6L-06

オブジェクト図のアニメーション

山崎　翔[1]　　久保田　吉彦[1]　　紫合　治[1]

東京電機大学 情報環境学部[1]

1 はじめに

オブジェクト指向のプログラムを理解する際に，インスタンスの動きを理解することは重要である．インスタンスの状態を表現するならばオブジェクト図が最適であり，それを astah*UML[1]などのモデリングツールを用いて描くことができる．また UML 自動生成ツール[2][3]を用いることでプログラムからオブジェクト図を自動生成することもできる．

しかし，それらで描かれるオブジェクト図はある時点での状態を抜き出した静止画であり，インスタンスの変化を表現し，理解の助けにするのは難しい．そこで我々は Java 言語を対象としてプログラム実行時に発生するイベントからデータを抽出し，それをもとにオブジェクト図の変化をアニメーション的に表現するツールの開発を行った．

2 システム構成

本ツールは大きくデータ抽出部とアニメータ部の２つに分けることができる．その構成を図１に示す．

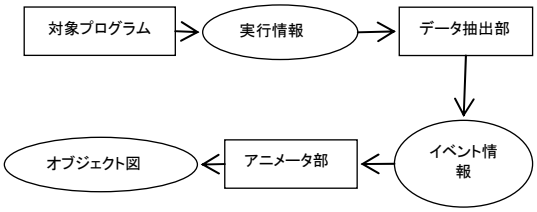

図１ 全体の構成

データ抽出部では Java 言語で記述された監視対象のプログラムから実行情報を抽出し，そこからアニメーションに必要な情報を抽出してアニメータ部へ送る．

アニメータ部ではデータ抽出部から送られてくるイベント情報(バッチ式ではテキストファイル，リアルタイム式ではストリーム)をもとに処理を行い，オブジェクト図の描画をする．描画には processing[4]を利用した．

3 プログラム実行情報の抽出

Java言語で記述されたプログラムを実行情報の監視対象とし，Java Platform Debugger Architecture を使用して実行情報を抽出する．

Object Diagram Animation
Sho Yamazaki[1] Yoshihiko Kubota[1] Osamu Shigo[1]
[1] School of Information Environment, Tokyo Denki University

監視対象のプログラムを実行している Java Virtual Machine(JVM)の Java Virtual Machine Tool Interface(JVM TI)へ Java Debugging Wire Protocol(JDWP)を通して接続し，Java Debugging Interface(JDI)から渡されてくるイベントオブジェクトからアニメーションに必要な情報を抽出する(図２)．

監視対象の JVM はシステムのクラスやオジェクトに関係するイベントも通知する為，あらかじめ監視対象の JVM にフィルタを設定し，監視対象のプログラムで定義されたクラスとそのクラスのオブジェクトに限定してイベントを通知させる．

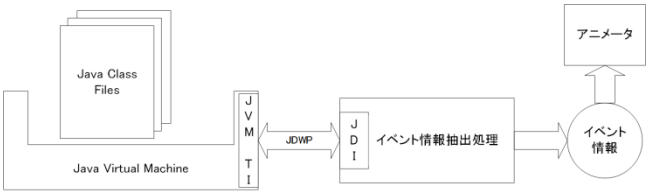

図２システム構成図

イベント情報抽出処理ではアニメーションに必要なメソッド呼び出し時・メソッド終了時・フィールドの値の変更時の情報をそれぞれ JDI のイベント(MethodEntryEvent,MethodExeitEvent, ModificationWatchpointEvent)からを文字列として抽出する．

各イベントからは共通して下記の情報を抽出する．

- クラス名
- メソッド名
- メソッドの引数
- オブジェクト ID
- スレッド名
- スレッド ID

MethodEntryEvent, MethodExitEvent からは以下の情報を追加抽出する．

- メソッドの引数の型・名称・値
- オブジェクトのフィールド変数の型・名称・値

ModificationWatchpointEvent からは以下の情報を追加抽出する．

- 変更前/変更後のフィールド変数の型・名称・値

メソッドの引数とフィールド変数については値がプリミティブ型と String 型については型・変数名・値を取得し，監視対象のプログラムで定義されたクラスのオブジェクトについてはオブジェクト ID のみを取得する．変数が ArrayList と HashMap の場合，要素を抽出し同様の処理を行なう．

4 アニメータ

4．1 処理の流れ

アニメータ部の処理の流れは図３に示す．

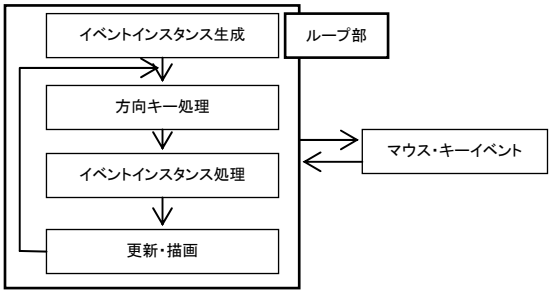

図３ アニメータ部の処理の流れ

まず，データ抽出部から送られてくるイベント情報をもとにメソッドの開始・終了・フィールド変更のイベントインスタンスを生成する．次に方向キーによる画面移動の処理を行う．イベントインスタンスの処理では前イベントによるアニメーションの変化が終了してからこの処理へ入るたびにカウンターを回し，その値が待ち時間を定める変数の値を超えたら次のイベントインスタンスを取り出して種類に応じた処理を行う．そして最後に画面の描画を行う．

イベントインスタンスの生成はデータ抽出部から情報が送られてきたときのみ行われ，通常時はそれ以外の３つのみでループする．また，4.4 節で挙げる操作が行われるとマウス・キーイベントの処理が行われてからループ部へと戻る．

4．2 レイアウトと関係線

図４に本ツールによる描画結果を示す．

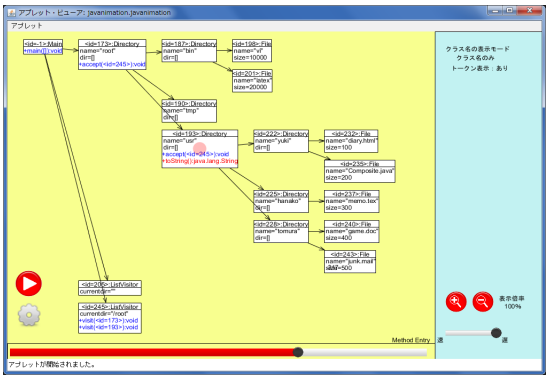

図４ オブジェクト図の描画

本ツールでは基本的にインスタンス，または static メソッドを持つクラスがオブジェクト図で描画され，それぞれがフィールドに持っているインスタンスに対して矢印をつなぐ形でツリー構造に図をレイアウトしている．しかし，あるインスタンスが複数のインスタンスに所有される場合はメッシュ構造に表示の形が切り替るようになっている．また，ArrayList や HashMap などはオブジェクト図が複雑化することを避けるために表示せずに，直接それらに格納されているインスタンスへと矢印をつないでいる．インスタンスの追加などによって親子関係が変更された場合は自動的にレイアウトも調整される．

4．3 メソッドの表現

メソッドの実行によってオブジェクト図が変化していく様子を表すために実行中のメソッドをフィールドに追加する．さらに，最も新しい実行メソッドは赤文字で表示し，赤い球(トークン)を表示している．それによってトークンを追いかけていくことでプログラム実行の流れを把握できる．そしてフィールドの変更やメソッドの実行開始・終了に合わせて各オブジェクト図のサイズも自動で調整される．

4．4 設定機能

本ツールにはいくつかユーザの手で操作できる部分がある．まず，画面左下に再生・停止ボタンと設定ボタン．下には現在の再生状態を示すシークバー．シークバーをドラッグによって任意の位置に移動させることで巻き戻しなどが行える．設定ボタンを押すことで開く設定画面（図４の水色部分）にはクラス名・トークンの表示，拡大・縮小，再生速度・待ち時間を自由に変更できる．さらにキー操作によってアニメーションを最初へ戻す，スクリーンショットの撮影，方向キーのよる図全体のスクロール移動を行うことができる．

5 おわりに

本稿では，プログラム実行時に発生するイベントからデータを抽出し，それをもとにオブジェクト図をアニメーション的に表現するツールの開発を行った．今後は，シーケンス図を表示することでプログラム全体の流れを把握しながらオブジェクト図を見ることができるようにし，操作・表現面での改良を行うことでさらに利便性や理解のしやすさを高めていきたい．

参考文献

[1] astah*UML, http://astah.change-vision.com/ja/
[2] Paul Gestwicki, Bharat Jayaraman: “Methodology and Architecture of JIVE,” SoftVis ’05, 2005/5.
[3] 中原 進，紫合 治: “オブジェクト指向プログラムの動作の可視化,” 情報処理学会ソフトウェア工学研究会報告 No.SIGSE167, 2010/3.
[4] Processing, https://processing.org/