
FastCopy PROC NEAR
  cld
  cmp Fat386, 0 ; are we on a 386?
  jne FastCopy_Copy32 ; if so, do 32-bit moves.

; We have the 16-bit case in the straight-line path, optimized
; for slow machines to avoid taking excess branches.

  shr cx, 1 ; do word copies first.
  rep movsw ; copy all the words we can.
  rcl cx, 1 ; now do the last byte if there is one.
  rep movsb ; there it is.
  ret ; we’re done.

; We are running on at least a 386 machine, so do 32-bit
; transfers.

FastCopy_Copy32:
  .386
  shr cx, 1 ; (CX) = # words to copy, (CY)=# bytes.
  pushf ; save CY for # bytes to copy.
  shr cx, 1 ; (CX) = # dwords to copy, (CY)=# words.
  rep movsd ; copy doublewords.
  rcl cx, 1 ; (CX) = # words to copy.
  rep movsw ; copy one word if necessary.
  popf ; (CY) = set if we must copy one more byte.
  rcl cx, 1 ; (CX) = # bytes to copy.
  rep movsb ; copy the last byte if (CX)!-0.
  .8086
  ; BEGIN ENHANCED INSTRUCTION SET USAGE.
  ret
FastCopy ENDP

Methods for Software Quality

David Kalinsky
Echelon
Palo Alto, Calif.

David Kalinsky is training manager at Echelon Corp., and has been a popular lecturer
and seminar leader on embedded systems development technologies. He was the
director of software engineering technologies and head of technical training at Ready
Systems Corp. and cofounder of Medicom Computers (Israel). He holds an AB in
physics from Brown University and an MPhil and PhD in nuclear physics from Yale
University.
SOFTWARE QUALITY

- Responsibility of Software Developers
  - not of SQA guys
- Planned and Systematic Actions
- Adherence to Technical Requirements
- Satisfaction of Needs and Expectations
- Adherence to Objective Quality Standards

- PREREQUISITE

ENGINEERING DISCIPLINE

- Data Flow Diagrams
- Control Flow Diagrams
- Data Dictionary
- State Transition Diagrams & Matrices
- Performance/Timing Requirements

- PREREQUISITE

ENGINEERING DOCUMENTATION

- RFP
- SRS
- IRS
- SDD

- PREREQUISITE

METHODIC SOFTWARE TESTING

- Hierarchical Design
- Object-Oriented
- Tasking Modeling
- Formal Testing
- Diagrams
- Algorithm Design
- Acceptance Testing
SOFTWARE QUALITY

• Responsibility of Software Developers
  - not of SQA guys

• Planned and Systematic Actions

• Adherence to Technical Requirements

• Satisfaction of Needs and Expectations

• Adherence to Objective Quality Standards

PREREQUISITE

ENGINEERING DISCIPLINE

Data Flow Diagrams
Control Flow Diagrams
Data Dictionary
State Transition Diagrams & Matrices
Performance/Timing Requirements
Interface Requirements

Hierarchical Design
Object-Orientation
Testing Modeling
Timing Analyses
Data Design
Algorithm Design

Test Planning
Software Integration
Formal Testing
System Integration
Acceptance Testing

PREREQUISITE

ENGINEERING DOCUMENTATION

RFP
SRS
IRS

SDD

PREREQUISITE

METHODIC SOFTWARE TESTING


**SQA ACTIVITY**

**THE MAJOR REVIEWS**

**REQUIREMENTS SPECIFICATION**
- System Requirements Reviewed
- Design Requirements Reviewed

**SOFTWARE DESIGN**
- High-Level Design Reviewed
- Detailed Design Reviewed

**SQA ACTIVITY**

**CONFIGURATION MANAGEMENT**
- Configuration Identification
- Change Control
- Configuration Accounting
- Configuration Audit

**SQA ACTIVITY**

**SMALL TECHNICAL REVIEWS**
- Structured Walk-Throughs
- Fagan Inspections

**SQA ACTIVITY**

**I. U. & U.**

Independent Verification and Validation
THE MAJOR REVIEWS

- Requirements Specification
- Allocation (Software)
- Software Design
- High-Level Detailed Design
- Low-Level Detailed Design
- Software Tests

SMALL TECHNICAL REVIEWS

- Structured Walk-Throughs
- Fagan Inspections

CONFIGURATION MANAGEMENT

- Configuration Identification
- Change Control
- Configuration Accounting
- Configuration Audit

I. V. & V.

Independent Verification and Validation
• MEASUREMENT OF SOFTWARE QUALITY

MODELS AND METRICS

- MODEL = an abstraction

- METRIC = a scale for a property of a model

- MEASUREMENT = value of metric at given time.

- Musa's model:

  "Errors are emitted like radioactivity:"

  Half-Life equation!

  Reliability Grows with Use

  Meas. Time to Repair

  Cumulative Execution Time

• McCabe:

  "Software is like a graph"

  Cyclomatic Complexity:

  \[ V(G) = (\text{Edges}) - (\text{Vertices}) + 2 \cdot (\text{Pieces}) \]

  - Identify 'Spaghetti code'

  \[ V(G) = 9 - 6 + 2 = 5 \]

• MEASUREMENT OF SOFTWARE QUALITY

COMPLEXITY MODELLING

• Belady & Lehman:

  "Software is like a gas"

  -behavior analogous to Laws of Thermodynamics

  LAWS OF EVOLUTION OF LARGE SYSTEMS

  I. Law of Continuous Change
  II. Law of Increasing Entropy
  III. Law of Statistically Smooth Growth
**Measurement of Software Quality**

**Models and Metrics**

- **Model** ↔ an abstraction
- **Metric** ↔ a scale for a property of a model
- **Measurement** ↔ value of metric at given time.

**Reliability Modelling**

- Musa's model:
  
  "Errors are emitted like radioactivity:"

  Half-Life equation!

- Reliability grows with use

**Complexity Modelling**

- McCabe:
  
  "Software is like a graph"

  Cyclomatic Complexity:

  \[ V(G) = (\text{Edges}) - (\text{Vertices}) + 2^*(\text{Pieces}) \]

- Identify 'Spaghetti code'

**Maintenance of Large Systems**

- Belady & Lehman:
  
  "Software is like a gas"

  - Behavior analogous to Laws of Thermodynamics

**Laws of Evolution of Large Systems**

I. Law of Continuous Change

II. Law of Increasing Entropy

III. Law of Statistically Smooth Growth