Integrating SCOOP into EVE

PROJECT PLAN

Project Type	Master Thesis
Project period	9.3.2009 – 9.9.2009
Student name	Patrick Huber
Status	4th Semester
Email address	pahuber@student.ethz.ch
Supervisor name	Benjamin Morandi

1. PROJECT DESCRIPTION

Overview

SCOOP - Simple Concurrent Object-Oriented Programming - is a practical framework for the development of high-quality concurrent software which carries the advantages of object technology and Design by Contract to the concurrent context. Its simplicity relies on basic O-O concepts; its expressiveness and modelling power is due to the full support for advanced O-O mechanisms and Design by Contract. Unlike most existing concurrent O-O languages, SCOOP is a full-blown O-O language: it supports (multiple) inheritance, polymorphism, dynamic binding, genericity, and contracts. One main goal of SCOOP is to hide the synchronization problems - a major problem in the concurrent programming context - from the programmer.

The EVE project provides a research branch of EiffelStudio. The goal of the EVE project is to avoid individual modifications of EiffelStudio. Instead of having a single research branch of EiffelStudio, code-named Eve for ETH Verification Environment provides a research environment around the EiffelStudio which includes the outgrowth of CDD, AutoTest, Ballet, Origo plug-in, Escher, SCOOP, Proof-Transforming Compilation plug-in and anything else we may dream of in the future.

Scope of the work

This master thesis relies on the work of Piotr Nienaltowski presented in his dissertation [3]. The goal is to extend EVE with SCOOP support. This integration work consists of several steps:

- ⇒ lexer, parser
- ⇒ abstract syntax tree: Integrate the type extension and the transformation.
- ⇒ type checker: Introduce new validity rules and their four-letter codes. Integrate modifications (including additions) to existing validity rules.
- ⇒ editor: Make sure they work with SCOOP code. In particular include automatic completion.
- ⇒ views: Make sure they work with SCOOP code.

⇒ context tool (metrics): We might add a metric on the number of separate types. Make sure it works with SCOOP code.
⇒ documentation: Update required settings, restrictions, mixing with multithreading, etc.
⇒ example: Provide usage example covering both basic and advanced usage.

- context tool (class, feature): Make sure it works with SCOOP code.
- navigation tool (feature, cluster): Make sure it works with SCOOP code.
- refactoring: Make sure it works with SCOOP code.
- interface to external code: How will the new construct appear via external features, CECIL, .NET, external features?
- integration into EiffelWeasel: At least existing tests should run.

⇒ Part of the project
• Optional

The SCOOP extension should only be applied within the compilation process if the particular project code contains also SCOOP code.

Intended results

The intended results contain a correct integration and implementation of at least the major project parts (parsing, AST adaption and transformation, type checking, editor and view changes, metrics). Optional marked items can also be integrated when there is enough time. Finally, basic SCOOP examples should compile and run.

2. BACKGROUND MATERIAL

Reading list

- Parts of [2]
- [3]

Further reading

- [5] – [9]

3. PROJECT MANAGEMENT

Objectives and priorities

In a first step the changes on the parser and the AST should be implemented to get an initial running version which already compiles existing SCOOP projects. In the next step the other objectives like type checking, view adaption etc. should be implemented. High priority gets of course the basic functionalities like parser, AST and type checking – lower priority the optional points.

Criteria for success

- Running SCOOP implementation in EVE as mentioned in ‘Intended results’

Method of work

- A new EVE SVN brunch will serve as working platform.
- Briefings with my supervisor and Volkan Arslan will take place regularly to discuss important details.
- Continuous documentation of the project parts.

Quality management

Documentation

- All work related to this project will be documented.

Validation steps

- Briefings with my supervisor and Volkan Arslan will take place regularly.
- Continuous code review
- Testing (Running examples, running EVE)

4. PLAN WITH MILESTONES

Project steps

- Understand the major parts of the SCOOP project & project plan
- Initial running SCOOP version of EVE: Parser and AST changes
- First testing series
- Integrating of type checking
- Editor support & automated completion
- EiffelStudio Views
- Metrics
- Optional Points

Deadline

9. September 2009

Tentative schedule

The following schedule is tentative – the complexity and needed work for the particular project parts are not so obvious at the outset.

Week	11	12	13	14	15	**16**	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
Reading & proj. plan	■																							▨	▨	▨	▨
Parser		■	■																					▨	▨	▨	▨
AST			■	■	■	■																		▨	▨	▨	▨
Testing						■																		▨	▨	▨	▨
Type checking							■	■	■	■														▨	▨	▨	▨
Editor support										■	■	■	■											▨	▨	▨	▨
Aut. completion													■	■	■	■								▨	▨	▨	▨
Eiffel views																■	■	■	■					▨	▨	▨	▨

Metrics																			■	■	■	■		▨	▨	▨	▨
Opt Feature view																						■	■	■	■	■	▨
Opt. Context tool																						■	■	■	■	■	▨
Other optional points																						■	■	■	■	■	▨

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.

[3] Piotr Nienaltowski: Practical framework for contract-based concurrent object-oriented programming.

[4] Maurice Herlihy, Nir Shavit: The Art of Multiprocessor Programming. Morgan Kaufmann, 2008.

[5] Allen B. Downey: The Little Book of Semaphores Second Edition. Green Tea Press, 2005.

[6] Piotr Nienaltowski: Flexible locking in SCOOP. CORDIE'06, July 2006, York, UK.

[7] Piotr Nienaltowski, Bertrand Meyer: Contracts for concurrency, CORDIE'06, July 2006, York, UK.

[8] Nienaltowski P.: Refined access control policy for SCOOP, Technical Report tr511, ETH Zurich, February 2006

[9] Nienaltowski P.: Efficient data race and deadlock prevention in concurrent object-oriented programs, Doctoral Symposium, 19th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA), Vancouver, Canada, October 2004. Appears in OOPSLA 2004 Companion: 56-57.

Zürich, 20.03.2009

Patrick Huber   Benjamin Morandi