6X−3

知的プログラミング環境における学習者モデル構築法

清口政彦　豊村真二　中島歩　上野晴樹
東京電機大学

1．　はじめに

知的CAIシステムは，高度個別教育の実現を目指すものであり，学習者モデルの構築が必要となる．我々が開発している知的プログラミング環境 1)は学生への教育的支援を目指すもので，学習者モデルの構築が必要となる．

これまで，学習者モデルに関しては様々な研究がされてきたが 2)，我々の学習者モデルは，パータベーションモデルの再構成型に分類される 3) ものである．

知的プログラミング環境における学習者モデルの構築は，バグの再現によって誤り原因を同定し，学生の理解状態を把握することにより行なわれる．そのために，プログラミングにおいて発生するバグの解析を行ないそれを体系化する．この体系化されたバグ知識を用い，システムが学生のバグを再現することが可能となり，学習者モデルを構築することができる．このとき，必要最小限の対話で学習者モデルを構築する．以下に，学習者モデルの構築法について述べる．

2．　バグについて

今までに行なわれた認知科学的実験（クイックソートプログラムの作成）の解析結果 4)より得られたバグは，1つのバグ原因によるものであった．しかし，バグは複数のバグ原因によって生じる場合がある．例えば，クイックソートにおける走査の概念を誤解しているうえに，それを実現するために用いた制御構造の知識を混同していたというバグである．バグの再現によって学習者モデルを構築するならば，このようなバグ原因を同定できなければならない．そこで，バグ知識の検討が必要となった．

2．1．　バグ知識

バグ知識は，バグ原因を同定するために用いる知識であるが，従来のものは 5)は，誤り原因とその振舞いが1つのパターンとして管理されている．そのため，いくつかの異なる誤り原因によってバグが説明できる場合や，同時に複数のバグ原因が発生しているときに，個々のバグ原因を同定することはできない．

この問題を解決するには，プログラム作成に用いられるプログラミング知識 1), 5)の個々の知識にバグ知識を管理させればよいと考えられる．つまり，バグ知識をよりプリミティブな表現で管理することによって，個々の知識に対する誤り原因の責任分担をより明確にするということである．

2．2．　バグ知識の表現

図1に示すように，バグ知識は，個々のプログラミング知識に対応している概念知識のレベルで管理され，対応するプログラミング知識を用いたときの，学生のもつバグの意図に対する振舞いとバグ原因で表現される．同様に，正しい意図に対する振舞いも概念知識で管理される．

バグの意図に対応するバグ原因は，1）プログラミング知識の誤解（misunderstood），2）不安なため冗長な表現をした（felt-uneasy），3）他の知識と混同した（confused-knowledge），4）プログラミング知識を用いる必要性がないと考えた(lacked)，の4つがある．ただし，4）の場合は，知識の欠落，知っているのに使わなかったの2つを含む．ここで，プログラミング知識を用いるということは，それに対応している概念知識に記述されたいずれかの意図を学生が適用したと考える．

図1のような概念知識の関係により，プログラミング知識間には，1）共通概念がある，2）類似概念があるという2つの関係が存在する．これらの知識関係を関係知識と呼ぶ．

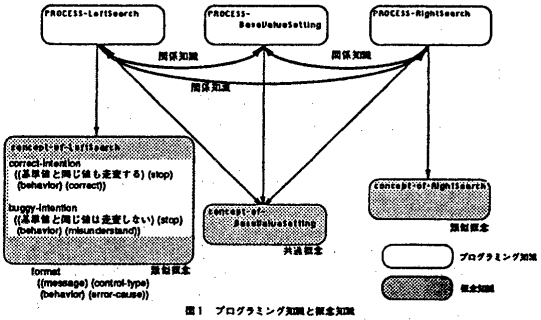

図1　プログラミング知識と概念知識

3．　学習者モデルの構築

学習者モデルは，バグの再現によってバグ原因が同定され，その結果から理解状態を把握することによって構築される．

3．1．　バグの再現

バグの再現は，ALPUS 4), 5)で意図理解された結果，学生の作成したプログラムにバグがあると認識されたプロセスに対して行なう．このとき，ALPUSで用いられる内部表現されたプロセス単位のデータがバグ再現のゴールとなる．なお，ここでのプロセスとは，アルゴリズムを階層表現したHPG 4), 5)を構成する1プロセスを指す．図2に示すように，バグの再現にはバグ知識のほかに，ALPUSで管理されているプログラミング知識が用いられる．これは，プログラムを実現するためのポインタを参照するためである．

3．2．　学習者モデルへの設定

学習者モデルは，プログラミング知識の個々の知識に対して設けられており，図3に示すような8つの項目（correct-count,use-count,miss-count,previous-state,past-record,used-problem,status,teaching-material）にデータをセットすることにより構築される．これによって学生の理解状態を把握したものとする．

理解状態を示すマークがstatusにセットされるが，マークは，アルゴリズムの知識とアルゴリズムに依存しない知識とでは設定されるマークが異なる．それは，上述した認知科学的実験の方法によるもので，実験の前にクイックソートアルゴリズムの

Construction Method of Student Model for Intelligent Programming Environment
Masahiko MIZUGUCHI　Shinji TOYOMURA　Ayumu NAKAJIMA and Haruki UENO
Tokyo Denki University

説明は行なうが，プログラム作成に必要なテクニック（交換の方法など）は教えていないからである．

アルゴリズムの知識には，「understand（学生は，当該知識を理解している）」と「don't-understand（理解していない）」がセットされ，アルゴリズムに依存しない知識には，「know（学生は，当該知識を知っている）」，「don't-know（知らない）」，「err（間違っている）」がセットされる．また，両知識に関係なく，初期状態は「initial-state」がセットされている．また，理解状態の把握ができないときには「not-know」が設定され，この場合は学生との対話によって理解状態が決定される．

バグの再現が終了しバグ原因の仮説が立てられると，より正確に学生の理解状態を把握するために，バグ原因となっているプログラミング知識に対して上述した関係知識の理解状態を考慮する．これにより，バグ原因であると診断されても正しく用いたという事実が分れば，バグ原因は，ケアレスミスであると指摘することができる．

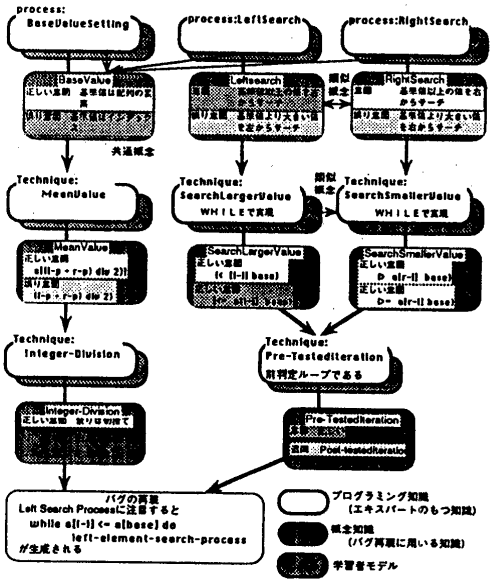

図2　バグの再現過程

スロットネと スロットタイプ	役割
correct-count	当該知識を正しく使用した回数
use-count	当該知識を用いた回数
miss-count	当該知識を誤った回数
previous-state	1つ前に診断した学生の振舞い
past-record	振舞いの履歴 (previous-stateの履歴)
used-problem	どの問題で用いたか
status	理解状態のマークの設定
teaching-material	教材知識

previous-state,past-recordのフォーマット
（（概念知識名　意図　誤り原因　使用プロセス名））

図3　学習者モデルにセットされる項目

4．　実行例

図4は，BASE-VALUE-SETTNG-PROCESSにはバグがない（基準値の概念は正しい）が，LEFT,RIGHT-SEARCH-PROCESSにおいて，基準値の概念と走査の概念の誤解であると診断されたプログラムに対して構築された学習者モデルの一部（LEFT-SEARCH-PROCESS）を示している．ここで，関係知識の参照によって，LEFT,RIGHT-SEARCH-PROCESSの基準値設定（共通概念）のバグはケアレスミスであり，操作の概念（類似概念）を誤解しているというもので，アルゴリズムを理解していないことがわかる．

```
sm-leftsearch
FRAME NAME:sm-leftsearch
correct-count       integer   0
use-count           integer   1
miss-count          integer   1
previous-state      list      ((concept-of-leftsearch
                               intention2
                               misunderstand
                               process-leftsearch)
                              (concept-of-basevalue
                               intention2
                               careless-mistake
                               process-leftsearch))
past-record         list      ((concept-of-leftsearch
                               intention2
                               misunderstand
                               process-leftsearch)
                              (concept-of-basevalue
                               intention2
                               careless-mistake
                               process-leftsearch))
used-problem        list      (recursive-quicksort)
status              atom      do-not-understand
teaching-material   atom      process-leftsearch

sm-technique-searchlargervalue
FRAME NAME:sm-technique-searchlargervalue
correct-count       integer   1
use-count           integer   1
miss-count          integer   0
previous-state      list      ((concept-of-technique-searchlargervalue
                               intention1
                               correct
                               process-leftsearch))
past-record         list      ((concept-of-technique-searchlargervalue
                               intention1
                               correct
                               process-leftsearch))
used-problem        list      (recursive-quicksort)
status              atom      know
teaching-material   atom      technique-searchlargervalue
```

図4　LEFT-SEARCH-PROCESSに対する学習者モデルの構築例の一部

5．　おわりに

このバグの再現による学習者モデル構築法は，HPGの構造が極端に異なるようなバグには対応することができないため，より柔軟なバグの再現を行なうためには，プログラミング知識を含めた検討が必要であると考えている．また，履歴や知識の難易度を考慮して理解状態を設定することや，学習者モデルを参照して学生に教授するという機能の実現を検討している．

参考文献

1)　上野晴樹：知的プログラミング環境ープログラム理解を中心に一，情報処理 VOL. 28,NO. 10, PP. 1280-1295 (1987)
2)　E.Wenger：知的CAIシステム　オーム社　(1990)
3)　溝口，角所：知的CAIにおける学習者モデル，情報処理 VOL. 29,NO. 11,PP. 1275-1282 (1988)
4)　関本，上野：プログラム理解システムALPUSにおける学生の意図理解の方法，信学技法AI91-75,pp. 9-16 (1992)
5)　中島，上野：プログラム知識の表現およびプログラム理解システムALPUSへの応用，信学技法AI91-76,pp. 17-24 (1992)