

CLAIMS

1. A trace replay execution control system, comprising:
 - a processor;
 - a memory in operable communication with the processor;
 - a trace recorded during an execution of a program in conjunction with a runtime and a kernel, the trace including trace entries which include a record of activity by native code instructions which executed in at least one thread of the program;
 - a source code of at least a portion of the program;
 - a source-intermediate mapping which maps between the source code and an intermediate representation of the source code;
 - an intermediate-native mapping which maps between the intermediate representation and the trace; and
 - an execution controller in a software tool, the execution controller configured to upon execution with the processor control a replay execution of the trace in response to requests, in which the source code is correlated with the trace entries through the source-intermediate mapping and the intermediate-native mapping.
2. The trace replay execution control system of claim 1, wherein the execution controller is configured to translate requests for at least the following operations into corresponding performance of trace native code instructions:
 - a backward execution step-in operation,
 - a backward execution step-out operation, and
 - a backward execution step-over operation.
3. The trace replay execution control system of claim 1, wherein the system is configured to step into a part of the source code which during the replay execution has not yet been compiled into native code instructions.
4. The trace replay execution control system of claim 1, wherein the intermediate representation comprises at least one of the following: a symbol table which correlates at least two identifiers used in the source code with data types, an abstract syntax tree derived from at least a portion of the source code, or a translation into an intermediate language of at least a portion of the source code.
5. The trace replay execution control system of claim 1, wherein the trace does not contain any executing instance of the runtime, in that the system is not configured

to call into the trace to execute any code that is specific to the runtime.

6. The trace replay execution control system of claim 1, further comprising a live process of the program with a callable instance of the runtime, wherein the trace is recorded from the live process and the system is configured to replay execution of the trace.

7. The trace replay execution control system of claim 1, wherein the system is configured to skip over replay execution of a portion of the trace, in that the execution controller moves the replay execution from a first execution time point of the trace to a second execution time point of the trace without the system performing trace native code instructions that would have been performed by continued execution from the first location to the second location.

8. A trace replay execution control method, comprising:
identifying a traced managed program, namely, a program which was configured to execute in conjunction with calls into a runtime and configured to be traced;
receiving a trace replay execution control request, the trace replay execution control request received in a software tool running on a computing system;
based on the trace replay execution control request, automatically mapping between a source code of the traced managed program and an intermediate representation of the source code, and automatically mapping between the intermediate representation and a trace, wherein the trace includes a record of activities by at least one processor, the activities traced as the traced managed program executed, and wherein the trace does not include any executing instance of the runtime; and
performing trace native code instructions which correspond to the trace replay execution control request, thereby replaying a portion of the trace.

9. The trace replay execution control method of claim 8, wherein the trace replay execution control request identifies a backward execution step-out operation to step out of a routine, and the method comprises setting a breakpoint on a trace native code instruction which has a location that shortly precedes a location of a return address of a call to the routine, wherein a location A shortly precedes a location B in a trace when A is encountered prior to B during forward execution of the trace and there are at most five trace native code instructions between A and B.

10. The trace replay execution control method of claim 8, wherein the source

code contains statements and expressions, and wherein the method comprises setting a breakpoint on a trace native code instruction which is on an edge of a group of all trace native code instructions that are mapped by the method to a given statement or a given expression in the source code, wherein an instruction X of a group of instructions G in a trace is on an edge of G when X satisfies at least one of the following conditions: X is the first instruction of G encountered during forward execution of the trace, X is the last instruction of G encountered during forward execution of the trace, X is the first instruction of G encountered during backward execution of the trace, or X is the last instruction of G encountered during backward execution of the trace.

11. The trace replay execution control method of claim 8, wherein the trace replay execution control request identifies a forward execution step-in operation to step into a user code routine, wherein replay execution of the user code routine includes replay execution of at least one call to a runtime routine, wherein the call to the runtime routine does not have a corresponding call statement or call expression in the source code, and wherein the method comprises at least one of the following:

- setting a breakpoint range which specifies an entire memory range that is

- designated to receive dynamically compiled code; or

- setting a breakpoint to perform a step-over operation on the call to the runtime routine.

12. The trace replay execution control method of claim 8, wherein the trace replay execution control request calls for at least one of the following operations:

- continue trace replay execution forward through the trace until a breakpoint is reached;

- continue trace replay execution forward through the trace until a location in a breakpoint range is reached;

- continue trace replay execution backward through the trace until a breakpoint is reached; or

- continue trace replay execution backward through the trace until a location in a breakpoint range is reached;

- a backward execution step-in operation;

- a backward execution step-out operation;

- a backward execution step-over operation;

- a forward execution step-in operation;

- a forward execution step-out operation;

a forward execution step-over operation.

13. The trace replay execution control method of claim 8, wherein during execution of the traced managed program a maximum number of breakpoints was constrained by a hardware breakpoint constraint that a processor can monitor at most N addresses at once, and wherein the method further comprises setting more than N breakpoints during the replay execution of the trace, thereby exceeding the hardware breakpoint constraint.

14. A computer-readable storage medium configured with data and instructions to perform, upon execution with a processor in operable communication with a memory, a trace replay execution control method, the trace replay execution control method comprising:

receiving a trace replay execution control request, the trace replay execution control request received in a software tool running on a computing system;
based on the trace replay execution control request, mapping between a source code of a traced managed program and an intermediate representation of the source code, and mapping between the intermediate representation and a trace, wherein the trace includes a record of native code instructions which were traced as the traced managed program executed, and wherein the trace does not include any executing instance of the runtime; and
performing trace native code instructions which correspond to the trace replay execution control request or setting a breakpoint on a trace native code instruction, thereby controlling replay of the trace.

15. The computer-readable storage medium of claim 14, wherein the method comprises at least three of the following execution control activities:

controlling execution of the runtime at a high level using machine-level information in the trace, where controlling at a high level means controlling based on a user selection that specifies an identifier or a statement or expression in the source code, and machine-level information means information which is specific to a processor architecture;
debugging both forward execution and backward execution of a process of the traced managed program;
stepping into a call shown in the source code at least in part by setting a large breakpoint range, where a breakpoint range is large when it covers at least a thousand contiguous addresses or instructions in the trace;

mapping to a particular version of native code of a routine when the traced
includes at least two versions of native code of the routine; or
recording trace data from a live process of the traced managed program while
debugging the live process, and then performing reverse execution
debugging on the live process using at least some of the recorded trace
data.