

Eiffel and Design by Contract

Ein Kompaktkurs für die Praxis

Kursleitung:

Prof. Dr. Bertrand Meyer

17./18. Juni 2004

2

KURS

**Departement
Informatik**

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Eiffel and Design by Contract

Target groups

Software developers; project managers; information technology executives.

Prerequisites

General knowledge of software engineering. Familiarity with object-oriented techniques is desirable but not required.

The course will be held in English.

The Eiffel method and the supporting language and tools have become an approach of choice for professional applications focusing on quality and economy, especially in the financial industry, the defense/aerospace community, health care and other areas where companies are interested in developing software systems that will stay around for a long time and undergo many changes, while wanting to see results quickly and with fewer developers. Eiffel uses a seamless lifecycle integrating tasks of requirements analysis, design, implementation and evolution; the approach systematically relies on reuse, and produces systems that can be easily adapted and extended. The techniques of Design by Contract, based on the systematic specification of the properties of software elements, play a central role in the process, affecting specification, design, documentation, testing and project management.

The course presents a detailed description of Eiffel techniques, allowing the participants to start developing systems in Eiffel and to understand the concepts of Design by Contract. It is meant both for software developers and project leaders, as well as for decision makers interested in trends of software engineering.

Course goal

The course provides participants with concrete techniques for producing better software faster and cheaper. It includes an in-depth description of important principles of modern software engineering, object-oriented methodology, and the design of successful reusable components. Its emphasis is on practical techniques, which can be readily applied in an everyday setting. For teams using Eiffel, the course provides the necessary information to get started immediately; for those using other approaches, it provides a complementary perspective enabling participants to use their existing object-oriented tools better.

Kursprogramm

Donnerstag, 17. Juni 2004

Part A 9:00 - 12:30 h: The Eiffel method

- **1: Overview**
Goals and scope of the Eiffel method. Survey of existing Eiffel usage.
- **2: The environment**
Applying O-O concepts to an IDE. Development objects. Pick and Drop. Eiffel in the Microsoft .NET framework: ENViSION. External language interfaces: using C++, C#, Java with Eiffel. Eiffel as component combinator. Multi-language development. Interfaces to databases, XML, UML and other technologies. Graphical representations: the Business Object Notation.
- **3: The lifecycle model**
Seamless development. Reversibility. The role of contracts in requirements, analysis and design. The cluster model of the software lifecycle; comparison with other approaches (waterfall, spiral). The Eiffel method and agile development.

Part B 14:00 - 17:00 h: Eiffel programming basics

- **4: Language basics**
Program structure, the role of classes, LAce.
- **5: The dynamic object model**
Object creation. Creation procedures. Cloning, copying, equality. Information hiding. Object persistence. Garbage collection
- **6: Design by Contract**
The role of contracts in software development. The notion of assertion. Preconditions, postconditions, class invariants. Contracts for analysis and design. Contracts for quality assurance and testing. Contracts for documentation. Contracts for project management. Role in the software lifecycle. From tests to proofs.

Freitag, 18. Juni 2004

Part C 9:00-12:30: Key development techniques

- **7: Agents, tuples, event-driven programming and functional programming**
Typical structure of an event-driven application. GUI programming. Observer-like patterns and their limitations. Agents: open arguments, closed arguments. An example: the Event Library. Comparison with C# "delegates", Java techniques.
- **8: Genericity**
Container structure and the need for generic classes. Examples from EiffelBase. The role of genericity for static typing.
- **9: Inheritance**
The Eiffel inheritance model. Polymorphism, dynamic binding. Using multiple inheritance properly. Deferred classes and features; applications to taxonomy, analysis, design. Redefinition, renaming, effecting, undefinition, join. Inheritance and Design by Contract: rules on redefining assertions.
- **10: Covariance & anchored types**
Reconciling flexibility with safe typing. Catcalls. Current approaches.

Part D 14:00-17:30: Advanced techniques and assessment

- **11: Once routines**
How to obtain global data without destroying the modular structure of programs.
- **12: Exception handling**
Handling abnormal situations. Exception Principle. Retrying vs. failure. Connection with Design by Contract principles.
- **13: Designing for reuse**
Issues of successful component design. The EiffelBase and EiffelVision experience. Consistency principles. Naming policy. Reuse rules.
- **14: Principles of the Eiffel method**
The Eiffel analysis, design, implementation and maintenance methodology. Comparison with other approaches (e.g. Unified Process). Eiffel for education: ETH experiences, "Touch of Class" project.
- **15: Discussion and conclusion**

