BUG TAXONOMY AND STATISTICS

by Boris Beizer, as amended[1] by Otto Vinter

EXPLANATION AND NOTATION -- This document is a taxonomy for bugs. Bugs are categorized by a 4-digit number, perhaps with subnumbers using the point system; e.g., "1234.1.6." The "x" which appears is a place holder for possible future filling in of numbers as the taxonomy is (inevitably) expanded. For example:

3xxx -- structural bugs in the implemented software
  32xx -- processing bugs
    322x -- expression evaluation bugs
      3222 -- arithmetic expressions bugs
        3222.1 -- wrong operator bugs

The last digit in a set is always a 9: e.g., 9xxx, 39xx, 329x, 3229, 3226.9. This is the category to use when a finer breakdown is not available, e.g. for bugs that do not fit in the other (sub)categories:

9xxx -- other or unspecified bugs
  39xx -- other structural bugs
    329x -- other processing bugs
      3229 -- other expression evaluation bugs
        3222.9 -- other arithmetic bugs

[1] This document is an amended version of the appendix from the referenced book by Boris Beizer. Otto Vinter has performed the following changes to the original appendix: The 114x category listed in the statistics' section has been included in the taxonomy text also. Inconsistencies between category names in the statistics' section and the taxonomy text have been corrected. A superfluous SUM-column in the statistics' section has been removed. And a few small text changes have been made to fit the present document text.
Otto Vinter has made the following additions: Categories 78xx, 8111, and 8112 introduced during the PRIDE project (PRIDE Final Report, Appendix A: http://ottovinter.dk/Finalrp3.doc) have been added to the taxonomy text. And a statistics' section covering bug data from a number of projects at Brüel & Kjaer has been added. Please address questions about this document to vinter@ottovinter.dk, bbeizer@localnet.com or bsquare@sprintmail.com

BUG TAXONOMY

1xxx: **FUNCTIONAL BUGS: REQUIREMENTS AND FEATURES:** bugs having to do with requirements as specified or as implemented.

11xx: **REQUIREMENTS INCORRECT:** the requirement or a part of it is incorrect.

111x: **Incorrect:** requirement is wrong.

112x: **Undesirable:** requirement is correct as stated but it is not desirable.

113x: **Not needed:** requirement is not needed.

114x: **Ambiguous:** requirement is ambiguous.

12xx: **LOGIC:** the requirement is illogical or unreasonable.

121x: **Illogical:** illogical, usually because of a self-contradiction which can be exposed by a logical analysis of cases.

122x: **Unreasonable:** logical and consistent but unreasonable with respect to the environment and/or budgetary and time constraints.

123x: **Unachievable:** requirement fundamentally impossible or cannot be achieved under existing constraints.

124x: **Inconsistent, incompatible:** requirement is inconsistent with other requirements or with environment.

1242: **Internal:** the inconsistency is evident within the specified component.

1244: **External:** the inconsistency is with external (to the component) components or the environment.

1248: **Configuration sensitivity:** the incompatibility is with one or more configurations (hardware, software, operating system) in which the component is expected to work.

13xx: **COMPLETENESS:** the requirement as specified is either ambiguous, incomplete, or overly specified.

131x: **Incomplete:** the specification is incomplete; cases, features, variations or attributes are not specified and therefore not implemented.

132x: **Missing, unspecified:** the entire requirement is missing.

133x: **Duplicated, overlapped:** specified requirement totally or partially overlaps another requirement either already implemented or specified elsewhere.

134x: **Overly generalized:** requirement as specified is correct and consistent but is overly generalized (e.g., too powerful) for the application.

137x: **Not downward compatible:** requirement as specified will mean that objects created or manipulated by prior versions can either not be processed by this version or will be incorrectly processed.

138x: **Insufficiently extendable:** requirement as specified cannot be expanded in ways which are likely to be needed -- important hooks left out of specification.

14xx: **VERIFIABILITY:** specification bugs having to do with verifying that the requirement was correctly or incorrectly implemented.

141x: **Unverifiable:** the requirement, if implemented, cannot be verified by any means or within available time and budget. For example, it is possible to design a test but the outcome of the test cannot be verified as correct or incorrect.

142x: **Untestable:** it is not possible to design and/or execute tests which will verify the requirement. Untestable is stronger than unverifiable.

15xx: **PRESENTATION:** bugs in the presentation or documentation of requirements. The requirements are presumed to be correct, but the form in which they are presented is not. This can be important for test design automation systems which demand specific formats.

152x: **Presentation, documentation:** general presentation, documentation, format, media, etc.

153x: **Standards:** presentation violates standards for requirements.

16xx: **REQUIREMENT CHANGES:** requirements, whether or not correct, have been changed between the time programming started and testing ended.

162x: **Features:** requirement changes concerned with features.

1621: **Feature added:** a new feature has been added.

1622: **Feature deleted:** previously required feature deleted.

1623: **Feature changed:** significant changes to feature, other than changes in cases.

163x: **Cases:** cases within a feature have been changed. Feature itself is not significantly modified except for cases.

1631: **Cases added:**

1632: **Cases deleted:**

1633: **Cases changed:** processing or treatment of specific case(s) changed.

164x: **Domain changes:** input data domain modified: e.g., boundary changes, closure, treatment.

165x: **User messages and diagnostics:** changes in text, content, or conditions under which user prompts, warning, error messages, etc. are produced.

166x: **Internal interfaces:** direct internal interfaces such as call sequences, or indirect interfaces (e.g., via data structures) have been changed.

167x: **External interfaces:** external interfaces, such as device drivers, protocols, etc. have been changed.

168x: **Performance and timing:** changes to performance requirements (e.g., throughput) and/or timings.

2xxx: **FUNCTIONALITY AS IMPLEMENTED:** requirement known or assumed to be correct, implementable, and testable, but implementation is wrong.

21xx: **CORRECTNESS:** having to do with the correctness of the implementation.

211x: **Feature misunderstood, wrong:** feature as implemented is not correct -- not as specified.

218x: **Feature interactions:** feature is correctly implemented by itself, but has incorrect interactions with other features, or specified or implied interaction is incorrectly handled.

22xx: **COMPLETENESS, FEATURES:** having to do with the completeness with which features are implemented.

221x: **Missing feature:** an entire feature is missing.

222x: **Unspecified feature:** a feature not specified has been implemented.

223x: **Duplicated, overlapped feature:** feature as implemented duplicates or overlaps features implemented by other parts of the software.

23xx: **COMPLETENESS, CASES:** having to do with the completeness of cases within features.

231x: **Missing case:**

232x: **Extra case:** cases which should not have been handled are.

233x: **Duplicated, overlapped case:** duplicated handling of cases or partial overlap with other cases.

234x: **Extraneous output data:** data not required is output.

24xx: **DOMAINS:** processing case or feature depends on a combination of input values. A domain bug exists if the wrong processing is executed for the selected input-value combination.

241x: **Domain misunderstood, wrong:** misunderstanding of the size, shape, boundaries, or other characteristics of the specified input domain for the feature or case. Most bugs related to handling extreme cases are domain bugs.

242x: **Boundary locations:** the values or expressions which define a domain boundary are wrong: e.g., "X>=6" instead of "X>=3."

243x: **Boundary closures:** end points and boundaries of the domain are incorrectly associated with an adjacent domain: e.g., "X>=0" instead of "X>0."

244x: **Boundary intersections:** domain boundaries are defined by a relation between domain control variables. That relation, as implemented, is incorrect: e.g., "IF X>0 AND Y>0 .." instead of "IF X>0 OR Y>0..."

25xx: **USER MESSAGES AND DIAGNOSTICS:** user prompt or printout or other form of communication is incorrect. Processing is assumed to be correct: e.g., a false warning, failure to warn, wrong message, spelling, formats.

26xx: **EXCEPTION CONDITIONS MISHANDLED:** exception conditions such as illogicals, resource problems, failure modes, which require special handling, are not correctly handled or the wrong exception-handling mechanisms are used.

3xxx: **STRUCTURAL BUGS:** bugs related to the component's structure: i.e., the code.

31xx: **CONTROL FLOW AND SEQUENCING:** bugs specifically related to the control flow of the program or the order and extent to which things are done, as distinct from what is done.

311x: **General structure:** general bugs related to component structure.

3112: **Unachievable path:** a functionally meaningful processing path in the code for which there is no combination of input values which will force that path to be executed. Do not confuse with unreachable code. The code in question might be reached by some other path.

3114: **Unreachable code:** code for which there is no combination of input values which will cause that code to be executed.

3116: **Dead-end code:** code segments which once entered cannot be exited, even though it was intended that an exit be possible.

312x: **Control logic and predicates:** the path taken through a program is directed by control flow predicates (e.g., boolean expressions). This category addresses the implementation of such predicates.

3122: **Duplicated logic:** control logic which should appear only once is inadvertently duplicated in whole or in part.

3124: **Don't care:** improper handling of cases for which what is to be done does not matter either because the case is impossible or because it really doesn't matter: e.g., incorrectly assuming that the case is a don't-care case, failure to do case validation, not invoking the correct exception handler, improper logic simplification to take advantage of such cases.

3126: **Illogicals:** improper identification of, or processing of, illogical or impossible conditions. An illogical is stronger than a don't care. Illogicals usually mean that something bad has happened and that recovery is needed. Examples of bugs include: illogical not really so, failure to recognize illogical, invoking wrong handler, improper simplification of control logic to take advantage of the case.

3128: **Other control flow predicate bugs:** control-flow problems which can be directly attributed to the incorrect formulation of a control flow predicate: e.g., "IF A>B THEN..." instead of "IF A<B THEN..."

313x: **Case selection bug:** simple bugs in case selections, such as improperly formulated case selection expression, GOTO list, or bug in assigned GOTO.

314x: **Loops and iteration:** bugs having to do with the control of loops.

3141: **Initial value:** initial iteration value wrong: e.g., "FOR I = 3 TO 17..." instead of "FOR I = 8 TO 17."

3142: **Terminal value or condition:** value, variable, or expression used to control loop termination is incorrect: e.g., "FOR I = 1 TO 7..." instead of "FOR I = 1 TO 8."

3143: **Increment value:** value, variable, or expression used to control loop increment value is incorrect: e.g., "FOR I = 1 TO 7 STEP 2..." instead of "FOR I = 1 TO 7 STEP 5.."

3144: **Iteration variable processing:** where end points and/or increments are controlled by values calculated within the loop's scope, a bug in such calculations.

3148: **Exception exit condition:** where specified values or conditions or relations between variables force an abnormal exit to the loop, either incorrect processing of such conditions or incorrect exit mechanism invoked.

315x: **Control initialization and/or state:** bugs having to do with how the program's control flow is initialized and changes of state which affect the control flow: e.g., switches.

3152: **Control initialization:** initializing to the wrong state or failing to initialize.

3154: **Control state:** for state-determined control flows, incorrect transition to a new state from the current state: e.g., input condition X requires a transition to state B, given that the program is in state A, instead, the transition is to state C. Most incorrect GOTOs are included in this category.

316x: **Incorrect exception handling:** any incorrect invocation of a control-flow exception handler not previously categorized.

32xx: **PROCESSING:** bugs related to processing under the assumption that the control flow is correct.

321x: **Algorithmic, fundamental:** inappropriate or incorrect algorithm selected, but implemented correctly: e.g., using an incorrect approximation, using a shortcut string search algorithm that assumes string characteristics which may not apply.

322x: **Expression evaluation:** bugs having to do with the way arithmetic, boolean, string, and other expressions are evaluated.

3222: **Arithmetic:** bugs related to evaluation of arithmetic expressions.

3222.1: **Operator:** wrong arithmetic operator or function used.

3222.2: **Parentheses:** syntactically correct bug in placement of parentheses or other arithmetic delimiters.

3222.3: **Sign:** bug in use of sign.

3224: **Logical or boolean, not control:** bug in the manipulation or evaluation of boolean expressions which are not (directly) part of control-flow predicates: e.g., using wrong mask, AND instead of OR, incorrect simplification of boolean function.

3226: **String manipulation:** bug in string manipulation.

3226.1: **Beheading:** the beginning of a string is cut off when it should not have been or not cut off when it should be.

3226.2: **Curtailing:** as for beheading but for string end.

3226.3: **Concatenation order:** strings are concatenated in wrong order or concatenated when they should not be.

3226.3.1: **Append instead of precede:**

3226.3.2: **Precede instead of append:**

3226.4: **Inserting:** having to do with the insertion of one string into another.

3226.5: **Converting case:** case conversion (upper to lower, say) is incorrect.

3226.6: **Code conversion:** string is converted to another code incorrectly or not converted when it should be.

3226.7: **Packing, unpacking:** strings are incorrectly packed or unpacked

3228: **Symbolic, algebraic:** bugs in symbolic processing of algebraic expressions.

323x: **Initialization:** bugs in initialization of variables, expressions, functions, etc. used in processing, excluding initialization bugs associated with declarations and data statements and loop initialization.

324x: **Cleanup:** incorrect handling of cleanup of temporary data areas, registers, states, etc. associated with processing.

325x: **Precision, accuracy:** insufficient or excessive precision, insufficient accuracy and other bugs related to number representation system used.

326x: **Execution time:** excessive (usually) execution time for processing component.

4xxx: **DATA:** bugs in the definition, structure, or use of data.

41xx: **DATA DEFINITION, STRUCTURE, DECLARATION:** bugs in the definition, structure and initialization of data: e.g., in DATA statements. This category applies whether the object is declared statically in source code or created dynamically.

411x: **Type:** the data object type, as declared, is incorrect: e.g., integer instead of floating, short instead of long, pointer instead of integer, array instead of scalar, incorrect user-defined type.

412x: **Dimension:** for arrays and other objects which have a dimension (e.g., arrays, records, files) by which component objects can be indexed, a bug in the dimension or in the minimum or maximum dimensions, or in redimensioning statements.

413x: **Initial, default values:** bugs in the assigned initial values of the object (e.g., in DATA statements), selection of incorrect default values, or failure to supply a default value if needed.

414x: **Duplication and aliases:** bugs related to the incorrect duplication or failure to create a duplicated object.

4142: **Duplicated:** duplicated definition of an object where allowed by the syntax.

4144: **Aliases:** object is known by one or more aliases but specified alias is incorrect; object not aliased when it should have been.

415x: **Scope:** the scope, partition, or components to which the object applies is incorrectly specified.

4152: **Local should be global:** a locally defined object (e.g., within the scope of a specific component) should have been specified more globally (e.g., in COMMON).

4154: **Global should be local:** the scope of an object is too global, it should have been declared more locally.

4156: **Global/local inconsistency or conflict:** a syntactically acceptable conflict between a local and/or global declaration of an object (e.g., incorrect COMMON).

416x: **Static/dynamic resources:** related to the declaration of static and dynamically allocated resources.

4162: **Should be static resource:** resource is defined as a dynamically allocated object but should have been static (e.g., permanent).

4164: **Should be dynamic resource:** resource is defined as static but should have declared as dynamic.

4166: **Insufficient resources, space:** number of specified resources is insufficient or there is insufficient space (e.g., main memory, cache, registers, disc, etc.) to hold the declared resources.

4168: **Data overlay bug:** data objects are to be overlayed but there is a bug in the specification of the overlay areas.

42xx: **DATA ACCESS AND HANDLING:** having to do with access and manipulation of data objects that are presumed to be correctly defined.

421x: **Type:** bugs having to do with the object type.

4212: **Wrong type:** object type is incorrect for required processing: e.g., multiplying two strings.

4214: **Type transformation:** object undergoes incorrect type transformation: e.g., integer to floating, pointer to integer, specified type transformation is not allowed, required type transformation not done. Note, type transformation bugs can exist in any language, whether or not it is strongly typed, whether or not there are user-defined types.

4216: **Scaling, units:** scaling or units (semantic) associated with object is incorrect, incorrectly transformed, or not transformed: e.g., FOOT-POUNDS to STONE-FURLONGS.

422x: **Dimension:** for dynamically variable dimensions of a dimensioned object, a bug in the dimension: e.g., dynamic redimension of arrays, exceeding maximum file length, removing one more than the minimum number of records.

423x: **Value:** having to do with the value of data objects or parts thereof.

4232: **Initialization:** initialization or default value of object is incorrect. Not to be confused with initialization and default bugs in declarations. This is a dynamic initialization bug.

4234: **Constant value:** incorrect constant value for an object: e.g., a constant in an expression.

424x: **Duplication and aliases:** bugs in dynamic (run time) duplication and aliasing of objects.

4242: **Object already exists:** attempt to create an object which already exists.

4244: **No such object:** attempted reference to an object which does not exist.

426x: **Resources:** having to do with dynamically allocated resources and resource pools, in whatever memory media they exist: main, cache, disc, bulk RAM. Included are: queue blocks, control blocks, buffer blocks, heaps, files, etc.

4262: **No such resource:** referenced resource does not exist.
4264: **Wrong resource type:** wrong resource type referenced.

428x: **Access:** having to do with the access of objects as distinct from the manipulation of objects. In this context, accesses include read, write, modify, and (in some instances) create and destroy.
4281: **Wrong object accessed:** incorrect object accessed: e.g., "X := ABC33" instead of "X := ABD33."
4282: **Access rights violation:** access rights are controlled by attributes associated with the caller and the object. For example, some callers can only read the object, others can read and modify, etc. Violations of object access rights are included in this category whether or not a formal access rights mechanism exists: that is, access rights could be specified by programming conventions rather than by software.
4283: **Data-flow anomaly:** data-flow anomalies involve the sequence of accesses to an object: e.g., reading or initializing an object before it has been created, or creating and then not using.
4284: **Interlock bug:** where objects are in simultaneous use by more than one caller, interlocks and synchronization mechanisms may be used to assure that all data are current and changed by only one caller at a time. These are not bugs in the interlock or synchronization mechanism but in the use of that mechanism.
4285: **Saving or protecting bug:** application requires that the object be saved or otherwise protected at different program states, or alternatively, not protected. These are bugs related to the incorrect usage of such protection mechanisms or procedures.
4286: **Restoration bug:** application requires that a previously saved object be restored prior to processing: e.g., POP the stack, restore registers after interrupt. This category includes bugs in the incorrect restoration of data objects and not bugs in the implementation of the restoration mechanism.
4287: **Access mode, direct/indirect:** object is accessed by wrong means: e.g., direct access of an object for which indirect access is required, call by value instead of name or vice-versa, indexed instead of sequential or vice versa.
4288: **Object boundary or structure:** access to object is partly correct, but the object structure and its boundaries are handled incorrectly: e.g., fetching 8 characters of a string instead of 7, mishandling word boundaries, getting too much or too little of an object.

5xxx: **IMPLEMENTATION:** bugs having to do with the implementation of the software. Some of these, such as standards and documentation, may not affect the actual workings of the software. They are included in the bug taxonomy because of their impact on maintenance.
51xx: **CODING AND TYPOGRAPHICAL:** bugs which can be clearly attributed to simple coding and typographical bugs. Classification of a bug into this category is subjective. If a programmer believed that the correct variable, say, was "ABCD" instead of "ABCE," then it would be classified as a 4281 bug (wrong object accessed). Conversely, if E was changed to D because of a typewriting bug, then it belongs here.
511x: **Coding wild card, typographical:** all bugs which can be reasonably attributed to typing and other typographical bugs.
512x: **Instruction, construct misunderstood:** all bugs which can be reasonably attributed to a misunderstanding of an instruction's operation or HOL statement's action.

52xx: **STANDARDS VIOLATION:** bugs having to do with violating or misunderstanding the applicable programming standards and conventions. The software is assumed to work properly.

521x: **Structure violations:** violations concerning control-flow structure, organization of the software, etc.

5212: **Control flow:** violations of control-flow structure conventions: e.g., excessive IF-THEN-ELSE nesting, not using CASE statements where required, not following dictated processing order, jumping into or out of loops, jumping into or out of decisions.

5214: **Complexity:** violation of maximum (usually) or minimum (rare) complexity guidelines as measured by some specified complexity metric: e.g., too many lines of code in module, cyclomatic complexity greater than 200, excessive Halstead length, too many tokens.

5215: **Loop nesting depth:** excessive loop nesting depth.

5216: **Modularity and partition:** modularity and partition rules not followed: e.g., minimum and maximum size, object scope, functionally dictated partitions.

5217: **Call nesting depth:** violations of component (e.g., subroutine, subprogram, function) maximum nesting depth, or insufficient depth where dictated.

522x: **Data definition, declarations:** the form and/or location of data object declaration is not according to standards.

523x: **Data access:** violations of conventions governing how data objects of different kinds are to be accessed, wrong kind of object used: e.g., not using field-access macros, direct access instead of indirect, absolute reference instead of symbolic, access via register, etc.

524x: **Calling and invoking:** bug in the manner in which other processing components are called, invoked, or communicated with: e.g., direct subroutine call that should be indirect, violation of call and return sequence conventions.

526x: **Mnemonics, label conventions:** violations of the rules by which names are assigned to objects: e.g., program labels, subroutine and program names, data object names, file names.

527x: **Format:** violations of conventions governing the overall format and appearance of the source code: indentation rules, pagination, headers, ID block, special markers.

528x: **Comments:** violations of conventions governing the use, placement, density, and format of comments. The content of comments is covered by 53xx, documentation.

53xx: **DOCUMENTATION:** bugs in the documentation associated with the code or the content of comments contained in the code.

531x: **Incorrect:** documentation statement is wrong.

532x: **Inconsistent:** documentation statement is inconsistent with itself or with other statements.

533x: **Incomprehensible:** documentation cannot be understood by a qualified reader.

534x: **Incomplete:** documentation correct but important facts are missing.

535x: **Missing:** major parts of documentation are missing.

6xxx: **INTEGRATION:** bugs having to do with the integration of, and interfaces between, components. The components themselves are assumed to be correct.

61xx: **INTERNAL INTERFACES:** bugs related to the interfaces between communicating components with the program under test. The components are assumed to have passed their component level tests. In this context, direct or indirect transfer of data or control

information via a memory object such as tables, dynamically allocated resources, or files, constitute an internal interface.

611x: **Component invocation:** bugs having to do with how software components are invoked. In this sense, a "component" can be a subroutine, function, macro, program, program segment, or any other sensible processing component. Note the use of "invoke" rather than "call," because there may be no actual call as such: e.g., a task order placed on a processing queue is an invocation in our sense, though (typically) not a call.

6111: **No such component:** invoked component does not exist.

6112: **Wrong component:** incorrect component invoked.

612x: **Interface parameter, invocation:** having to do with the parameters of the invocation, their number, order, type, location, values, etc.

6121: **Wrong parameter:** parameters of the invocation are incorrectly specified.

6122: **Parameter type:** incorrect invocation parameter type used.

6124: **Parameter structure:** structural details of parameter used are incorrect: e.g., size, number of fields, subtypes.

6125: **Parameter value:** value (numerical, boolean, string) of the parameter is wrong.

6126: **Parameter sequence:** parameters of the invocation sequence in the wrong order, too many parameters, too few parameters.

613x: **Component invocation return:** having to do with the interpretation of parameters provided by the invoked component on return to the invoking component or on release of control to some other component. In this context, a record, a subroutine return sequence, or a file can qualify for this category of bug. Note that the bugs included here are *not* bugs in the component that created the return data but in the receiving component's subsequent manipulation and interpretation of that data.

6131: **Parameter identity:** wrong return parameter accessed.

6132: **Parameter type:** wrong return parameter type used. That is, the component using the return data interprets a return parameter incorrectly as to type.

6134: **Parameter structure:** return parameter structure misinterpreted.

6136: **Return sequence:** sequence assumed for return parameters is incorrect.

614x: **Initialization, state:** invoked component not initialized or initialized to the wrong state or with incorrect data.

615x: **Invocation in wrong place:** the place or state in the invoking component at which the invoked component was invoked is wrong.

616x: **Duplicate or spurious invocation:** component should not have been invoked or has been invoked more often than necessary.

62xx: **EXTERNAL INTERFACES AND TIMING:** having to do with external interfaces, such as I/O devices and/or drivers, or other software not operating under the same control structure. Data passage by files or messages qualify for this bug category.

621x: **Interrupts:** bugs related to incorrect interrupt handling or setting up for interrupts: e.g., wrong handler invoked, failure to block or unblock interrupts.

622x: **Devices and drivers:** incorrect interface with devices or device drivers or incorrect interpretation of return status data.

6222: **Device, driver, initialization or state:** incorrect initialization of device or driver, failure to initialize, setting device to the wrong state.

6224: **Device, driver, command bug:** bug in the command issued to a device or driver.

6226: **Device, driver, return/status misinterpretation:** return status data from device or driver misinterpreted or ignored.

623x: **I/O timing or throughput:** bugs having to do with timings and data rates for external devices such as: not meeting specified timing requirements (too long or too short), forcing too much throughput, not accepting incoming data rates.

7xxx: **SYSTEM AND SOFTWARE ARCHITECTURE:** bugs that are not attributable to a component or to the interface between components but affect the entire software system or stem from architectural errors in the system.

71xx: **OS bug:** bugs related to the use of operating system facilities. Not to be confused with bugs in the operating system itself.

711x: **Invocation, command:** erroneous command given to operating system or OS facility incorrectly invoked.

712x: **Return data, status misinterpretation:** data returned from operating system or status information ignored or misinterpreted.

714x: **Space:** required memory (cache, disc, RAM) resource not available or requested in wrong way.

72xx: **Software architecture:** architectural problems not elsewhere defined.

721x: **Interlocks and semaphores:** bugs in the use of interlock mechanisms and interprocess communications facilities. Not to be confused with bugs in these mechanisms themselves: e.g., failure to lock, failure to unlock, failure to set or reset semaphore, duplicate locking.

722x: **Priority:** bugs related to task priority: e.g., priority too low or too high, priority selected not allowed, priority conflicts.

723x: **Transaction-flow control:** where the path taken by a transaction through the system is controlled by an implicit or explicit transaction flow-control mechanism, these are bugs related to the definition of such flows. Note that all components and their interfaces could be correct but this class of bug could still exist.

724x: **Resource management and control:** bugs related to the management of dynamically allocated shared resource objects: e.g., not returning a buffer block after use, not getting an object, failure to clean up an object after use, getting wrong kind of object, returning object to wrong pool.

725x: **Recursive calls:** bugs in the use of recursive invocation of software components or incorrect recursive invocation.

726x: **Reentrance:** bugs related to reentrance of program components: e.g., a reentrant component which should not be reentrant, a nonreentrant component which should be, a reentrant call which should be nonreentrant.

73xx: **RECOVERY, ACCOUNTABILITY:** bugs related to the recovery of objects after failure and to the accountability for objects despite failures.

74xx: **PERFORMANCE:** bugs related to the throughput-delay behavior of software under the assumption that all other aspects are correct.

741x: **Throughput inadequate:**

742x: **Response time, delay:** response time to incoming events too long at specified load or too short (rare), delay between outgoing events too long or too short.

743x: **Insufficient users:** maximum specified number of simultaneous users or task cannot be accommodated at specified transaction delays.

748x: **Performance parasites:** any bug whose primary or only symptom is a performance degradation: e.g., the harmless but needless repetition of operations, fetching and returning more dynamic resources than needed.

75xx: **INCORRECT DIAGNOSTIC, EXCEPTION:** diagnostic or error message incorrect or misleading. Exception handler invoked is wrong.

76xx: **PARTITIONS AND OVERLAYS:** memory or virtual memory is incorrectly partitioned, overlay to wrong area, overlay or partition conflicts.

77xx: **SYSGEN OR ENVIRONMENT:** wrong operating system version, incorrect system generation, or other host environment problem.

78xx: **EXTERNAL AND OTHER THIRD-PARTY SOFTWARE:** bugs in the interface to third-party software or other software developed externally. Due to a misunderstanding or wrong interpretation of the features and operation of the third-party software; or due to problems in the third-party software which the vendor does not correct.

8xxx: **TEST DEFINITION OR EXECUTION BUGS:** bugs in the definition, design, execution of tests or the data used in tests. These are as important as "real" bugs.

81xx: **DESIGN BUGS:** bugs in the design of tests.

811x: **Requirements misunderstood:**

8111: **Genuine misunderstanding:** test and component are mismatched because test designer did not understand requirements.

8112: **Suggestion for improvement:** ideas or suggestions for another specification than the one correctly implemented. Proposals for improvements which are not accepted on the current version of the product, but which may be included in a later version.

812x: **Incorrect outcome predicted:** predicted outcome of test does not match required or actual outcome.

813x: **Incorrect path predicted:** outcome is correct but achieved by wrong predicted path. The test is only coincidentally correct.

814x: **Test initialization:** specified initial conditions for test are wrong.

815x: **Test data structure or value:** data objects used in tests or their values are wrong.

816x: **Sequencing bug:** the sequence in which tests are to be executed, relative to other tests or to test initialization, is wrong.

817x: **Configuration:** the hardware and/or software configuration and/or environment specified for the test is wrong.

818x: **Verification method, criteria:** the method by which the outcome will be verified is incorrect or impossible.

82xx: **EXECUTION BUGS:** bugs in the execution of tests as contrasted with bugs in their design.

821x: **Initialization:** tested component not initialized to the right state or values.

822x: **Keystroke or command:** simple keystroke or button hit error.

823x: **Data base:** data base used to support the test wrong.

824x: **Configuration:** configuration and/or environment specified for the test was not used during the run.

828x: **Verification act:** the act of verifying the outcome was incorrectly executed.

83xx: **TEST DOCUMENTATION:** documentation of test case or verification criteria is incorrect or misleading.

84xx: **TEST CASE COMPLETENESS:** cases required to achieve specified coverage criteria missing.

BUG STATISTICS
by Boris Beizer

This is a spreadsheet dump for combined bug statistics gathered from many different sources by Boris Beizer. The sources are credited in the book: SOFTWARE TESTING TECHNIQUES, second edition, Van Nostrand Reinhold, New York, 1990.

The data were collected before 1989. Primarily US defense, aerospace, and telecommunications companies provided the data. The software is typically Systems Software, e.g. operating systems, compilers, and utility software etc. The programming languages comprise: Assembler, FORTRAN, COBOL, ADA, and C.

The columns denote:

1. *Category Number* -- e.g., 1xxx, 11xx, 119x....
2. *Bug Category Name* -- e.g., requirements incorrect.
3. *Bug Category Count* -- e.g., 111x has a count of 222.

4-6. *Subgroup Count* -- e.g., 11xx = 649, 13xx = 224, 15xx = 13.

5-7. *Group Count* -- e.g., 1xxx=1317, 2xxx=2624.
Note that category, subgroup, and group counts progress from left to right. How many columns are used depends how fine the group breakdown is.

8. *Total Percentage*.
The column shows the percentage of this bug category compared to the total number of bugs: e.g., 1xxx = 8.12% of the whole, 11xx = 4.00% and 111x = 1.37%.

9. *Group Percentages*.
The column percentages relative to the group. You can determine the group by going up to the nearest 100% entry: e.g., 11xx (incorrect requirements) is 49.28% of all requirements bugs, while ambiguous requirements are 0.15% of all requirements bugs.

10. *Subgroup Percentages*.
As for group percentages, except for subgroups: e.g., case completeness (23xx) can be subdivided into missing (75.13%), duplicated or overlapped (5.18%), extraneous output (18.65%), and miscellaneous (1.04%).

		TOTAL	SUM1	SUM2	SUM3	SUM4	Percent of total	Percent Group	Percent Subgroup
	KLOC(WITH COMMENTS)	6877.26							
	TOTAL NUMBER OF BUGS	16209	16209	16209	16209	16209			
	BUGS/KLOC	2.36							
1xxx	FUNCTIONAL BUGS: REQUIREMENTS AND FEATURES				1317	1317	8.12%	100.00%	
11xx	REQUIREMENTS INCORRECT		649	649			4.00%	49.28%	100.00%
111x	Incorrect	222					1.37%	16.86%	34.21%
114x	Ambiguous	2					0.01%	0.15%	0.31%
119x	Other incorrect requirements	425					2.62%	32.27%	65.49%
12xx	REQUIREMENTS LOGIC			153			0.94%	11.62%	100.00%
124x	Inconsistent, incompatible		62				0.38%	4.71%	40.52%
1249	Other inconsistencies	62					0.38%	4.71%	40.52%
129x	Other requir. logic problems	91	91				0.56%	6.91%	59.48%
13xx	REQUIREMENTS COMPLETENESS		224	224			1.38%	17.01%	100.00%
131x	Incomplete	138					0.85%	10.48%	61.61%
132x	Missing, unspecified	84					0.52%	6.38%	37.50%
139x	Other completeness problems	2					0.01%	0.15%	0.89%
15xx	REQUIREMENTS PRESENTATION		13	13			0.08%	0.99%	100.00%
152x	Presentation, documentation	1					0.01%	0.08%	7.69%
159x	Other presentation problems	12					0.07%	0.91%	92.31%
16xx	REQUIREMENT CHANGES			278			1.72%	21.11%	100.00%
162x	Features		175				1.08%	13.29%	62.95%
1621	Feature added	37					0.23%	2.81%	13.31%
1622	Feature deleted	3					0.02%	0.23%	1.08%
1623	Feature changed	110					0.68%	8.35%	39.57%
1629	Other feature changes	25					0.15%	1.90%	8.99%
164x	Domain changes	5	5				0.03%	0.38%	1.80%
165x	User messages and diagnostics	8	8				0.05%	0.61%	2.88%
167x	External interfaces	22	22				0.14%	1.67%	7.91%
169x	Other requirements changes	68	68				0.42%	5.16%	24.46%
2xxx	FUNCTIONALITY AS IMPLEMENTED			2624	2624	2624	16.19%	100.00%	
21xx	CORRECTNESS		456				2.81%	17.38%	100.00%
211x	Feature misunderstood, wrong	70					0.43%	2.67%	15.35%
218x	Feature interactions	32					0.20%	1.22%	7.02%
219x	Other feature bugs	354					2.18%	13.49%	77.63%
22xx	COMPLETENESS, FEATURES		231				1.43%	8.80%	100.00%
221x	Missing feature	56					0.35%	2.13%	24.24%
223x	Duplicated, overlapped feat.	155					0.96%	5.91%	67.10%
229x	Other feature completeness	20					0.12%	0.76%	8.66%
23xx	COMPLETENESS, CASES		193				1.19%	7.36%	100.00%
231x	Missing case	145					0.89%	5.53%	75.13%
233x	Duplicated, overlapped case	10					0.06%	0.38%	5.18%
234x	Extraneous Output data	36					0.22%	1.37%	18.65%
239x	Other case completeness bugs	2					0.01%	0.08%	1.04%
24xx	DOMAIN BUGS		778				4.80%	29.65%	100.00%
241x	Domain misunderstood, wrong	306					1.89%	11.66%	39.33%
242x	Boundary locations	457					2.82%	17.42%	58.74%
243x	Boundary closure	11					0.07%	0.42%	1.41%
249x	Other domain bugs	4					0.02%	0.15%	0.51%
25xx	USER MESSAGES AND DIAGNOSTICS	857	857				5.29%	32.66%	100.00%
26xx	EXCEPTION CONDITION MISHANDLED	79	79				0.49%	3.01%	100.00%
29xx	OTHER FUNCTIONAL BUGS	30	30				0.19%	1.14%	100.00%

		TOTAL	SUM1	SUM2	SUM3	SUM4	Percent of total	Percent Group	Percent Subgroup
3xxx	STRUCTURAL BUGS					4082	25.18%	100.00%	
31xx	CONTROL FLOW AND SEQUENCING			2078	2078		12.82%	50.91%	100.00%
311x	General structure		155				0.96%	3.80%	7.47%
3119	Other general structure	155					0.96%	3.80%	7.47%
312x	Control logic and predicates		561				3.46%	13.74%	26.99%
3128	Other control-flow predicate bugs	268					1.65%	6.56%	12.90%
3129	Other control logic and predicate bugs	293					1.81%	7.18%	14.10%
314x	Loops and iterations		120				0.74%	2.94%	5.77%
3142	Terminal value or condition	54					0.33%	1.32%	2.60%
3144	Iteration variable processing	1					0.01%	0.02%	0.05%
3149	Other loop and iteration	65					0.40%	1.59%	3.13%
315x	Control initialization and/or state		10				0.06%	0.24%	0.48%
3159	Other control state bugs	10					0.06%	0.24%	0.48%
319x	Other control flow and sequencing	1232	1232				7.60%	30.18%	59.28%
32xx	PROCESSING				2004		12.36%	49.09%	100.00%
321x	Algorithmic, fundamental	121	121	121			0.75%	2.96%	6.04%
322x	Expression evaluation			445			2.75%	10.90%	22.21%
3222	Arithmetic expressions		278				1.72%	6.81%	13.87%
3222.3	Sign	30					0.19%	0.73%	1.50%
3222.9	Other arithmetic	248					1.53%	6.08%	12.38%
3224	Logic or boolean, not control	167	167				1.03%	4.09%	8.33%
323x	Initialization	303	303	303			1.87%	7.42%	15.12%
324x	Cleanup	10	10	10			0.06%	0.24%	0.50%
325x	Precision, accuracy	88	88	88			0.54%	2.16%	4.39%
326x	Execution time	47	47	47			0.29%	1.15%	2.35%
329x	Other processing	990	990	990			6.11%	24.25%	49.40%
4xxx	DATA				3638	3638	22.44%	100.00%	
41xx	DATA DEFINITION, STRUCTURE, DECLARATION			1805			11.14%	49.62%	100.00%
413x	Initial, default value	157	157				0.97%	4.32%	8.70%
414x	Duplication and aliases		11				0.07%	0.30%	0.61%
4149	Other duplication and aliases	11					0.07%	0.30%	0.61%
419x	Other data definition, structure, declaration bugs	1637	1637				10.10%	45.00%	90.69%
42xx	DATA ACCESS AND HANDLING			1831			11.30%	50.33%	100.00%
421x	Type		359				2.21%	9.87%	19.61%
4212	Wrong type	10					0.06%	0.27%	0.55%
4214	Type transformation	84					0.52%	2.31%	4.59%
4216	Scaling, units	237					1.46%	6.51%	12.94%
4219	Other type bugs	28					0.17%	0.77%	1.53%
423x	Value		236				1.46%	6.49%	12.89%
4232	Initialization	236					1.46%	6.49%	12.89%
424x	Duplication and aliases		10				0.06%	0.27%	0.55%
4249	Other duplication and aliases	10					0.06%	0.27%	0.55%
426x	Resources		11				0.07%	0.30%	0.60%
4269	Other dynamic resource	11					0.07%	0.30%	0.60%
428x	Access		677				4.18%	18.61%	36.97%
4281	Wrong object accessed	244					1.51%	6.71%	13.33%
4282	Access rights violation	8					0.05%	0.22%	0.44%
4283	Data-flow anomaly	115					0.71%	3.16%	6.28%
4285	Saving or protecting bug	10					0.06%	0.27%	0.55%
4287	Access mode, direct/indirect	113					0.70%	3.11%	6.17%
4288	Object boundary or structure	115					0.71%	3.16%	6.28%
4289	Other access bug	72					0.44%	1.98%	3.93%
429x	Other access and handling	538	538				3.32%	14.79%	29.38%
49xx	OTHER DATA BUGS	2	2	2			0.01%	0.05%	100.00%

		TOTAL	SUM1	SUM2	SUM3	SUM4	Percent of total	Percent Group	Percent Subgroup
5xxx	IMPLEMENTATION				1601	1601	9.88%	100.00%	
51xx	CODING AND TYPOGRAPHICAL		322	322			1.99%	20.11%	100.00%
511x	Coding wild card, typographical	26					0.16%	1.62%	8.07%
519x	Other, general coding bugs	296					1.83%	18.49%	91.93%
52xx	STANDARDS VIOLATION			318			1.96%	19.86%	100.00%
527x	Format	15	15				0.09%	0.94%	4.72%
528x	Comments	68	68				0.42%	4.25%	21.38%
529x	Other standards and style violation bugs	235	235				1.45%	14.68%	73.90%
53xx	DOCUMENTATION		960	960			5.92%	59.96%	100.00%
531x	Incorrect	550					3.39%	34.35%	57.29%
532x	Inconsistent	9					0.06%	0.56%	0.94%
534x	Incomplete	146					0.90%	9.12%	15.21%
539x	Other documentation, general	255					1.57%	15.93%	26.56%
59xx	OTHER IMPLEMENTATION	1	1	1			0.01%	0.06%	100.00%
6xxx	INTEGRATION				1455	1455	8.98%	100.00%	
61xx	INTERNAL INTERFACES			859			5.30%	59.04%	100.00%
611x	Component invocation		27				0.17%	1.86%	3.14%
6119	Other component invocation	27					0.17%	1.86%	3.14%
612x	Interface parameter, invocation		128				0.79%	8.80%	14.90%
6121	Wrong parameter	75					0.46%	5.15%	8.73%
6126	Parameter sequence	2					0.01%	0.14%	0.23%
6129	Other invocation param. bugs	51					0.31%	3.51%	5.94%
613x	Component return		70				0.43%	4.81%	8.15%
6131	Parameter identity wrong	37					0.23%	2.54%	4.31%
6139	Other parameter bugs on return	33					0.20%	2.27%	3.84%
614x	Initialization, state	221	221				1.36%	15.19%	25.73%
619x	Other, internal interfaces	413	413				2.55%	28.38%	48.08%
62xx	EXTERNAL INTERFACES AND TIMING			518			3.20%	35.60%	100.00%
621x	Interrupts	94	94				0.58%	6.46%	18.15%
622x	Devices and drivers		137				0.85%	9.42%	26.45%
6222	Device, driver, initialization or state	3					0.02%	0.21%	0.58%
6224	Device, driver, command bug	28					0.17%	1.92%	5.41%
6226	Device, driver, return/status misinterpretation	24					0.15%	1.65%	4.63%
6229	Other, devices and drivers	82					0.51%	5.64%	15.83%
623x	I/O timing or throughput	23	23				0.14%	1.58%	4.44%
629x	Other, external interfaces and timing	264	264				1.63%	18.14%	50.97%
69xx	OTHER INTEGRATION	78	78	78			0.48%	5.36%	100.00%
7xxx	SYSTEM, SOFTWARE ARCHITECTURE			282	282	282	1.74%	100.00%	
71xx	O/S CALL, USE BUG		47				0.29%	16.67%	100.00%
711x	Invocation, command	5					0.03%	1.77%	10.64%
714x	Space	3					0.02%	1.06%	6.38%
719x	Other OS call, use bugs	39					0.24%	13.83%	82.98%
72xx	SOFTWARE ARCHITECTURE		139				0.86%	49.29%	100.00%
721x	Interlocks and semaphores	56					0.35%	19.86%	40.29%
724x	Resource Management and Control	8					0.05%	2.84%	5.76%
729x	General software architecture	75					0.46%	26.60%	53.96%
73xx	RECOVERY AND ACCOUNTABILITY	4	4				0.02%	1.42%	100.00%
74xx	PERFORMANCE		64				0.39%	22.70%	100.00%
742x	Response time, delay	44					0.27%	15.60%	68.75%
749x	Other performance, unspecified	20					0.12%	7.09%	31.25%
75xx	INCORRECT DIAGNOSTIC, EXCEPTION	16	16				0.10%	5.67%	100.00%
76xx	PARTITIONS AND OVERLAYS	3	3				0.02%	1.06%	100.00%
77xx	SYSGEN OR ENVIRONMENT	9	9				0.06%	3.19%	100.00%

		TOTAL	SUM1	SUM2	SUM3	SUM4	Percent of total	Percent Group	Percent Subgroup
8xxx	TEST DEFINITION OR EXECUTION			447	447	447	2.76%	100.00%	
81xx	TEST DESIGN BUGS		11				0.07%	2.46%	100.00%
811x	Requirements misunderstood	3					0.02%	0.67%	27.27%
819x	Other test design bugs	8					0.05%	1.79%	72.73%
82xx	TEST EXECUTION BUGS		355				2.19%	79.42%	100.00%
823x	Data base, wrong	6					0.04%	1.34%	1.69%
824x	Configuration	66					0.41%	14.77%	18.59%
828x	Verification act incorrect	28					0.17%	6.26%	7.89%
829x	Other test execution bugs	255					1.57%	57.05%	71.83%
83xx	TEST DOCUMENTATION	11	11				0.07%	2.46%	100.00%
84xx	TEST CASE COMPLETENESS	64	64				0.39%	14.32%	100.00%
89xx	OTHER TEST DEFINITION OR EXECUTION BUGS	6	6				0.04%	1.34%	100.00%
9xxx	OTHER BUGS, UNSPECIFIED	763	763	763	763	763	4.71%	100.00%	100.00%

BUG STATISTICS
by Otto Vinter

This is a spreadsheet dump for the bug statistics gathered by Otto Vinter from different projects at Brüel & Kjaer in Denmark, a leading manufacturer of high-precision measurement instruments for sound and vibration measurement applications (www.bk.dk). KLOCs and BUGS/KLOC cannot be published in this statistic.

The data were collected from 1992 – 1998. The type of software ranges from embedded real-time systems to MS Windows NT applications. The programming languages comprise: Pascal, C, and C++. The project sizes range from 50 – 100 KLOCs. They were developed by 5-7 persons in 12 – 18 months, with an average of seven person years.

The columns denote:

1. *Category Number* -- e.g., 1xxx, 11xx, 119x....
2. *Bug Category Name* -- e.g., requirements incorrect.
3. *Bug Category Count* -- e.g., 111x has a count of 222.

4-6. *Subgroup Count* -- e.g., 11xx = 649, 13xx = 224, 15xx = 13.

5-7. *Group Count* -- e.g., 1xxx=1317, 2xxx=2624.
Note that category, subgroup, and group counts progress from left to right. How many columns are used depends how fine the group breakdown is.

8. *Total Percentage*.
The column shows the percentage of this bug category compared to the total number of bugs: e.g., 1xxx = 8.12% of the whole, 11xx = 4.00% and 111x = 1.37%.

9. *Group Percentages*.
The column percentages relative to the group. You can determine the group by going up to the nearest 100% entry: e.g., 11xx (incorrect requirements) is 49.28% of all requirements bugs, while ambiguous requirements are 0.15% of all requirements bugs.

10. *Subgroup Percentages*.
As for group percentages, except for subgroups: e.g., case completeness (23xx) can be subdivided into missing (75.13%), duplicated or overlapped (5.18%), extraneous output (18.65%), and miscellaneous (1.04%).

		TOTAL	SUM1	SUM2	SUM3	SUM4	Percent of total	Percent Group	Percent Subgroup
	KLOC(WITH COMMENTS)	-							
	TOTAL NUMBER OF BUGS	982	982	982	982	982			
	BUGS/KLOC	-							
1xxx	REQUIREMENTS AND FEATURES				177	177	18,0%	100,0%	
11xx	REQUIREMENTS INCORRECT		5	5			0,5%	2,8%	100,0%
112x	Undesirable	5					0,5%	2,8%	100,0%
12xx	LOGIC		12	12			1,2%	6,8%	100,0%
122x	Unreasonable	3					0,3%	1,7%	25,0%
123x	Unachievable	6					0,6%	3,4%	50,0%
124x	Inconsistent	3					0,3%	1,7%	25,0%
13xx	COMPLETENESS		65	65			6,6%	36,7%	100,0%
131x	Incomplete	31					3,2%	17,5%	47,7%
132x	Missing, Unspecified	31					3,2%	17,5%	47,7%
133x	Duplicated,overlapped	1					0,1%	0,6%	1,5%
134x	Overly Generalized	1					0,1%	0,6%	1,5%
139x	Completeness, other	1					0,1%	0,6%	1,5%
16xx	REQUIREMENT CHANGES			95			9,7%	53,7%	100,0%
162x	Features		50				5,1%	28,2%	52,6%
1621	Feature Added	29					3,0%	16,4%	30,5%
1622	Feature Deleted	2					0,2%	1,1%	2,1%
1623	Feature Changed	19					1,9%	10,7%	20,0%
163x	Cases		10				1,0%	5,6%	10,5%
1631	Case Added	5					0,5%	2,8%	5,3%
1633	Case Changed	4					0,4%	2,3%	4,2%
1639	Other cases	1					0,1%	0,6%	1,1%
164x	Domain Changes	10	10				1,0%	5,6%	10,5%
165x	User Messages and Diagnostics	13	13				1,3%	7,3%	13,7%
167x	External Interfaces	2	2				0,2%	1,1%	2,1%
168x	Performance and Timing	6	6				0,6%	3,4%	6,3%
169x	Requirements changes, other	4	4				0,4%	2,3%	4,2%
2xxx	FUNCTIONALITY AS IMPLEMENTED			294	294	294	29,9%	100,0%	
21xx	CORRECTNESS		129				13,1%	43,9%	100,0%
211x	Feature misunderstood, Wrong	82					8,4%	27,9%	63,6%
218x	Feature interactions	47					4,8%	16,0%	36,4%
22xx	COMPLETENESS, FEATURES		28				2,9%	9,5%	100,0%
221x	Missing	24					2,4%	8,2%	85,7%
222x	Unspecified	4					0,4%	1,4%	14,3%
23xx	Completeness, Cases		77				7,8%	26,2%	100,0%
231x	Missing	72					7,3%	24,5%	93,5%
232x	Extra	4					0,4%	1,4%	5,2%
234x	Extraneous output data	1					0,1%	0,3%	1,3%
24xx	DOMAINS		38				3,9%	12,9%	100,0%
241x	Domain misunderstood, Wrong	27					2,7%	9,2%	71,1%
242x	Boundary locations	3					0,3%	1,0%	7,9%
243x	Boundary closures	7					0,7%	2,4%	18,4%
244x	Boundary intersections	1					0,1%	0,3%	2,6%
25xx	USER MESSAGES AND DIAGNOSTICS	17	17				1,7%	5,8%	100,0%
26xx	EXCEPTION CONDITIONS MISHANDLED	5	5				0,5%	1,7%	100,0%

		TOTAL	SUM1	SUM2	SUM3	SUM4	Percent of total	Percent Group	Percent Subgroup
3xxx	STRUCTURAL BUGS					165	16,8%	100,0%	
31xx	CONTROL FLOW AND SEQUENCING			54	54		5,5%	32,7%	100,0%
311x	General structure		17				1,7%	10,3%	31,5%
3112	Unachievable path	3					0,3%	1,8%	5,6%
3114	Unreachable code	3					0,3%	1,8%	5,6%
3119	Other general structure bugs	11					1,1%	6,7%	20,4%
312x	Control Logic and Predicates		17				1,7%	10,3%	31,5%
3128	Other control flow predicate bugs	7					0,7%	4,2%	13,0%
3129	Other logic and predicate bugs	10					1,0%	6,1%	18,5%
313x	Case Selection Bug	8	8				0,8%	4,8%	14,8%
314x	Loops and Iteration		5				0,5%	3,0%	9,3%
3142	Terminal value or condition	4					0,4%	2,4%	7,4%
3144	Iteration variable processing	1					0,1%	0,6%	1,9%
315x	Control Initialization		4				0,4%	2,4%	7,4%
3152	Control initialization	3					0,3%	1,8%	5,6%
3159	Other control initialization	1					0,1%	0,6%	1,9%
316x	Incorrect Exception Handling	3	3				0,3%	1,8%	5,6%
32xx	PROCESSING				111		11,3%	67,3%	100,0%
321x	Algoritmic, fundamental	13	13	13			1,3%	7,9%	11,7%
322x	Expression evaluation			13			1,3%	7,9%	11,7%
3222	Arithmetic	1	1				0,1%	0,6%	0,9%
3224	Logic or Boolean, not control	6	6				0,6%	3,6%	5,4%
3226	String manipulation		5				0,5%	3,0%	4,5%
3226.2	Curtailing	1					0,1%	0,6%	0,9%
3226.6	Code conversion	1					0,1%	0,6%	0,9%
3226.9	Other string manipulation bugs	3					0,3%	1,8%	2,7%
3229	Other expression evaluation bugs	1	1				0,1%	0,6%	0,9%
323x	Initialization	32	32	32			3,3%	19,4%	28,8%
324x	Cleanup	43	43	43			4,4%	26,1%	38,7%
325x	Precison, accuracy	3	3	3			0,3%	1,8%	2,7%
39xx	OTHER STRUCTURAL BUGS	7	7	7			0,7%	4,2%	6,3%
4xxx	DATA				81	81	8,2%	100,0%	
41xx	DATA DEFINITION, STRUCTURE, DECLARATION			37			3,8%	45,7%	100,0%
411x	Type	4	4				0,4%	4,9%	10,8%
412x	Dimension	2	2				0,2%	2,5%	5,4%
413x	Initial, default values	28	28				2,9%	34,6%	75,7%
415x	Scope		3				0,3%	3,7%	8,1%
4152	Local should be global	1					0,1%	1,2%	2,7%
4154	Global should be local	1					0,1%	1,2%	2,7%
4156	Global/local inconsistency or conflict	1					0,1%	1,2%	2,7%
42xx	DATA ACCESS AND HANDLING			43			4,4%	53,1%	100,0%
421x	Type		5				0,5%	6,2%	11,6%
4212	Wrong type	1					0,1%	1,2%	2,3%
4216	Scaling, units	4					0,4%	4,9%	9,3%
423x	Value		5				0,5%	6,2%	11,6%
4232	Initialization	5					0,5%	6,2%	11,6%
428x	Access		33				3,4%	40,7%	76,7%
4281	Wrong object accessed	22					2,2%	27,2%	51,2%
4283	Data-flow anomaly	10					1,0%	12,3%	23,3%
4289	Other access bugs	1					0,1%	1,2%	2,3%
49xx	OTHER DATA BUGS	1	1	1			0,1%	1,2%	100,0%

		TOTAL	SUM1	SUM2	SUM3	SUM4	Percent of total	Percent Group	Percent Subgroup
5xxx	IMPLEMENTATION			40	40	40	4,1%	100,0%	
51xx	CODING AND TYPOGRAPHICAL	1	1				0,1%	2,5%	100,0%
52xx	STANDARDS VIOLATION		2				0,2%	5,0%	100,0%
522x	Data definition, declaration	2					0,2%	5,0%	100,0%
53xx	DOCUMENTATION		37				3,8%	92,5%	100,0%
531x	Incorrect	8					0,8%	20,0%	21,6%
532x	Inconsistent	2					0,2%	5,0%	5,4%
533x	Incomprehensible	1					0,1%	2,5%	2,7%
534x	Incomplete	8					0,8%	20,0%	21,6%
535x	Missing	4					0,4%	10,0%	10,8%
539x	Other documentation bugs	14					1,4%	35,0%	37,8%
6xxx	INTEGRATION				19	19	1,9%	100,0%	
61xx	INTERNAL INTERFACES			10			1,0%	52,6%	100,0%
611x	Component Invocation	1	1				0,1%	5,3%	10,0%
612x	Interface Parameter, Invocation		2				0,2%	10,5%	20,0%
6122	Parameter type	1					0,1%	5,3%	10,0%
6129	Other parameter invocation	1					0,1%	5,3%	10,0%
613x	Component Invocation Return		3				0,3%	15,8%	30,0%
6132	Parameter type	1					0,1%	5,3%	10,0%
6139	Other component invocation return	2					0,2%	10,5%	20,0%
619x	Other internal interface bugs	4	4				0,4%	21,1%	40,0%
62xx	EXTERNAL INTERFACES AND TIMING			8			0,8%	42,1%	100,0%
621x	Interrupts	2	2				0,2%	10,5%	25,0%
622x	Devices and Drivers		1				0,1%	5,3%	12,5%
6222	Device, driver, initialization or state	1					0,1%	5,3%	12,5%
623x	I/O Timing or Throughput	2	2				0,2%	10,5%	25,0%
629x	Other external interface bugs	3	3				0,3%	15,8%	37,5%
69xx	OTHER INTEGRATION BUGS	1	1	1			0,1%	5,3%	100,0%
7xxx	SYSTEM AND SOFTWARE ARCHITECTURE			61	61	61	6,2%	100,0%	
71xx	OS bug		2				0,2%	3,3%	100,0%
711x	Invocation, command	1					0,1%	1,6%	50,0%
719x	Other OS bugs	1					0,1%	1,6%	50,0%
72xx	SOFTWARE ARCHITECTURE		19				1,9%	31,1%	100,0%
721x	Interlocks and semaphores	9					0,9%	14,8%	47,4%
722x	Priority	5					0,5%	8,2%	26,3%
724x	Resource management and control	1					0,1%	1,6%	5,3%
726x	Reentrance	2					0,2%	3,3%	10,5%
729x	Other architecture bugs	2					0,2%	3,3%	10,5%
74xx	PERFORMANCE		11				1,1%	18,0%	100,0%
741x	Throughput inadequate	2					0,2%	3,3%	18,2%
742x	Response time, delay	5					0,5%	8,2%	45,5%
749x	Other performance bugs	4					0,4%	6,6%	36,4%
77xx	SYSGEN OR ENVIRONMENT	14	14				1,4%	23,0%	100,0%
78xx	EXTERNAL SOFTWARE, THIRD PARTY	15	15				1,5%	24,6%	100,0%
8xxx	TEST DEFINITION AND EXECUTION BUGS				86	86	8,8%	100,0%	
81xx	DESIGN BUGS			71			7,2%	82,6%	100,0%
811x	Requirements misunderstood		66				6,7%	76,7%	93,0%
8111	Genuine misunderstanding	27					2,7%	31,4%	38,0%
8112	Proposal for improvement	26					2,6%	30,2%	36,6%
8119	Other requirements misunderstood	13					1,3%	15,1%	18,3%
812x	Incorrect Outcome Predicted	1	1				0,1%	1,2%	1,4%
814x	Test initialization	1	1				0,1%	1,2%	1,4%
817x	Configuration	2	2				0,2%	2,3%	2,8%
818x	Verification method, criteria	1	1				0,1%	1,2%	1,4%
83xx	TEST DOCUMENTATION	1	1	1			0,1%	1,2%	100,0%
89xx	OTHER TEST BUGS	14	14	14			1,4%	16,3%	100,0%
9xxx	OTHER BUGS, UNSPECIFIED	59	59	59	59	59	6,0%	100,0%	