Visualization of Heaps with GCSpy

GCSpy [1] is a generic heap visualization framework for collection and replay of memory management behavior. Its architecture allows easy incorporation into memory management systems, not limited to garbage-collected languages. With only a few changes to the system in which it is incorporated it provides a data-gathering API and a tool to visualize the information in a number of ways.

GCSpy has already been successfully incorporated into many existing environments (e.g. Hotspot JVM [3], Jikes RVM [4], Rotor [5], ...) showing the adequateness of its data gathering API for a wide range of applications. Although the visualization proved quite useful over time as well, there are still many areas where improvements are possible.

The main task of this bachelor's thesis is to further improve the existing freely available code base, in particular to work on interesting ways to visualization the gathered data.

The project should consist of the following tasks:

- analyze the current GCSpy framework: describe the architecture, communication mechanism, available functionality and main use cases (workflow) of the tools.
- discuss and compare the main differences of the GCSpy framework and toolset with those of an existing tracing framework (e.g. log4j [2])
- improve the visualizer
  - analyze the current (visualizer) source code, try to find and document possible design, coding and implementation related issues
  - play with the visualizer on existing traces and think of interesting (new) use cases for the GCSpy visualizer for a particular type of user
  - refactor and improve the existing code with these new use cases in mind. Ideas for enhancements include:
    - general user interface enhancements
    - improve visualization for enumerations: coloring, value stacking in history graph, …
    - alternatives for the default tile view
    - history graphs: value stacking, graphing multiple streams at the same time, augmentation with auxiliary data, …
    - presentation of multiple streams in a single view
    - visualization of multiple heaps
    - additional visualization plugins: summary graphs, …
    - CSV export of data for further analysis
    - custom graphs: visualize data generated from simple user-defined expressions

There will be no need to augment any existing system with data collection routines. Example traces can either be gathered using existing implementations for e.g. Jikes RVM or provided by the advisor.

The focus of this bachelor's project is to add new features, but you will also have to spend some time on getting familiar with the existing sources and on improving the existing code base by smaller extensions or fixing bugs.

Basic experience in working with source control systems is useful. In particular, Mercurial[6] will be used for versioning. Basic knowledge and interest in memory management is recommended, e.g. previous attendance of the "System Software" lecture of the institute is more than sufficient.

Programming language: Java. Depending on the actual task it may be required to do minor C/C++ changes (for changes in the data-gathering API only; if required these can be done with support from the advisor).

Procedure, deliverables:

- framework analysis including a short document describing the new ideas for enhancements (2-4 pages)
- implementation
- after a more intensive beginning phase, short weekly (possibly later bi-weekly) meetings about recent issues, the current progress, and planning for the next time are expected to be held.
- at the end of the bachelor's project a written report of at least 30-40 pages has to be handed in. It should at least contain the following contents: problem description, framework analysis, comparison with another framework, description of the new functionality, description of the implementation, user documentation, evaluation and further work.
- since the efforts will be contributed back to the original project the written documentation should be in English.

Advisor: Dipl.-Ing. Thomas Schatzl (thomas.schatzl@jku.at, room HF305 at the SSW institute)

References:

[1] GCSpy homepage: http://www.cs.kent.ac.uk/projects/gc/gcspy/
[2] Log4j homepage: http://logging.apache.org/log4j/
[3] Hotspot/OpenJDK homepage: http://www.openjdk.org/
[4] Jikes RVM: http://jikesrvm.org/
[5] Rotor: http://msdn.microsoft.com/de-de/library/cc405435.aspx
[6] Mercurial: http://mercurial.selenic.com/wiki/