Profiling SCOOP Programs

PROJECT PLAN

Master Thesis	Profiling SCOOP Programs
Project period	November 2nd, 2009 – April 30th, 2010
Student name	Martino Trosi
Status	3rd semester Master
Email address	trosim@student.ethz.ch
Supervisor name	Benjamin Morandi

1. PROJECT DESCRIPTION

Overview

SCOOP (Simple Concurrent Object-Oriented Programming) [4] is a model and practical framework for building concurrent applications. It comes as a refinement of the Eiffel [2] programming language and is in the process of being integrated into the research version of EiffelStudio, called EVE [3].

Performance is a key objective of any concurrent application. Any model for concurrent programming should have performance as one of its main goals.

Many examples have proven SCOOP's simplicity, but some of these show bad performance. Although the user-friendliness of SCOOP eases the development of concurrent programs, there is no systematic way to measure its performance yet.

A SCOOP profiler would be useful in situations where it is unclear why a particular SCOOP program is running slowly, in order to find out the bottlenecks of the running application. Moreover, the profiler would show us eventual performance problems in the SCOOP model and implementation.

Furthermore the profiler will help comparing SCOOP to the traditional semaphore-based approach [5], used for example in Java.

Scope of the work

We start by defining SCOOP example programs which we want to improve in terms of performance. In order to achieve this we have the define the profiling metrics that we want to use. As an example, unnecessary locking of processors could be detected with the profiler.

As next step we want to design the profiling framework. The profiler might be built upon the EiffelStudio profiler, which is already available for sequential programs. If the use of the provided profiler is not possible, we will include profiling entirely in the SCOOP compiler and the SCOOP library.

The design will be implemented in EVE as part of its SCOOP extension. The SCOOP profiler will only be activated for SCOOP programs.

Finally we will evaluate the SCOOP profiler by applying it to the example programs. The profiler should enable us to detect performance problems and improve the example programs with respect to this aspect.

Intended result

The results include:
- example programs
- profiling metrics
- a version of EVE with an implementation of the profiler
- improved example programs
- a project report

2. BACKGROUND MATERIAL

Reading list

See references [2] - [7]. More material about profiling will be added during our literature research phase.

3. PROJECT MANAGEMENT

Objectives and priorities

These are the main objectives of this project, listed with decreasing priority:
- design a profiling framework and integrate it into EVE
- profile a few example applications and improved versions of them in terms of performance

Criteria for success

The thesis is successful if the SCOOP profiler enables us to improve the performance of the example programs.

The design choices and the metrics have to be properly documented and justified. In case the EiffelStudio profiler could not be enhanced, documentation to support this decision has to be provided.

The code has to be properly documented and tested.

Method of work

Weekly meetings will be held to report on the progress of the project and to make decisions about next steps. Continuous supervisor feedback is expected.

Collaboration with the other members of the SCOOP team is necessary, because of the ongoing work on the SCOOP compiler and the SCOOP library.

Quality management

Documentation

The design choices, the metrics and the implementation have to be properly documented.

A final project report will serve as a container for all documentation except the documentation that is included in the code.

Validation steps

All the source code will be extensively tested.

Bug reports concerning the SCOOP implementation will be filed.

4. PLAN WITH MILESTONES

Project steps

1. Preliminaries (setting up working environment, installing software, Origo setup)
2. Literature research
3. Definition of example programs
4. Design of the profiling metrics for SCOOP
5. Design the profiler framework
6. Understand the SCOOP compiler and design the necessary changes
7. Implement the code
8. Profile and improve example programs
9. Produce documentation and project report

Deadline

The deadline for this project is April 30th, 2010.

Tentative schedule

Description	**Start**	**End**	**Duration**
1. Preliminaries	November 2nd	November 6th	1 weeks
2. Literature research	November 8th	November 20th	2 weeks
3. Example applications	November 23th	December 4th	2 weeks
4. Profiling metrics	December 7th	December 18nd, 2009	2 weeks
5. Profiler framework	January 4th, 2010	January 22th	3 weeks
6. SCOOP compiler	January 25th	February 12th	3 weeks
7. Implementation	February 15th	March 19th	5 weeks
8. Profile examples	March 22th	April 2nd	2 weeks
9. Documentation	April 5th	April 30th	4 weeks

REFERENCES

[1] Chair of Software Engineering: *Semester-/Diplomarbeiten*; Online at: http://se.inf.ethz.ch/projects/index.html, consulted in October 2009.

[2] Bertrand Meyer: *Object-Oriented Software Construction, 2nd edition*, Prentice Hall, 1997.

[3] Patrick Huber: *Integrating SCOOP into EVE*, 2009, http://se.inf.ethz.ch/projects/patrick_huber.

[4] Piotr Nienaltowski: *Practical framework of contract-based concurrent object-oriented programming*, ETH Zürich, 2007.

[5] Allen. B. Downey: *The Little Book of Semaphores*, second edition, Green Tea Press, 2005.

[6] Maurice Herlihy, Nir Shavit: *The Art of Multiprocessor Programming*, Morgan Kaufman, 2008.

[7] ECMA. Ecma-367 eiffel: Analysis, design and programming language 2nd edition. Technical report, ECMA International, 2006.