Design by Contract and the Component Revolution

Bertrand Meyer

If you are involved in software development, either as a manager or as a developer, you know how crucial it is to improve software quality and productivity. In spite of the major advances of recent years, especially Object Technology, it is still much too hard to produce correct, reliable systems at a reasonable cost.

This state of the art seminar is a unique opportunity to hear about the approach that is revolutionizing the software industry - not from an outside observer, but from one of the key originators of the discipline, Dr. Bertrand Meyer, one of a handful of object-oriented movers and shakers whose work made modern object-oriented technology possible.

"Design by Contract" is the major technological breakthrough in software engineering in the past 15 years. This course introduces Design by Contract and shows why it can, like no other technique since the invention of classes and objects, dramatically affect the way we develop software, and why it is a required condition for achieving the promise of the Component Revolution.

Design by Contract tackles head-on the issue of software reliability, by taking a holistic approach to the construction of systems so that they will work correctly the first time around. Based on a simple yet powerful metaphor — software systems as collections of components that interact through contracts, similar to those between people or companies — it has a profound effect on almost every aspect of software development, from analysis and design to implementation, documentation, debugging, quality assurance, maintenance, and project management.

In one dense, information-packed day, Dr. Bertrand Meyer, one of the pioneers of modern software engineering and the creator of Design by Contract, will present the full power of the method for improving the software process. You will understand why major companies the world over are gearing up to be ready for this revolutionary approach.

Go beyond buzzwords and simplistic views. Learn how to take the best advantage of Design by Contract for your own team in your own environment: assertions, invariants, automatic documentation, contract-based testing, quality assurance, handling errors and abnormal cases, writing useful specifications, retaining the work of the best developers, maintaining legacy code, ensuring successful components.

If you keep more than a casual interest in software technology, you cannot afford to miss the Design by Contract revolution. This course is a unique opportunity to learn firsthand how the next generation of software will be developed.

Course Material:
The material distributed to participants includes more than 100 slides and supporting articles.

Course Outline:
Part 1: Issues
- Software reliability: how important is it? Can we get away with Good Enough Software? How does the industry cope with bugs and other reliability problems?
- Components of reliability: correctness, robustness; the role and limits of quality assurance. Role and limits of formal methods.
- Reliability and the software process: what is the role of each phase?
- Reliability and the component revolution.

Part 2: Principles
- The theoretical basis: assertions and formal semantics.
- The notion of contract: human contracts, software contracts. How far does the metaphor extend? What is special about software contracts?
- Introducing contracts into software: preconditions, postconditions, class invariants and others. How does this fit in an object-oriented software architecture? What's special about objects and contracts?

Part 3: Applications
- Contracts and documentation: how to produce good software documentation (and live to tell the tale).
- Contracts and analysis: real developers do use bubbles!
- Contracts and implementation: killing the defects before they kill you.
- Contracts and debugging: rehabilitating the most shameful part of the business.
- Contracts and testing: a systematic approach.
- Contracts and quality assurance: a unifying concept.
- Contracts and components: making reuse succeed.
- Contracts and abnormal case: a sound basis for exception handling.
- Programming languages and contracts: Ada, C++, Eiffel, Sather.
- Java and C++ extensions for contracts: how much can be done?
- Contracts and UML: the Object Constraint Language.
- Contracts and component technologies: using Design by contract to take the best advantage of Microsoft's COM and DCOM and OMG's CORBA standard.
- Contracts and standards: ISO 9001, CMM.
- Development environments: their support for contracts.
- A window on research: beyond current approaches.

Key Questions:
- Find out the answers to the most pressing issues in this radically new approach to software construction:
  - What's Design by Contract beyond the buzzword?
  - How much of Design by Contract can be applied in Java and C++?
  - How much can you do in classical languages such as C?
  - What gains can you expect in terms of quality and productivity?
  - How can contracts be combined with component technologies such as COM/DCOM and CORBA.
  - How does Design by Contract fit with quality-enhancing standards such as ISO 9001 and the CMM?
  - How can you ascertain the quality of software components?
  - How can developers produce useful documentation without huge extra work?
  - How does Design by Contract affect the software lifecycle and project management?
  - What tools are available today to support Design by Contract?

Bertrand Meyer has played a major role in developing modern software technology and bringing it to software engineers working in production environments. As President of Interactive Software Engineering, Inc. (ISE), Bertrand was the major designer of the acclaimed Eiffel object-oriented method.
and language, and directed the implementation of Eiffel environment. Prior to founding Interactive he was head of software engineering in a large European company for nine years, and on the faculty of the University of California, Santa Barbara, for several years. In addition to his work at ISE he holds a position of adjunct Professor at Monash University (Melbourne, Australia), where he directs research on Design by Contract, Trusted Components and Distributed Systems. He serves as the editor of the Eiffel column for the JOOP magazine, the editor of the Object column for Computer, and chairman of the TOOLS conference series.

Active on both the industrial and academic scenes, Dr. Meyer consults for Fortune 500 companies, on architectural reviews, design and deployment of software quality plans, and software strategies. He has published 9 books on software engineering, object technology and programming languages, and edited or co-edited several others. His publications also include more than 100 articles in international journals as well as numerous conference presentations.

Dr. Meyer holds an MS in Computer Science from Stanford and a doctorate from the University of Nancy. His unique perspective on Design by Contract comes from the software engineering approach that he brings to all topics, backed by many years of practical involvement with the issues of developing high-quality software for projects large and small.