5ZC-01

レイアウト定義言語DASTを用いたオブジェクト構造可視化の使用性及び有効性の評価*

松崎 駿†，松澤 芳昭‡，酒井 三四郎§

1 はじめに

プログラミング教育ではデータ構造の概念を理解するためにオブジェクト構造の図示が行われることが一般的である．データ構造を学習するにあたり，学習者はまずデータ構造の概念を理解するために図を見ることで，そのデータ構造がどのような形であるかのイメージを確立させる．

このことから，実際にデータ構造を構築するプログラムを実行する際に生成されるオブジェクトが，あるタイミングではどの変数から参照されるかといったオブジェクトの状態を可視化し，実際にイメージに近い形を形成していく過程を見せることで，このようなプログラムの動作の把握，及び実装を補助することができると推測する．

2 先行研究と本研究の目的

プログラム実行時の状態を可視化する既存のツールは，Demian らによる Eclipse プラグイン Jive[1] のような，可視化を行うプログラムについての制限が無いが描画される図が理想と異なる体系を取ることがある「汎用目的のツール」と James らによる jGrasp[2] のような，対応するプログラムは限定されるが用途に合わせて適切な体系で可視化が行われる「特定用途に特化したツール」の 2 種類に分類できる．

既存の研究における問題点として，「確実にデータ構造の学習に適した学習者のイメージに近い体形をで描画を行う」ということ，「プログラムの内容，実装方法に関わらず可視化に対応する」ということの 2 点を両立することができないという点があげられる．

この 2 点を両立するための方法として，本研究では，可視化によって描画される図のレイアウトを，実行するプログラムごとに定義することで最適な図を描画するという方法の提案，そのための軽量なレイアウト定義言語「DAST」の設計，および DAST を読み込んだ状態で Java プログラムを実行することで，DAST で定義されたレイアウトに従って描画を行う可視化環境の開発を行った．[3]

```
Class:Tree, Node
Tree{
        root:v
}
Node{
        right:v>
        left:v<
}
```

図 1: 二分探索木の DAST 記述

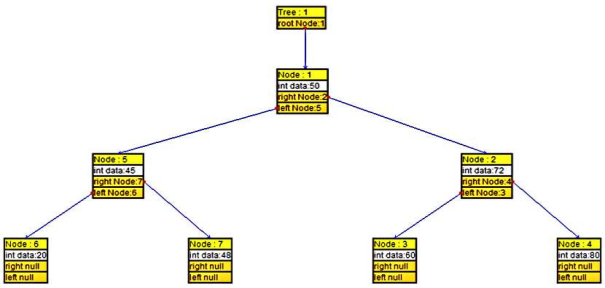

図 2: 二部探索木のプログラムの実行で得られる図

3 DAST の概要

Tree クラス，Node クラスの 2 つのクラスで構成される二分探索木のプログラムの配置を定義する場合の DAST の記述例を図 1 に示す．

図 1 のように，まず 1 行目で「Class:」に続けて図で表示したいクラスを列挙し，2 行目以降で各クラスが持つメンバ変数について，その変数で参照しているオブジェクトを，変数を持つオブジェクトを中心としてどの方向に配置するかの指定を行う．配列の変数の場合も同様に方向の指定が可能となっている．図 1 の DAST ファイルを読み込んだ状態で実際にプログラムを実行しすると図 2 が得られる．

DAST を読み込み，可視化を行う機能は Eclipse のプラグインとして実装されており，Eclipse でデバッグされる Java プログラムの可視化が行われる．また，Eclipse の機能であるブレークポイントやステップ実行によってプログラムを段階的に実行することで，描画される図もその時の状態に合わせて変化する．

4 評価実験

プログラミングの学習者が DAST を使いこなせるかを検証するため，情報系の大学生 6 名 (以降 A~F と表記) を対象とした評価実験を行った．なお，被験者は

*Evaluation of Usability and Effectiveness of Object Structures Visualizer Using Layout Definition Language “DAST”

†Matsuzaki Shun 静岡大学大学院　情報学研究科
‡Matsuzawa Yoshiaki 青山学院大学 社会情報学部
§Sakai Sanshiro 静岡大学　情報学部

全員プログラミング，アルゴリズムとデータ構造の授業の履修を終えており，本実験で扱うデータ構造について，全員がそれらの概念を把握していると言える．

4.1 実験 1

プログラムと出力したい図を提示し，その図を得るための DAST を記述をさせ，記述に要した時間の計測を行った．問いは全部で 3 問で，一つのプログラムに対し出力して欲しい図が 1 つ提示される．

被験者が DAST の記述に要した時間を表 1 に示す．

表 1: 実験 1 の結果 (分：秒)

	A	B	C	D	E	F	平均
問 1	3:40	4:47	3:11	2:43	3:55	1:12	3:14
問 2	4:03	8:11	7:47	6:19	10:40	1:25	6:24
問 3	4:16	9:13	8:24	6:14	10:03	3:06	6:53

4.2 実験 2

意図的にバグを混入したプログラムを可視化環境で実行しバグの修正をさせ，修正に要した時間の計測を行った．

被験者には 2 つの二分探索木のプログラム (P1, P2) の修正を行わせた．修正箇所を特定するために，一方では Jive を，もう一方では DAST を用いてプログラム状態の可視化を行った．また，DAST で可視化を行う場合は被験者自身が可視化する際の適切な配置を考え，DAST の記述を行った．なお，修正するプログラムと用いる可視化ツールの組み合わせは被験者ごとに異なる．

各被験者が DAST の記述に要した時間を表 2 に，DAST を用いた場合と Jive を用いた場合それぞれで修正箇所の特定までに要した時間を表 3 に示す．

表 2: DAST の記述に要した時間 (分:秒)

	A	B	C	D	E	F	平均
P1	3:37	-	-	5:14	5:50	-	4:54
P2	-	6:13	6:14	-	-	2:30	4:59

表 3: 修正箇所の特定に要した時間 (分:秒)

	A	B	C	D	E	F	平均
DAST P1	2:47	-	-	2:19	1:49	-	2:18
DAST P2	-	1:08	3:15	-	-	1:34	1:59
Jive P1	-	9:23	11:15	-	-	12:51	11:10
Jive P2	1:47	-	-	1:42	16:23	-	6:37

4.3 考察

表 1，表 2 より，長くとも 10 分程度で DAST の記述ができており，DAST の記述は期待ほどではないが短い時間で行えていると言える．

また，表 3 より，DAST を用いて P1，Jive を用いて P2 の修正を行った被験者 A，D，E のうち E の 1 名，Jive を用いて P1，DAST を用いて P2 の修正を行った被験者 B，C，D の 3 名全員，合計で 4 名が Jive を使った場合よりも DAST を使った場合よりも早くプログラムの修正箇所を特定しており，残りの 2 名も DAST との差は 1 分以内である．

Jive の可視化では，一つのオブジェクトが他のオブジェクトを参照している変数が一つだけある場合，参照先のオブジェクトは参照元の真下に配置される．また，変数による参照は矢印で表現されるが，矢印の根元の位置からその矢印がどの変数による参照なのかを一目で把握することはできなくなっている．

複数の被験者が Jive による修正が DAST での修正箇所特定よりも大幅に時間がかかった原因としては，これらの Jive の仕様が，誤った変数で参照されているオブジェクトを特定する妨げになっていたと考えられる．一方，DAST を使った修正箇所特定の場合は，あらかじめ自分で DAST の定義を行っているため，オブジェクトが配置される方向からそのオブジェクトを参照している変数の把握が可能であり，瞬時に誤った参照に気づくことができたと思われる．

5 おわりに

実験における DAST の記述に要した時間は想定より長かったものの，その後のアンケートでは DAST の記述が難しかったと回答する被験者はおらず，慣れればもっと楽に書けそうという感想が多数あったことから，繰り返し DAST の記述を重ねれば記述にかかる時間の短縮が見込まれる．

また，Jive で可視化を行う際に起こりうる問題の一部が DAST では解消されており，Jive と DAST のどちらが使いやすかったかという質問では，DAST の記述を行うという手間を加味しても全員が DAST のほうが使いやすかったと回答していた．

以上のことから，プログラミング学習において，DAST の使用性，有効性はともに十分であると考える．

参考文献

[1] Demian Lessa et al:JIVE:A Pedagogic Tool for Visualizing the Execution of Java Programs, Technical report, 2010, http://www.cse.buffalo.edu/tech-reports/2010-13.pdf, (2014/02/01 アクセス).

[2] Lacy Mortgomery et al: Testing the jGRASP Structure Identifier with Data Structure Examples from Textbooks, ACM-SE 46: Proceedings of the 46th Annual Southeast Regional Conference on XX, pp.198-203, (2008).

[3] 松崎駿:プログラミング教育現場で利用可能な軽量オブジェクトレイアウト言語「DAST」の設計と評価，情報処理学会情報教育シンポジウム SSS2014, pp.233-238,（2014）.