

HUMAN TESTING, CHECKING PROCEDURES

- paraphrasing / explaining
- symbolic execution
- assume different perspectives
 role of the users,
 testers,
 maintenance staff
- playing different scenarios
- elaboration of checklists

SCENARIOS

-> playing different application-oriented user scenarios

examples:

- (1) checking the requirements for a library system
 - borrowing a book
 - returning a requested book
 - ...
- (2) code inspection of a chained list
 - inserting in an empty list
 - inserting in the middle of list
 - search for the last list element
 - ...

CHECKLISTS OF COMMON ERRORS

- topic-related
 - > OOA- oriented
 - > programming guidelines / coding standards
 - > language independent
 - > language dependent
- distribution to different inspectors, each sublist no longer than one page
- updated after each inspection

OOA - PACKET

- (1) self-contained unit
 - > topic may be handled and understood separately
 - > comprises classes logically connected
 - > supports abstraction
 - > almost no cut-off of inheritance lines, at least all superclasses belong to the same packet
 - > no cut-off of aggregations
 - > almost no cut-off of associations
- (2) suitable packet name
 - > no verbs allowed
 - > derived from the packet's description
 - > packet description less than 25 words
- (3) packet too small

OOA - CLASS

- (1) expressive class name
 - > singular substantive
 - > reflects technical terminology
 - > express the same as all attributes as a whole
 - > distinguish from all other class names
- (2) suitable abstraction level
 - > too small
 - > user interface modelled
 - > design and implementation details only

OOA - ASSOCIATION

- (1) naming necessary or useful
 - > multiple associations
 - > prefer role names (substantives) above association names (verbs)
 - > reflexive associations need role names
- (2) 1:1 association
 - really necessary if
 - > optional connection in on/both directions
 - > changes in connection possible
 - > both classes quite complex
 - > different semantics in both classes
- (3) multiple associations between two classes
 - > different semantics/cardinalities
- (4) derived associations applied correctly
- (5) associative versus stand-alone class
- (6) confusion with inheritance

OOA - ATTRIBUTE

OOA - INHERITANCE

OOA - CARDINALITIES

**OOA - SIMPLE ASSOCIATIONS,
AGGREGATION, COMPOSITION**

OOA - SCENARIO

OOA - STATE AUTOMATON

OOA - OPERATION

FAULT CLASS: DATA REFERENCE

(1) unset variables used

(2) array/string limits exceeded
in any indexing operations

(3) off-by-one errors in indexing

-> lower bound: 0, 1 or something else

-> upper bound: size of the array or size -1

(4) noninteger indexing

(5) dynamic storage allocated correctly

-> dereferencing of nil addresses

(6) dynamic storage de-allocated
if no longer required

(7) dangling references

(8) incorrect storage attributes
referenced by pointers

FAULT CLASS: DATA DECLARATION

- (1) all variables declared
- (2) default attributes understood
- (3) correct types, length, attributes (storage class) assigned
- (4) variables, arrays, strings initialized properly
- (5) any variables with similar names

FAULT CLASS: COMPUTATION

- (1) computations on nonarithmetic inconsistent variables
- (2) mixed-mode computations
- (3) computations on variables of different lengths
- (4) target size less than size of assigned value
- (5) intermediate result overflow or underflow
- (6) division by zero
- (7) variables value outside of meaningful range
- (8) operator precedence understood
- (9) integer divisions correct

FAULT CLASS: COMPARISON

- (1) comparison between inconsistent variables
- (2) mixed-mode comparison
- (3) comparison relationships correct
- (4) Boolean expressions correct
- (5) comparison and Boolean expressions mixed
- (6) operator precedence understood
- (7) compiler evaluation of Boolean expressions understood

FAULT CLASS: CONTROL FLOW

- (1) will each loop terminate
- (2) will each procedure terminate
- (3) will program terminate
- (4) any loop bypasses because of entry conditions
- (5) are possible loop fallthroughs correct
- (6) of-by-one iteration errors
- (7) do-end statements match
- (8) correctly bracketed compound statements
- (9) any nonexhaustive decisions
 - > sequence of if statements
 - > case statement

FAULT CLASS: INTERFACES

- (1) equal (correct) number of formal and actual parameters
- (2) formal and actual parameters match in type and size
- (3) right order of the parameters
- (4) formal and actual parameters match in the units system
- (5) number, attributes, and order of parameters to library functions correct
- (6) any references to parameters not associated with current point of entry
- (7) input-only parameters altered
- (8) global variables consistent across modules
- (9) modules with shared memory use the same shared memory structure

FAULT CLASS: INPUT/OUTPUT

- (1) all input variables used
- (2) file attributes correct
- (3) attributes in open statement correct
- (4) format specification matches I/O statement
- (5) all files opened before use
- (6) end-of-file conditions handled
- (7) I/O errors handled
- (8) any spelling errors in output information

FAULT CLASS: MISCELLANEOUS

- (1) any unreferenced variables
(in cross-reference listing)
- (2) attribute list as expected
- (3) any warning or informational messages
- (4) input checked for validity
- (5) all possible exceptions handled
- (6) missing functionality
-> requirement specification