分散分析表

変動要因	SS	df	MS	F	p
個人間	15.22	1	15.22	6.80	0.0198
誤差	33.55	15	2.24		
個人内	22.56	1	22.56	19.47	0.0005
交互作用	13.38	1	13.38	11.55	0.0040
反復誤差	17.38	15	1.16		

表 5　構造化点評価における単純主効果の表

変動要因	SS	df	MS	F	p
個人間(事前)	0.03	1	0.03	0.02	0.8962
個人間(事後)	28.57	1	28.57	16.83	0.0003
誤差		30	1.70		
個人内(実験)	35.35	1	35.35	30.50	0.0001
個人内(統制)	0.60	1	0.60	0.51	0.4844
誤差		15	1.16		

5.3 アンケート結果

4 段階評価 (4 : 非常にそう思う～1 : 全く思わない) を用いたアンケートの結果を表 6,7,8 に示した．表 6 では各質問項目が重要だと思うかについてのアンケートを行っている．この結果では，各質問項目平均 3.5 以上と実験群，統制群共に各能力について重要だと感じていると考えられる．また，表 7,8 では，実験群，統制群に対して使用した学習教材について，各能力につながるかについてのアンケートを行っている．

表 6　両群に対する
能力に対する重要さのアンケート項目

質問項目	平均	S.D.
プログラムの塊を部品として認識する能力	3.64	0.49
プログラムごとの関係性を理解する能力	3.83	0.39
プログラムごとの再利用性を高める能力	3.76	0.44
構造的にプログラムを理解する能力	3.83	0.39

この結果より，実験群と統制群の間には，各項目に 0.50 以上の差がみられている．このことから実験群のほうが，学習教材が構造化に繋がるように感じていたと考えられる．

表 7　実験群に対する
本システムについてのアンケート項目

質問項目	平均	S.D.
プログラミングの理解につながるか	3.00	0.71
プログラムの塊を部品として認識することにつながるか	2.89	0.60
プログラムごとの関係性を理解することにつながるか	2.89	0.33
プログラムごとの再利用性を高めるプログラミングにつながるか	2.89	0.60
構造的にプログラムを理解することにつながるか	3.00	0.50

表 8　統制群に対する
紙媒体教材についてのアンケート項目

質問項目	平均	S.D.
プログラミングの理解につながるか	2.25	0.89
プログラムの塊を部品として認識することにつながるか	2.13	1.25
プログラムごとの関係性を理解することにつながるか	1.63	1.06
プログラムごとの再利用性を高めるプログラミングにつながるか	2.13	1.13
構造的にプログラムを理解することにつながるか	2.50	0.93

6. おわりに

本研究では，プログラミング学習において，知識の拡充だけでなく，拡充した知識の関係性を理解する構造的理解の重要性を主張した．また，この構造的理解を深めることで，コードの有意な塊ごとに部品化を行い，部品の再利用性を高めるといった構造的な設計が行えると考えた．よって，構造的理解の支援を行うために，先行研究の段階的抽象化を用いた「部品の段階

的拡張手法」の提案を行った.「部品の段階的拡張手法」では，プログラムにおける一行ずつの処理を組み合わせて部品の構築を行い，さらに構築した部品に対して新たな処理や既存の部品の追加・修正を行うことで部品の構造を変化させ，段階的に拡張していくことが有用であると考えた．加えて，提案手法を用いたシステムの開発を行った．本システムの評価実験では，プログラムの構造的理解に有用であることが示唆され，アンケート結果より，学習者が本システムを肯定的に受け入れていることが分かった.

今回は，提案手法において部品の組み合わせと拡張に着目してシステム開発を行ったため，修正による部品の変化はシステムに適応されなかった．また，問題系列にもより発展的な問題系列のために検討の余地があるといえる.

そのため,今後は部品の修正をシステムに取り入れ,問題系列の発展や，複数の問題系列の分岐・統合などについても検討していきたい.

謝辞

本研究の一部は科研費・基盤研究（C）（15K00492），科研費・基盤研究（B）（K26280127）の助成による.

参 考 文 献

(1) 兼宗進, 中谷多哉子, 御手洗理英, 福井眞吾, 久野靖：“教育用プログラミング言語におけるオブジェクト共有機能の導入,” *情報処理学会論文誌プログラミング（PRO）*, vol. 45, no. 5, pp. 81–81, (2004).

(2) 江木 鶴子, 竹内 章：“プログラミング初心者にトレースを指導するデバッグ支援システムの開発と評価,” 日本教育工学会論文誌, vol. 32, no. 4, pp. 369–381, (2009).

(3) 石川裕季子, 松澤芳昭, 酒井三四郎：“オブジェクト指向言語におけるポリモーフィズムの概念を理解するためのワークベンチの試作,” 教育システム情報学会誌, vol. 31, no. 2, pp. 208–213, (2014).

(4) 杉浦学, 松澤芳昭, 岡田健, 大岩元：“アルゴリズム構築能力育成の導入教育: 実作業による概念理解に基づくアルゴリズム構築体験とその効果,” 情報処理学会論文誌, vol. 49, no. 10, pp. 3409–3427, (2008).

(5) 松澤 芳昭, 保井 元, 杉浦 学, 酒井 三四郎：“ビジュアル-Java相互変換によるシームレスな言語移行を指向したプログラミング学習環境の提案と評価,” 情報処理学会論文誌, vol. 55, no. 1, pp. 57–71, (2014).

(6) 古宮 誠一, 今泉 俊幸, 橋浦 弘明, 松浦 佐江子：“プログラミング学習支援環境AZUR—ブロック構造と関数動作の可視化による支援—,” *研究報告ソフトウェア工学（SE）*, vol. 2014, no. 5, pp. 1–8, (2014).

(7) 金森春樹, 東本崇仁, 米谷雄介, 赤倉貴子：“プログラミングプロセスにおける 「プログラムを読む学習」 の提案及び 「意味理解」 プロセスの学習支援システムの開発,” *電子情報通信学会論文誌 D*, vol. 97, no. 12, pp. 1843–1846, (2014).

(8) T. Arai, H. Kanamori, T. Tomoto, Y. Kometani, T. Akakura:“Development of a learning support system for source code reading comprehension,” in *International Conference on Human Interface and the Management of Information*, 2014, pp. 12–19.

(9) 東本崇仁, 赤倉貴子：“プログラムトレース課題の提案と学習支援システムの開発,” *電子情報通信学会論文誌 D*, vol. 99, no. 8, pp. 805–808, (2016).

(10) 渡辺圭祐, 東本崇仁, 赤倉貴子：“段階的抽象化を用いたプログラムを読む学習の支援システムの開発とその評価 (教育工学),” *電子情報通信学会技術研究報告= IEICE Tech. Rep. 信学技報*, vol. 115, no. 50, pp. 49–54, (2015).

(11) 新開 純子, 炭谷 真也：“プロセスを重視したプログラミング教育支援システムの開発,” 日本教育工学会論文誌, vol. 31, Suppl., pp. 45–48, (2008).

(12) 木村 優那, 矢入 郁子：“段階的詳細化能力とプログラミング学習の関係に関する研究,” 人工知能学会全国大会論文集, vol. 29, pp. 1–4, (2015).