Uses of the Soot Framework

Laurie Hendren
`hendren@cs.mcgill.ca`

June 30, 2006

Introduction

This document briefly outlines known uses of the Soot analysis and transformation framework (`www.sable.mcgill.ca/soot`). This is not an exhaustive list, but has been compiled via documents available from the web and from feedback provided by Soot users. More additions to this list would be very welcome.

Please contact Laurie Hendren at `hendren@cs.mcgill.ca` with a brief description of your use of Soot, along with any relevant URLs and paper references. Updates to existing courses/projects on the list may also be sent to `hendren@cs.mcgill.ca`.

Graduate Courses and Projects

Soot can be used in various ways for graduate courses and graduate projects. It has been used extensively in the Optimizing Compilers course at McGill both for assignments on program analysis and instrumentation, and as the basis of many individual course projects (`http://www.sable.mcgill.ca/-~hendren/621`). Several of those projects have developed into interesting additions to Soot.

Many course instructors at other institutions have also adopted the use of Soot for course assignments and/or course projects. Soot has been used for advanced compiler courses and also for courses on analysis tools for software engineering.

These uses include the following:

- *198:515 - Programming Languages and Compilers I*, Barbara G. Ryder, Rutgers University.
  `http://www.cs.rutgers.edu/~ryder/515/f03/`
- *CMPS 551 (course project)*
  `http://www.cacs.louisiana.edu/~euk4141/report.pdf`
- *CSE 501 - Implementation of Programming Languages*, University of Washington, Craig Chambers.
- *CMPUT 680 - Compiler Design and Optimization*, University of Alberta, Jose Nelson Amaral.
  `www.cs.ualberta.ca/~amaral/courses/680/`
- *CS 8803H - Program Analysis and Testing*, Mary Jean Harrold, Georgia Tech.
  `www.cc.gatech.edu/~harrold/8803/`

- *CS577 - Compiler Construction*, Andrew Tolmach, Portland State University.
- *Proposed Projects*, Paul Kelly, Imperial College London.

  `www.doc.ic.ac.uk/~phjk/StudentProjects/CollectedProjectIdeas2001-2002.html`
- *CSCI-6967 - Program Analysis for Software Engineering*, Ana Milanova, RPI.

  `www.cs.rpi.edu/~milanova/csci6967`
- *CIS 788.A12: Analysis and Testing of Object-Oriented Software*, Nasko Rountev, Ohio State

  `www.cse.ohio-state.edu/~rountev/788/`
- *Principles of Software Development*, Gregory Kapfhammer, Allengheny College.

  `cs.allegheny.edu/~gkapfham/teach/cs290/`
- *CS614 - Theory and Construction of Compilers*, Joel Jones, University of Alabama.

  `http://a1.cs.ua.edu/cs614/`
- *Proposed undergraduate projects*, David Lacey, University of Warwick

  `www.dcs.warwick.ac.uk/people/academic/David.Lacey/projects.html`
- *ViDoC - Visuelles Design optimierender Compiler*, Bernhard Steffen and Oliver Ruthing, University of Dortmund, Germany.

  `ls5-www.cs.uni-dortmund.de/teaching/ViDoC/pg-antrag0304.pdf`
- *Translators II (CIS 801)* , Matthew Dwyer and John Hatcliff, Kansas State University.

  `http://www.cis.ksu.edu/~dwyer/courses/801/`
- *CS762 Compiler Construction II*, Amie Souter, Drexel University

  `http://www.cs.drexel.edu/~souter/cs762/project.html`
- *ECEN 621 Digital Circuits*, Mike Wirthlin, Brigham Young University

  `http://www.ee.byu.edu/ee/class/ee625/`

Research Projects

Soot has been used in a wide variety of research projects and also used for the development of other open source projects. The following list gives some of these projects which include work in traditional compiler analyses, analyses for software engineering, analysis for distributed programs and software verification.

- The Ptolemy project, developed at Berkeley, is a framework for component based design in Java. Particular concentration is on block-diagram languages, with components specified using Java APIs. The Ptolemy project uses Soot to transform these highly generic component specifications into more specialized ones. This allows efficient simulation, and someday, embedded hardware and software implementations.

  `http://ptolemy.eecs.berkeley.edu`

- The Aristotle group, Georgia Tech, uses Soot to implement DUSC, a tool to perforam dynamic updates of classes.

  `http://www.cc.gatech.edu/aristotle/Publications/index.html#icsm02_dusc`

- Bandera was one of the first large projects to use Soot and has been developed for program verification. The goal of the Bandera project is to integrate existing programming language processing techniques with newly developed techniques to provide automated support for the extraction of safe, compact, finite-state models that are suitable for verification from Java source code. The Bandera toolset is designed to be an open architecture in which a variety of analysis and transformation components may be incorporated.

  `http://bandera.projects.cis.ksu.edu/`

- Recently the Santos Laboratory at Kansas State has started the Indus project for a collection of program analyses and transformations to adapt Java programs. These tools are based on Jimple and the Soot framework.

  `http://indus.projects.cis.ksu.edu/`

- The Canvas Project (Component ANnotation, Verification And Stuff) aims to allow the component designer to specify component conformance constraints in a natural (yet still formal) way and to provide automated certification tools to determine whether the client satisfies the component's conformance constraints.

  `http://www.research.ibm.com/menage/canvas/`

- Archie Cobbs has developed an opensource Java virtual machine based on Soot. His project, `jcvm`, is a JVM that converts class files to C source and compiles them with GCC Opensource software.

  `http://jcvm.sourceforge.net/`

- At the University of Maryland, College Park, Ankush Varma and Shuvra S. Bhattacharyya have been using Soot to develop a Java-to-C compiler, and its integration with the Java-based Ptolemy II design environment. The overall objective is to translate dataflow-based models of signal processing systems into efficient C code for embedded processors.

- Mooly Sagiv's group from Tel Aviv have used Soot in order to build two Java-to-TVLA translators. Both translators take as input Java class files and a set of translation rules, and output a set of TVP files, which form the input to TVLA. The difference between the translators is technical and regards the way translation rules are specified.

  The first translator, JFE, developed by Alex Warshavsky was used in the CANVAS project (a project of IBM Watson joint with TAU), which is aimed at verification of certain properties of components, in particular, verifying the absence of concurrent modification of iterators. A paper "Deriving specialized program analyses for certifying component-client conformance" by G. Ramalingam, Alex Warshavsky, John Field, Deepak Goyal, Mooly Sagiv, appeared in PLDI'02. The second translator, J2TVLA, developed by Roman Manevich is currently used to translate JavaCard programs to TVLA in a project joint with Giesecke and Devrient.

- Michael I. Schwartzbach, Anders Mller, Christian Kirkegaard and Aske Simon Christensen at the Univeristy of Aarhus in Denmark have used Soot in three projects. Two of these are the JWIG project (http://www.jwig.org/) and the Xact project (http://www.brics.dk/Xact/), where Soot is used as a frontend in the program analyses. Both of these analyses are concerned with static checking of the validity of dynamically generated XML values. The third project is an analysis for string values (http://www.brics.dk/JSA/). The output of this analysis has the form of a mapping from Soot Values of type String, String array (of arbitrary dimension) or StringBuffer into finite automata, such that the language of the automaton includes all possible runtime values of the given Soot Value.

  `http://www.jwig.org`

  `http://www.brics.dk/Xact/`

  `http://www.brics.dk/JSA/`

- Barbara Ryders's PROLANGS research group at Rutgers University has used Soot for various context-sensitive and context-sensitive points-to analyses, implemented as annotated inclusion constraints that are solved by a specialized version of the BANE constraint solver from UC Berkeley (OOPSLA'01, ISSTA'02). SOOT has also provided the framework used for program fragment analysis applied to Java class analysis of incomplete programs (ICSE'03).

  Recently, SOOT has provided the framework used in exception-catch analysis of Java programs. First, this analysis calculates the 'def-uses' of exceptions due to failed operating system resource requests from a Java program. Second, the analysis instruments the Java program (again using SOOT) so that a run-time fault injection program can be called to simulate the exception and then coverage of the calculated def-uses can be measured. (ISSTA'04)

  `http://www.prolangs.rutgers.edu`

- Eric Bodden led a third year group project at the University of Kent at Canterbury to develop a static analyser for Java code. The user can input sourcecode or class files / jar-archives on which then an analysis will be performed using SOOT. Then the user can query the system e.g. for getting all callers of a specific method. Results are shown using an open source graph display package. This project was then further developed into an opensource project.

  `http://bodden.de/projects/janalyzer/`

- Richard Stahl, at IMEC vzw in Belgium, has used Soot in project on task-level parallelism extraction and optimisation for embedded heterogeneous systems. SOOT is used for analysis of Java applications, especially call-graphs, control- and data-dependence. It is also used for instrumentation of the program with profiler-specific code. Publications include:

  - R.Stahl et al.: Performance Analysis for Identification of (Sub)task-Level Parallelism in Java, Proceedings of SCOPES'03, Vienna, Austria, 2003
  - R.Stahl et al.: High-Level Data-Access Analysis for Characterisation of (Sub)task-Level Parallelism in Java, Proceedings of HIPS'04, Santa Fe, USA, 2004
  - R. Stahl, F. Catthoor, R. Lauwereins and D. Verkest: Design-Time Data-Access Analysis for Parallel Java Programs with Shared-Memory Communication Model, Proceedings of EUROPAR'04, Pisa, Italy

- Alan Donovan, Adam Kiezun, Matthew Tschantz and Michael D. Ernst, MIT, have used Soot in their Jiggetai project, which aims to automate the translation of existing Java to Java 1.5, inferring instantiations of generic types to exploit the new type system.

  Their analysis combines a whole-program (or "most of program") pointer analysis at the bytecode level, using SOOT's jimple IR, and a modular source-level type constraint solving component.

  `http://pag.csail.mit.edu/jiggetai/`

- Gregory Kapfhammer and Mary Lou Soffa, University of Pittsburgh, have used Soot in their research on a family of test adequacy criteria for database-driven applications. Their research on testing database-centric applications resulted in the publication of the paper "A Family of Test Adequacy Criteria for Database-Driven Applications" at ACM SIGSOFT ESEC/FSE 2003. This paper was selected to win the ACM SIGSOFT Distinguished paper award. All of their testing and analysis tools rely upon Soot.

  `http://cs.allegheny.edu/ gkapfham/research/diatoms/`

  Gregory Kapfhammer has also authored a chapter on "Software Testing" chapter in the CRC Press Computer Science Handbook where he specifically mentions the Soot program analysis framework in his discussion of "Program Building Blocks".

- Walkinshaw, Roper and Wood, from Strathclyde University in Glasgow have developed a new intermediate representation, the Java System Dependence Graph (JSysDG), based on Soot.

  `www.cis.strath.ac.uk/~nw/JSysDG.ppt`

  N. Walkinshaw, M. Roper, and M. Wood. The Java System Dependence Graph. In SCAM, 2003.

- Another use of Soot for verification comes from Lars-Ake Fredlunk of the Swedish Institute of Computer Science, Sweden. His work is in guaranteeing correctness properties of a Java Card applet.

- Gould, Su and Devanbu from the University of California, Davis, have used Soot to develop their JDBC Checker, a static analysis tool for SQL/JDBC applications.

  `csdl.computer.org/comp/proceedings/icse/2004/2163/00/21630697.pdf`

- Boulifa and Madelaine of the Oasis Team from INRIA have used Soot for model generation for distributed Java programs.

  `www-sop.inria.fr/oasis/Vercors/slides/Fidji03.ppt`

- Cherem and Rugina from Cornell University have used Soot for region analysis and transformation for Java programs.

  `www.cs.cornell.edu/~siggi/papers/ismm04.pdf`

  They have also used Soot for developing techniques to transform Java programs by adding `free` statements to safely reclaim memory.

  `www.cs.cornell.edu/~siggi/papers/ismm06.pdf`

- A research group at Polytechnic University has developed a design obfuscator for Java based on Soot.

  `isis.poly.edu/projects/obfuscator/`

- Patrick Cousot and and Radhia Cousot have used Soot in building a prototype for an abstract interpretation-bases framework for software watermarking (POPL 2004).

- Gagan Agrawal and Liang Guo, University of Delaware, used Soot to experiment with explicitly context-sensitive program slicing (PASTE 2001).

- Qi Su and Gagan Agarwal, University of Deleware, have used Soot for research on efficient and accurate interprocedural program slicing through simultaneous call graph analysis.

  `http://www.cis.udel.edu/~su/research.html`

- Anatas Rountev, Scott Kagan and Michael Gibas from Ohio State University have used Soot to build a tool to support their research for static and dynamic call graphs for Java.

  `www.cse.ohio-state.edu/~rountev/pubs/issta04.pdf`

- Ciarán Bryce and Chrislain Razafimahefa from the University of Geneva have used Soot in their implementation of Safe Object Sharing.

- Jelte Janses from Radboud University Nijmegen in the Netherlands has used Soot to develop a tool for slicing Midlets.

  `www.student.kun.nl/j.r.p.jansen/sec/slicing_midlets.pdf`

- Dave Clark (Ultrecht University, The Netherlands), Michael Richmond (Purdue) and James Noble (Victoria University of Wellington, New Zealand) have developed techniques for statically determining component integrity in Enterprise JavaBeans based on the Soot infrastructure.

  `www.cs.uu.nl/~dave/papers/Beans.pdf`

- Andreas Martens (Imperial College, UK) used Soot in his M.Eng project on the optimization of Java Threads.

  `www.doc.ic.ac.uk/~ajf/Teaching/ Projects/Distinguished02/AndreasMartens.ps`

- Douglas J Brear, Thibaut Weise, Tim Wiffen, Kwok Cheung Yeung, Sarah A M Bennett and Paul H J Kelly, Imperial College London, have used Soot to support the fragmentation process in their work on search strategies for Java bottleneck location by dynamic instrumentation.

  `http://citeseer.ist.psu.edu/581663.html`

- Roman Manevich (Tel Aviv University), Mooly Sagiv (Tel Aviv University), G. Ramalingam (IBM Watson) and John Field (IBM Watson) have used Soot in the front-end processing in their work for three projects.

  - Verifying Safety Properties Using Separation and Heterogeneous Abstractions, Yahav E. and Ramalingam G. PLDI 2004 (joint work of TAU and IBM Watson)

  - Establishing Local Temporal Heap Safety Properties with Application to Compile-Time Memory Management, Shaham R., Yahav E., Kolodner E.K., and Sagiv M., SAS 2003 (join work of TAU and IBM Haifa Research Lab)
  - Partially disjunction heap abstraction. (SAS 2004)

    `www.math.tau.ac.il/~rumster/sas04.ps`

- Dong Zhou, Yuan Chen, Greg Eisenhauer and Karsten Schwan, Georia Institute of Technology, have used Soot to implement the extraction algorithm in their work on active brokers and their runtime deployment.

  `http://www.cc.gatech.edu/systems/papers/schwan/Zhou01AB.pdf`

- Joel Jones and Randy Smith, University of Alabama, used the points-to analysis for Soot in their work on *Automated Auditing of Design Principle Adherence.*

  `a1.cs.ua.edu/~jones/papers/OOOACMSE2003v3.pdf`

- Polyvios Pratikakis, Jaime Spacco and Mike Hicks, University of Maryland, modified the points-to analysis in Soot in their work on transparent proxies for Java futures. (OOPSLA 2004)

  `www.cs.umd.edu/users/mwh/papers/pratikakis04transparent.htmlransparent-proxies.pdf`

- Davide Balzarotti and Mattia Monga, Dip. Informatica e Comunicazione, Milano, have used Soot to perform the analysis phase of their work on Using Program Slicing to Analyze Aspect Oriented Composition. (FOAL 2004)

  `http://citeseer.ist.psu.edu/633907.html`

- Xiaotong Zhuang and Santosh Pande, Georgia Tech, have used Soot to create the CFGs used in their work on compiler scheduling of mobile agents.

  `http://www.cc.gatech.edu/~santosh/papers/icdcs2003.pdf`

- Pierre-Luc Brunelle (Ecole Polytechnique, Montreal), Ettore Merlo (Ecole Polytechnique, Montreal) and Giuliano Antoniol (University of Sannio, Italy) have used Soot to investiage different Java Type Analyses.

  `http://alpha.rcost.unisannio.it/antoniol/publications/papers/issre03vta.pdf`

- Scott Hemmert, Justin L. Tripp, Brad L. Hutchings and Preston A. Jackson, Birgham Young University, have used Soot to perform bytecode reading for their low-level debugger for the Sea Cucumber Synthesizing Compiler.

  `http://csdl.computer.org/comp/proceedings/fccm/2003/1979/00/19790228abs.htm`

- Suhabe Bugrara and Alex Salcianu, MIT, have used Soot in a project aimed at analyzing the interactions between different aspects in a program. This work is presented in FSE'04, "A Classification System and Analysis for Aspect-Oriented Programs".

- Joh Froelich and Paul Dourish, University of California, Irvine, have used Soot to generate call-graph information in their Augur research project. They combine call-graph information with version control meta-data of the source code to analyze social structures/dependencies in the source code.

`http://drzaius.ics.uci.edu/~jfroehli/augur`

- Chris Pickett (McGill University), along with Clark Verbrugge (McGill) and Etienne Gagnon (UQAM) are working on speculative multithreading and return value prediction in SableVM, a Java Virtual Machine. He has extended Soot to analyse features of Java programs relevant to this work and convey the results to SableVM using Soot's attribute generation framework. (http://www.sablevm.org)

- Robert Mittermayr, Technical University of Vienna, is using Soot for his master's thesis work on the topic "statical analysis of multi-threaded java-programs". He is using the CFG and the Dominator-Tree to build the DJ-Graph. He has modified BlockGraph because he needs to have a new Basic Block after a method call from and to a Thread. And with all the DJ-Graphs from the methods he build a TDJG (Thread-DJ-Graph), thus representing the whole program in one DJ-Graph. With this data structure he performs static analysis to find deadlocks and busy waiting problems in multithreaded java programs. For this he implemented a elimination based data flow framework[Sreedhar 95 and 98].