Reproducible executions of SCOOP programs

PROJECT PLAN

Project period:	18. June 2010 – 31.August 2010
Student names:	Andrey Nikonov, Andrey Rusakov
Email addresses:	nikonov.andrey@gmail.com, rusakov_andrei@mail.ru
Supervisor names:	Benjamin Morandi, Sebastian Nanz, Scott West

1. PROJECT DESCRIPTION

Overview

Testing is the most important technique to ensure the production of quality software in industry, but is not applied to concurrent programs in a satisfactory way. Since the nondeterministic execution of concurrent programs leads to very large state spaces and subtle variations in run-time behavior, testing in a concurrent setting has to be based on repeatable execution schedules, which should be generated in a systematic way to cover the program's state space. Without such a technique, program errors that are discovered are difficult to track down and to reproduce, making debugging a nightmare.

This project explores testing of concurrent programs written in SCOOP based on the idea of controlling execution schedules.

Scope of the work

This project includes developing and implementation of the following techniques for debugging SCOOP programs:

- Deterministic replay
- Schedule generation

Depending on progress, the second project part can be treated as optional.

Intended results

Implementation of the mentioned techniques and demonstration of their benefits by applying them to SCOOP programs with bugs will be the main result of this project.

2. BACKGROUND MATERIAL

Reading list

- Benjamin Morandi, Sebastian S. Bauer, and Bertrand Meyer. SCOOP - A contract-based concurrent object-oriented programming model. In Proceedings of LASER Summer School on Software Engineering 2007/2008, 2010.

- Martino Trosi. Profiling SCOOP programs. Master Thesis. November 2009 – April 2010.

- EIFFEL website. http://www.eiffel.com/, 2010.

- SCOOP website. http://scoop.origo.ethz.ch/, 2010.

3. PROJECT MANAGEMENT

Objectives and priorities

The priority lies in testing of concurrent programs written in SCOOP based on using the following technique:

- Deterministic replay

Criteria for success

This project will be successful if one at least technique for debugging SCOOP programs is developed, implemented and demonstrated on faulty parallel programs.

Method of work

- Weekly meetings with the supervisors

- Communicate via email and Skype if there is any question.

Quality management

Documentation

- Project plan

- Project report

Validation steps

Continuous feedback from the supervisors will guide the development process.

4. PLAN WITH MILESTONES

Project steps

The project involves the following milestones:

1. Familiarization with Eiffel and SCOOP; investigating programs with concurrency bugs.

2. Deterministic replay: a technique for recording and replaying the decisions of the SCOOP scheduler will be investigated and implemented; the technique for recording can reuse recent work on a profiler for SCOOP programs (by Martino Trosi).

3. Schedule generation and debugging (optional):

   a. Schedule generation: one or more techniques for schedule generation will be investigated and implemented (e.g. starting with generation of random schedules).

   b. Debugging: a technique for debugging SCOOP programs will be developed and implemented (e.g. display of object/processor associations or locks).

4. Evaluation: The strength of the developed techniques is demonstrated by applying it to the faulty programs from (1) and other examples.

Project Deadline

30. August 2010

Tentative schedule

Week	1	2	3	4	5	6	7	8	9	10	11
Familiarization with Eiffel and SCOOP	■	■									
Investigating programs with concurrency bugs		■									
Investigating profiler for SCOOP programs, deterministic replay implementation		■	■	■	■						
Implementing one or more techniques for schedule generation					■	■	■	■			
Developing and implementation technique for debugging SCOOP programs					■	■	■	■			
Applying developed techniques to the faulty programs									■	■	
Report writing							■	■			■

REFERENCES

[1] Chair of Software Engineering: *Semester-/Diplomarbeiten*; Online at: http://se.inf.ethz.ch/projects/index.html, consulted in October 2002.

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