

# The Programmers' Stone

Hi, and welcome to The Programmers' Stone. The purpose of this site is to recapture, explore and celebrate the Art of Computer Programming. By so doing we hope to help the reader either become a better programmer, understand what less experienced programmers are struggling with, or communicate more effectively with other experienced programmers.

We know from work with individuals that by doing this we put the fun back into the work and greatly extend the boundaries of the possible, so building much smarter and stronger systems.

The present structure is planned around an eight day course, delivered two days a week for four weeks. Each chapter corresponds to the course notes for one day's material. The eighth day should be free discussion, so no prepared notes, meaning that there are seven chapters. We've deliberately made each chapter a single HTML page because it makes it much easier to print the text. Sorry there are no internal anchors yet, there are big headings, so use your slider!

We'd very much like to hear from you!

Alan & Colston

[alan@melloworld.com](mailto:alan@melloworld.com) [colston@shotters.dircon.co.uk](mailto:colston@shotters.dircon.co.uk)

---

## Chapter 1 - Thinking about Thinking

- Roots of the Approach
- Mapping and Software Engineering
- Mapping and TQM
- Mandate Yourself!
- The Undiscovered Country
- Knowledge Packets, Daydreams, Maps and Understanding
- Mappers and Packers
- How to Regain Mapping
- The Ways of Mappers and Packers
- Packing as a Self-Sustaining Condition
- The Mapper/Packer Communication Barrier

## Chapter 2 - Thinking about Programming

- What is Software Engineering For?
- Software Engineering is Distributed Programming
- What is Programming?
- Programming is a Mapper's Game
- General Tips on Mapping
- Mapping and the Process
- Angels, Dragons and the Philosophers' Stone

- Literary Criticism and Design Patterns
- Cognitive Atoms
- The Quality Plateau
- Knowledge, Not KLOCS
- Good Composition and Exponential Benefits

## Chapter 3 - The Programmer at Work

- Approaches, Methodologies, Languages
- How to Write Documents
- The Knight's Fork
- The Personal Layered Process
- To See the World in a Line of Code
- Conceptual Integrity
- Mood Control
- Situation Rehearsals

## Chapter 4 - Customs and Practices

- The Codeface Leads
- Who Stole My Vole?
- Reviews and Previews
- Code Inspections and Step Checks
- Coding Standards and Style Guides
- Meaningful Metrics
- Attitude to Tools
- Software Structures are Problem Structures
- Root Cause Analysis
- Complexity Matching and Incremental Boildown
- The Infinite Regress of 'Software Architectures'
- The Quality Audit

## Chapter 5 - Design Principles

- Simple and Robust Environments
- System Types
- Error Handling - a Program's Lymphatic System
- Modalism and Combinatorial Explosion
- Avoid Representative Redundancy
- Look at the State of That!

- The Reality of the System as an Object
- Memory Leak Detectors
- Timeouts
- Design for Test
- Dates, Money, Units and the Year 2000
- Security

## Chapter 6 - Prudence and Safety

- Brain Overload
- Brain Overrun
- Overwork
- Cultural Interface Management
- Individual Responsibility and Leadership
- The False Goal of Deskillling
- Escape Roads
- New Member Integration

## Chapter 7 - Some Weird Stuff...

- Richard Feynman
- George Spencer-Brown
- Physics Textbook as Cultural Construct
- Are Electrons Conscious?
- Teilhard de Chardin and Vernor Vinge
- Society of Mind
- Mapping and Mysticism
- Mapping and ADHD
- How The Approach Developed
- Complexity Cosmology
- The Prisoners' Dilemma, Freeware and Trust
- Predeterminism

## Appendix

- Stoned! Sites
- User Reports
- Additional Materials
- Links
- References