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

松崎 駿[1,a)]　酒井 三四郎[1,b)]　松澤 芳昭[1,c)]

概要：初学者のアルゴリズム理解支援を目的としたプログラム動作過程の可視化ツールが提案されている．しかし既存のツールには，(1) 汎用目的で設計されたツールでは図の配置が学習者の期待どおりに行われない，(2) ある用途に特化して設計されたツールではその用途にしか利用できない，という問題がある．そこで本研究では，プログラムで生成されるオブジェクトのレイアウト定義言語「DAST」の設計を行った．DAST の設計目標は (1) 教育者/学習者が大きな負担なく利用できる程度に簡潔に記述できること，(2) データ構造とアルゴリズムの教科書に掲載される程度のサンプルプログラムについてはレイアウト可能なこと，である．Java プログラムを対象とする DAST の言語処理，およびレイアウト処理システムを Java で開発した．アルゴリズムとデータ構造の教科書 3 冊に掲載されている 5 種類のデータ構造を対象に DAST による記述実験を行った．その結果，全てのプログラムで学習に有用なレイアウトの配置をすることができることを確認した．また，その記述は量はいずれも 10 行以内であった．

キーワード：可視化，オブジェクト，レイアウト，言語

DAST: Lightweight Object Layout Language for Object-Oriented Programming Education

Abstract: The visualization tool of the program execution process to help beginner to understand algorithm is proposed. Conventional tools have problems, (1) In the case it's designed to be usable generally, the layout of obtained figure is different from the expectation, (2) In the case it's designed to specialized in a certain pattern, it's available only in the pattern. This study was designed an original language "DAST", defining layout of the object. Design goal of DAST is as follows. (1) The description make simple not to be burdensome for teacher and student, (2) At least support all programs published in a textbook of algorithm and the data structure as a sample. I developed language handling of DAST for JAVA programs and a layout processing system by Java. I was performed description experiment targeting five kinds of data structures placed in three textbooks of algorithm and the data structure .As a result, I confirmed that DAST could place the layout that was useful for learning all programs In addition, the quantity of description was all less than ten lines.

Keywords: visualization, object, layout, language

1. 研究の背景と目的

プログラミング教育では，データ構造理解の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．例えば，**図 1** に二分探索木の説明で用いられる一般的な図を示す．図を見た学習者は，キーの値が低いオブジェクトが左に，高いオブジェクトが右に置かれるというイメージを始めに持つことが考えられる．

このことから，実際にデータ構造を構築するプログラムの実行過程で生成される各オブジェクトの参照関係を可視化し，そのデータ構造のイメージに近い形態を取れているかを確認することでより理解を深めることができると推測する．

プログラムの実行過程を可視化するツールはいくつか存

[1] 静岡大学大学院情報学研究科
Graduate School of Informatics, Shizuoka University
[a)] matsuzaki-s@sakailab.info
[b)] sakai@inf.shizuoka.ac.jp
[c)] matsuzawa@inf.shizuoka.ac.jp

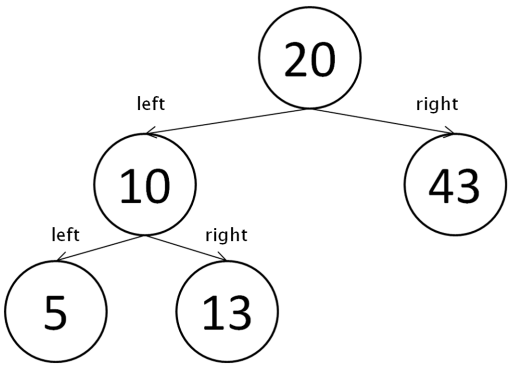

図 1　一般的な二分探索木の図

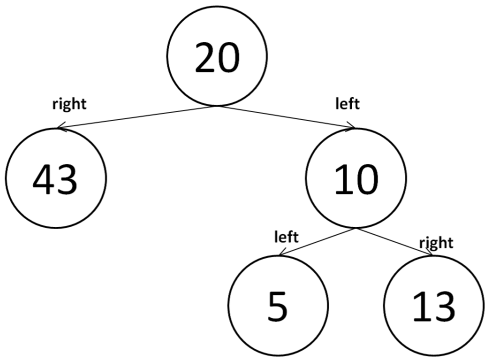

図 2　一部の right が左，left が右に配置された二分探索木

在しており，どのようなプログラムにも対応できる汎用目的のもの，ある用途に特化したものの 2 種類に分類できる．

汎用目的のツールによる可視化では図の配置が学習者のイメージ通りに行われず，かえって混乱を招く可能性がある．例えば**図 2** のように，あるオブジェクトが別のオブジェクトの right という名前の変数に参照されているにも関わらず可視化した際左側に表示されるといったことが起こり得る．

一方で，ある用途に特化した可視化ツールの場合，対応しているプログラムについては正確な配置による可視化が行えるが，可視化が行えるプログラムは限られてしまう．

そこで本研究では，任意のプログラムの実行時に生成されるオブジェクトを，学習者が持つデータ構造のイメージに近い形態で可視化するためのオブジェクトレイアウト言語の設計，及び言語処理，レイアウト処理システムの開発を行った．

2. 先行研究

2.1 汎用目的のツール

Demian らによって教育用ツールとして開発された Eclipse プラグイン Jive[1] は Java のプログラムデバッグ時に生成されたオブジェクトが持つ変数の値や参照先，実行中のメソッドといった情報を図で表示する．

この時，オブジェクトが持つ変数が別のオブジェクトを参照している場合，参照先のオブジェクトは参照元オブジェクトの下に配置され，2 つのオブジェクトは矢印で結ばれる．また，一つのオブジェクトが複数の変数を持ち複数のオブジェクトを参照している場合，参照先のオブジェクトは左から生成順に並べられる．例えば right，left の 2 種類の変数を持つオブジェクトがあり，right が参照しているオブジェクトの方が先に生成されている場合は**図 3** のように right が参照しているオブジェクトにもかかわらず左側に配置されてしまう．

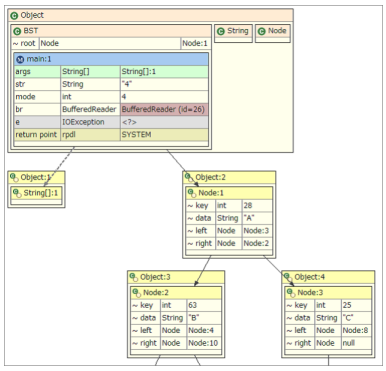

図 3　Jive で二分探索木のプログラムを実行して得られる図

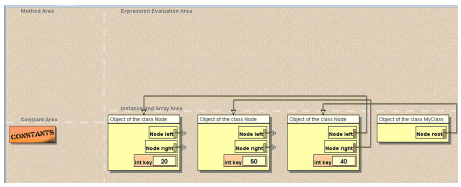

図 4　Jeliot3 で二分探索木のプログラムを実行して得られる図

Andrs らによって開発された Jelot3[2] は手続き型プログラミングやオブジェクト指向プログラミングの初学者を支援するために設計された開発環境で，オブジェクトの生成，変数の値や参照先の変化といった Java プログラム実行時の挙動をアニメーションで表現する機能を持つ．

実際に Jelot3 で二分探索木のプログラムを実行すると**図 4** のようになる．アニメーションが再生される画面は 4 つのエリアに分かれており，プログラムで生成されたオブジェクトは右下の Instance and Array Area に横一列に並べられる．各オブジェクトが持つ変数による他のオブジェクトとの関連は矢印により表現されるが，オブジェクトが横一列に並んでいるため木構造のような関係を一目で把握するのは困難である．

2.2 特定の用途に特化したツール

James らによって開発された jGrasp[3] は特定のデータ構造のプログラムを実行と同時に可視化する機能を持つ．この機能の目的は，データ構造を構築するオブジェクトの理解を支援することであり，配列，連結リスト，二分木，ハッシュマップに対応している．jGrasp で二分探索木のプログラムを実行すると**図 5** が得られる．

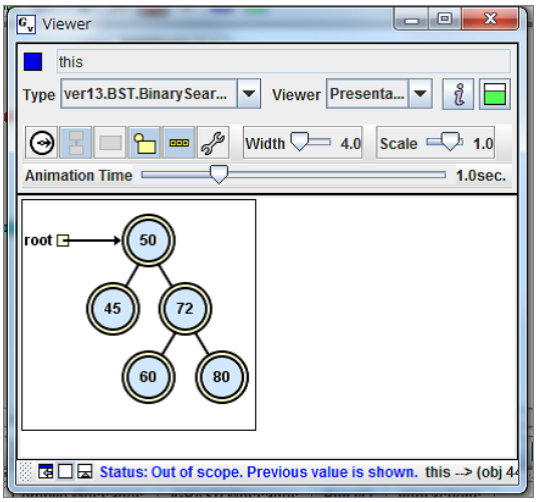

図 5　jGrasp による二分探索木の図

2.3　既存研究の問題点

現状の可視化ツールの問題として，Jive や Jeliot3 では，可視化の際ツールごとに決められたルールの下で図の配置が行われる．そのため，「変数 right は木構造の右部分木を参照する」，「変数 next はリストの次の要素を参照する」といったプログラムの作成者によって変数に込められた意味は無視した配置が行われる．

一方で jGrasp のような特定のデータ構造を対象とすることが前提の可視化ツールの場合，図の配置はそれらのデータ構造の学習者のイメージに近い形態となる配置が行われるが，対応していない構造を持つプログラムを可視化することはできない．対応しているデータ構造のプログラムの場合も，実装方法によっては可視化できない場合があり，可視化するために実装の手段が制限されることがある．

このように，既存のプログラム実行時の状態を可視化するツールは「学習者のイメージに近い形態の描画を行う」ということ，「プログラムの内容，実装方法にかかわらず可視化ができる」ということの 2 点を両立することができていないというのが現状である．

3.　レイアウト言語 DAST の提案

3.1　DAST

既存のプログラム可視化ツールにおける問題を解決する手段として，プログラムの実行者が図に表示するクラスを指定し，オブジェクトの配置のレイアウトを自分で定義を行うという方法を提案する．

そこで，本研究では独自のオブジェクトレイアウト言語「DAST」の設計，及び Java のプログラム実行時に生成されるオブジェクトを DAST ファイルの記述を基にした配置で描画を行う可視化ツールの開発を行った．

本システムは，レイアウトの定義を記述した DAST ファイルをツールが読み込んだ状態で Java のプログラムを実行することで，通常の実行結果に加え各オブジェクトの状態やオブジェクト同士の関連を明示した図が得られるものとなっている．この図は対象のプログラムによって生成されたオブジェクトの情報をリアルタイムで描画しており，ステップ実行によって段階的にプログラムを実行した場合やコンソール等での入力によりオブジェクトのメンバ変数の値や参照先が変化していく場合は連動して図も変化していく．

3.2　想定している利用場面

本システムの利用場面としては，以下のような状況を想定している

- 教師がプログラムと DAST ファイルを作成し，プログラムを実行することでデモンストレーションを行う．
- 教師が DAST ファイルを作成しクラス名，変数名などの仕様を指定したプログラム作成の課題を出す．生徒は課題のプログラムを作成し，DAST ファイルを用いて自身のプログラムを可視化し正しく動作しているかを確認する．
- 作成したプログラムに対して自身で DAST を記述し，作成したプログラムを可視化することで，そのプログラムが正常に動作しているかを確認する．

3.3　設計目標

DAST を設計するに当たり，以下の 2 点を設計目標とする．

(1) 教育者/学習者が大きな負担なく利用できる程度に簡潔に記述できるようにする．

(2) データ構造とアルゴリズムの教科書に掲載される程度のサンプルプログラムについてはレイアウト可能にする．

先行研究として挙げられた Jive や Jelot3 の場合，対象となるプログラムを実行するだけで可視化が行えるのに対し，本システムではソースコードのほかに DAST ファイルを用意する必要がある．この DAST ファイルの作成が利用者にとって大きな負担となることがないよう，これらの記述は極力簡単にするべきである．

本システムに対応させるためにプログラムの設計方法が制限されるということがなく，自分が作成したプログラムがどのような方法で設計された場合でも，DAST により定義を行い可視化が行えるようにすべきである．現段階としては，データ構造の学習者が最低限理解すべき範囲のデータ構造を網羅することを目標とする．

3.4　DAST の記述方法

3.4.1　基本的な記述方法

DAST の記述例を**図 6** に示す．

図 6 では，2 つのクラス，BinarySearchTree クラスと Node クラスで構成される二分探索木のプログラムのレイ

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

図 6　二分探索木の DAST 記述

上	^	右上	^> または >^
下	v	左上	^< または <^
右	>	右下	v> または >v
左	<	左下	v< または <v

表 1　各方向を指定する際の記述

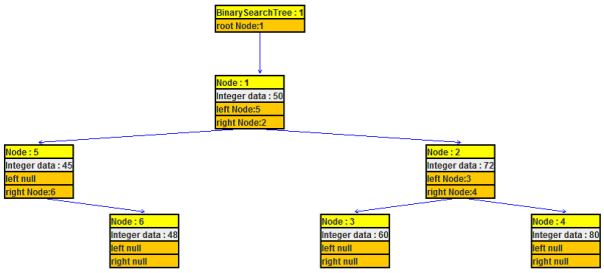

図 7　二部探索木のプログラムの実行で得られる図

アウトを定義している．

DAST は，まず 1 行目で「Class:」に続けて図で表示したいクラスを列挙する．

そして，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を指定する．方向の指定では「v，^，<，>，^>，v>，^<，v<」のいずれかを選択し記述する．指定する方向と DAST における記述の対応関係を**表 1** に示す．

この例の場合，BinarySearchTree クラスのオブジェクトが持つ変数 root が参照するオブジェクトを下に，Node クラスのオブジェクトが持つ変数 right が参照するオブジェクトを右下に，left が参照するオブジェクトを左下に配置すると定義している．

図 6 の DAST ファイルを読み込んだ状態で実際に BinarySearchTree クラスと Node クラスで構成される二部探索木のプログラムを実行し，整数をキーとする 5 つのデータを挿入した状態で得られる図が**図 7** となる．

なお，DAST ファイル内で列挙されていないクラスのオブジェクトは図には表示されず，参照関係についても DAST ファイル内で方向の指示が行われている変数によるもののみが表示され，指示のない変数による参照は描画されない．また，オブジェクトが持つ int や String といった基本型の変数の値も図中に表示される．

図 7 より，実行したプログラムは変数 data の値が大きいオブジェクトを right で，そうでないオブジェクトを left で参照するという二部探索木の性質が正しく実装されているということが読み取れる．

```
Class:HashC, Cell, MyKey
HashC{
                table[]:v
}
Cell{
                key:>
                next:v
}
MyKey{
}
```

図 8　ハッシュ (チェイン法) の DAST 記述

3.4.2　配列を持つ場合の記述方法

描画の対象に配列が含まれる場合の DAST の記述例を**図 8** に示す．

こちらの DAST の記述は HashC, Cell, MyKey の 3 つのクラスで構成されるハッシュのチェイン法のプログラムのレイアウトを定義している．オブジェクトが配列を持っている場合，その配列が参照しているオブジェクトを配置する方向を変数の場合と同様に指定する．変数名と区別するため，配列は名前の後に [] を付けて記述する．この場合は HashC クラスのオブジェクトが持つ配列 table を下に，Cell クラスのオブジェクトが持つ変数 key が参照するオブジェクトを右に，next が持つオブジェクトを下に配置する．

図 8 の DAST ファイルを読み込んだ状態で，HashC クラス，Cell クラス，MyKey クラスで構成されるハッシュのプログラムを実行し，5 つのデータを挿入した状態で得られる図が**図 9** となる．

3.5　DAST の設計思想

DAST ではオブジェクトが持つ参照型変数の参照先のオブジェクトをどの方向に配置するかをクラスごとに指定する．

オブジェクトを配置する方向は，上下左右に右上，右下，左上，左下の 8 方向いずれかから選択する．

DAST は記述する内容が主にこの，「変数の参照先のオブジェクトを配置する方向」のみであり，方向の種類も 8 通りから選択する非常にシンプルなものとなっている．

この仕様における問題点としては以下が考えられる．

- 参照型変数を 9 個以上持つプログラムに対応できない
- 同じクラスから生成されるオブジェクトについて，個別に参照先の方向指定ができない

設計目標（2）より，今回 DAST が対象とするプログラムの範囲は，データ構造とアルゴリズムの教科書に掲載される程度のサンプルプログラムとしている．この範囲のプログラムについて，参照型変数の上限は 8 つで十分であると考えられる．また，同じクラスから生成されたオブジェクトについて，それぞれ個別に参照先の方向指定をする必要があるという状態は考えにくい．

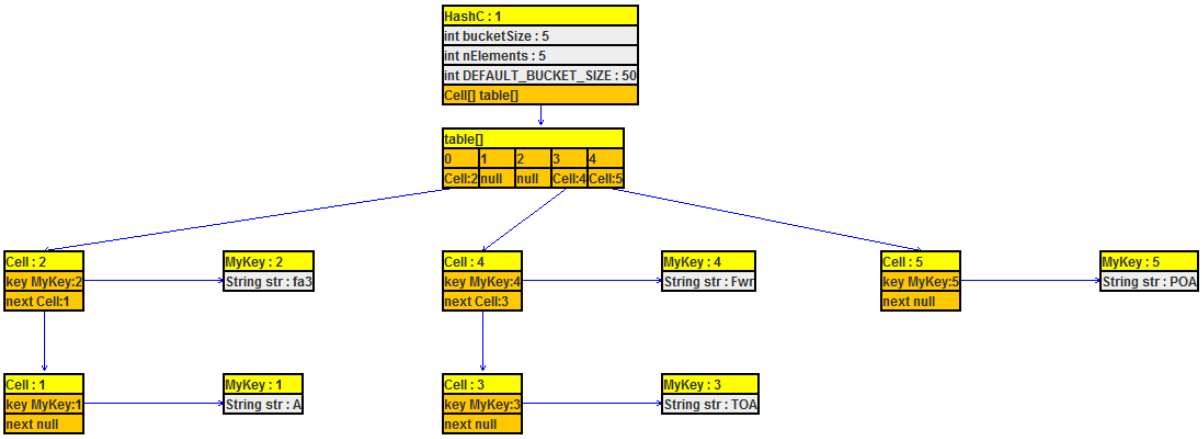

図 9 ハッシュ (チェイン法) のプログラム実行で得られる図

以上のことから，設計目標における，なるべく小さな負担で簡潔に記述することができるという点と，対象とするプログラムの範囲の兼ね合いから本仕様が適切であると考える．

こ

4. 評価

レイアウト言語 DAST について，設計目標を満たしているか評価を行う．

文献 [4]，[5]，[6] に掲載されている連結リスト，二分木，B 木，ハッシュ（チェイン法，オープンアドレス法）の計 5 種類のデータ構造を構築するプログラムについて，筆頭著者が DAST によるレイアウト定義を行う．

これにより，全てのデータ構造を構築するプログラムに対して DAST による配置の定義が行えるか，同様のデータ構造が異なる方法で実装されている場合に同様に DAST による定義が行えるか，そして DAST ファイルの行数はどれほどになるかの 3 点を確認した．

5. 結果

確認結果の概要を**表 2** に示す．表 2 において，○は DAST を用いたレイアウト定義ができたこと，-は該当のプログラムが対象の書籍に非掲載だったことを示す．括弧内の数値はそのプログラムに対する DAST ファイルの行数を示す．

表 2 を見てわかる通り，3 冊の書籍に掲載されてる対象のデータ構造の全てのプログラムについて，DAST によるレイアウト定義が行え，いずれの場合も DAST ファイルの行数は 10 行以内で済んでいる．

また，同じデータ構造を構築するプログラムでもプログラムによって実装のアルゴリズムは異なっている．しかしながら，プログラムを構成するクラスやクラスが持つ変数については役割が共通している場合が多く，そのような場合は DAST の記述もほぼ同じで済む．

表 2 DAST による定義の可否
Table 2

	文献 [4]	文献 [5]	文献 [6]
連結リスト	○ (7)	○ (7)	○ (7)
二分木	○ (8)	○ (8)	○ (8)
B 木	○ (8)	○ (8)	-
ハッシュ（オープンアドレス）	○ (9)	-	○ (5)
ハッシュ（チェイン）	○ (10)	-	○ (10)

3.4.1 の例示で用いた図 6 は書籍 [4] に掲載されている二部探索木のプログラムに対する DAST 記述である．[6] に掲載されている二部探索木のプログラムに対する DAST の記述，及びプログラム実行によって描画される図を**図 10** に示す．この二つのプログラムはそれぞれ異なるアルゴリズムで二部探索木を構築しているが，DAST の記述はほとんど同じで済んでいる．

6. 考察

6.1 設計目標（1）ついて

いずれのプログラムについても DAST ファイルの本文は 10 行以内で済んでいるということから，DAST は少ない記述でレイアウト定義を行うことができている．

同じデータ構造で実装方法の異なるプログラムでも，DAST の記述はクラス名，変数名を変えたことを除くとほとんど同じで済むことから，DAST によるレイアウト定義はオブジェクトが持つ変数の役割が理解できていればデータ構造構築のアルゴリズムをほとんど考慮することなく行えると言える．

このことから，この設計目標については．記述の量は少く済ませるという点は達成できており，記述の難易度を低くするという点でも部分的に達成できていると言える．

6.2 設計目標（2）について

文献 [4] に掲載されている 5 種類全てのデータ構造のプログラムについて DAST によるレイアウト定義が行え，更

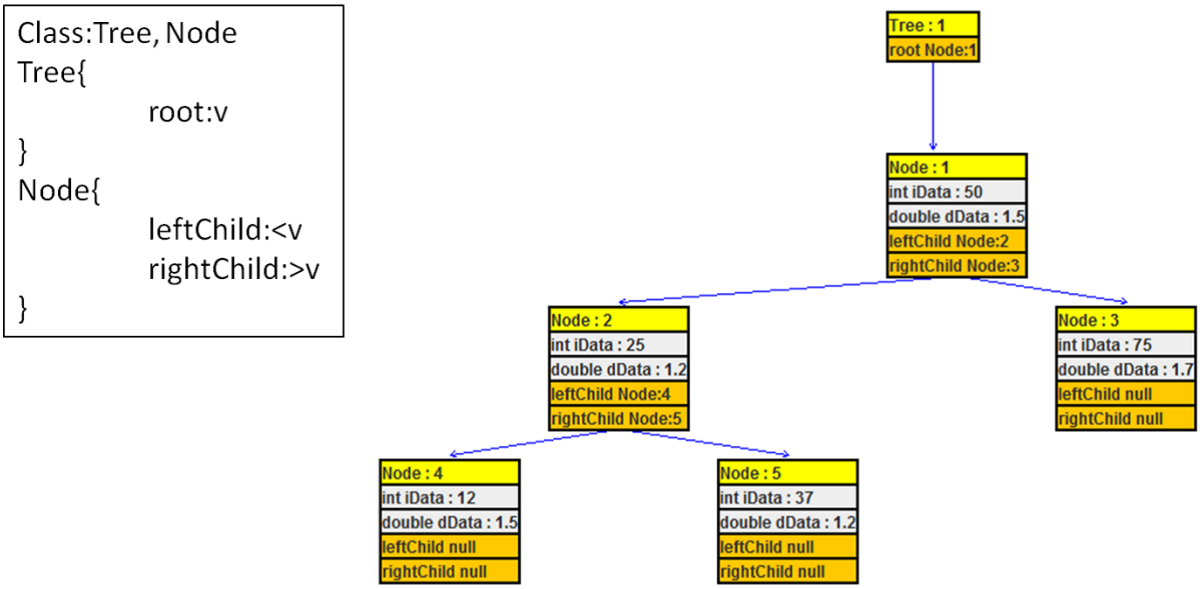

図 10 [6] に掲載されている二部探索木のプログラムに対する DAST の記述，及び描画される図

に文献 [5]，[6] に掲載されているこれら 5 種類のデータ構造を構築するプログラムについて，いずれも DAST でレイアウトを定義することができた．

このことから，アルゴリズムとデータ構造を学習する学習者が理解すべき全てのデータ構造のプログラムについて DAST でレイアウトを定義することができる．

また，これら 5 種類のデータ構造はプログラムがどのように実装されていても DAST は対応することができる．

以上のことから，この設計目標については達成できていると言える．

6.3 評価の限界と今後の課題

今後の課題としては，設計目標（1）については，今回行った確認は著者自身が DAST によるプログラムのレイアウト定義を行ったため，DAST の記述の難易度について客観的なデータが得られていない．DAST による配置定義が実際のアルゴリズムとデータ構造の学習者でも容易に行えるかを確認する必要がある．そのうえで，DAST で配置を指定して可視化を行うことがアルゴリズムとデータ構造の学習者の補助に繋がることを証明していく．

設計目標（2）については，あらゆるオブジェクト指向プログラムの実行状況の可視化に対応させるための改良を行い，配置を指定したプログラムの可視化がプログラミングの学習，及び開発の場において利用することができるものにしていきたいと考えている．

参考文献

[1] Demian Lessa, Jeffrey K. Czyz, Bharat Jayaraman :JIVE:A Pedagogic Tool for Visualizing the Execution of Java Programs, Technical report, State University of New York at Buffalo, 2010, http://www.cse.buffalo.edu/tech-reports/2010-13.pdf, (2014.02.01)

[2] Andrs Moreno, Niko Myller, Erkki Sutinen :Visualizing programs with Jeliot 3, AVI 04:Proceedings of the working conference on Advanced visual interfaces, pp.373-376. (2004)

[3] Lacy Mortgomery, James H. Cross, T. Dean Hendrix, Larry A. Barowski: 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)

[4] 近藤嘉雪 [著]，『定本 Java プログラムのためのアルゴリズムとデータ構造』，2011，ソフトバンククリエイティブ 487pp

[5] 奥村晴彦，首藤一幸，杉浦方紀，土村展之，津留和生，細田隆之，松井吉光，光成滋生 [著]，『Java によるアルゴリズム辞典』，2003，技術評論社，500pp

[6] Robert Lafore[著]，岩谷宏 [訳]，『Java でまなぶアルゴリズムとデータ構造』，1991，ソフトバンク株式会社，628pp