Code Oriented Diagram Editorを用いた並行バグの可視化

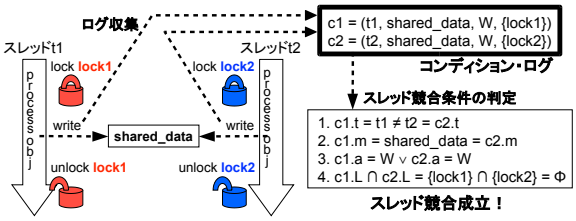
大村 裕・荒堀喜貴・権藤克彦 ・渡部 卓雄
東京工業大学

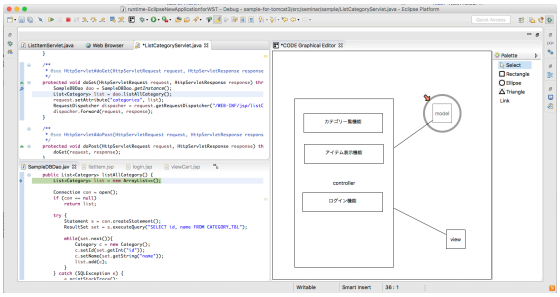
ソースコードリーディング

既存ソフトウェアの改善・修正となる保守タスクでは既存ソフトウェアの理解のためにソースコードを読む必要がある．このソースコードリーディングの支援のための様々な研究が存在する．特に，可視化によるソースコードリーディング支援の研究としてCode bubbles [Bragdon, A. ICSE'10] やSoftwareReflectionModels [G.C.Murphy et al. SIGSOFT'95]等が存在する．

競合の動的検出

ソフトウェアの競合を実行時情報による検出する手法が存在する．この手法では各メモリアクセスごとにアクセス・コンディションのイベント履歴として収集し，同時にスレッドの競合条件の成否を判定して行う．





Webアプリケーションとユーザ作成ダイアグラムが同期して実行している様子

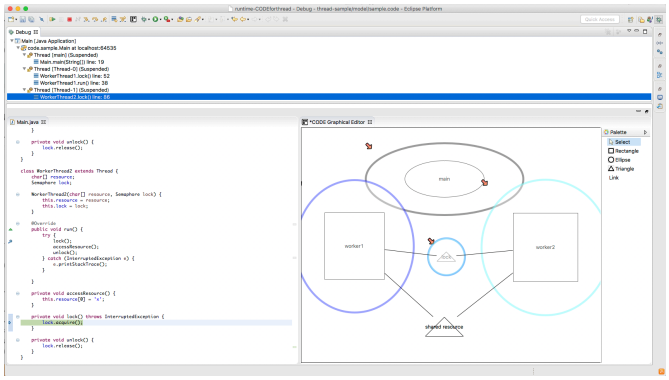
CODE(Code Oriented Diagram Editor)

開発者は様々な視点とソースコードを同期しながら，ソースコードを読み進めていく必要がある．しかし，開発者自身がその同期を維持・管理していくのは効率的ではない．そこで我々はユーザが作成したダイアグラムとソースコードのマッピングを維持・管理し，プログラム実行時に実行箇所をダイアグラム上で表示することでソースコードリーディングの支援を行う環境(CODE)を開発した．

RaceTrace

既存の動的競合検出ツールは複数スレッドのイベント履歴の過度な簡約により検出精度が低いという問題がある．
そのため，詳細な履歴情報を記録するRaceTraceを開発した．本ツールでは拡張GNU Cコンパイラを利用し，抽象構文木のメモリ操作部に境界管理/検査機構を挿入することで実行時に詳細なイベント履歴を記録する．競合検出結果は詳細でプリミティブなメモリアクセス列を列挙したものとなる．

```
yuu — yuu@cobuntu: ~/macos/src/vp/eval/datarace — ssh cobuntu — 80×24
yuu@cobuntu:~/macos/src/vp/eval/datarace$ ./eval_datarace_region_heap
ERROR (DATA RACE (thread)):
  access: loc="eval_datarace_region_heap.c:23:3 (process_obj)",
          addr=[0x025ea5a0, 0x025ea5a0],
          tid/iid=67/1,
          tatype/iatype=write/1,
          lockset/intmask={0x00000043, 0x00601860}
  object: loc="eval_datarace_region_heap.c:32:3 (main)"
          addr=[0x025ea5a0, 0x025ea5bf], owner=00000000
  subobj: addr=[0x025ea5a0, 0x025ea5a0], owner=00000065
    access conditions (thread):
    cond: loc="eval_datarace_region_heap.c:23:3 (process_obj)",
          addr=[0x025ea5a0, 0x025ea5a0],
          tid/iid=66/1,
          tatype/iatype=write/1,
          lockset/intmask={0x00000042, 0x006018a0}
    access conditions (interrupt):
    cond: loc="eval_datarace_region_heap.c:38:3 (main)",
          addr=[0x025ea5a0, 0x025ea5a0],
          tid/iid=1/1,
          tatype/iatype=write/1,
          lockset/intmask={}
yuu@cobuntu:~/macos/src/vp/eval/datarace$
```



CODEによる並行バグの可視化

上記の通り，並行バグを理解するには出力結果の実行の流れをソースコード上で理解する必要がある．我々はCODEに以下の機能を追加することでVPの出力結果の可視化およびソースコードとの連携を提案する．

・マルチスレッド対応
・VPの出力結果を入力とした対象プログラム制御