

Catching CATs

Towards a fully typesafe Eiffel

Diploma Project Plan

Project Period: 2003-03-10 Mo – 2003-07-09 Wed

Student: Markus Keller (markus.keller@student.ethz.ch)

Student-No.: 98-918-758

Supervising Assistant: Bernd Schoeller

Supervising Professor: Bertrand Meyer

1 Project Description

1.1 Overview

Object-oriented programming languages strive to be expressive and safe at the same time. There are situations where the standard ingredients of fully object-oriented languages interfere and expose a hard problem for language designers. Pierre America's conjecture [Ame90] was rather drastic: Of the three protagonists *static typing*, *substitutivity* (subtype polymorphism) and *covariance*, at most two can be on the stage at the same time.

Covariance is “The policy allowing a feature redeclaration to change the signature so that the new types of both arguments and result conform to the originals” [Mey97]. This flexibility is required by the dual role of a class as type and reusable module. Furthermore, Eiffel allows changing the export status of features in descendant classes.

Unfortunately, it is easy to devise redefined routines which cannot be used polymorphically in every context the original routine could (and thus break static typing). The term *CAT* describes features that *C*hange *A*vailability or *T*ype. *Catcalls* are calls to routines which are CAT in any redefinition. Calling them on a polymorphic entity breaks static typing.

The Catcall problem potentially appears in every object-oriented language, but is only identified as such in advanced languages with a strong yet flexible static type system, which allows covariant redefinitions.

1.2 Scope of the work

There are several ways to cope with the issue: I will give an overview over the solutions in various languages, ranging from "no solution" (Java, C, C++, ...) over "dynamic" (Smalltalk, scripting languages, ...) to real improvements (system validity, banning polymorphic catcalls, matching, Generative Programming, separating subclassing and subtyping, ...).

One part of my thesis will be a thorough analysis of the problems that covariant parameter redefinitions impose. The exposition will lead us to the question as to when and in what context those redefinitions occur and what intentions of the programmer lead to the problem. The differences between subtyping and subclassing are discussed. There's hope that minor changes to the interpretation of "static typing" are enough to overcome America's conjecture. On the basis of a carefully chosen small set of representative (yet immediately understandable) example problems, I will come up with a set of requirements that a "final" solution must meet.

In parallel, I will create a testbed for the proposed solutions. This infrastructure generates an AST, on which the various approaches can be implemented. Existing Eiffel programs will be examined on the testbed, which gives valuable information on the impact of the restrictions a certain approach imposes.

The ultimate goal is a revised type system and conformance mechanism, which is no more restrictive than needed to ensure a sound type system, but on the other hand leaves as much expressive freedom as possible to the programmer, while still being understandable for clients. The solution will be proposed to the ECMA Committee for the Standardization of the Eiffel language and is expected to be a valuable contribution to the final Eiffel standard.

1.3 Intended results

1.3.1 Report: Discussion of Covariance problems

- The report will cover existing and possibly new approaches to make Eiffel typesafe.
- A collection of exemplary situations where covariance causes problems in practice.
- Description of the testbed (implementation and user manual)
- Results from the analysis of existing code on the testbed.

1.3.2 Testbed for checking Eiffel programs against custom typing rules

- The testbed will be an open framework for testing different typechecking policies for Eiffel dialects. The facility produces an annotated AST on which the different solutions can operate to check a given source project.
- Starting with system validity, I'll implement checkers for some of the proposed solutions.
- Each type checker tells the user whether the given source is valid (with respect to its policy) or not. For detected errors, it indicates the position in the source and the kind of error.

2 Background Material

2.1 Reading list

Eiffel: System validity and banning CATcalls: [Mey97]; [Mey]

Other Eiffel solutions: [CL00], [Coo89], and an unpublished idea by Mark Howard.

Software: [Bez]

Other contributions in the field: [Cas95], [Sim95], [CW85], [Sim03], [BCC⁺95], [Bru96], [AC95]

3 Project Management

3.1 Objectives and priorities

The project should finish on time.

3.2 Criteria for success

3.2.1 Report

The report should explain the problem and various approaches to its solution to an Eiffel programmer.

3.2.2 Testbed

The resulting software generates an AST from Eiffel source and is a framework for the development of different Eiffel type checkers. At least two type checking policies are implemented and runnable on given source.

The testbed must be usable by others. A clean architecture and the manual should enable anybody to use the provided checkers and to implement new ones.

3.3 Method of work

The thesis will evolve in a *Seamless Development* [Mey97] process. Report and testbed are written in parallel, influencing each other. Since my Diploma Thesis is an individual job, I will do the work alone in an office near the Software Engineering group at ETH. Major contributions of others will be indicated. Discussions with the SE group members may influence my work.

3.4 Quality management

3.4.1 Documentation

The produced code is documented by *Design By ContractTM*. Comments and indexing clauses make the source almost self-explanatory. A document describing the whole system completes the documentation. The report includes a description of the framework and its application to the solutions.

3.4.2 Validation steps

Design By ContractTM, a unit test suite and a high compilation/test frequency ensure that the code is always in an executable state. Frequent discussions with the supervisors should help me to stay “on the path”.

4 Plan with Milestones

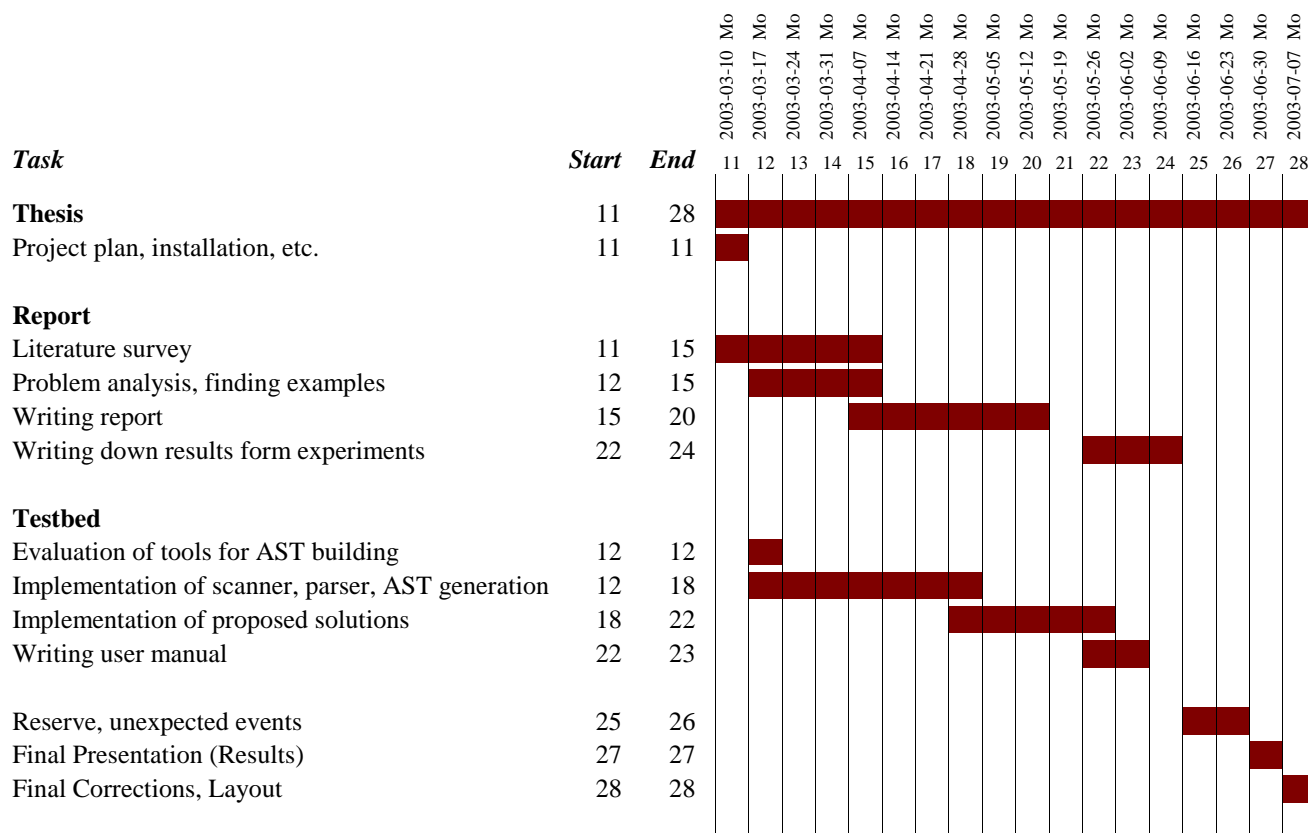
4.1 Fixed parameters

Project start: Mo, 2003-03-10 (week 11)

Project end: Wed, 2003-07-09 (week 28)

Total work time: 4 months, i.e. 17 weeks + 3 days = 88 days

4.2 Tentative schedule



References

- [AC95] Martin Abadi and Luca Cardelli. On subtyping and matching. In *Lecture Notes in Computer Science (ECOOP '95)*, volume 952, pages 147–167. Springer, 1995. <http://research.microsoft.com/Users/luca/Papers/PrimObjMatching.A4.ps> (2003-03-12). 3
- [Ame90] Pierre America. *The America Conjecture*. Unpublished lecture slides by Bertrand Meyer, citing dictum at TOOLS Europe, 1990. 1
- [BCC⁺95] Kim B. Bruce, Luca Cardelli, Giuseppe Castagna, Jonathan Eifrig, Scott F. Smith, Valery Trifonov, Gary T. Leavens, and Benjamin C. Pierce. On binary methods. *Theory and Practice of Object Systems*, 1(3):221–242, 1995. <ftp://ftp.cs.williams.edu/pub/kim/binary.ps> (2003-03-12). 3
- [Bez] Eric Bezault. *Gobo Eiffel Project*. <http://www.gobosoft.com> (2003-03-12). 3
- [Bru96] Kim B. Bruce. Typing in object-oriented languages: Achieving expressiveness and safety. Technical report, Williams College, September 1996. <ftp://ftp.cs.williams.edu/pub/kim/Static.ps.gz> (2003-03-12). 3

- [Cas95] Giuseppe Castagna. Covariance and contravariance: conflict without a cause. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 17(3):431–447, 1995. ISSN 0164-0925. 3
- [CL00] Dominique Colnet and Luigi Liquori. Match-o, a dialect of eiffel with match-types. In *35th conference on Technology of Object-Oriented Languages and Systems (TOOLS Pacific'2000)*, pages 190–201. IEEE Computer Society, Sydney, Australia, November 2000. <http://www.loria.fr/~colnet/publis/tools-pacific-2000.ps.gz> (2003-03-12). 3
- [Coo89] W. R. Cook. A proposal for making eiffel type-safe. In *Proceedings of the Third European Conference on Object-Oriented Programming (ECOOP)*. Cambridge University Press, 1989. <http://www.ifs.uni-linz.ac.at/~ecoop/cd/papers/ec89/ec890057.pdf> (2003-03-12). 3
- [CW85] Luca Cardelli and Peter Wegner. On understanding types, data abstraction, and polymorphism. *ACM Computing Surveys*, 17(4):471–522, December 1985. <http://research.microsoft.com/Users/luca/Papers/OnUnderstanding.A4.pdf> (2003-03-12). 3
- [Mey] Bertrand Meyer. *Eiffel: The Language*, third edition. Work in progress: <http://www.inf.ethz.ch/personal/meyer/#Progress> (2003-02-12). 3
- [Mey97] Bertrand Meyer. *Object-Oriented Software Construction*. Prentice Hall, 2nd edition, 1997. <http://www.eiffel.com/>. 1, 3
- [SE] Chair of Software Engineering. *Semester-/Diplomarbeiten*. <http://se.inf.ethz.ch/projects/index.html> (2003-03-06).
- [Sim95] A J H Simons. Rationalizing eiffel's type system. In C. Mingins, R. Duke, and B. Meyer, editors, *18th Conf. Technology of Object-Oriented Languages and Systems (TOOLS Pacific)*, pages 365–377. Prentice-Hall, Melbourne, 1995. <http://www.dcs.shef.ac.uk/~ajhs/papers/eiffelty.ps.gz> (2003-03-12). 3
- [Sim03] Anthony J.H. Simons. The theory of classification (article series). *Journal of Object Technology*, 2002–2003. <http://www.jot.fm/>. 3