Programming Practices

By Joe Feliu in conjunction with Harris Kern's Enterprise Computing Institute

Description: Is programming an art or a science? This debate, dating back to the early days of software development, still makes for good water cooler discussion. What is generally accepted, however, is that certain standard operating procedures in the development and maintenance of software are essential to an effective operation of an IT organization. What constitutes programming best practices? This is a broad topic, which depends heavily on the language and operating systems that exist within your infrastructure. An in depth discussion is beyond the scope of this book. However, certain fundamental concepts apply across most environments, from basic code design through the release management process that transitions code to the production environment. In the example below, you will find a comprehensive set of topics that should be considered in constructing your programming practices and standards document.

Benefits: Among the seemingly countless benefits you can derive from developing and enforcing good programming practices are:

1. Easier code maintainability resulting from less complex, more readable and more reusable code
2. Greater programmer interchangeability as standards are followed
3. Improved productivity resulting in more timely delivery of higher quality code
4. Fewer "surprises"
5. Improved communication among development teams and with customers
6. Higher quality; reduced rework

First Steps: If there are no standard programming practices in place in your development group, take the following actions

1. Convene a team of senior developers who would be willing to engage a standards/practices effort; if facilitation is needed, contract with a expert in the field to assist the team
2. Select a small number of high impact standards/practices to start (typically, we would suggest you start with naming and coding standards, along with formal code reviews), and have the team document and disseminate them
3. Ensure that management enforces the standards/practices
4. Systematically expand the effort

Example: Perhaps the most comprehensive categorization of software development practices is included in the IEEE Software Development Professional Certification examination specification. Figure 1 includes these categories along with their relative weighting in the examination.

I. Business Practices and Engineering Economics (3-4%)

A. Engineering Economics
B. Ethics
C. Professional Practice
D. Standards

II. Software Requirements (13-15%)

A. Requirements Engineering Process
B. Requirements Elicitation
C. Requirements Analysis
D. Software Requirements Specification
E. Requirements Validation
F. Requirements Management

III. Software Design (22-24%)

A. Software Design Concepts
B. Software Architecture
C. Software Design Quality Analysis and Evaluation
D. Software Design Notations and Documentation
E. Software Design Strategies and Methods
F. Human Factors in Software Design
G. Software and System Safety

IV. Software Construction (10-12%)

A. Construction planning
B. Code design
C. Data design and management
D. Error processing
E. Source code organization
F. Code documentation
G. Construction QA
H. System integration and deployment
I. Code tuning
J. Construction tools

V. Software Testing (15-17%)

A. Types of Tests
B. Test Levels
C. Testing Strategies
D. Test Design
E. Test Coverage of Code
F. Test Coverage of Specifications

G. Test Execution
H. Test Documentation
I. Test Management

VI. Software Maintenance (3-5%)

A. Software Maintainability
B. Software Maintenance Process
C. Software Maintenance Measurement
D. Software Maintenance Planning
E. Software Maintenance Management
F. Software Maintenance Documentation

VII. Software Configuration Management (3-4%)

A. Management of SCM Process
B. Software Configuration Identification
C. Software Configuration Control
D. Software Configuration Status Accounting
E. Software Configuration Auditing
F. Software Release Management and Delivery

VIII. Software Engineering Management (10-12%)

A. Measurement
B. Organizational Management and Coordination
C. Initiation and Scope Definition
D. Planning
E. Software Acquisition
F. Enactment
G. Risk Management
H. Review and Evaluation
I. Project Close Out
J. Post-closure Activities

IX. Software Engineering Process (2-4%)

A. Process Infrastructure
B. Process Measurement
C. Process Definition
D. Qualitative Process Analysis
E. Process Implementation and Change

X. Software Engineering Tools and Methods (2-4%)

A. Management Tools and Methods
B. Development Tools and Methods
C. Maintenance Tools and Methods
D. Support Tools and Methods

XI. Software Quality (6-8%)

A. Software Quality Concepts
B. Planning for SQA and V&V
C. Methods for SQA and V&V
D. Measurement Applied to SQA and V&V