Concurrency Patterns in SCOOP

Master Thesis – Project Plan

Project period:	10. March to 8. September 2014
Student name:	Roman Schmocker, 09-911-215
Status:	4. semester, Msc in Computer Science
Email address:	romasch@student.ethz.ch
Supervisor name:	Alexey Kolesnichenko and Prof. Dr. Bertrand Meyer

1 Project Description

1.1 Overview

The SCOOP [14] [5] mechanism in Eiffel [1] [13] is a language extension for concurrency. Many important concurrency problems, such as the Dining Philosophers, can be solved in a simple and elegant way [4].

In practice programmers have learned to avoid tricky concurrency issues in most cases. In order to make use of modern multi-core machines, most applications use thread-safe data structures and some well-known patterns, like a worker pool.

In the case of Eiffel primitive operations like individual method calls are thread-safe thanks to SCOOP. There is little experience in constructing higher-level concurrency patterns however. As a consequence, whenever a programmer wants to use a pattern to solve a specific problem he or she needs to figure out a way to do it in SCOOP.

The goal of the project is therefore to determine how some interesting concurrency problems, for which a well-known pattern exists in other languages, can be solved in Eiffel with the SCOOP extension.

An example is the idea that when two threads proceed concurrently and need to update each other, as in the case of a user interface thread and a downloading thread, where the UI should be updated regularly to show the progress of the download. The standard solution is for the download process to trigger a repaint operation in the UI thread and to store the progress value in a synchronized shared object. With SCOOP however it is useful to introduce a third processor with which they both communicate to avoid undue waiting or even deadlock [11].

The result of the previous analysis is a new set of SCOOP specific patterns that each solve a typical concurrency problem. As a last step, those patterns should be turned into reusable library components.

1.2 Scope of the work

The project consists of four main tasks:

1. A survey of popular concurrency problems and the standard patterns used to solve them. The goal of the survey is to have a broad overview over widely used concurrency patterns and to determine if the underlying problem they are solving is valid as well in a SCOOP context.

2. An analysis of several selected patterns. In this task the goal is to investigate if it's possible to devise a new pattern in Eiffel and SCOOP solving the same problem.

3. The design and implementation of a new library which provides the new SCOOP patterns as reusable components wherever possible.

4. A written report with descriptions about the patterns and the problems that may have occurred.

1.3 Written Report

The report should cover the following topics:

- A short introduction to SCOOP and how it differs from other concurrency mechanisms.
- A section about concurrency patterns and an overview over all patterns collected in the survey.
- A detailed description of all analyzed patterns. For each pattern this part should answer the following questions:
  - What is it used for?
  - What is the general architecture?
  - Is it possible to efficiently implement it with SCOOP, and if not, why?
  - Is it possible to provide the pattern in a library, and if not, why?
  - What were the problems that occurred and how could they be solved?
  - What is the difference in implementation between Eiffel and other languages?
- A description (manual) of the new concurrency pattern library.

1.4 Intended results

The result of the project should be an Eiffel library with the analyzed concurrency patterns and a written report describing the patterns and the library. If a specific pattern can not be turned into a library component, the report should provide a reason why it's not possible.

2 Background Material

2.1 Reading list

- P. Nienaltowski. *Practical Framework for Contract-based Concurrent Object-oriented Programming.* PhD thesis, ETH Zürich, 2007
- J. Bloch J. Dowbeer D. Holmes D. Lea B. Goetz, T. Peierls. *Java concurrency in practice.* Addison-Wesley, 2006
- J. Reinders M. McCool, A. Robison. *Structured Parallel Programming: Patterns for Efficient Computation.* Morgan Kaufmann, 2012
- H. Rohnert . Buschmann D. Schmidt, M. Stal. *Pattern-Oriented Software Architecture, Volume 2, Patterns for Concurrent and Networked Objects.* Wiley, 2000
- M. Grand. *Patterns in Java, Volume 1: A Catalog of Reusable Design Patterns Illustrated with UML.* Wiley, 2002
- S. Toub. Patterns for parallel programming, 2010. Microsoft Corporation
- Karine Arnout. *From Pattern to Components.* PhD thesis, ETH Zürich, 2004
- K. Arnout B. Meyer. Componentization: The visitor example. *Computer*, 39(7):23–30, 2006

2.2 Software Libraries

- Microsoft task parallel library. http://msdn.microsoft.com/en-us/library/dd460717(v=vs.110).aspx
- Oracle java concurrency package. http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/package-summary.html

3 Project Management

3.1 Objectives and priorities

Due to the nature of the project prioritization on the individual tasks doesn't make a lot of sense. Instead the amount of work can be measured by the number of analyzed concurrency problems.

There is one optional low priority task, which is to analyze existing Eiffel / SCOOP applications and libraries and checking if they can be refactored to using the new pattern library.

3.2 Criteria for success

At least five concurrency problems are analyzed and the report is finished. For each analyzed problem there should be an Eiffel pattern solving the problem and either a fully tested library component or a written statement with the reason why it's not possible to turn the pattern into a library. The report is complete when all points in Section 1.3 are covered.

3.3 Method of work

There will be regular meetings and email communication. Important design decisions or project plan changes will be discussed in advance.

3.4 Quality management

3.4.1 Documentation

The written report should give a detailed description about the patterns. For the source code, a header comment for each class and feature is required. All SVN commits have a meaningful commit message. The source code will be code-reviewed.

3.4.2 Validation steps

The concurrency pattern library should be tested and equipped with some example applications. If possible, performance tests should be included.

4 Plan with Milestones

Date	Duration (weeks)	Milestone
31. March	3	Literature study and initial survey complete.
afterwards	-	Selection of relevant concurrency problems and patterns to be implemented in SCOOP.
9. June	10	Analysis of selected patterns and experimental implementations in Eiffel. If there's enough time: analysis of more patterns or optional tasks.
14. July	5	Design and implementation of the new pattern library, including testing, example applications and a code review at the end.
28. July	2	Implementation of code review suggestions and finalization of the pattern library.
25. August	4	The final report is written and ready for submission.
8. September	2	Submission of thesis. Last two weeks are reserved for delays.

4.1 Deadline

8. September 2014

References

[1] Eiffel ECMA-367 Standard. http://www.ecma-international.org/publications/standards/Ecma-367.htm.

[2] Microsoft task parallel library. http://msdn.microsoft.com/en-us/library/dd460717(v=vs.110).aspx.

[3] Oracle java concurrency package. http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/package-summary.html.

[4] Scoop examples. http://docs.eiffel.com/book/solutions/scoop-examples.

[5] Scoop official website. http://docs.eiffel.com/book/solutions/concurrent-eiffel-scoop.

[6] Karine Arnout. *From Pattern to Components*. PhD thesis, ETH Zürich, 2004.

[7] J. Bloch J. Dowbeer D. Holmes D. Lea B. Goetz, T. Peierls. *Java concurrency in practice*. Addison-Wesley, 2006.

[8] K. Arnout B. Meyer. Componentization: The visitor example. *Computer*, 39(7):23–30, 2006.

[9] H. Rohnert . Buschmann D. Schmidt, M. Stal. *Pattern-Oriented Software Architecture, Volume 2, Patterns for Concurrent and Networked Objects*. Wiley, 2000.

[10] M. Grand. *Patterns in Java, Volume 1: A Catalog of Reusable Design Patterns Illustrated with UML*. Wiley, 2002.

[11] Alexey Kolesnichenko, Sebastian Nanz, and Bertrand Meyer. How to cancel a task. 2013. To appear.

[12] J. Reinders M. McCool, A. Robison. *Structured Parallel Programming: Patterns for Efficient Computation*. Morgan Kaufmann, 2012.

[13] B. Meyer. *Touch of Class*. Springer, 2009.

[14] P. Nienaltowski. *Practical Framework for Contract-based Concurrent Object-oriented Programming*. PhD thesis, ETH Zürich, 2007.

[15] S. Toub. Patterns for parallel programming, 2010. Microsoft Corporation.