Newtech Software's HEURISTICS
Code Analyzer For C / C++

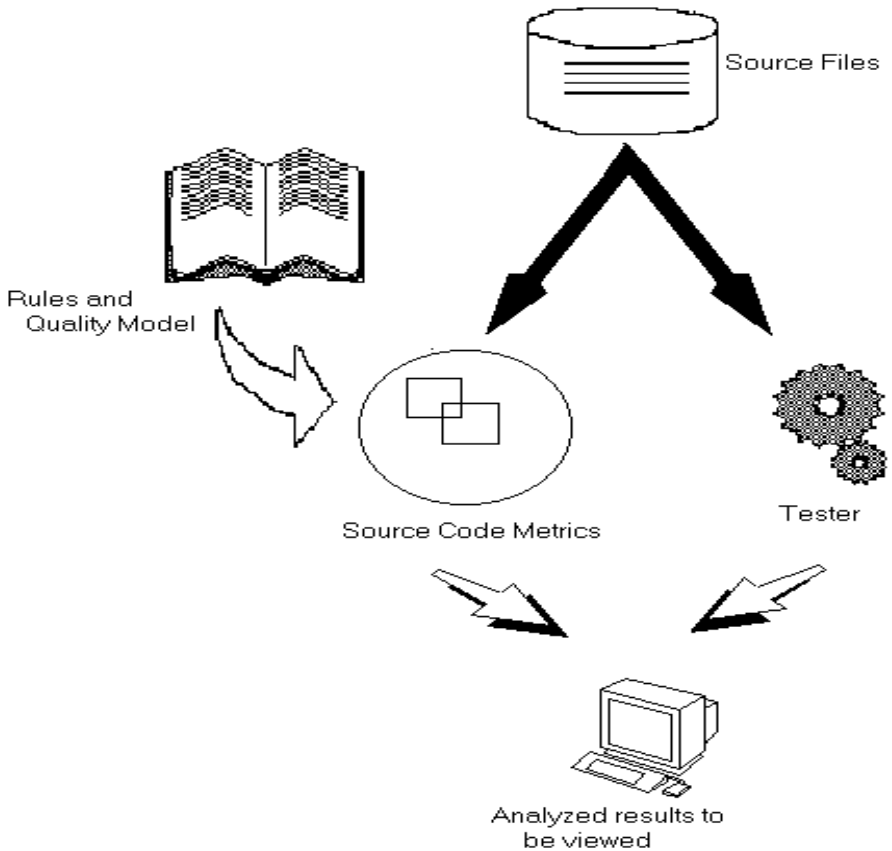
Code Analyzer

Code Analyzer is a set of tools addressing code quality through

- source code metrics
- coding standards and
- resource usage analysis.

Newtech Software's HEURISTICS
Code Analyzer For C / C++

Code Analyzer



Benefits

Early availability of metrics is a key factor to a successful management of software development, since it allows for

- Early detection of problems
- Better software quality monitoring
- More accurate planning of resource allocation based upon the predicted error proneness of the system and its components

contd...

Benefits

The Code Analyzer

- increases efficiency of the code
- evaluates quality and complexity of the software
- identifies error prone software modules
- provides visual representation of the code

Newtech Software's HEURISTICS
Code Analyzer For C / C++

Beneficial to

- Managers
- Software engineers
- Developers
- Testers
- Users
- Organization and
- System as a whole

Newtech Software's HEURISTICS
Code Analyzer For C / C++

Managers

The Code Analyzer

- Presents visual, filtered, high-level overview of the current state of the software.
- Highlights potential problems without going into any technical details.
- Allows to have a weekly glance at the code to make sure that the complexity of the system is tamed.
- Helps to identify the potentially problematic areas of the code through visual screens.
- Makes restructuring and improvement easier.

Software Engineers

- Helps to assess the quality and robustness of their design.
- Facilitates a simple visual navigation through complex software system showing symmetries and design decisions built into it.
- Outlines in a simple, yet comprehensive way, the issues of system stability, flexibility and maintenance.
- Shows the key sources of maintenance issues enabling the engineers to eliminate them before they turn into problems.

Developers

Newtech Software's HEURISTICS
Code Analyzer For C / C++

- Offers what-if scenarios which let the developer see the possible impact of any change.
- Makes them understand the structure of the system created by the dependencies between components.
- Prevents problems from unexpected places by inferring all possible components that may be affected, when a change takes place.
- Helps them communicate more effectively through visual large-scale views of the system.
- Allows to view the current dynamic state of the system not the static design

Newtech Software's HEURISTICS
Code Analyzer For C / C++

Testers

- Provides them with a definitive way to capture all system components which can be affected from a bug fix or larger code change.
- Helps to view the whole structure to plan the test effectively.

Newtech Software's HEURISTICS
Code Analyzer For C / C++

Organization

- Reduces cost of software maintenance by settling the related issues.
- Cuts cost of software development.
- Lowers the risk during production releases.
- Facilitates more stable corporate software.
- Software libraries are corporate assets.The analyzer makes effective code reuse possible.
- Allows more precise project budgeting.
- It saves bad vendor choices. A third party product can be put through scrutiny of our tool.
- Allows companies to save money by reducing the overall testing time.

Software System

Newtech Software's HEURISTICS
Code Analyzer For C / C++

- Helps prevent failures by allowing the participants of software development to gain critical insights into stability issues in their system.
- Restructuring defective code has a tremendous impact on the performance of the system.
- Helps maintenance and system improvement.

Newtech Software's HEURISTICS
Code Analyzer For C / C++

Software System

- Supplements the verbal explanations and low level diagrams with a broader conceptual pictures of the system which simplifies the knowledge transfer process.
- Helps the system members to build more stable and robust applications by becoming more aware of the structure of the system.
- Helps system testing.
- Shows prospects for evolution

Newtech Software's HEURISTICS
Code Analyzer For C / C++

Applications

- This tool finds application in
  - maintenance and
  - evolution of existing software systems.
- Also useful in
  - testing and
  - post maintenance testing to
    - generate cross-reference information
    - perform data flow/control flow analysis
    - derive definition-use chains.

Heuristics-Code Analyzer for C/C++

- Heuristics is an invaluable tool in the software engineer's arsenal that brings out
  - the complexity
  - inherent quality
  - potential faults and
  - redundant code in a software package.

Newtech Software's HEURISTICS
Code Analyzer For C / C++

Heuristics-Code Analyzer for C/C++

Heuristics

- gives valuable insight, for the engineer to fine tune the design/code to meet the required specifications in a better way.
- is a windows based package for C/C++ source code.

Newtech Software's HEURISTICS
Code Analyzer For C / C++

Features

- Analysis of source code by automatic code inspection.
- Ease of use and high- level performance.
- Quick and easy GUI.
- Requires no programming knowledge.

Newtech Software's HEURISTICS
Code Analyzer For C / C++

Features

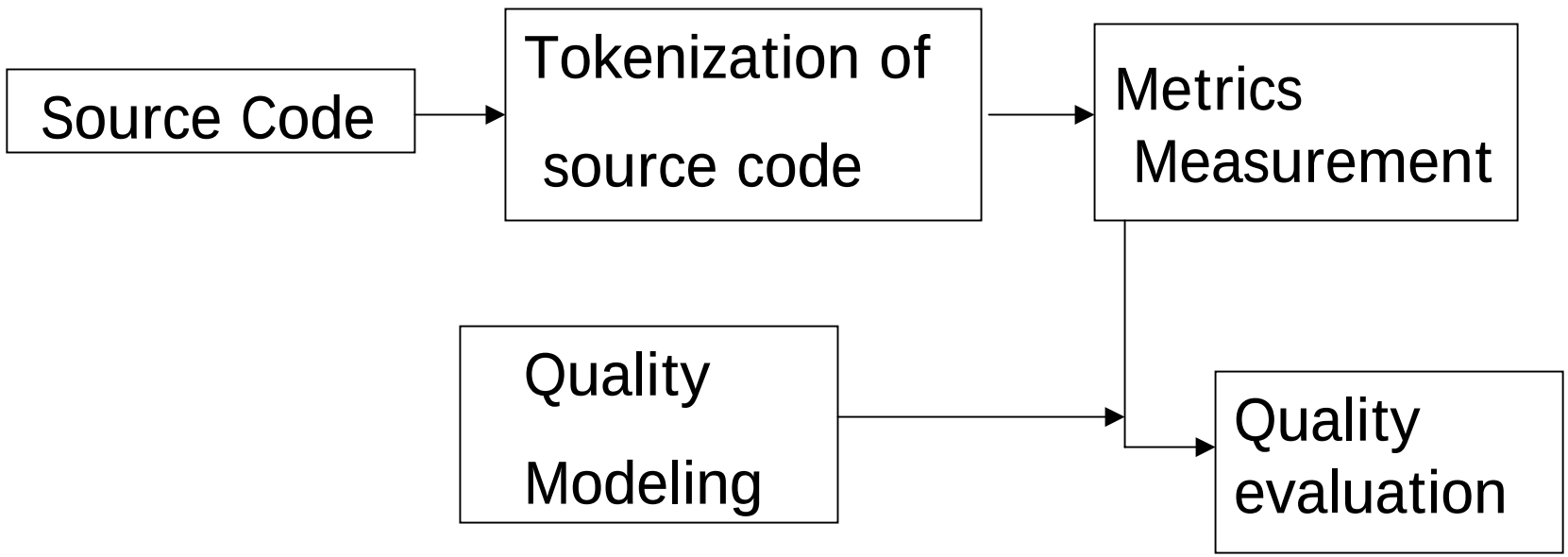
Heuristics provides solutions that address

- spectrum of software development and
- quality assurance issues.
- It will automate the following :
  - Static Analysis
  - Dynamic Analysis

Static Analyzer

Newtech Software's HEURISTICS
Code Analyzer For C / C++

Block Diagram:



Newtech Software's HEURISTICS
Code Analyzer For C / C++

Static Analyzer

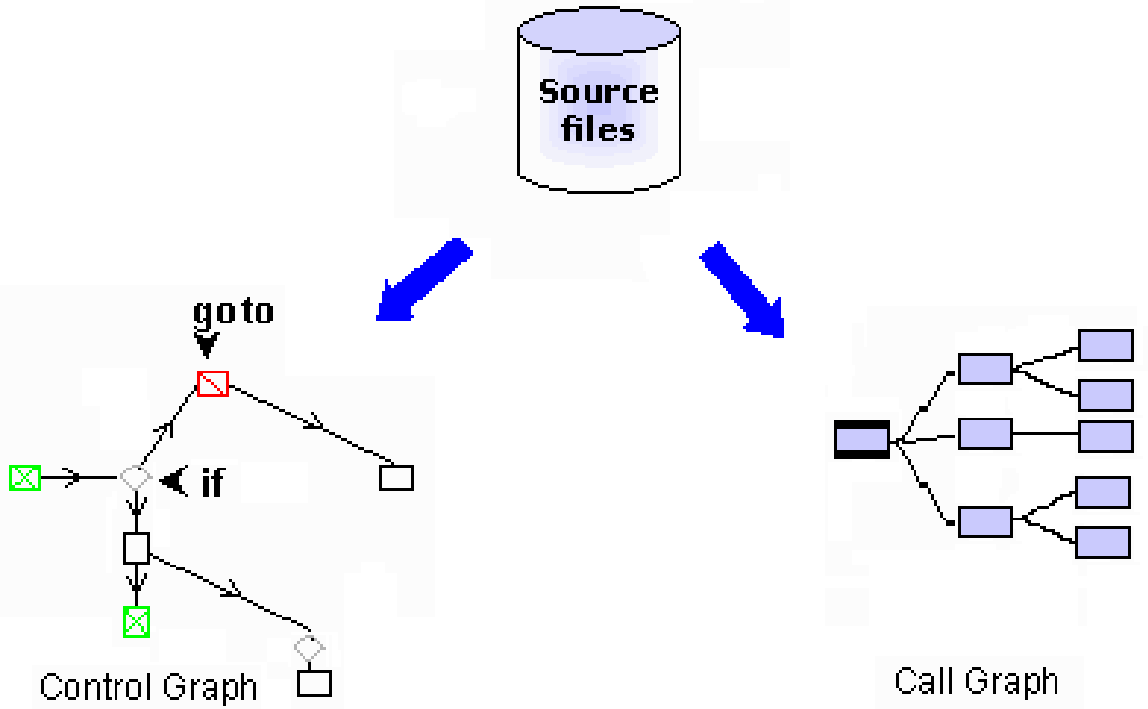
Static Analyzer provides

- qualitative assessment of the code through call graph and control graph
- quantitative assessment of the code through complexity measurements.

Static Analyzer

Newtech Software's HEURISTICS
Code Analyzer For C / C++

QUALITATIVE ASSESSMENT



Newtech Software's HEURISTICS
Code Analyzer For C / C++

Quality Attributes

- Specific quality attributes are selected based on their importance to the project and their ability to be quantified.
- The quality attributes are used to derive a core set of metrics relating to the development process and the products such as requirements and design documents, code and test plans.
- There is a need for a comprehensive model for the evaluation of quality and the management of risk.

Newtech Software's HEURISTICS
Code Analyzer For C / C++

Quality Standard

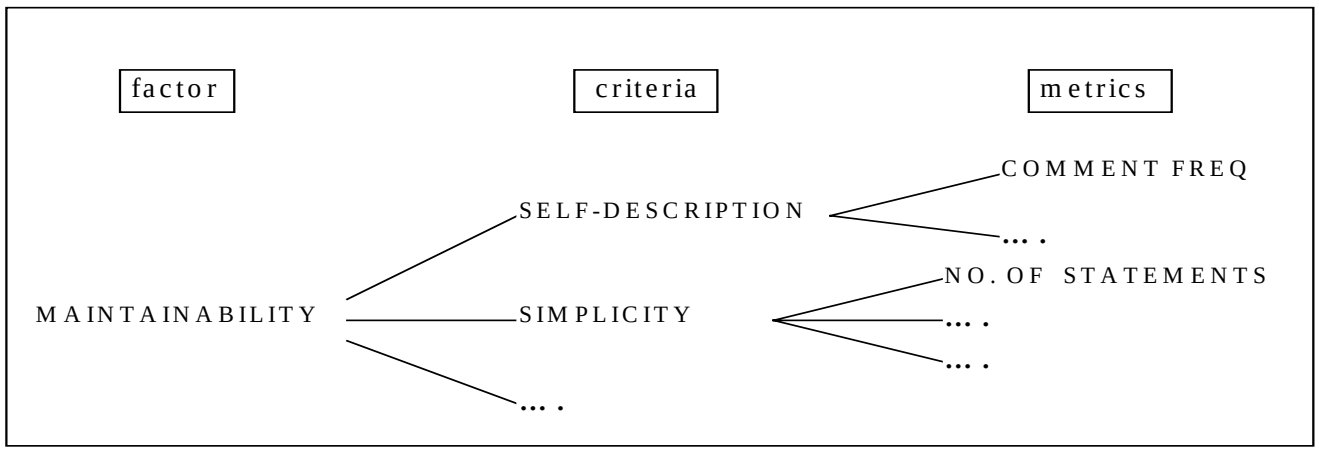
- Our quality model incorporates the factors specified in ISO 9126, each having a number of attributes which the respective metrics constitute.
- Following the structure of ISO 9126, the factors are related to software product and process attributes that allow indications of the probability of success in meeting the goals/factors. A set of metrics is chosen or developed that measure the selected attributes.

Newtech Software's HEURISTICS
Code Analyzer For C / C++

Quality Model

Quality Model is defined for Static Analysis only for C. It

- Evaluates Software quality.
- Quality is
  - specified in terms of factors.
  - designed in terms of criteria.
  - built with the help of programming rules.
  - assessed by means of metrics.



Newtech Software's HEURISTICS
Code Analyzer For C / C++

Software Metrics

A software metric

- is a measure of some aspect of a program, design, or algorithm.
- can be systematically calculated.
- can be used to make inferences about that program, design, or algorithm.
- We can infer the complexity of other programs from the calculated values of one program.

Software Complexity

Static Analysis of the code is performed through automatic code inspection. For C, Complexity of the source code is given through

- Architectural Complexity
- Structural Complexity
- Textual Complexity

Architectural Complexity

Architectural Complexity gives the complexity related to the call graph of the software. Metrics associated with this are

- **Accessibility of component**:is the measure of the ease with which a component may be accessed.
- **Testability of a path**: is the indication of the ease with which a path may be tested.
- **No. of call paths :** paths going from root to the final component.
- **No. of levels :** maximum length of call paths.
- **No. of edges :** total number of call paths.
- **Hierarchial Complexity:** mean number of components per level.

Structural Complexity

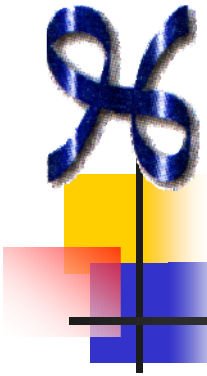
Structural Complexity gives the complexity related to control structure of a given code. Metrics associated with this are

- **Cyclomatic Complexity :** is the maximum number of linearly independent circuits in a strongly connected graph. It is also a measure of the number of basic paths in a component.
- **Degree of nesting :** is the total number of nesting of the control structure.
- **No. of in-out points :** is the number of entry and exit points in a given code.

Textual Complexity

Textual Complexity gives the complexity related to the text of the code. Metrics associated with this are

- **Program length** : is the total usage of all operands and operators in the program.
- **Program volume :** is the number of bits needed to code the program.
- **Programming effort :** is the number of 'elementary mental discriminations' required to code the program.
- **Coding time :** is the time to code a preconceived algorithm in the language used.



Newtech Software's HEURISTICS
Code Analyzer For C / C++

Results

The results of Static Analysis are given in

- Graphical and
- Textual representation

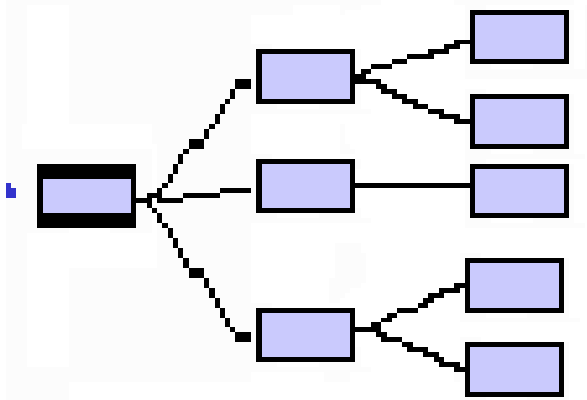
Graphical representation

In graphical representation Static Analyzer provides the following

- Call graph
- Control graph
- Kiviat graph
- Criteria graph
- Quality graph

Call Graph

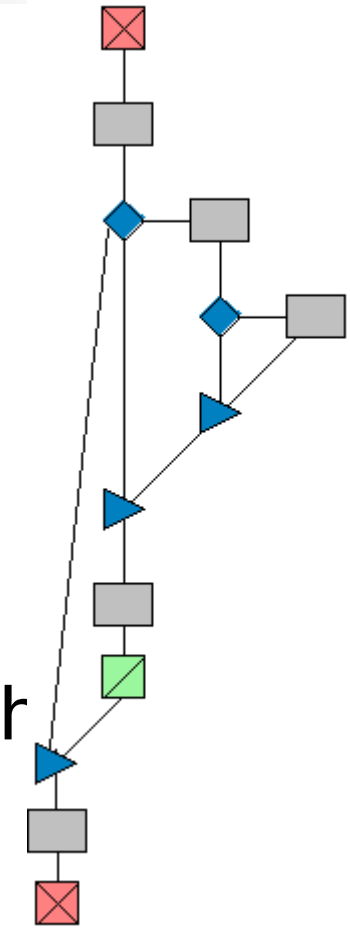
- Gives an architectural view of the program.



- Points out recursive calls and critical resources, such as frequently called functions.

Control Graph

- Displays the logical structure of a component.
- Consists of nodes, representing statements, and edges representing the transfer of the control flow between nodes.
- Finds code duplication, unstructured switch dead code, etc through the uniquely represented nodes.



Kiviat Graph

Newtech Software's HEURISTICS
Code Analyzer For C / C++

- Exhibits the behaviour of a component for the metrics defined in the model.

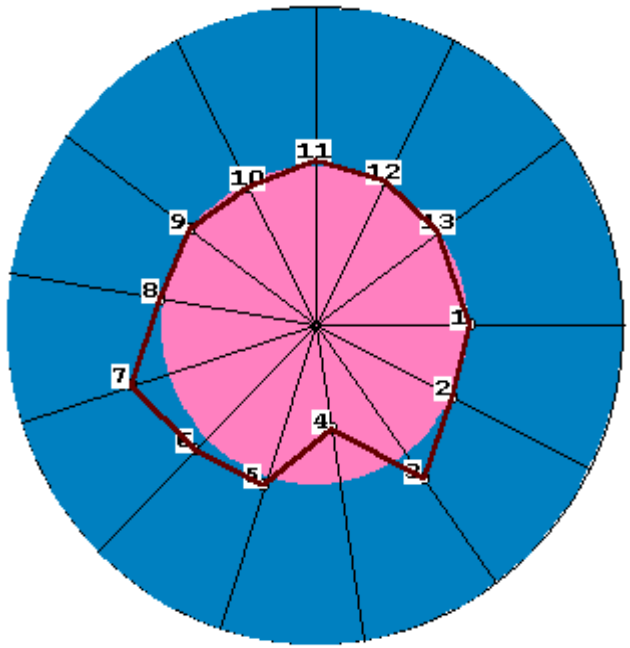
- Points out components that are outside the metric limits.

Newtech Software's HEURISTICS
Code Analyzer For C / C++

Kiviat Graph

In Kiviat Graph

- each axis represents a metric
- the limits are indicated by two circles : the inner circle corresponds to the minimum value accepted, and the outer circle corresponds to the maximum value accepted.
- the polygon links all the values obtained for the object analyzed.

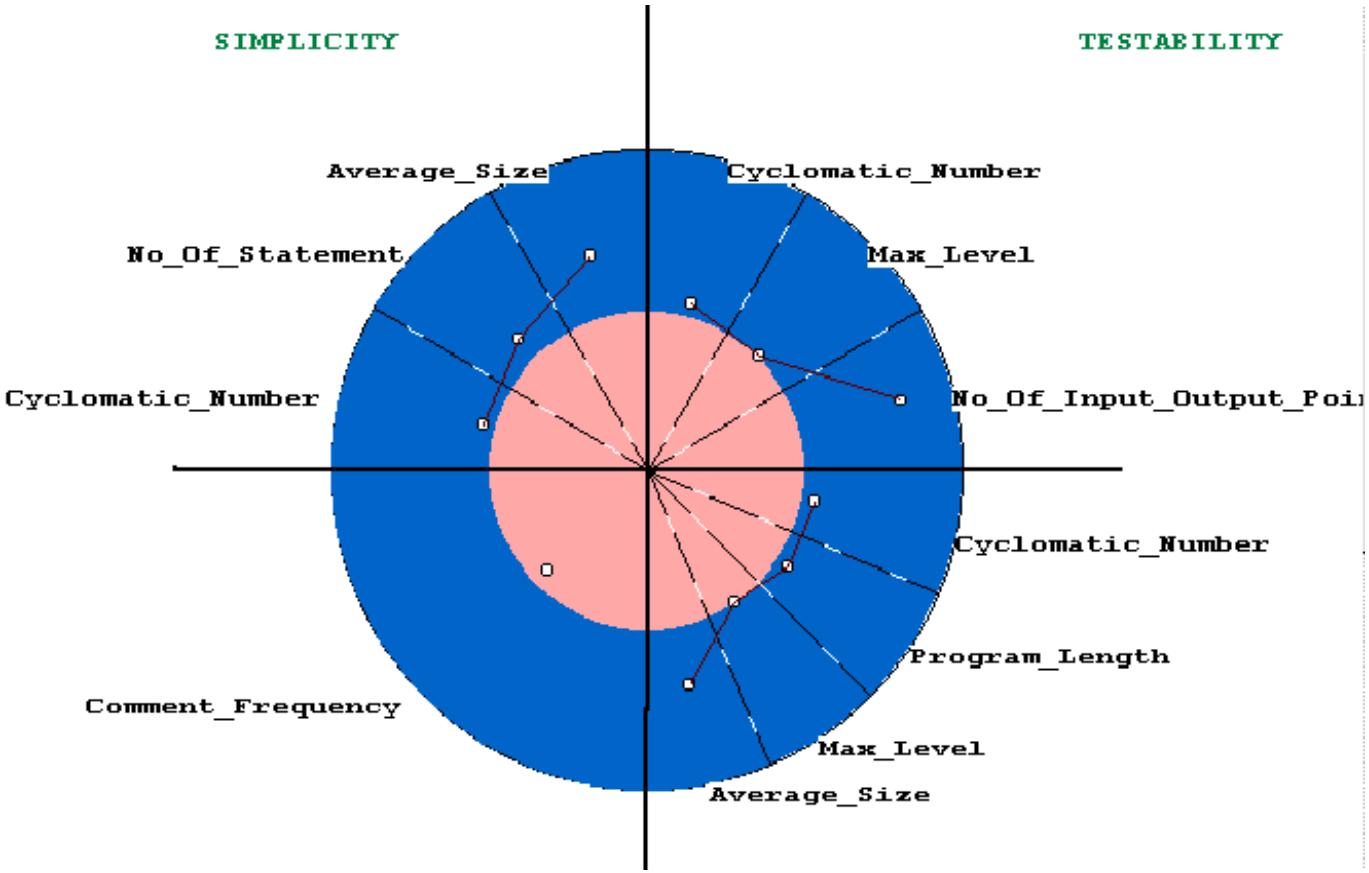


Criteria Graph

- Ranks an object with respect to set of quality criteria defined in the model.
- It presents
  - criterion/metric associations.
  - the metrics position with respect to the limit values.
  - the category of the object is given for each criterion.

Newtech Software's HEURISTICS
Code Analyzer For C / C++

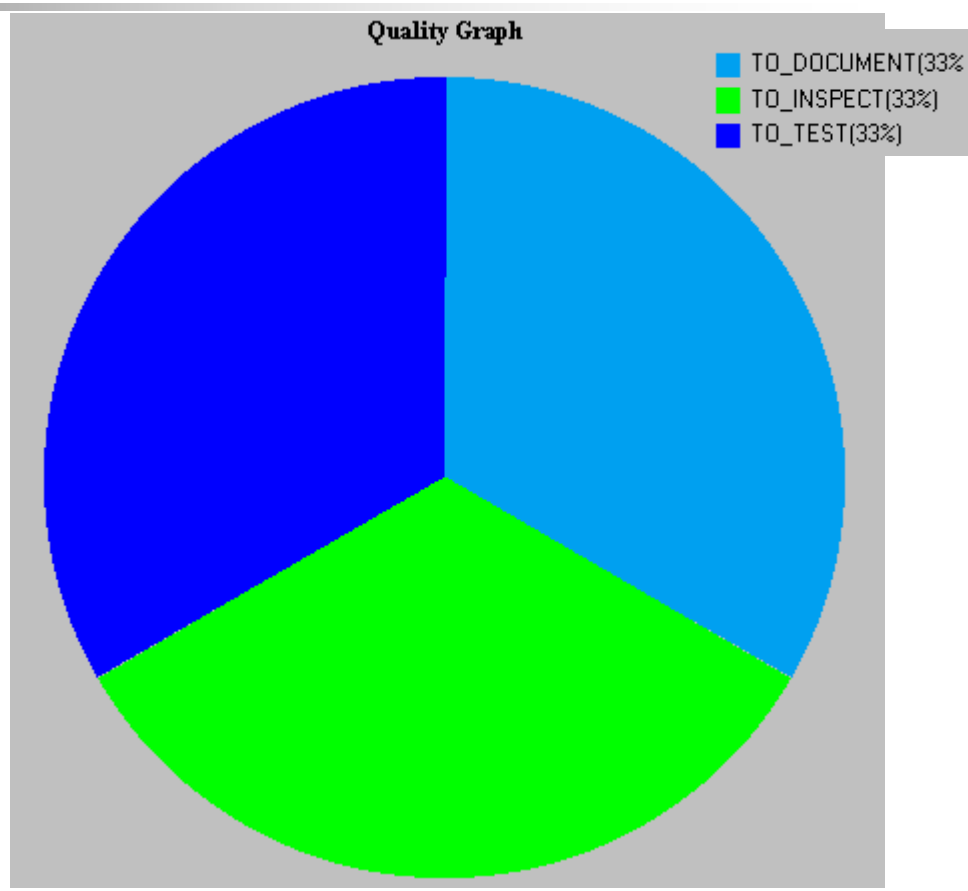
Criteria Graph



Quality graph

- Is a pie chart showing the distribution of the components according to the quality factor.

- Indicates the percentage of components belonging to each category defined.



Newtech Software's HEURISTICS
Code Analyzer For C / C++

Textual Reports

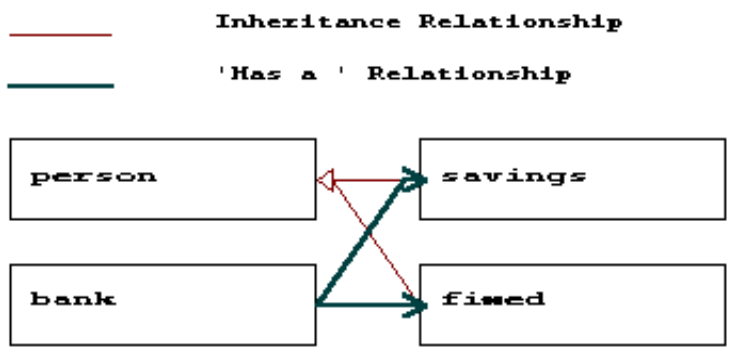
Results in textual form :

- Call Structure metrics
- Control Structure metrics
- Textual metrics
- Quality report
- Criteria Distribution report

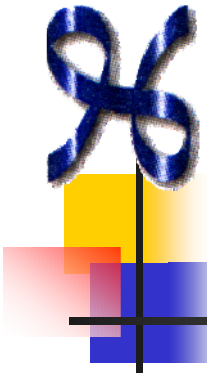
Quality measures for C++

The measures that are exclusive for C++ are as follows :

- Class diagram
- Depth of inheritance report
- Coupling of objects report



Class Diagram



Newtech Software's HEURISTICS
Code Analyzer For C / C++

Measures for C++

- Depth of Inheritance :

  exhibits the value of inheritance of each class in the code.

- Coupling of Objects:

  exhibits the number of relationships each class has with the other classes in the code.

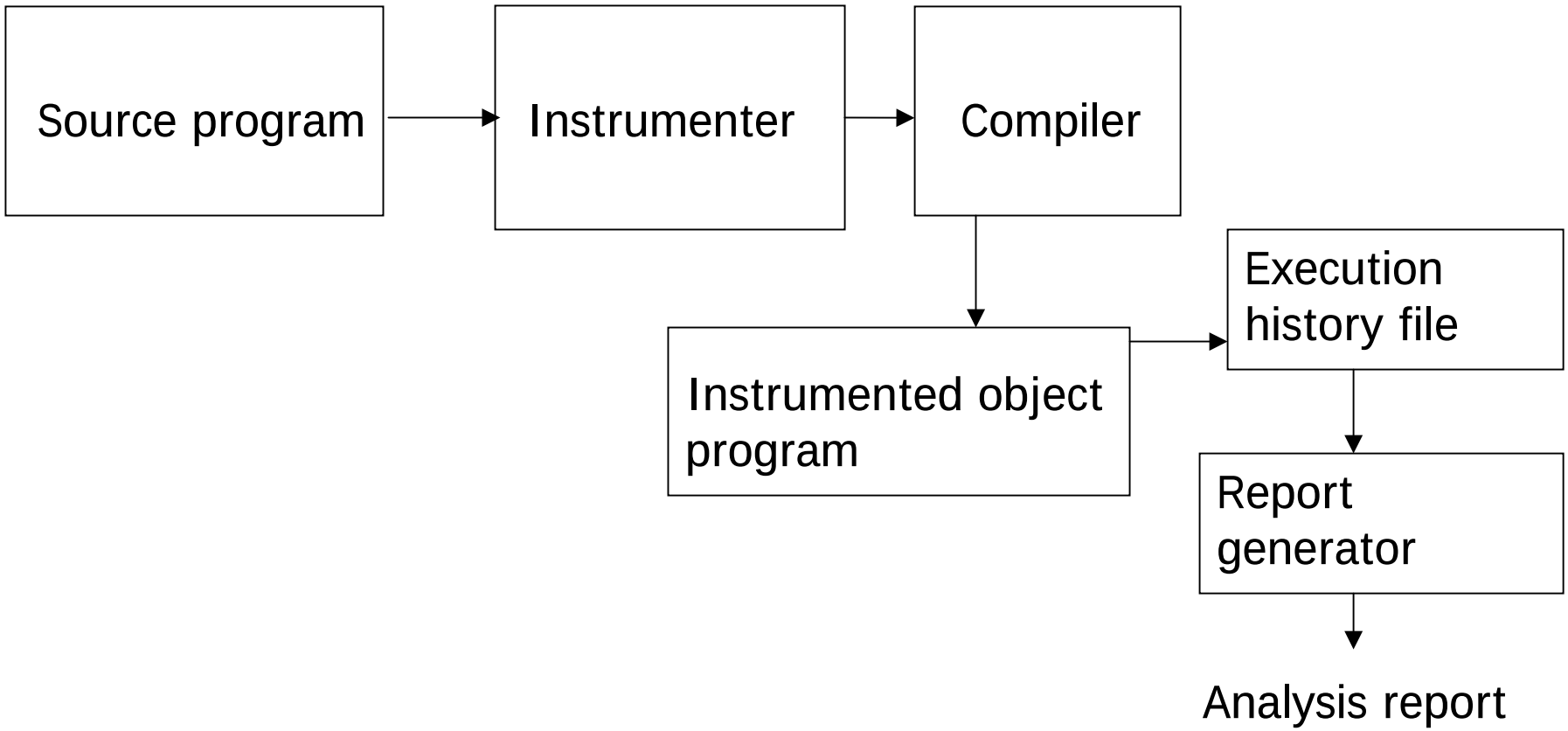
Newtech Software's HEURISTICS
Code Analyzer For C / C++

Metrics for C/C++

Procedural Metrics [C]	Object Oriented Metrics [C++]
Cyclomatic Number	Number Weighted methods per class
Number of execution paths	Number of classes
Number of nesting levels	Number of children, of ancestors
Number of comments	Depth of inheritance tree
Number of statements	Coupling between object

Dynamic Analyzer

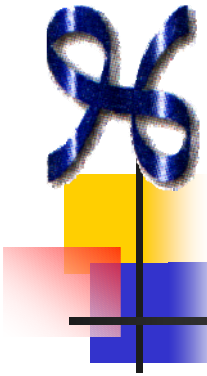
Block Diagram:



Newtech Software's HEURISTICS
Code Analyzer For C / C++

Dynamic Analyzer

- Instruments the given source code to provide run time performance or evaluation of the software package.
- Identifies memory leakage in the program.
- Produces event trace diagrams, which shows the flow of the program.
- Provides information on how often each statement in a program has been executed.



Newtech Software's HEURISTICS
Code Analyzer For C / C++

Dynamic Analyzer

It has two logical components

1. **Instrumentation part:** adds instrumentation statements to a program.When the program is run, these statements gather and collect information on how often each program statement is executed.
2. **Monitoring and Display part:** collects information provided by the instrumentation statements and prints an execution report.

Dynamic Analyzer

Newtech Software's HEURISTICS
Code Analyzer For C / C++

The results are given through

- Event Trace analysis
  - Event trace diagram
  - Event replay diagram
- Memory analysis
  - Heap variation report
  - Stack variation report

Newtech Software's HEURISTICS
Code Analyzer For C / C++

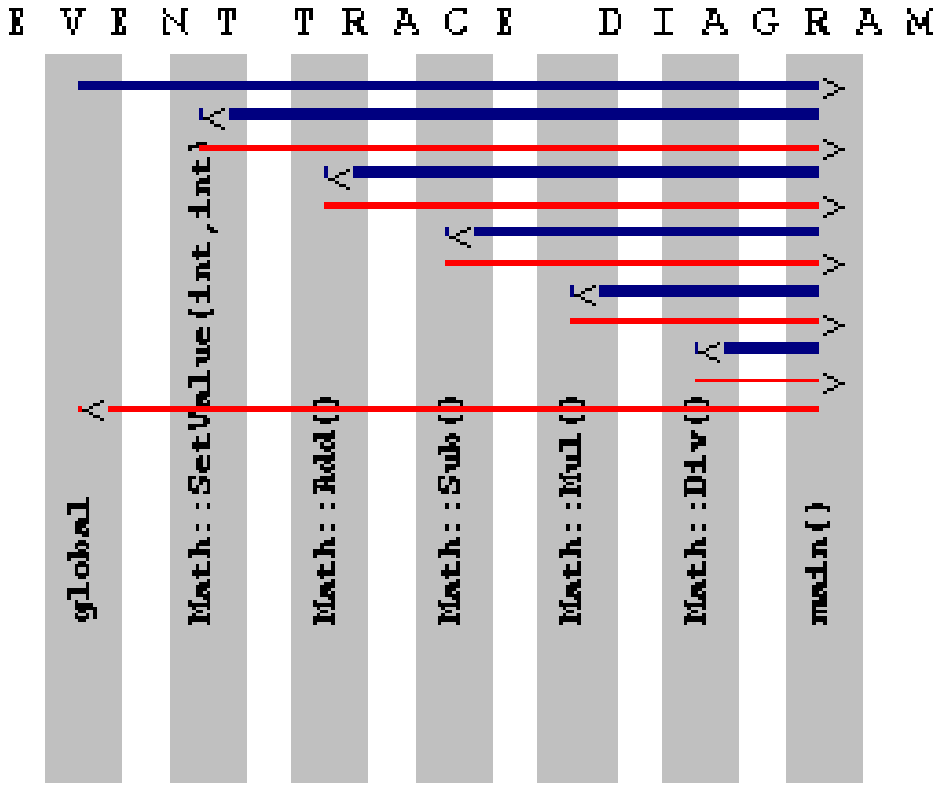
Event Trace Analysis

Event Trace diagram

•Shows the execution path of the source code

Event Replay Diagram

•Shows the execution trace with actual time taken by the execution of the source code.



Newtech Software's HEURISTICS
Code Analyzer For C / C++

Memory Analysis

Heap Variation Chart

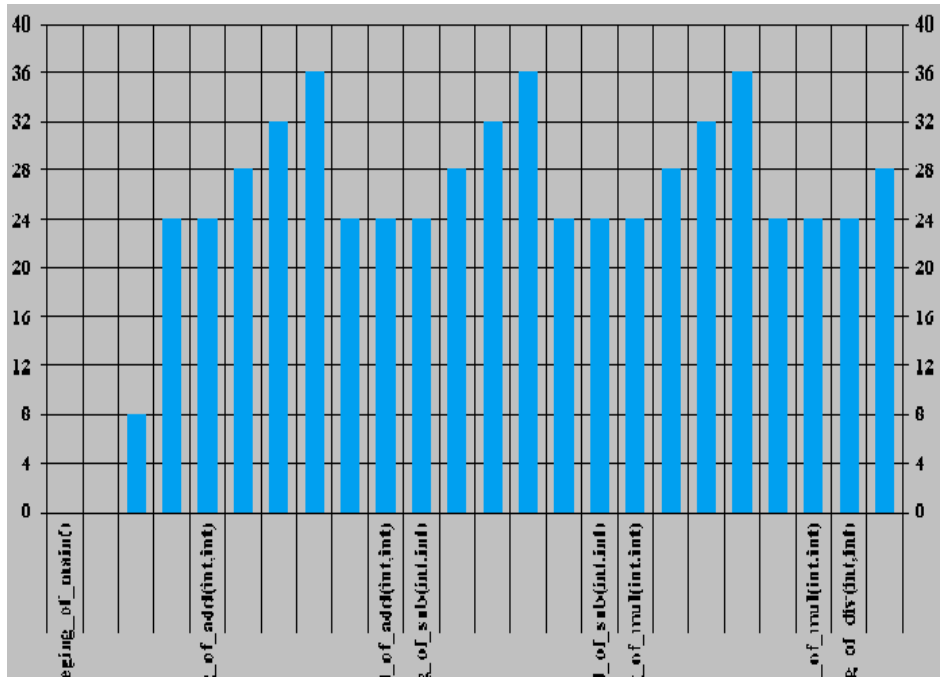
- shows the utilization

of dynamic memory allocation.

Stack Variation Chart

- shows the utilization

of static memory allocation



Memory Analysis Chart

Newtech Software's HEURISTICS
Code Analyzer For C / C++

Heuristics

Heuristics benefit the users in a variety of ways:

- Get a view of software quality.
- Avoids errors, eases maintenance by ensuring compliance with coding standards.
- Provides demonstrable, quantitative measurements of software quality.
- Makes efficient use of software resources.

Code Coverage

- Code coverage tools measure how thoroughly tests exercise programs.
- Testers who read the source code while testing.
- It also describes coverage's relevance to the independent product tester (someone who doesn't look at the code) and to managers of developers and testers.

Code Coverage Process

- Code coverage analysis is the process of:
- Finding areas of a program not exercised by a set of test cases,
- Creating additional test cases to increase coverage, and
- Determining a quantitative measure of code coverage, which is an indirect measure of quality.

Newtech Software's HEURISTICS
Code Analyzer For C / C++

Optical Aspect

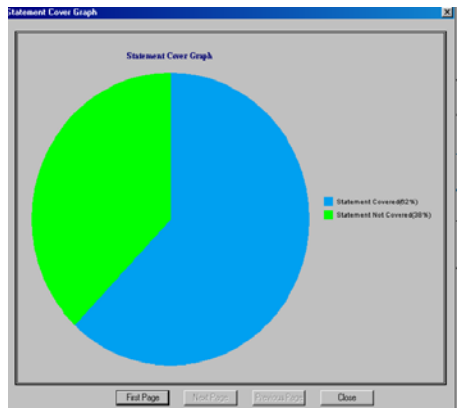
- Identifying redundant test cases that do not increase coverage.

Code Coverage for C

- For C Analyzer, we have implemented the following types of coverages.
  1. Statement Coverage
  2. Function Coverage
  3. Call Coverage
  4. Decision Coverage
  5. Data Coverage

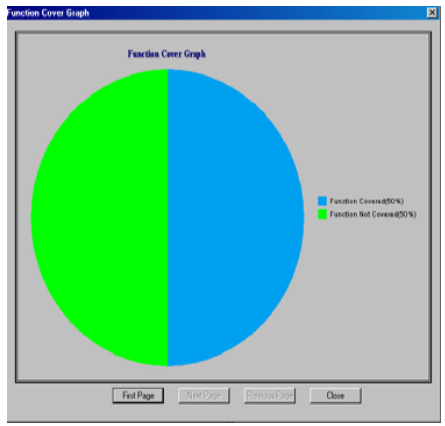
Statement Coverage

- Statement coverage does not report whether loops reach their termination condition - only whether the loop body was executed.
- It gives the detailed information about the percentage of covered Statements in a program and the percentage of uncovered Statements in a program.



Function Coverage

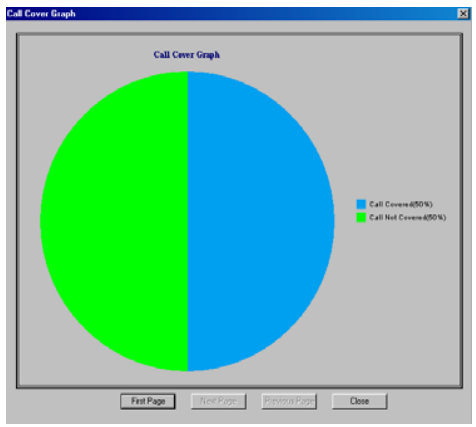
- This measure reports whether you invoked each function or procedure.
- It is useful during preliminary testing to assure at least some coverage in all areas of the software.
- It gives the detailed information about the percentage of covered function in a program and the percentage of uncovered function in a program.



Newtech Software's HEURISTICS
Code Analyzer For C / C++

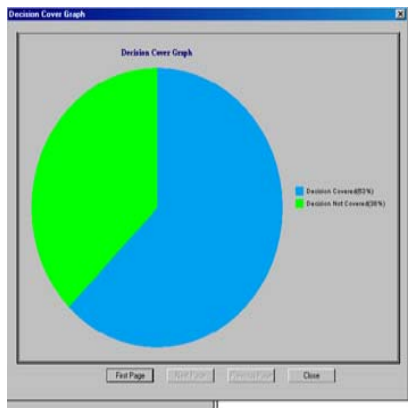
Call Coverage

- This measure reports whether you executed each function call.
- The hypothesis is that faults commonly occur in interfaces between modules. It gives the detailed information about the percentage of covered call function in a program and the percentage of uncovered call function in a program.



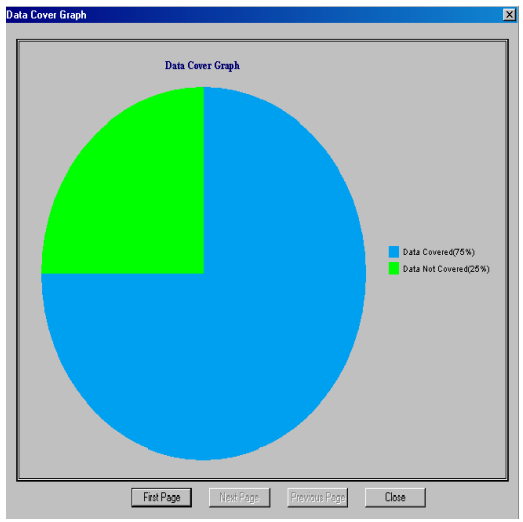
Decision Coverage

- This measure reports whether Boolean expressions tested in control structures (such as the if-statement and while-statement) evaluated to both true and false.
- The entire Boolean expression is considered one true-or-false predicate regardless of whether it contains logical-and or logical-or operators.
- It gives the detailed information about the percentage of covered Decision statements in a program and the percentage of uncovered Decision statements in a program.



Data Coverage

- It gives the detailed information about the percentage of covered data in a program and the percentage of uncovered data in a program.

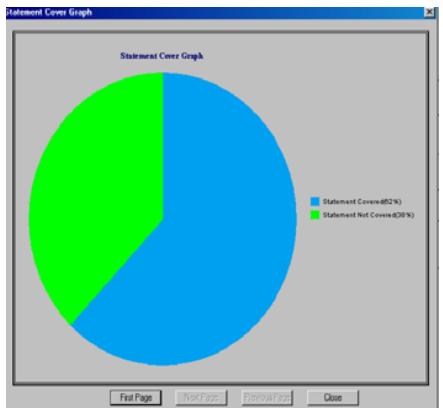


Code Coverage For C++

- For C++ Analyzer, we have implemented the following types of coverages.
    1. Statement Coverage
    2. Function Coverage
    3. Decision Coverage
    4. Inheritance Coverage

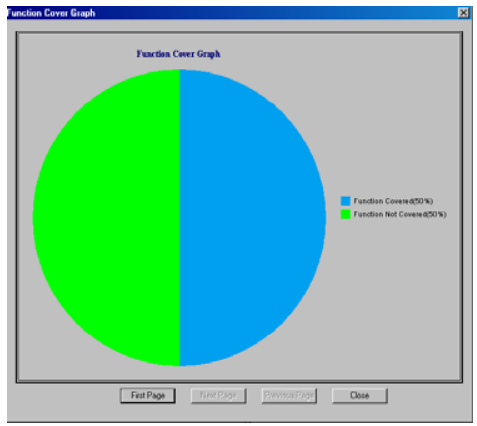
Statement Coverage

- Statement coverage does not report whether loops reach their termination condition - only whether the loop body was executed.
- It gives the detailed information about the percentage of covered Statements in a program and the percentage of uncovered Statements in a program.



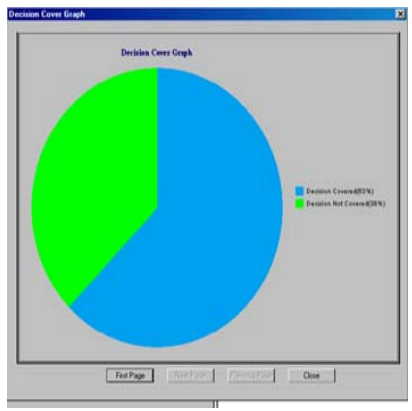
Function Coverage

- This measure reports whether you invoked each function or procedure.
- It is useful during preliminary testing to assure at least some coverage in all areas of the software.
- It gives the detailed information about the percentage of covered function in a program and the percentage of uncovered function in a program.



Decision Coverage

- This measure reports whether Boolean expressions tested in control structures (such as the if-statement and while-statement) evaluated to both true and false.
- The entire Boolean expression is considered one true-or-false predicate regardless of whether it contains logical-and or logical-or operators.
- It gives the detailed information about the percentage of covered Decision statements in a program and the percentage of uncovered Decision statements in a program.



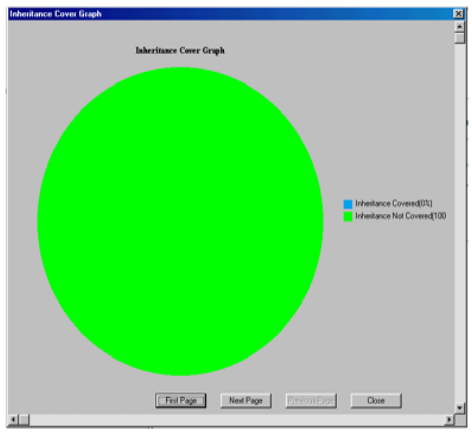
Inheritance Coverage

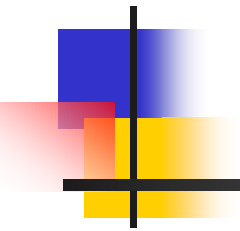
- Inheritance context coverage is not a single metric, but rather a way of extending the Interpretation of (any of) the traditional structural coverage metrics to take into Account the additional interactions, which occur when methods are inherited.
- Inheritance context coverage provides alternative metric definitions, which consider
- The levels of coverage achieved in the context of each class as separate measurements.

Cont....

Inheritance Coverage

- The inheritance context definitions regard execution of the routine in the context of the base class as separate from execution of the routine in the context of a derived class.





ADDITIONAL SLIDES ON

Condition

Decision

Modified Condition/Decision

Coverage Analysis

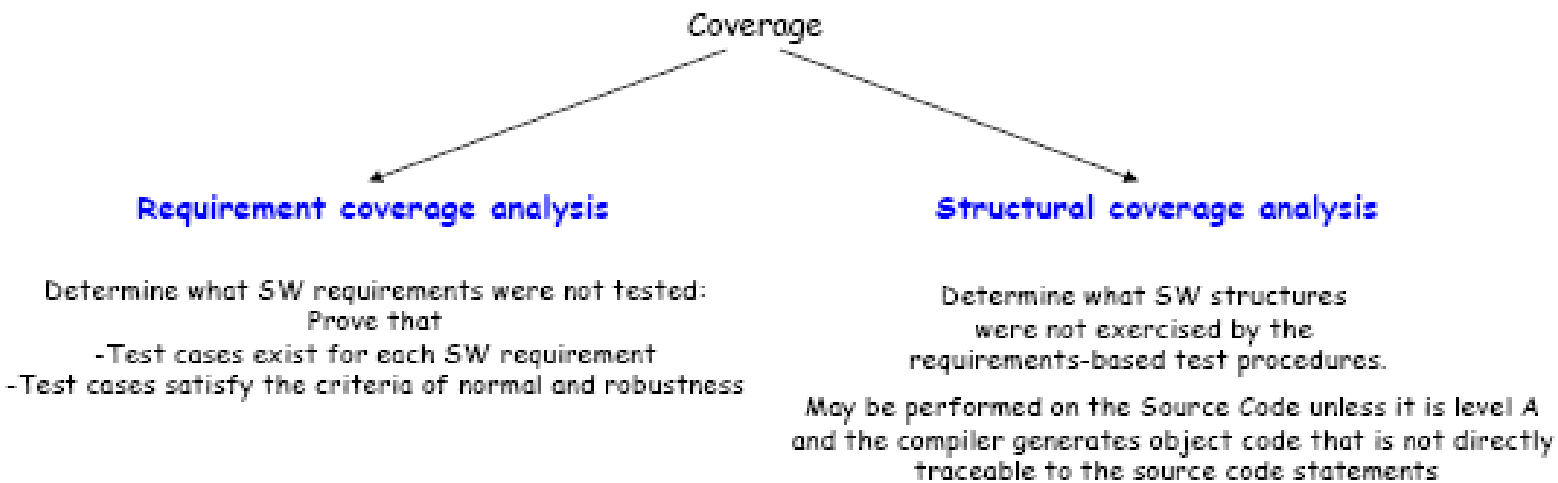
Implemented in HEURISTICS

Coverage Analyses

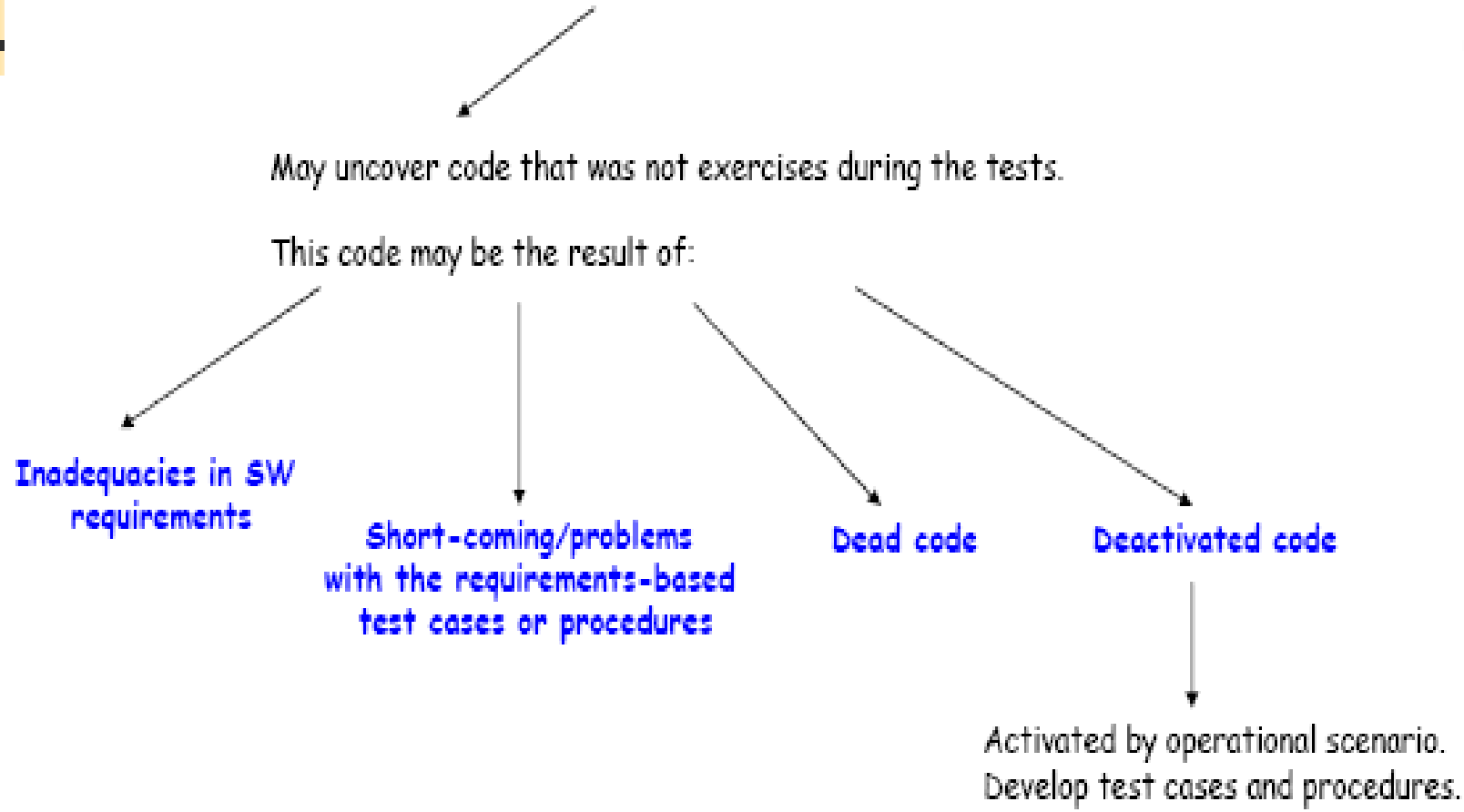
Coverage refers to the extend to which a given verification activity satisfies its objectives.

Coverage is most often applied to the testing activities.

→ Coverage by itself is just a measure.



Structural Coverage



Structural Coverage Analyses

- Statement Coverage
- Decision Coverage
- Condition Coverage
- Decision/Condition Coverage
- Multiple Decision Coverage
- Modified Condition/Decision Coverage
  - Unique-cause
  - Masking

Required to assure that the as-implemented code structure has been adequately tested and does not contain any unintended functionality.

DO 178A/B Requirements

Table A-7
Verifications of Verification Process Results

	Objective		Applicability by SW level				Output		Control cat. by SW level			
	Description	Ref.	A	B	C	D	Description	Ref.	A	B	C	D
1	Test procedures are correct.	6.3.6b	●	○	○		Software Verification Cases and Procedures	11.13	②	②	②	
2	Test results are correct and discrepancies explained.	6.3.6c	●	○	○		Software verification results	11.14	②	②	②	
3	Test coverage of high-level requirements is achieved.	6.4.4.1	●	○	○	○	Software verification results	11.14	②	②	②	②
4	Test coverage of low-level requirements is achieved.	6.4.4.1	●	○	○		Software verification results	11.14	②	②	②	
5	Test coverage of software structure (Modified Condition/Decision Coverage) is achieved.	6.4.4.2	●				Software verification results	11.14	②			
6	Test coverage of software structure (Decision Coverage) is achieved.	6.4.4.2a 6.4.4.2b	●	●			Software verification results	11.14	②	②		
7	Test coverage of software structure (Statement Coverage) is achieved.	6.4.4.2a 6.4.4.2b	●	●	○		Software verification results	11.14	②	②	②	
8	Test coverage of software structure (data coupling and control coupling) is achieved.	6.4.4.2c	●	●	○		Software verification results	11.14	②	②	②	

Conditions versus Decisions (1)

Decision Coverage

Every decision in the program has taken all possible outcomes + every point of entry and exit in the program has been invoked at least once. (required for SW levels A and B)

Condition Coverage

Every condition in the program has taken all possible outcomes + every point of entry and exit in the program has been invoked at least once.

→ So, what is the difference?

- Decision coverage looks at the two cases: the outcome of a decision is TRUE and the outcome of the decision is FALSE.
- Condition coverage looks at the conditions within the decision and let them take all possible outcomes (TRUE and FALSE).

Conditions versus Decisions (2)

IF (A or B) THEN Proc1()
ELSE Proc2()

Decision coverage:

2 test cases:

For example:
Test case #1 (TF) A=T, B=F
Test case #2 (FF) A=F, B=F

Condition coverage:

2 test cases:

For example:
Test case #1 (TF) A=T, B=F
Test case #2 (FT) A=F, B=T

These test cases do NOT cause the decision
To take on all possible outcomes !!

T = TRUE, F = FALSE

Conditions versus Decisions (3)

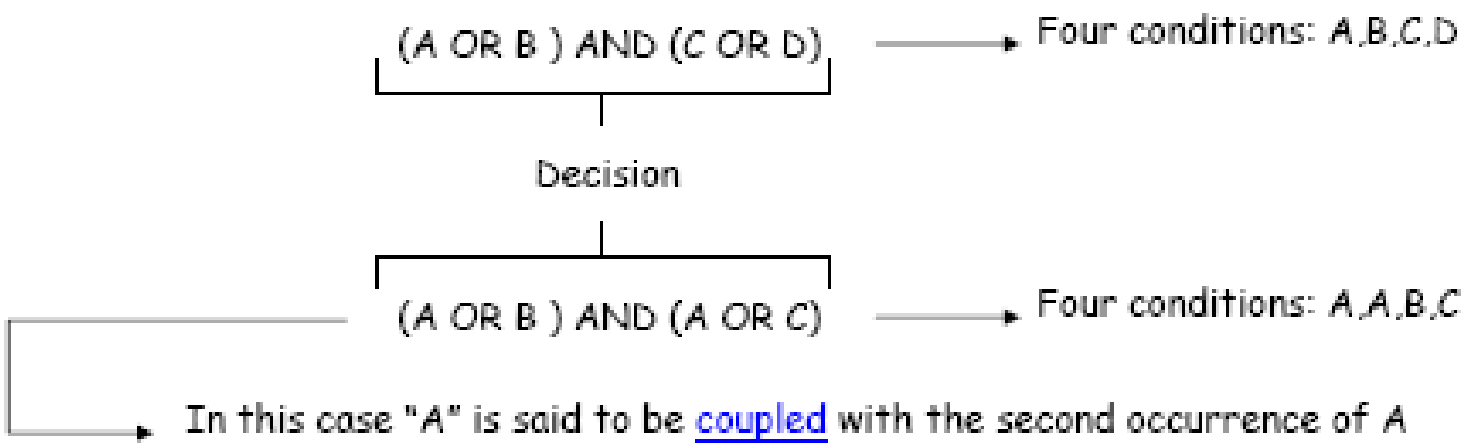
In the Glossary section of DO-178B "decision" and "condition" are defined as follows:

Decision

"A Boolean expression composed of conditions and zero or more Boolean operators. A decision without a Boolean operator is a condition. If a condition appears more than once in a decision each occurrence is a distinct condition"

Condition

"A Boolean expression containing no Boolean operators"



Conditions versus Decisions (4)

Furthermore ...

Set 1

A := B or C ← Statement 1
E := A and D ← Statement 2

Set 2

E := (B or C) and D ← Statement 3

Logically, statements 1 and 2 are equivalent to statement 3.
All three these statements are decisions even though none of the statements are branch points such as an "if" statement.

Condition/Decision Coverage

Condition/Decision Coverage
Every decision in the program has taken all possible outcomes +
Every condition in the program has taken all possible outcomes +
every point of entry and exit in the program has been invoked at least once.

```
IF (A or B) THEN Proc1()
ELSE Proc2()
```

2 test cases will give Condition/Decision coverage:

Test case #1: TT (decision outcome should be TRUE)
Test case #2: FF (decision outcome should be FALSE)

Problem: what is the programmer makes a mistake and implements –

```
if (A && B) Proc1();
else proc2();
```

Multiple Decision Coverage

Multiple Decision Coverage
Test cases require each possible combination of inputs to the decision to be executed at least once

Very, very labor intense

IF (A or B) THEN Proc1()
ELSE Proc2()

Test cases:

#1:	FF
#2:	TF
#3:	FT
#4:	TT

If a decision has N inputs conditions:
Multiple Decision coverage requires 2^N test cases.

Modified Condition/Decision Coverage (MC/DC)

Modified Condition/Decision Coverage
Every decision in the program has taken all possible outcomes +
Every condition in the program has taken all possible outcomes +
Each condition has been shown to affect the outcome of the decision independently +
every point of entry and exit in the program has been invoked at least once.

→ In unique-cause MC/DC (as defined in DO-178B, pp83), this statement means:
"A condition is shown to independently affect a decision's outcome
by varying just that conditions while holding fixed all other possible conditions."

Anybody foresees any problems?

Problem: What about (A or B) and (A or C) ?

→ How do we vary A, while holding fixed A ?

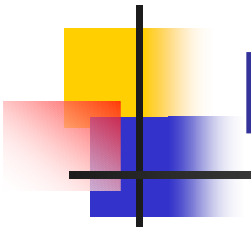
Masking MC/DC

Instead of

"A condition is shown to independently affect a decision's outcome by varying just that conditions while holding fixed all other possible conditions."

in "Unique-cause MC/DC", "Masking MC/DC" allows more than one input to change in an independence pair, as long as the condition of interest is the only condition that affects the value of the decision outcome.

→ **Analysis of the internal logic** is used to show that the condition of interest is the only condition causing the value of the decision's outcome to change.



Heuristics

Newtech Software's HEURISTICS
Code Analyzer For C / C++

NEWTECH SOFTWARE

THANK YOU
FOR BEING WITH US

HEURISTICS

Newtech Software's HEURISTICS
Code Analyzer For C / C++

For further information contact:

Newtech Software(Pvt.) Ltd.,
#572, 6th 'F' Cross,
17th 'A' main, 6th Block,
Bangalore-95.India.
Phone :91- 80-25529336,
91- 80-25529338.
Fax : 91- 80- 25529338
E-mail: info.newtech@newtechsoftware.net