Progress on CAFFEINE

Yann-Gaël Guéhéneuc

File started May 13, 2002
Copy of October 14, 2003
Priorities: 7, 8

Fixed bugs and implemented features The following list points out the bugs and missing features in CAFFEINE, and the corrections made. (The list is given in reverse-chronological order).

1. Rename the packages to replace *fr.emn* with *caffeine* for consistency.

   2003/08/20 ▶ Done!

2. Convert the tests into JUNIT *real* test-cases!

   2003/08/15 ▶ I heavily refactored CAFFEINE to make it a Singleton and to add an Observer. Then, building the tests what just a matter of rewriting the existing test-cases into JUNIT test-cases.

3. Improve performances with respect to the remote loading mechanism (which uses JAVASSIST [1]).

   2003/07/30 ▶ The load-time byte-codes modification are now performed only if required. In particular, the extra `finalize()` methods are added only if needed, the `System.exit()` method-calls are intercepted only if needed... This dramatically improves performances!

4. The `OutputMonitor` generates a superfluous carriage return.

   2003/07/30 ▶ The problem was that the flow contains carriage returns. These carriage returns are now skipped for a cleaner output.

5. Implement a trace-generation simulator. The simulator shall improve the performances when testing Prolog code...

   2002/06/03 ▶ The `caffeine.monitor.simulator` package contains the needed classes to perform a simulation of a Java program execution. The simulation requires a trace file and a rule file. The trace file must contains event as data string (see `caffeine.monitor.logic.Event.toData()`). There are no simple way (right now) to generate such a trace file, the generation code must be added and removed by hand from the `EventManager` source code.

2002/06/23 ▶ The `Caffeine` class proposes a new `run` method that takes as parameter a file name and a boolean, for the finalizations.

6. Evaluate the performance when setting dynamically JDI filters.

   2002/06/03 ▶ It is now possible to set up dynamically the filter when requesting for the next event. (See `nextEvent/*` predicate definitions in the `Caffeine/Monitor/Logic/Rules.pl` file.) This improves dramatically the performances of the event generation mechanism (up to a 3-factor!).

7. Clean up the `CaffeineMethodReturnedValueWrapper` and `Loader` classes to remove the code about `caffeineUniqueExit`, because I perform the detection of a program exit by monitoring the relevant threads.

   2002/06/23 ▶ This is actually no true... See the ASE'02 article for details on program end in Java.

8. Remove the negative filter list parameter because it is not needed anymore!

   2002/06/03 ▶ Done.

9. Rename packages to make them consistent.

   2002/06/03 ▶ Done.

10. Remove class `CaffeineMethodReturnedValueWrapper` and add its methods directly to the main class being analyzed. This shall improve the performances.

    2002/05/22 ▶ I need to keep this class because of a bug (one more...) in the JDI. I improve the performances by *renaming* the package to include the class into a monitored package.

    2002/06/03 ▶ I kept the class and I don't rename it anymore. I now create a dedicated method exit request when required.

11. The `Caffeine.query(Class, String)` method did not consult properly the query file.

    2002/05/24 ▶ The method now uses a `StringBuffer` for the query file.

12. I need to have an event for the program end.

    2002/05/24 ▶ The `programEnd` event indicates that the program just terminated; The JVM may still run for a while to process the finalizers. I catch this event by monitoring the system threads `Signal Dispatcher` and `AWT-Windows`.

13. The events defined in JIProlog [2] include useless parameters. Need to define within Prolog predicates with just the right number of parameters each.

2002/05/20 ▶ The `Rules.pl` Prolog file now convert the events `<name of the event>0` into `<name of the event>` with the only appropriate parameters.

14. The `EventManger` builds an erroneous constructor event when encountering a static initializer.

    2002/05/15 ▶ The problem was the test to distinguish between method event, constructor event, and finalizers: I tested only the first char of the method name, *i.e.*, '<'. A static initializer as for name '<clinit>', thus I generated a constructor event for it. I now test the complete names for constructor and static initializers.

Bugs and missing features The following list points out the bugs and missing features in CAFFEINE.

1. Verify that CAFFEINE resists to un-handled exceptions. (Especially with regards to the trick to get the finalizers in the right order, on exit.)
2. The method entry and exit events must include the types of formal parameter and possibly the parameter values, to distinguish among polymorphic methods.
3. Add an interface to keep all constants in one unique (and clean) place!
4. Implement in the `EventManger` a filter mechanism at the method level?
5. Manage deferred composition relationships, *i.e.*, composition relationships that exist through a container class, such as `Vector`.
6. The `EventManger` does not handle static field correctly (the `accessWatchpointEvent.object()` method returns `null`).
7. Improve performances by replacing `java.util.Vector` by `fr.emn.caffeine.remote.Vector` only for fieldAccess-related classes.
8. Improve predicate `updateEXProperties/7` in the `Composition.pl` file.

References

[1] Shigeru Chiba. Javassist – A reflection-based programming wizard for Java. In Jean-Charles Fabre and Shigeru Chiba, editors, *proceedings of the OOPSLA workshop on Reflective Programming in C++ and Java*. Center for Computational Physics, University of Tsukuba, October 1998. UTCCP Report 98-4.

[2] Ugo Chirico. JIProlog, April 2002.