Vol.2010-SE-167 No.9
2010/3/18

オブジェクト指向プログラムの動作の可視化

中原　進[†]　紫合　治[†]

本論文では，オブジェクト指向プログラムの動作を可視化するために，従来のオブジェクト図にメソッド呼出しやスタティック属性等の情報を追加した"拡張オブジェクト図"を規定し，プログラムの実行が進むに従ってオブジェクト全体の状態が変化していく様子を，拡張オブジェクト図の変化(スライド描画)によって分かりやすく示す方式と，それを実装したシステムについて述べる．システムは，JPDA(Java Plat form Debugger Architecture) を用いてプログラム実行時のメソッド呼出しや属性への代入等のイベントを捕らえ，オブジェクトの状態についての情報を抽出する．さらに，配列やコレクションの要素の値や，ローカル変数の値の変化情報等も抽出している．これらの抽出情報をもとに，プログラム実行の各動作時点でのオブジェクト全体の様子を，拡張オブジェクト図の描画によって分かりやすく可視化することが出来る．

Visualization of Object-Oriented Program Execution

SUSUMU NAKAHARA[†]　OSAMU SHIGO[†]

This p aper p resents ou r p roposed method and th e support s ystem for object o riented program visualization by drawing extended object diagrams at each step in the program execution. The extended object diagra m sho ws a whole objects s tates and relations among the objects by UML object diagram with additional information, such as method calls and static field va lues. The system obtains object state data by catching program execution ev ents, lik e method inv ocations and f ield as signments, u sing J PDA (Java Platform Debugg er Architecture). It also obtains v alue ch anges of arr ay and collection elements and method lo cal v ariables. After obtaining the execution data, the s ystem draws th e extended object d iagram for each p rogram ex ecution step, wh ich shows an instance of whole object status in understandable diagram.

1. はじめに*

近年ソフトウェアの開発が活発になって来ている．プログラムの大規模・複雑化，プログラミングの普及により，プログラムの動作の不可視性から開発やデバッグ，教育が難しいという問題が出てきた．プログラムの動作が入力（引数）と結果（戻り値）との間で不可視である事はデバッグや教育にとって障害となる．プログラムの動作を可視化するために一般的に用いられるやり方として，ソースコード中に過程を出力するコードを挿入する方法があるが，導入及び解除のコストが膨大となり，またコードの可読性が低下するので現実的では無い．特にオブジェクト指向ではオブジェクトやその関係等の複雑な概念がありよりプログラムへの理解を妨げている．これによりオブジェクトの生成やオブジェクト間の関係，オブジェクトの状態の遷移を的確に把握することは困難となり,デバッギングやオブジェクト指向の教育が難しくなっている．

そこで，デバッグツールやプログラム可視化ツールの研究・開発が活発に行われてきた．可視化には静的・動的の 2 種類がある．静的なものとして UML[1]がある．UML はプログラムの設計や構築に用いられている．反対に，構築後に実行時情報を取得するプログラムの分析ツールとして動的なものが研究されている．ミクロな範囲では変数の変化を監視するツール[2]が，マクロな範囲では実行時情報からシーケンス図の自動生成[3]ツールや，プロファイラー[4]等がある．

しかし，既存の動的な可視化システムではオブジェクトに注目したものは無く，オブジェクト指向プログラムの可視化にはあまり向いていない．オブジェクト指向の本質はオブジェクト同士の関係である．オブジェクト指向プログラムの振る舞いはオブジェクト同士の関係とオブジェクトの状態が変化する事により成り立っており，その変化を可視化するにはオブジェクト図を用いる事が最適であると考えられる．

そこで本論文ではプログラム実行時のオブジェクト同士の関係及びオブジェクトの状態の変化を解りやすくするためにオブジェクト図をベースに拡張した図（拡張オブジェクト図）を定め，プログラム実行過程に伴うオブジェクトの変化の様子を拡張オブジェクト図を用い可視化する事によりデバッグやプログラミング教育を支援するシステムについて述べる．このシステムの概念を図 1 に示す．オブジェクト指向プログラムの実行開始から終了までの間の任意の時点のオブジェクトの様子を拡張オブジェクト図により可視化することができる．

以下，第 2 節では考案したシステムの概要を，第 3 節では実行時情報の履歴取得方法について，第 4･5 節では第 3 節で取得した実行履歴を元に拡張オブジェクト図を生成する方法について述べる．第 5 節で今後の課題を示し，最後に第 6 節でまとめを述べる．

*[†] 東京電機大学
Tokyo Denki University

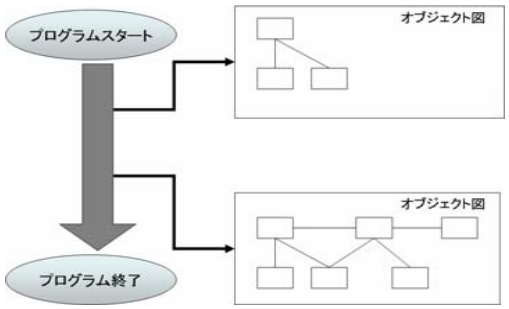

図 1　オブジェクト指向プログラムの実行過程の可視化

2. オブジェクト指向プログラムの実行可視化システム

オブジェクト指向プログラムにおいて，オブジェクト同士の関係及びオブジェクトの状態は，それぞれのオブジェクトの属性値で示され操作（メソッド）によって管理されている．メソッドによって属性が変化しオブジェクトが造られる．よって，対象プログラム内の属性の変化およびメソッドの実行を監視する事により，オブジェクト指向プログラムの動作について情報を得ることができる．

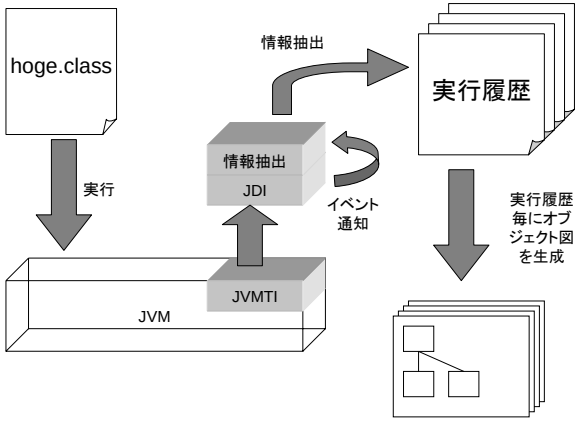

図 2　システムの構成図

属性の変化及びメソッド実行情報を実行履歴として記録し，実行履歴一件に付き一枚の拡張オブジェクト図生成に必要な情報を作成，これを元にオブジェクト図を作成する．

2.1 システム構成

本システムは対象を java 言語[5]とする．属性の変化及びメソッド実行情報を取得するために実行時情報抽出手段として Java Platform D ebugger Ar chitecture（JPDA）[6]のインターフェイスの一つである Java Debuggin g I nterface（JDI）を使用した．JDI を使用する事により対象プログラムのソースコードを変更することなく Java V irtual M achine（JVM）内部の情報を閲覧・制御できる．システムの構成を図 2 に示す．

2.2 拡張オブジェクト図

UML に規定されているオブジェクト図をベースに拡張した．UML に規定されているオブジェクト図は構成要素としてオブジェクト及びその関連を示すリンクがあるが，Map オブジェクトやコレクションオブジェクト，配列オブジェクトでは key（又は index）と value の対応が重要だがオブジェクト図では考慮されていない．また，ローカル変数及びスタティック属性・スタティックメソッドについても同様に考慮されておらず，メソッド実行情報も無く，本システムにそのまま用いるのには不十分である．

そこで通常のオブジェクトに加え，スタティックメソッドオブジェクト，スタティック属性ブジェクト，配列・コレクション・Map オブジェクトを新たに創り，オブジェクト及びスタティックメソッドにメソッド実行情報を付加した拡張オブジェクト図を規定した．本システムで用いる拡張オブジェクト図の構成要素について以下に述べる．

2.2.1 メソッド実行情報

従来のオブジェクト図のオブジェクトにメソッド実行情報を付加する．メソッド実行情報はローカル変数情報と，そのメソッドを呼び出したメソッド及びそのメソッドが呼び出したメソッド情報を持っている．メソッド実行情報は同一オブジェクト内で複数のメソッドが実行された場合は下に継ぎ足していく．オブジェクト名は〔オブジェクト ID〕とする．図 3 にメソッド実行情報付のオブジェクトの描画を示す．

2.2.2 スタティックメソッド

スタティックメソッドはインスタンスに属さずクラスから呼び出されるため，独立したオブジェクトとして描画する．オブジェクトと区別するために，角を丸くした長方形を用いている．オブジェクトと同様にメソッド実行情報を持つが，インスタンス．変数を持たない．オブジェクトと同様に複数のメソッド実行時には継ぎ足していくまた，オブジェクト名は[メソッド名：クラス名]とする．図 4 にスタティックメソッドの描画を示す．

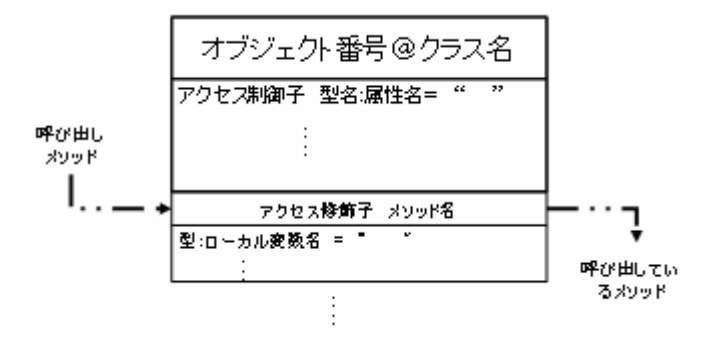

図 3 メソッド実行情報付のオブジェクト

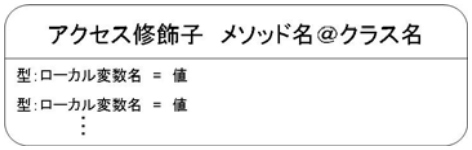

図 4 スタティックメソッド

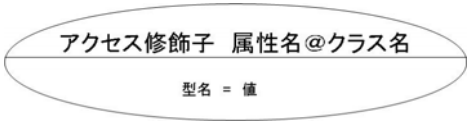

図 5 スタティック属性

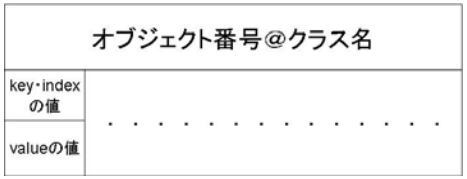

図 6 配列・コレクション・Map オブジェクト

2.2.3 スタティック属性

インスタンスに属さないスタティック属性も，拡張オブジェクト図ではオブジェクトとして扱う．図 5 にスタティック属性の描画を示す．

オブジェクトと区別するため，楕円形にしている．メソッド実行情報は持たない．オブジェクト名は[属性名：クラス名]とする．

2.2.4 配列・コレクション・Map オブジェクト

オブジェクトのように属性記述欄を設けず，key（又は index）と value を記述する欄を設けた．上を key（または index），下を value とする．オブジェクト名は[オブジェクト ID：クラス名]とする．図 6 に，配列・コレクション・Map オブジェクトの描画を示す．

2.3 拡張オブジェクト図の描画例

図 7 は拡張オブジェクト図の描画例を示す．メソッド実行欄にあるそのメソッドを呼び出したメソッド及びそのメソッドが呼び出したメソッド情報を元にし，main メソッドからその時点で呼び出しているメソッドまでをメソッド実行線で結んでいる．また，リンクをオブジェクトからではなく，属性欄の右端を始点として伸ばすことにより，どの属性がどのオブジェクトを参照しているかを判り易くした．また，終点をオブジェクト名の左端に統一している．

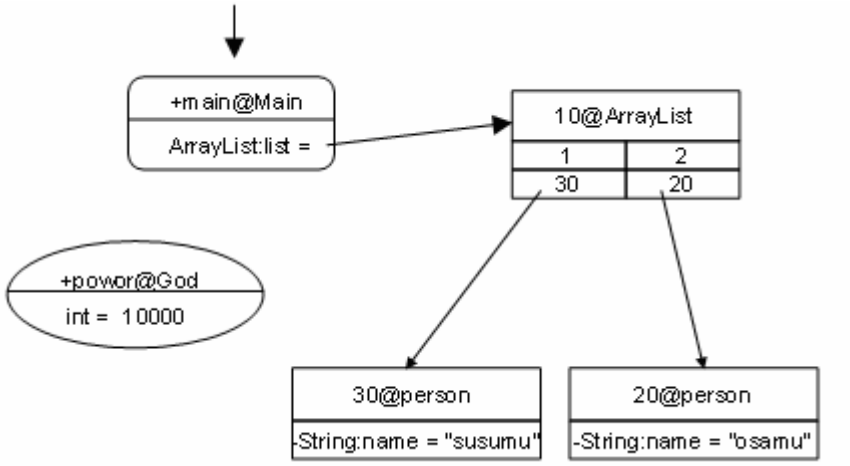

図 7 拡張オブジェクト図の例

3. 実行時情報の抽出

JDI には監視対象の JVM 内で発生した特定の動作（イベント）発生時にこれを検知し JDI を利用するデバッガに通知する機能がある．これを利用し，対象プログラム（を含む JVM）内でイベントが発生したときに情報を取得する事で実行時情報を抽出する．

JDI を使用することによって、アスペクト指向（例えば javassist[7]）等他の手法では取得できない JVM 内のオブジェクト番号 等の必要とする実行時情報が取得できる．

JDI によって監視対象であるプログラムのオブジェクトが変更するイベントを監視し，その時点の JVM 内部の情報及びイベント情報を抽出し一件の実行履歴を作成する．これをプログラムの任意の時点あるいは終了時点まで行う．JDI ではイベント通知を要求しなければ通知を受けられない．あるイベントが発生した場合，発生したスレッドあるいは全てのスレッドを停止させることが出来る．イベント発生時に，JVM 内の情報を正確に取得するため，すべてのスレッドを停止させ，情報を抽出した後でスレッドの動作を再開させる．また，イベント通知自体にそのイベントに関する情報が含まれているので，それらの情報を抽出していく．

以下に取得するイベント及びイベントが発生した時に抽出する情報を記述する．なお，抽出した全ての情報に通し番号であるイベントナンバー，ソースファイル名，そのイベントが発生したソースコード上の行番号をつける．本システム起動時にプログラム引数として対象プログラムのパスを含むクラスファイル名を入力する．また，これらのイベントの収集期間は対象プログラムの main メソッド開始から終了までの間である．対象プログラムが動作する JVM 内では，対象プログラムの main メソッド開始前よりイベントが発生しているが，これらについては情報を抽出しない．

なお，実行時情報をより詳しく抽出するために，デバッグ情報を付加する”-g”オプションをつけコンパイルしたクラスファイルを用いる．これにより，ローカル変数・行番号・ソースファイル名がクラスファイルに付加される．

対象プログラム内で監視し通知を受けるイベントと，実行履歴として出力するイベントを以下に記述する．

- JDI により対象プログラム内で補足するイベント
  - ●クラス読み込みイベント
  - ●メソッド開始イベント
  - ●メソッド終了イベント
  - ●属性値変更イベント
  - ●ブレークポイントイベント
- 本システムが実行履歴として出力するイベント
  - ●クラス読み込みイベント
  - ●（メソッド開始イベント時に出力するイベント）
    - メソッド開始イベント
    - コンストラクタ開始イベント
  - ●（メソッド終了イベント時に出力するイベント）
    - メソッド終了イベント
    - コンストラクタ終了イベント
  - ●属性値変更イベント
  - ●（ブレークポイントイベント処理で発生するイベント）
    - 配列オブジェクト生成イベント
    - 配列要素変更イベント
    - コレクション・Map オブジェクト生成イベント
    - コレクション・Map 要素変更イベント
    - ローカル変数変更イベント

3.1 クラス情報の抽出

JVM がクラスをロードした際に監視対象となるプログラムのクラスの場合，そのクラスに含まれるメソッド実行情報および値変更の監視設定を行う．また，3.4 で述べるコレクションなどの要素の値変更を抽出するために、クラス内全ての行にブレークポイントイベントを設定し通知要求をする．以下にクラス情報抽出時に抽出する情報を記述する．

- ●クラス番号
- ●クラス名
- ●スーパークラス名
- ●実装インターフェイス名
- ●（スーパークラスを含めた）実装インターフェイス名
- ●サブクラス名
- ●属性（属性番号・属性名・属性の型・スタティックの肯否）
- ●メソッド（メソッド番号・名前・引数の型・引数名・スタティックの肯否）
- ●アクセス制御子

3.2 属性値変更情報の抽出

オブジェクトの属性の変化を捉える．その際に以下の情報を抽出する．また，配列・コレクション・Map の型を持つ属性に対し変化があった場合，そのオブジェクトのオブジェクト番号と要素を変化検出用リストに記録しておく．記録に無いオブジェクト番号があった場合，その配列・コレクション・Map オブジェクトが生成されたものと判断し配列オブジェクト生成イベント/コレクション・Map オブジェクト生成イベントを出力する．また，記録にあるオブジェクト番号がありかつ要素が変更されていた場合配列要素変更イベント/コレクション・Map 変更イベントを出力する．出力する情報については 3.4.1 節を参照のこと．

以下に属性値変更情報抽出時に抽出する情報を記述する．

●属性番号
●属性名
●属性の型
●属性値
●アクセス制御子
●final 修飾子の有無

3.3 メソッド実行情報の抽出

対象プログラムのクラスに含まれる，コンストラクタを含むメソッドの開始及び終了を捕える．なお，メソッド開始時にはローカル変数情報（変数番号・変数名・変数の型）及びその値を，ローカル変数の変化を捉えるためにローカル変数スタックに抽出し記録しておく．（例えばメソッドの引数のように）既に値があった場合ローカル変数変更イベントを出力する．出力する情報については 3.4.2 節を参照のこと．

以下にメソッド実行情報抽出時に抽出する情報を記述する．

●開始時
- メソッド番号
- 実行されたメソッドを含むクラス番号
- 実行されたメソッドを持つオブジェクト番号
- 引数
- クラス初期子の肯否
- メインメソッドの肯否

●終了時
- メソッド番号
- 戻り値の値
- スタティック初期化子の肯否
- メインメソッドの肯否

また，メソッドがコンストラクタの場合以下の情報を抽出する．

●開始時
- メソッド番号
- 作成されたオブジェクト番号
- 作成されたオブジェクトのクラス番号
- コンストラクタの引数

●終了時
- メソッド番号

3.4 コレクション等の要素値変更及びローカル変数値変更情報の抽出

配列・コレクション・Map オブジェクトの生成及び要素の値の変化，ローカル変数の値の変化を検出し，その時点での情報を抽出し実行履歴に出力する．JDI に配列・コレクション・Map オブジェクトの要素値変化及びローカル変数の変化を検出する機能は無い. 以下にそれぞれについて変化を検出及び情報抽出する方法について述べる．

配列・コレクション・Map オブジェクトについては属性値変更イベント時に取得した情報及びブレークポイント時に検出し，ローカル変数変化情報についてはメソッド開始情報から生成されたローカル変数情報及びその値とブレークポイント時に検出する．

3.4.1 配列・コレクション・Map オブジェクトの要素値変更情報の検出

配列・コレクション・Map オブジェクトの生成及び属性の変化を検出するフローを図 8 に示す．配列については JDI の要素に対し取得・操作できる機能を用いる．コレクション・Map オブジェクトについては，JDI の対象オブジェクトの要素を取得するメソッドを実行しその戻り値を得る機能を利用し，戻り値から要素を取得する．

コレクションオブジェクト（Collection インターフェイスを実装しているクラス由来のオブジェクト）はその要素を配列で返す toArray メソッドを実行することにより，その戻り値として要素を含んだ配列オブジェクトを得ることができる．

Map オブジェクト（Map インターフェイスを実装しているクラス由来のオブジェクト）はその要素を戻り値として key と value を含む Entry オブジェクトの配列オブジェクトを返す EntrySet メソッドを用いる．この配列オブジェクトより要素である Entry オブジェクトを取り出し, そのオブジェクトに対し getKey メソッド及び getValue メソッドを実行させる事で Map オブジェクトより要素（key 及び value）を得ることができる．

それぞれのオブジェクト番号とそれらの要素（key 又は index と value）を記録して置き，ブレークポイント毎に記録されたオブジェクト番号に相当するオブジェクトに対しその要素が変更されていないか検査している．変更があった場合，また，記録に無い配列・コレクション・Map オブジェクトが検査中にあった場合これを変化検出用リストに記録する．また，要素値変更があった場合は配列要素変更イベント/コレクション・Map 要素変更イベントを，記録に無いオブジェクトがあった場合にはコレクション生成イベント/Map 生成イベントを実行履歴へ出力する．

図 8 の変化検出用リストは，属性値変更イベント時に配列・コレクション，Map オブジェクト型の属性だった場合その値及び要素を上書きされる．

以下に配列オブジェクト生成イベント，配列要素変更イベント，コレクション・Map オブジェクト生成イベント，コレクション・Map 要素変更イベント発生時に出力するイベント情報を記述する

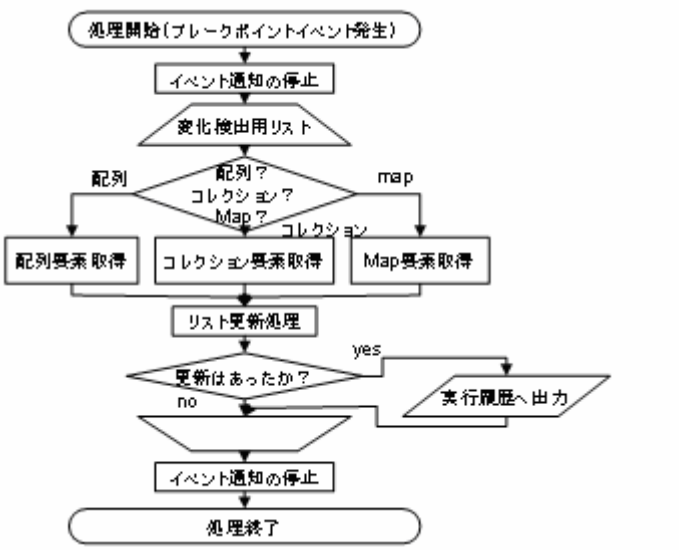

図 8　配列・コレクション・Map オブジェクトの生成及び要素値変化の検出

●配列オブジェクト生成イベント時
- オブジェクト番号
- 配列の型
- 配列のサイズ
- 要素リスト

●配列要素値変更イベント
- オブジェクト番号
- 変更後の要素リスト

●コレクション・Map オブジェクト生成イベント
- オブジェクト番号
- クラス名
- コレクションか Map かのフラグ
- 要素（key(又は index)と value）のリスト

●コレクション・Map 要素変更イベント
- オブジェクト番号
- 変更後の要素（key(又は index)と value）のリスト

コレクション情報取り出しのために対象プログラム内のメソッドを実行するには，そのメソッドを持つスレッドを動かさなくてはならない．その時にイベントが発生すると，イベント発生時にスレッドを停止させるためデッドロックが発生する．これを回避するために，対象メソッドを呼び出す前に全てのイベント発生を止め，呼び出した後にイベント発生を再開させている．

3.4.2 ローカル変数の値変更情報の検出

JDI ではローカル変数にアクセスする機能が提供されているため，これを用いローカル変数の値の変化を検知する．検知にはメソッド開始時に記録した変数情報を用い，メソッド開始時及びブレークポイント時に検査する．ここで言うローカル変数とはメソッドの引数及びそのメソッド内で宣言されたメソッドスコープの変数である．

ローカル変数を管理するためにスタックを用意し，メソッド開始時にローカル変数情報群を一つにしスタックにプッシュし，メソッド終了時にポップする．一番上のローカル変数情報群が常にその時に実行されているメソッドのローカル変数情報が収められている．ブレークポイントの度に peek（一番上のオブジェクトを取り出すが削除しない）し，そこに収められているローカル変数について検査する．JVM 内（のそのローカル変数を持つメソッドが含まれるスレッド）からローカル変数番号を用い最新の値を抽出し，記録されているローカル変数番号と値と照合，変更があった場合ローカル変数値変更イベントとして出力する．以上の動作を図 9 に示す．

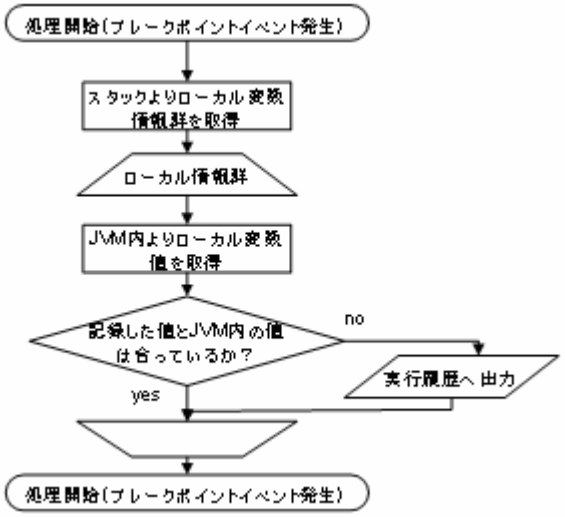

図 9　ローカル変数の値変更情報の検出フロー

4. オブジェクト図の生成

4.1 オブジェクト情報の生成

実行履歴を元に，一件の実行イベントに対して一枚のオブジェクト図を描画できるようにオブジェクト情報を作成する．オブジェクト情報作成の概要を図 10 に示す．オブジェクト情報の生成により，オブジェクト図を検索あるいはフィルタリングにかけることができる．ある一件の実行イベントに対応するオブジェクト情報を作成するとき，それまでの実行履歴を全て考慮して作成する．オブジェクト情報は以下の情報を含んでいる．

- ●イベントナンバー
- ●イベントが起きたクラスファイル名
- ●イベントが起きた行番号
- ●イベントが起きたオブジェクト番号（またはメソッド番号）
- ●クラス情報
- ●オブジェクト情報
  - オブジェクト番号
  - クラス番号
  - 属性情報（属性番号・値）
  - 実行メソッド情報（メソッド番号・ローカル変数情報・戻り値・呼び出しメソッドの番号・呼び出しメソッドを持つオブジェクト番号・呼び出されているメソッド番号・呼び出しているオブジェクト番号）
- ●スタティックメソッドオブジェクト情報
  - メソッド番号
  - クラス番号
  - 実行メソッド情報（オブジェクト情報のものと同一）
- ●スタティック属性ブジェクト情報
  - 属性番号
  - クラス番号
  - 属性情報（属性番号・値）
- ●配列・コレクション・Map オブジェクト情報
  - オブジェクト番号
  - クラス名
  - 要素

まず main メソッドのメソッド実行イベントよりオブジェクト情報を生成し，これに次のイベントを適用し，そのイベントのオブジェクト情報を生成する．これを任意の時点でのイベントまで行う．このように，実行履歴内のイベント一つに付オブジェクト情報を一つ生成していく．

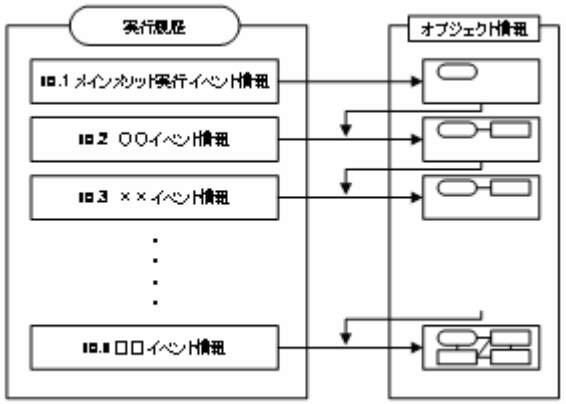

図 10　オブジェクト情報の生成

4.2 オブジェクト図の描画

オブジェクト情報を元にオブジェクト図を描画する．図 11 に本システムの描画結果を示す．左ペインは実行履歴を用い，main メソッド実行を根ノード，メソッド実行/コンストラクタ実行を内部ノード，その他イベントを葉ノードのツリー構造でプログラムの実行過程を表している．これを用い，プログラムの任意の時点を指定し，その時点でのオブジェクトの状態を右ペインに表示させる．これは，デザインパターンのサンプルプログラム[8]のうち，コンポジットパターンのプログラムに適用した結果を示す．この場合，実行ステップは 427 件であった．

5. 今後の課題

5.1 コレクション及び Map オブジェクトからの要素取得方法の改良

コレクション及び Map オブジェクトから要素を取得する際に対象オブジェクトのメソッドを実行しているが，これにはイベント発生に伴い停止していた対象オブジェクトを含むスレッドを動かさなければならない．スレッド動作最中になんらかのイベントが発生しスレッドが止まる場合，本システムは対象オブジェクトからメソッドの帰りを待ち対象スレッドは本システムによる再動作を待つデッドロックがかかってしまう問題があるためイベント通知要求を全て解除しなければならない欠点がある．そ

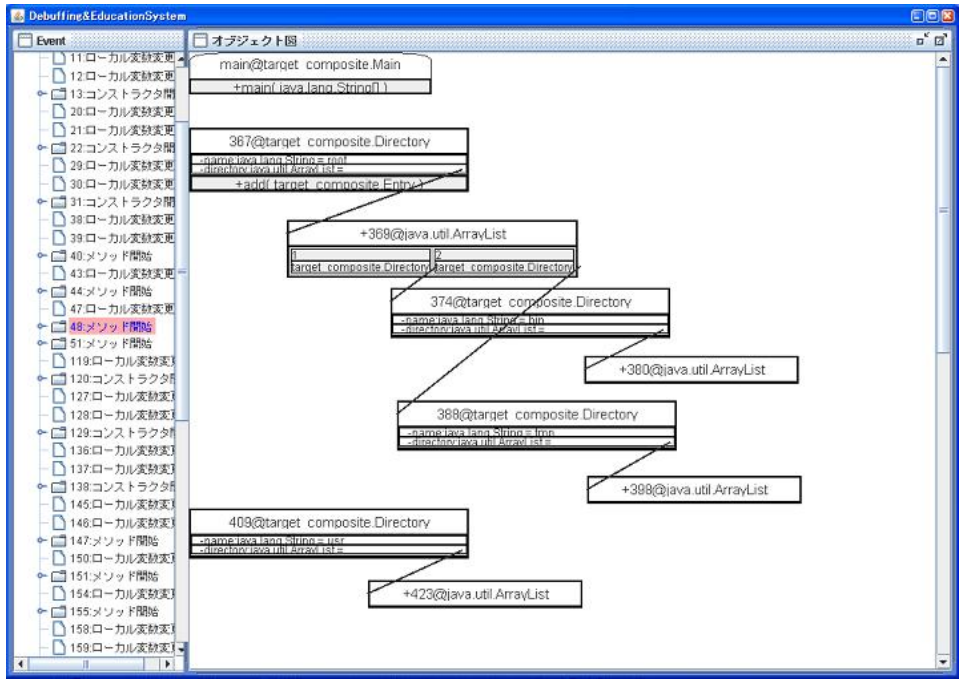

図 11　オブジェクト図の描画

のため，いくつかのイベント（及び其処で検知されるはずだった変化）を見逃す可能性がある．また，ブレークポイントの設定及び解除に対象プログラムの起動からオブジェクト図の生成まで時間の９０％ 以上を占めるオーバーヘッドが確認された. 新たなコレクション及び Map オブジェクトからの要素取得を研究したい．

5.2 例外への対応

本システムでは例外処理を考慮していない．例外処理はしばしばプログラムの働きにおいて重要な役割を果たしているし，また実行時エラーを明示的に表している．このためデバッガとしての機能を高めるために必要であると考える．JDI では例外発生イベントを検知する機能があるため，今後は例外への対応をしていきたい．

5.3 実行時値書き換えへの補助ツールとしての活用

JDI ではプログラム実行中の属性の書き換えをサポートしている．プログラムの破壊が可能になるものの，デバックやテストに強力な機能となると考える．任意の時点でプログラムを止め，その時点での拡張オブジェクト図を通し属性の書き換えを行う事により，実行時の値を書き換えられるツールとして改良できると構想している．

5.4 必要な情報のみを抽出

本論文では全てのイベント（及びそれに付随する情報）を抽出し，全てのイベントについてオブジェクト図を作成している．しかし，それらの情報は膨大となり本当に見たい情報が埋もれてしまう可能性がある．あまり重要で無いイベント（例えば if 文ループを制御するカウンタの増減イベントやオブジェクト同士の関係及びオブジェクトの状態変化を起こさないメソッドの実行イベントなど）を排除し，必要な情報を抽出できる様，特定のクラス名や属性名をキーとしてオブジェクト図を検索或いはフィルタリングする機能を実装したい．

5.5 描画レイアウト方法

現在は main メソッドを頂点とする，オブジェクトの関係について左から右へのツリー構造としてオブジェクト図を自動配置しているが，これがオブジェクト図の表現方法として適切かどうか議論する余地がある．オブジェクト図の自動配置については研究されている[9]ため，これを参考にし，更に見やすいオブジェクト図の自動配置をしていきたい．

6. おわりに

本論文ではオブジェクト指向プログラムの動作をオブジェクト図を用い可視化する手法の提案・実装を行った．このシステムを用いることで，オブジェクト指向プログラムの動作が解りやすく可視化ができ，デバッグやプログラミング教育を行いやすくなると考える．今後の予定としては，統合開発環境への組み込みや，実際にユーザに使っていただきアンケートをとりたいと思っている．

参考文献

1) UML, http://www.uml.org/
2) insight,http://sources.redhat.com/insight/
3) 谷口孝治他，プログラム実行履歴からの簡潔なシーケンス図の生成手法，コンピュータソフトウェア Vol.24, No.3, pp153-169, 2007
4) eclipse,TPTP http://www.eclipse.org/tptp/
5) java ,http://java.com/ja/
6) jpda ,http://java.sun.com/javase/ja/6/docs/ja/technotes/guides/jpda/index.html
7) javassist, http://www.csg.is.titech.ac.jp/~chiba/javassist/
8) 結城浩，Java 言語で学ぶデザインパターン入門，ソフトバンクパブリッシング，2004.
9) 中島 哲，田中 二郎，オブジェクト指向方法論に基づくオブジェクト図の自動レイアウト，情報処理学会論文誌 39 (12), 3282-3293，1998.