Introduction to RESORT for Java(JSP)

Soft4Soft

www.soft4soft.com

Contents

- Background
- RESORT for Java Product & Solution
- RESORT for Java - SW Quality
- RESORT for Java - Code Inspection
- RESORT for Java - Test

Background

- 2001 Established Soft4Soft Co., Ltd.
- Main Business
  - SW Quality Solution Tool Development(QA Tool)
  - SW Quality Consultancy and Education Services
- Certifications – in KOREA
  - IT(excellent Information Technology) Mark
  - GS(GOOD Software) Mark
  - KT(Excellent Korean Technology) Mark
- Main Client List

excellent Information Technology

GOOD software

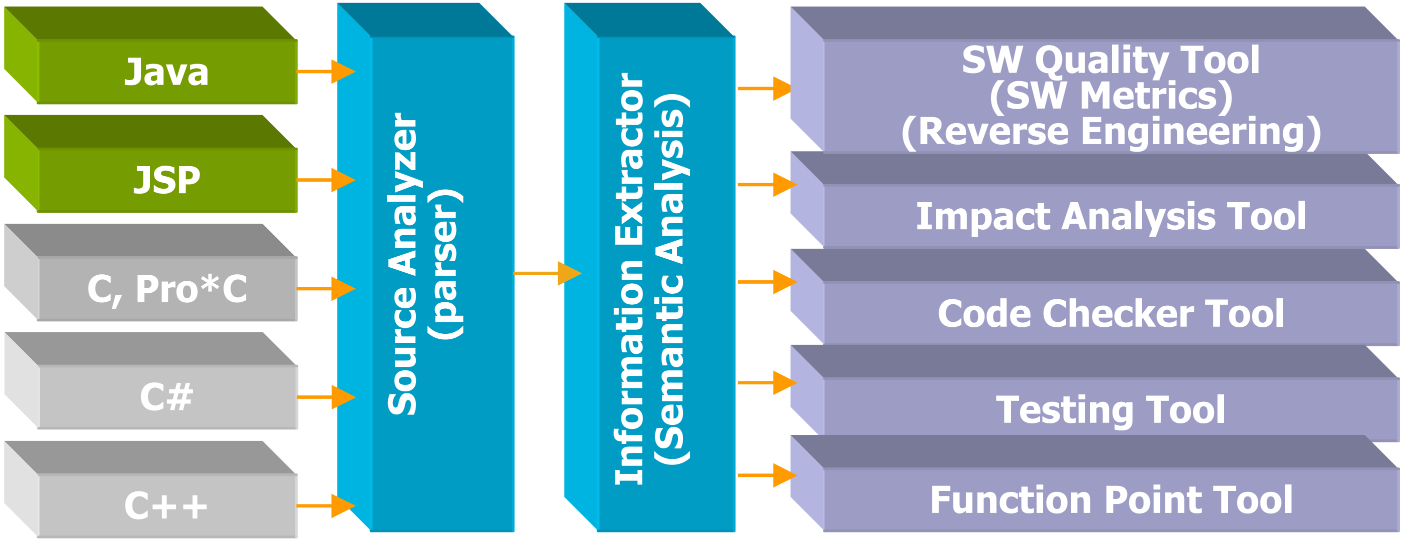
KT

Excellent Korean Technology

SAMSUNG 삼성전자	POSDATA	비씨카드	ETRI www.etri.re.kr
LG전자	LG CNS	농협	TTA 한국정보통신기술협회 Telecommunications Technology Association
DAEWOO ELECTRONICS	SK C&C	한국은행 THE BANK OF KOREA	KISTI
KIA 현대·기아자동차	AutoEverSystems	Banktown	대법원
HANJIN SHIPPING	CyberLogitec	MERITZ 메리츠화재	대한생명

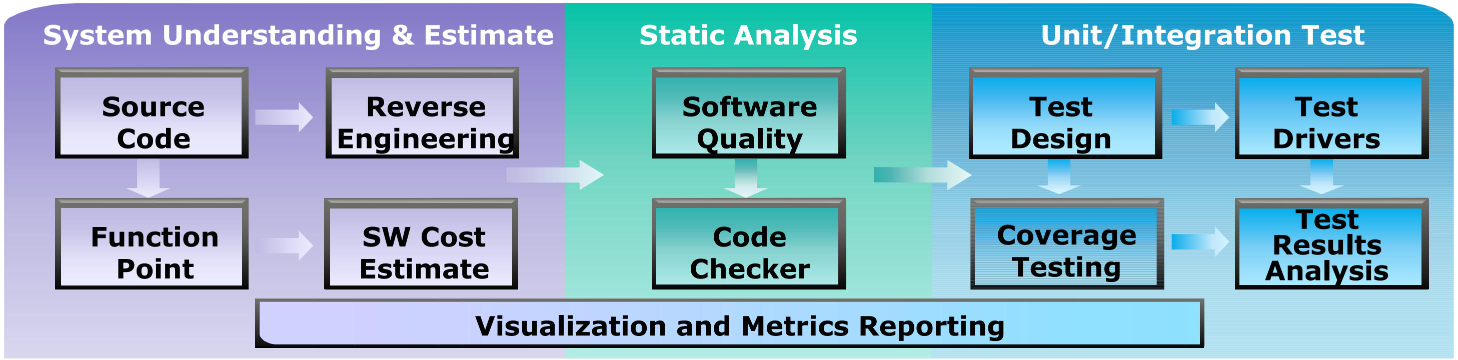
Products

- Soft4Soft – Software Quality Assurance Products
  - Source code analysis of a compile level
  - Information analysis which is various and correct
  - Second statistical information to need to manager, quality team, in addition to developers.

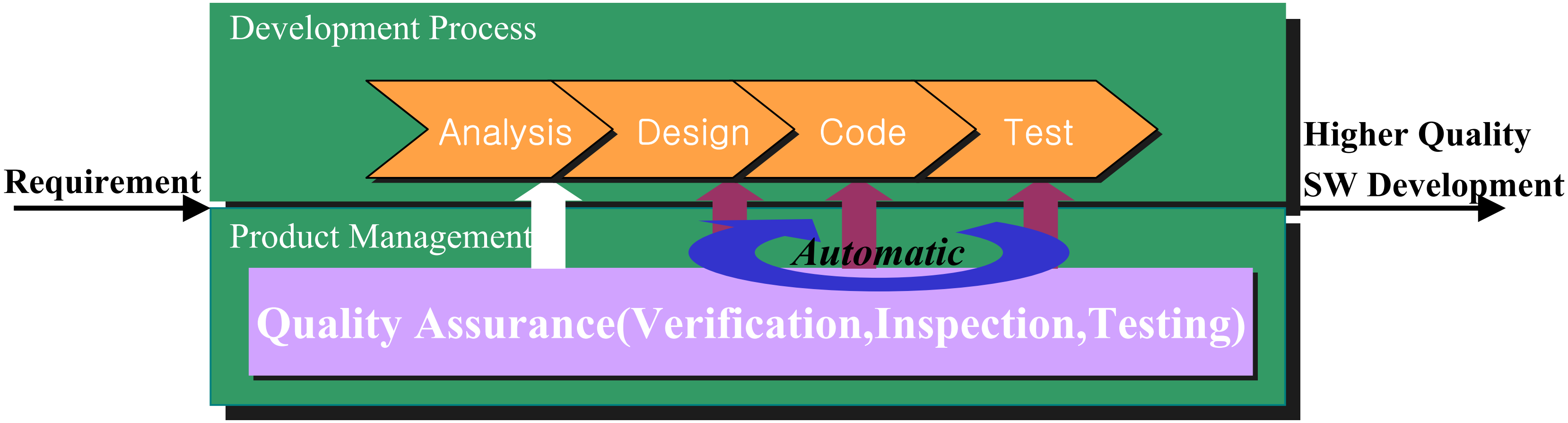


Solutions

- Soft4Soft – Software Quality Assurance Solution

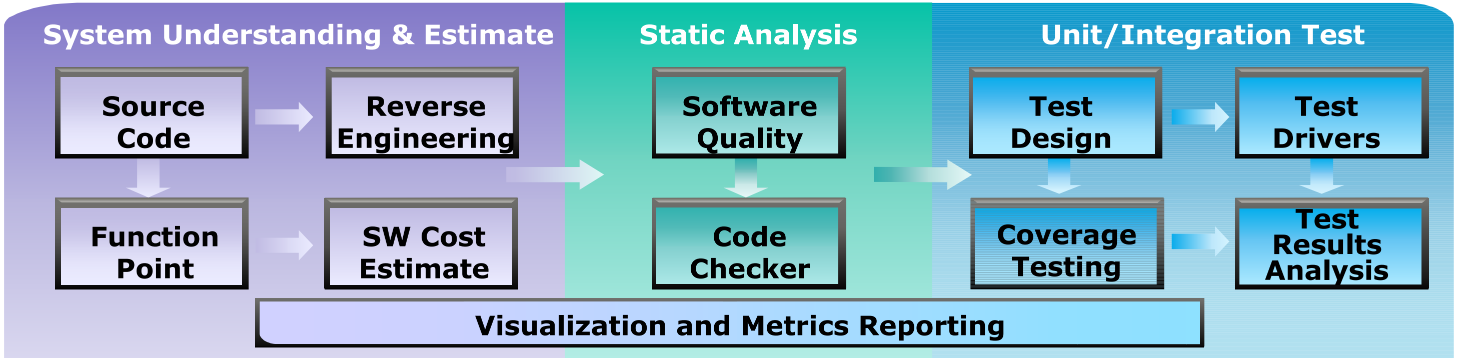


- Software Process Model and RESORT Products Map



RESORT for Java Toolset

- **Product Development and Management Process with RESORT**

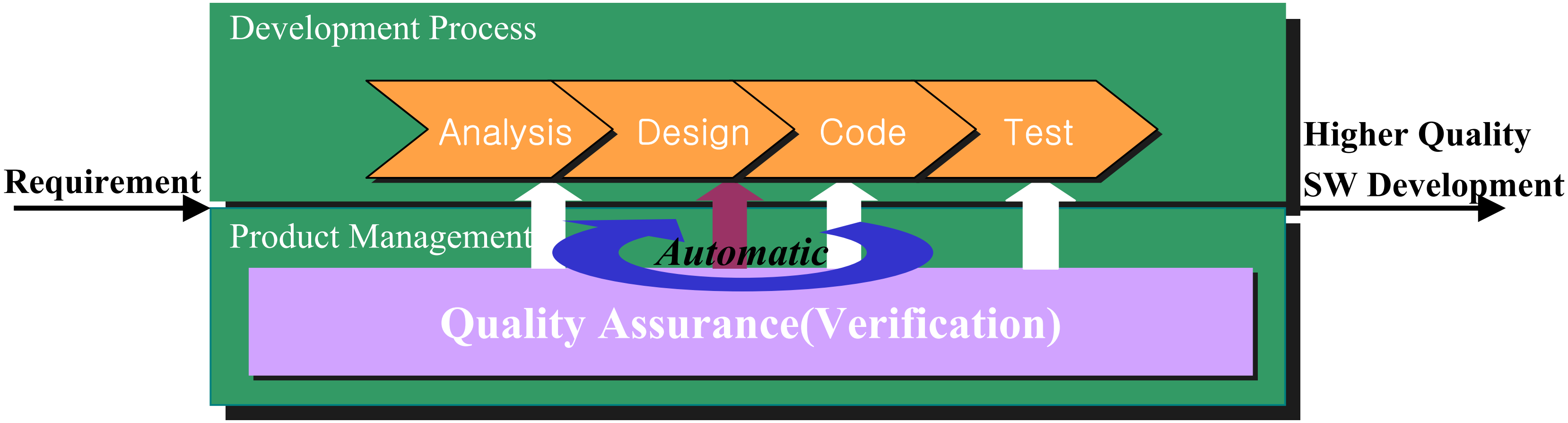


Tools	Features	View Type
SW Quality (Reverse)	Automatic Diagram Generation(UML Diagram) Understand both the Design and Architecture of SW Various OO Quality Measurement(size, structure) Multi-Reports to monitor&manage SW Quality	Diagram: Class/Package Sequence/Collaboration Control-Flow/Data-Flow Source Code Browser SW Quality: OO/Halstead/ Package/Quality/System
Code Inspection	Coding Error Detection (Application, DB Interface) Multi-Reports to monitor&manage Code Quality	Audit per File/Class/Method File/Class/Method Inspection
Test	Unit & Integration Testing Code Coverage & Performance Analysis Graphical Monitor of Historical Test Results	Test Case Design Control-Flow Testing Data-Flow Testing Sequence Testing
Function Point	Automatic Function Point Calculation SW Productivity and Cost Evaluation	FP Size FP Estimator FP Counter

Soft4Soft

SW Quality(Reverse Eng.) Tool

- SW Quality Improvement
  - Understandability
  - Maintainability
  - Performance
  - Code Optimization
- SW Quality and SW Process Model Map
  - Prevent SW Potential Problems early in the Development Cycle



SW Quality Solution

- SW Quality Measurement and Expected Effect
  - Identify high-risk components in the short-term
  - Consider portability and reusability in the long-term

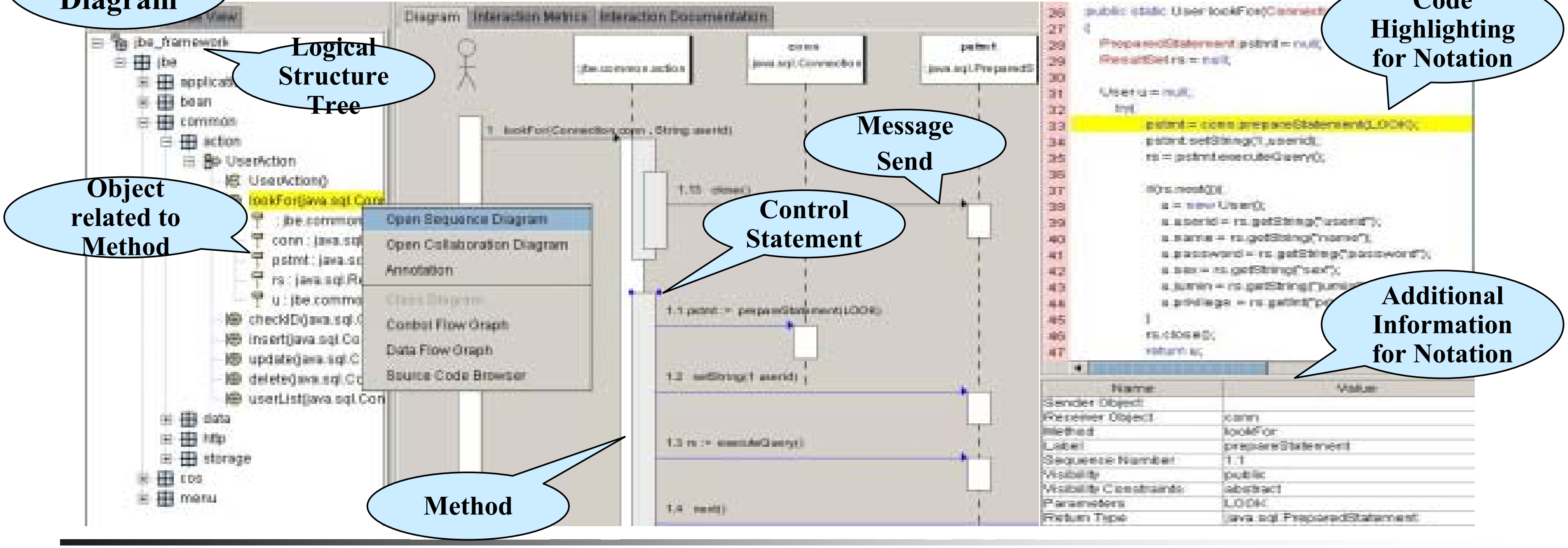
Quality characteristic	Quality sub-characteristic	Effect
High-risk Components	SW Potential Error Measurement -Architecture's Potential Error	-SW Problem Identification -High-risk Components Identification
SW Optimization	SW Optimization Measurement -Code Optimization	-Impurity Code Prevention -Program Size & Run-Time Reduction -Testing Cost-Saving
SW Complexity	SW Quality Measurement -Method Size & Structure Metrics -Class Size & Structure Metrics -Class OO Metrics	-SW Quality & Productivity Improvement -SW Maintainability Improvement -SW Reusability Improvement -Testing Cost-Saving
SW Usability	Understandability	-ISO 9126-3 Usability Evaluation
SW Maintainability	Analyzability , Changeability, Stability, Testability	-ISO 9126-3 Maintainability Evaluation
System Design	Package Quality Measurement -Package's balance between abstractness and stability	-Well Designed (Structured) Package Identification . Component Candidate Identification

Reverse Engineering Tool

SW Quality - Reverse Engineering

- Automatic UML Diagram Generation
  .Class(Package) Diagram
  .Sequence Diagram
  .Collaboration Diagram

-Support Various Visualization
-Analyze Detail Design&Architecture of SW
-Identify High-risk SW Structure
-Prevent SW Potential Problems



```
public static User lookFor(Connect
{
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    User u = null;
      try{
          pstmt = conn.prepareStatement(LOOK);
          pstmt.setString(1,userid);
          rs = pstmt.executeQuery();

          if(rs.next()){
            u = new User();
            u.userid = rs.getString("userid");
            u.name = rs.getString("name");
            u.password = rs.getString("password");
            u.sex = rs.getString("sex");
            u.jumin = rs.getString("jumi
            u.privilege = rs.getInt("p
          }
          rs.close();
          return u;
```

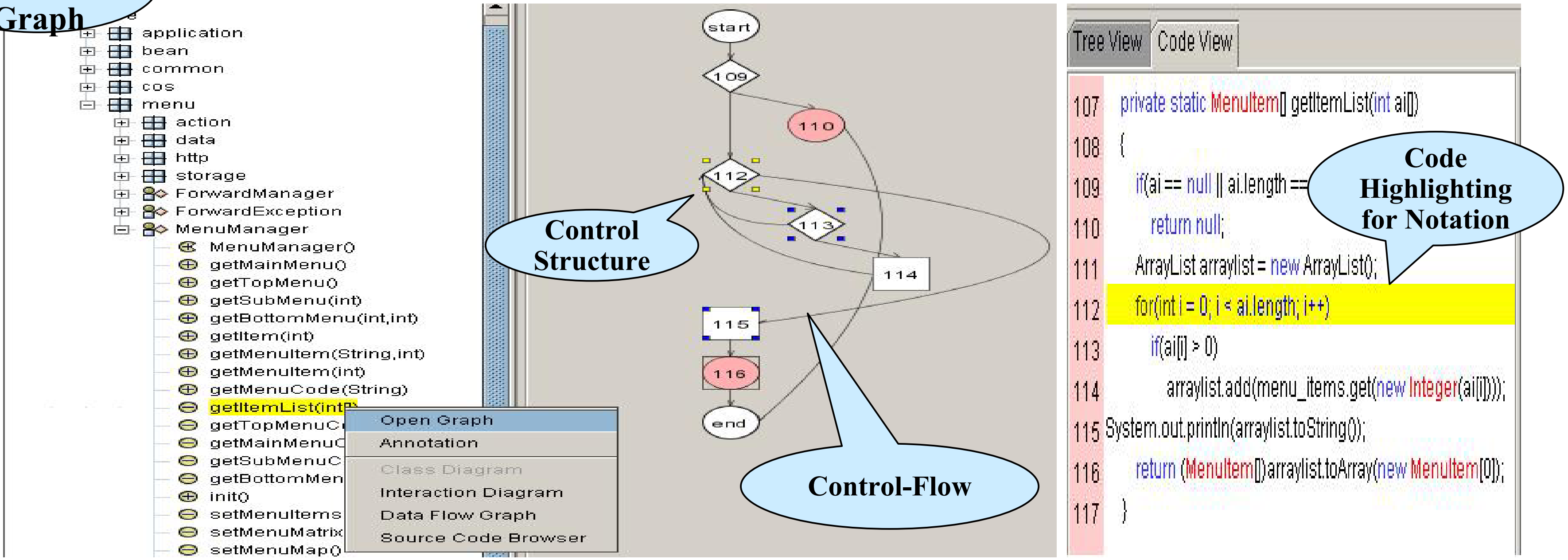
Name	Value
Sender Object	
Receiver Object	conn
Method	lookFor
Label	prepareStatement
Sequence Number	1.1
Visibility	public
Visibility Constraints	abstract
Parameters	LOOK
Return Type	java.sql.PreparedStatement

Reverse Engineering Tool

SW Quality - Reverse Engineering

- Automatic Graph Generation
  .Data-Flow Graph
  .Control-Flow Graph
  .Source Code Browser

-Control-Flow Analysis and Understand
  .Algorithm Structure Visualization
  .Structure Complexity Identification
  .Unconditional Structure(goto/exit/label)



```
private static MenuItem[] getItemList(int ai[])
{
  if(ai == null || ai.length ==
    return null;
  ArrayList arraylist = new ArrayList();
  for(int i = 0; i < ai.length; i++)
    if(ai[i] > 0)
      arraylist.add(menu_items.get(new Integer(ai[i])));
System.out.println(arraylist.toString());
  return (MenuItem[])arraylist.toArray(new MenuItem[0]);
}
```

Control-Flow Graph

Control Structure

Control-Flow

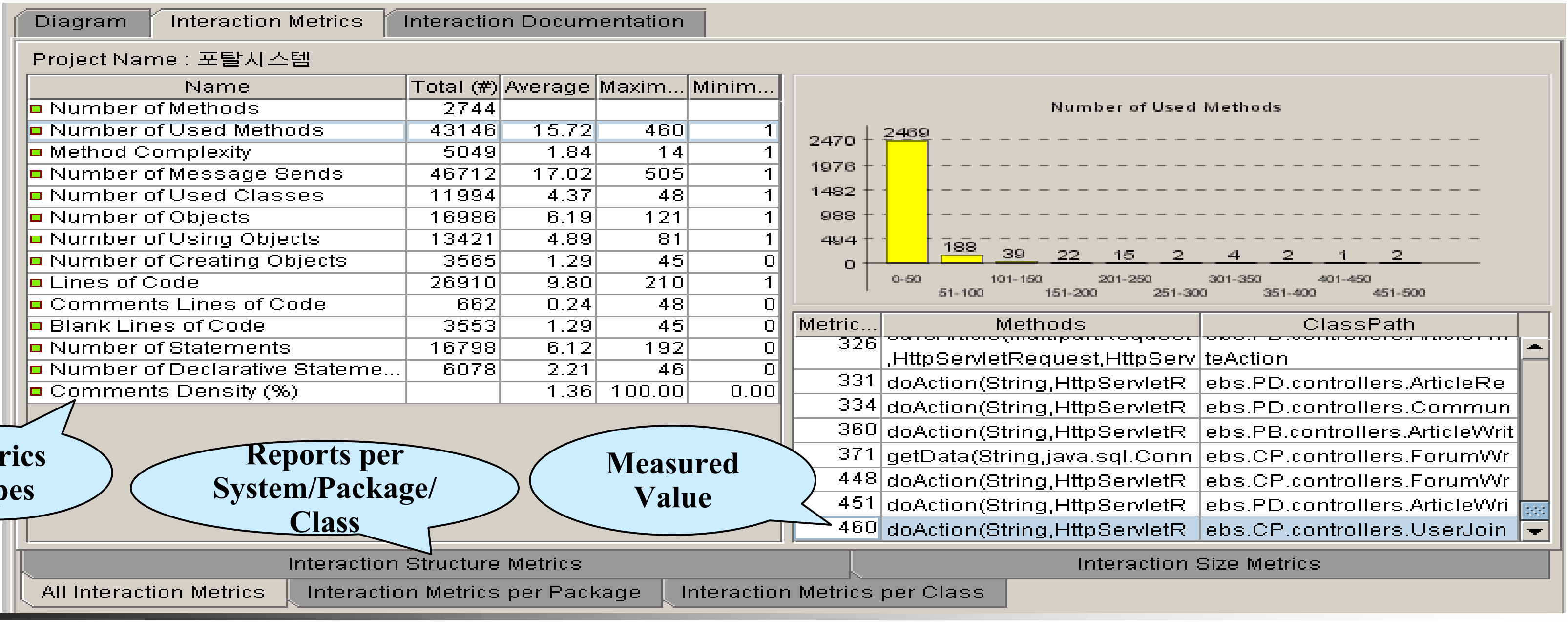
Code Highlighting for Notation

Reverse Engineering Tool

SW Quality – Quality Metrics(100+)

– Multi-level Statistical Metrics Reports
 .Class: Size, Coupling,Cohesion, etc.
 .Sequence: Method Call, Object, etc.
 .Control Flow: Complexity, Branch, etc.
 .Data Flow: Local/Global Variable, etc.

–Size, Structure, OO Metrics
 .SW Structure Analysis & Understand
 .High-risk Analysis of SW Structure
 .SW Complexity, Performance, etc.



Project Name : 포탈시스템

Name	Total (#)	Average	Maxim...	Minim...
🟩 Number of Methods	2744			
🟩 Number of Used Methods	43146	15.72	460	1
🟩 Method Complexity	5049	1.84	14	1
🟩 Number of Message Sends	46712	17.02	505	1
🟩 Number of Used Classes	11994	4.37	48	1
🟩 Number of Objects	16986	6.19	121	1
🟩 Number of Using Objects	13421	4.89	81	1
🟩 Number of Creating Objects	3565	1.29	45	0
🟩 Lines of Code	26910	9.80	210	1
🟩 Comments Lines of Code	662	0.24	48	0
🟩 Blank Lines of Code	3553	1.29	45	0
🟩 Number of Statements	16798	6.12	192	0
🟩 Number of Declarative Stateme...	6078	2.21	46	0
🟩 Comments Density (%)		1.36	100.00	0.00

Metric...	Methods	ClassPath
326	,HttpServletRequest,HttpServ	teAction
331	doAction(String,HttpServletR	ebs.PD.controllers.ArticleRe
334	doAction(String,HttpServletR	ebs.PD.controllers.Commun
360	doAction(String,HttpServletR	ebs.PB.controllers.ArticleWrit
371	getData(String,java.sql.Conn	ebs.CP.controllers.ForumWr
448	doAction(String,HttpServletR	ebs.CP.controllers.ForumWr
451	doAction(String,HttpServletR	ebs.PD.controllers.ArticleWri
460	doAction(String,HttpServletR	ebs.CP.controllers.UserJoin

Interaction Structure Metrics
Interaction Size Metrics
All Interaction Metrics
Interaction Metrics per Package
Interaction Metrics per Class

Metrics Types

Reports per System/Package/ Class

Measured Value

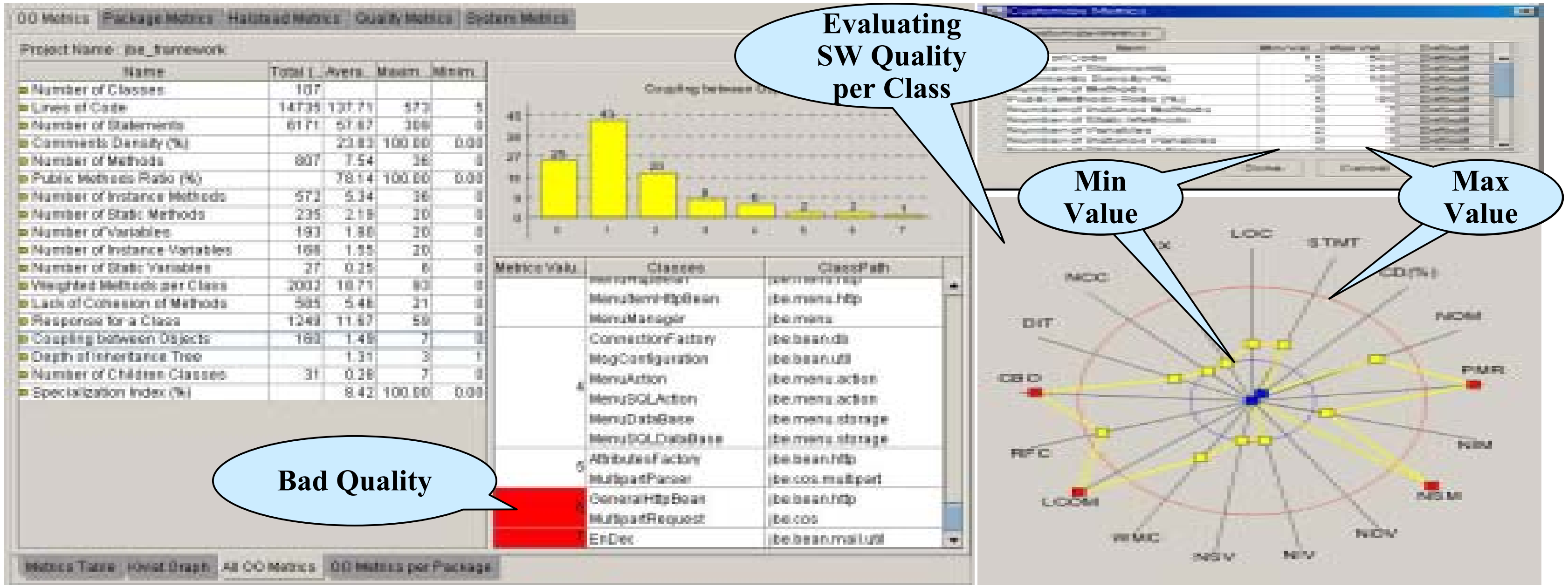
Soft4Soft

SW Quality Tool(Evaluation)

SW Quality - Software Metrics

-SW Quality Evaluation & Monitoring
 .80+Metrics(Customizing Quality Goals)
  .Measuring Good/Bad Quality
– Multi-level Statistical Metrics Reports

-OO Metrics
 .SW Size Metrics Evaluation
 .SW Structure Metrics Evaluation
 .Object-Oriented Metrics Evaluation



Project Name : jbe_framework

Name	Total ...	Avera...	Maxim...	Minim...
Number of Classes	107			
Lines of Code	14735	137.71	573	5
Number of Statements	6171	57.67	306	0
Comments Density (%)		23.83	100.00	0.00
Number of Methods	807	7.54	36	0
Public Methods Ratio (%)		78.14	100.00	0.00
Number of Instance Methods	572	5.34	36	0
Number of Static Methods	235	2.19	20	0
Number of Variables	193	1.80	20	0
Number of Instance Variables	166	1.55	20	0
Number of Static Variables	27	0.25	6	0
Weighted Methods per Class	2002	18.71	93	0
Lack of Cohesion of Methods	585	5.46	21	0
Response for a Class	1249	11.67	59	0
Coupling between Objects	160	1.49	7	0
Depth of Inheritance Tree		1.31	3	1
Number of Children Classes	31	0.28	7	0
Specialization Index (%)		8.42	100.00	0.00

Metrics Valu...	Classes	ClassPath
	MenuItemHttpBean	jbe.menu.http
	MenuManager	jbe.menu
4	ConnectionFactory	jbe.bean.db
	MsgConfiguration	jbe.bean.util
	MenuAction	jbe.menu.action
	MenuSQLAction	jbe.menu.action
	MenuDataBase	jbe.menu.storage
	MenuSQLDataBase	jbe.menu.storage
5	AttributesFactory	jbe.bean.http
	MultipartParser	jbe.cos.multipart
6	GeneralHttpBean	jbe.bean.http
	MultipartRequest	jbe.cos
7	EnDec	jbe.bean.mail.util

Evaluating SW Quality per Class

Min Value

Max Value

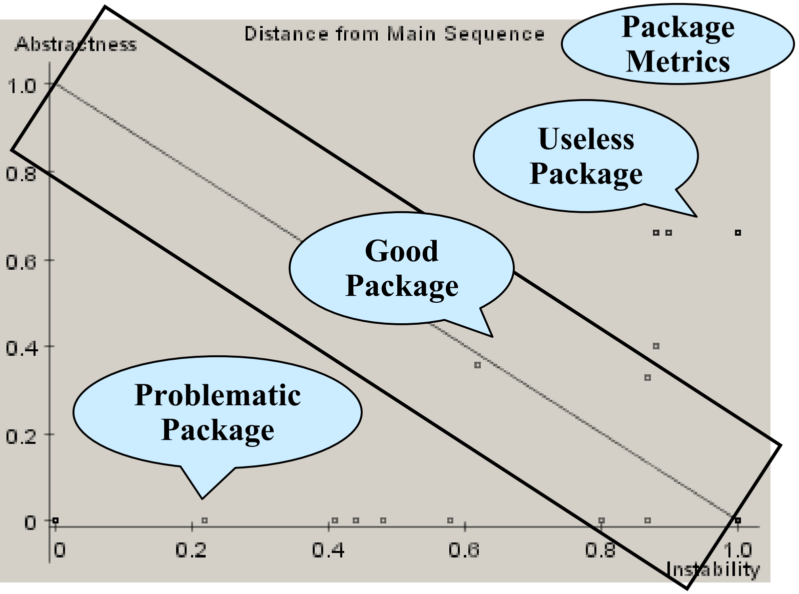
Bad Quality

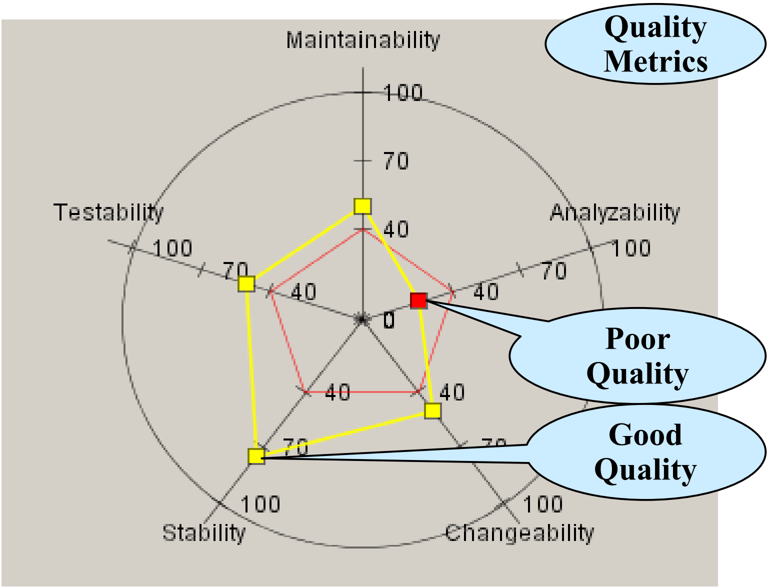
Soft4Soft

SW Quality Tool(Evaluation)

SW Quality - Software Metrics

-Package Metrics : Well Designed(Structured)/Reusability Package Identification
-Halstead Metrics : Non-optimized Code Evaluation
-Quality Metrics : ISO 9126-3 Maintainability Evaluation(Excellent, Good, Fair, Poor)
-System Metrics : Project Component Summary





SW Quality – Internal Complexity

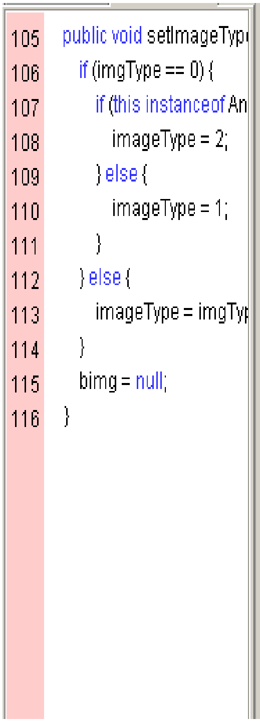
V(G)(Cyclomatic Complexity)

(Measure) measuring the control flow complexity of method in a method

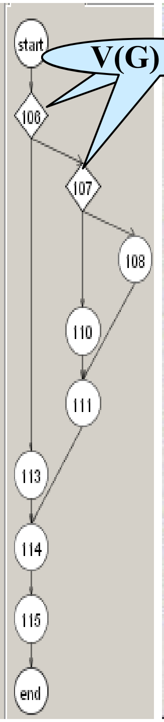
(Analysis) SW Complexity, SW Maintainability, Performance, Unit Testing Planning(Effort)

(Evaluation) 1% Bad Quality – Re-Design, Method Decomposition

# of Method	Recommended Value	Measure Value	# of Violation
2,100	1 ≤ V(G) ≤ 20	1 ≤ V(G) ≤ 65	32

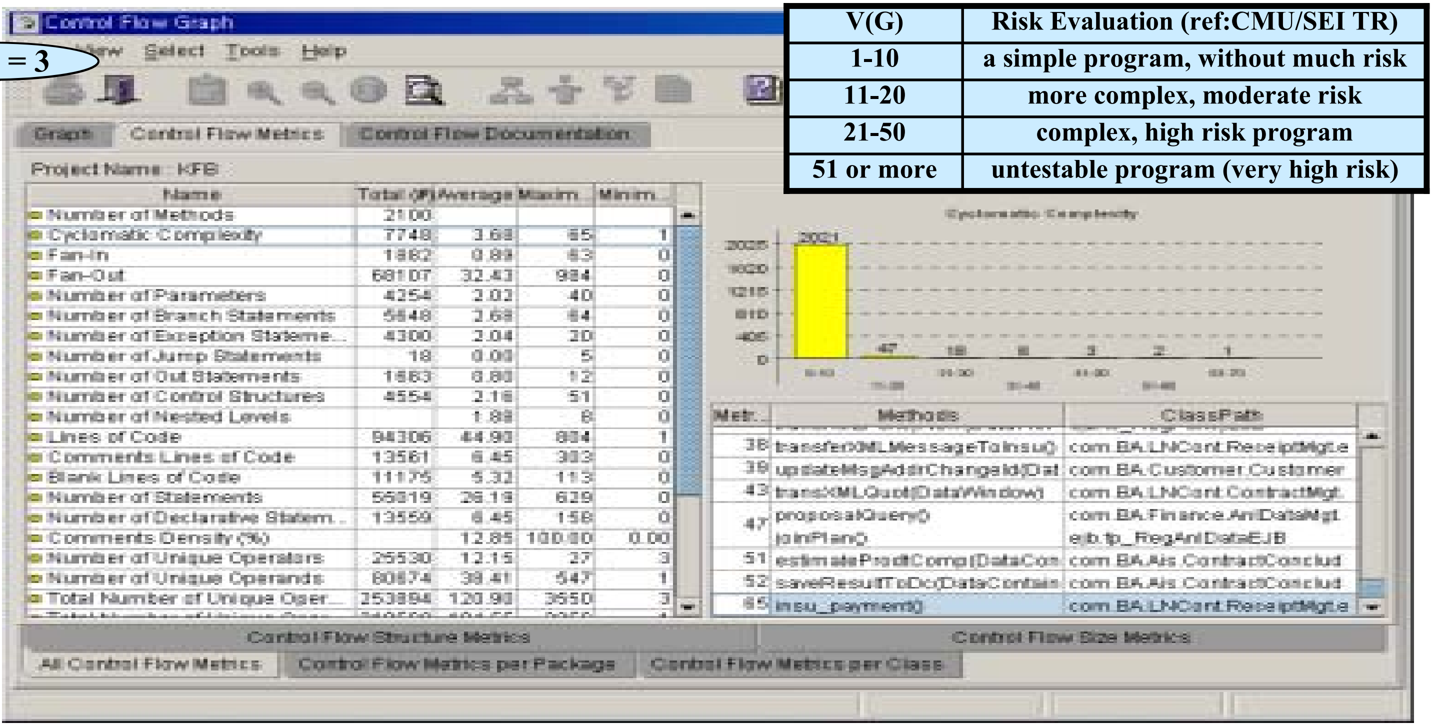


```
105  public void setImageTyp
106    if (imgType == 0) {
107      if (this instanceof An
108        imageType = 2;
109      } else {
110        imageType = 1;
111      }
112    } else {
113      imageType = imgTyp
114    }
115    bimg = null;
116  }
```



V(G) = 3

V(G)	Risk Evaluation (ref:CMU/SEI TR)
1-10	a simple program, without much risk
11-20	more complex, moderate risk
21-50	complex, high risk program
51 or more	untestable program (very high risk)



Project Name : KFB

Name	Total (#)	Average	Maxim...	Minim...
Number of Methods	2100			
Cyclomatic Complexity	7748	3.68	65	1
Fan-In	1882	0.89	63	0
Fan-Out	68107	32.43	984	0
Number of Parameters	4254	2.02	40	0
Number of Branch Statements	5648	2.68	64	0
Number of Exception Stateme...	4300	2.04	20	0
Number of Jump Statements	18	0.00	5	0
Number of Out Statements	1863	0.80	12	0
Number of Control Structures	4554	2.16	51	0
Number of Nested Levels		1.88	8	0
Lines of Code	94306	44.90	884	1
Comments Lines of Code	13561	6.45	303	0
Blank Lines of Code	11175	5.32	113	0
Number of Statements	55019	26.19	629	0
Number of Declarative Statem...	13559	6.45	158	0
Comments Density (%)		12.85	100.00	0.00
Number of Unique Operators	25530	12.15	27	3
Number of Unique Operands	80874	38.41	547	1
Total Number of Unique Oper...	253894	120.90	3550	3

Metr...	Methods	ClassPath
38	transferXMLMessageToInsu()	com.BA.LNCont.ReceiptMgt.e
39	updateMsgAddrChangeId(Dat	com.BA.Customer.Customer
43	transXMLQuot(DataWindow)	com.BA.LNCont.ContractMgt.

47	proposalQuery()
	joinPlan()
		com.BA.Finance.AnlDataMgt
		ejb.tp_RegAnlDataEJB

51	estimateProdComp(DataCon	com.BA.Ais.ContractConclud
52	saveResultToDc(DataContain	com.BA.Ais.ContractConclud
65	insu_payment()	com.BA.LNCont.ReceiptMgt.e

Control Flow Structure Metrics
Control Flow Size Metrics
All Control Flow Metrics
Control Flow Metrics per Package
Control Flow Metrics per Class

Soft4Soft

SW Quality – External Complexity

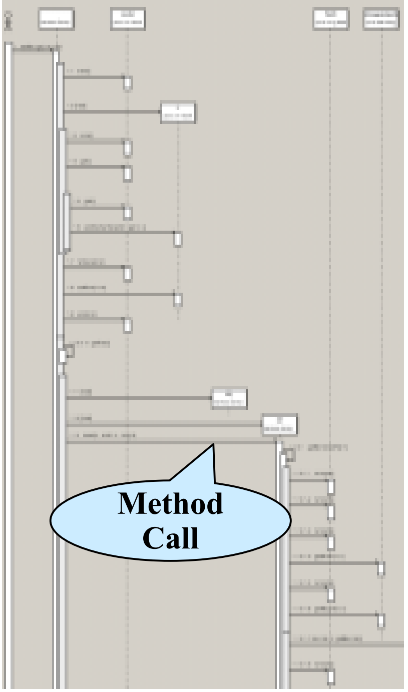
NOUM(Number of Used Methods)

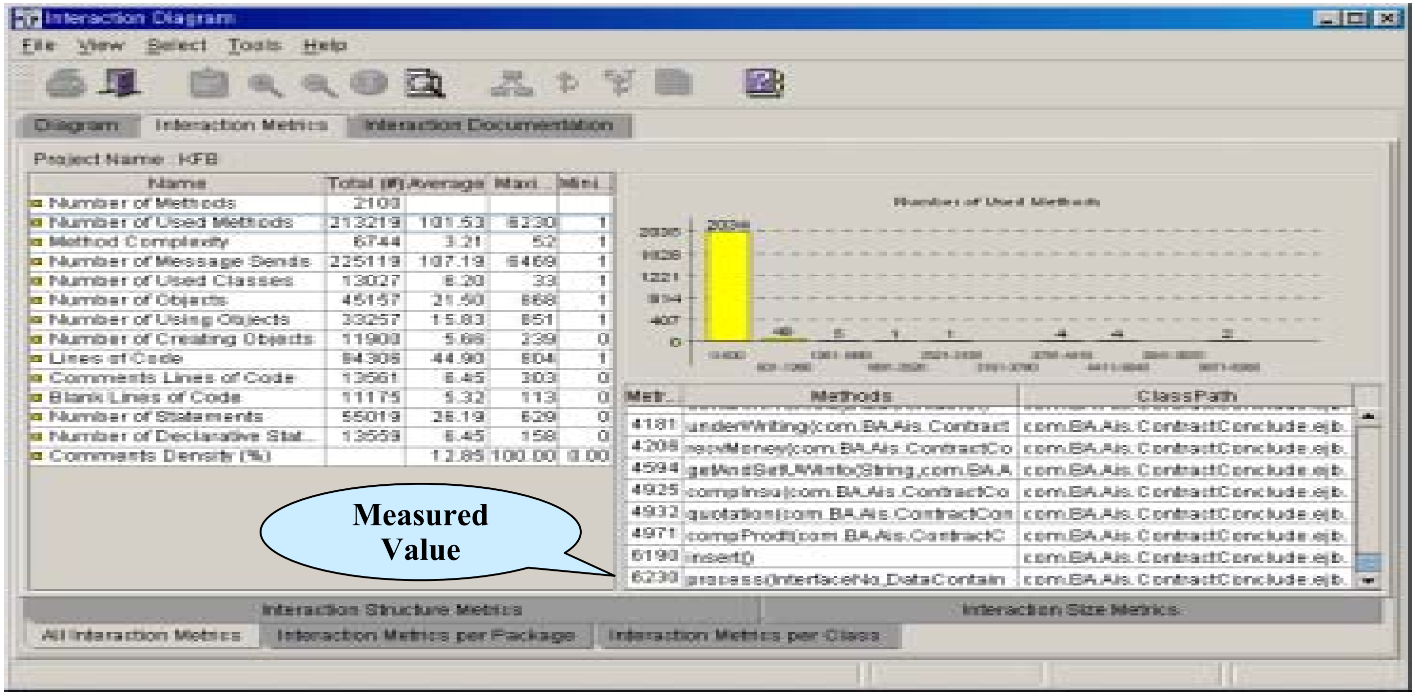
(Measure) measuring the set of all methods that can be executed by a method

(Analysis) SW Complexity, SW Maintainability, Performance, Integration Testing Effort

(Evaluation) 16% Bad Quality – Re-Design, Method Decomposition

# of Method	Recommended Value	Measure Value	# of Violation
2,100	1 ≤ NOUM ≤ 100	1 ≤ NOUM ≤ 6230	343


Method Call



Project Name : KFB

Name	Total (#)	Average	Maxi...	Mini...
Number of Methods	2100			
Number of Used Methods	213219	101.53	6230	1
Method Complexity	6744	3.21	52	1
Number of Message Sends	225119	107.19	6469	1
Number of Used Classes	13027	6.20	33	1
Number of Objects	45157	21.50	668	1
Number of Using Objects	33257	15.83	651	1
Number of Creating Objects	11903	5.66	239	0
Lines of Code	94306	44.90	604	1
Comments Lines of Code	13561	6.45	303	0
Blank Lines of Code	11175	5.32	113	0
Number of Statements	55019	26.19	629	0
Number of Declarative Stat...	13559	6.45	158	0
Comments Density (%)		12.85	100.00	0.00

Metr...	Methods	ClassPath
4181	underWriting(com.BA.Ais.Contract	com.BA.Ais.ContractConclude.ejb.
4208	recvMoney(com.BA.Ais.ContractCo	com.BA.Ais.ContractConclude.ejb.
4594	getAndSetLWInfo(String,com.BA.A	com.BA.Ais.ContractConclude.ejb.
4925	compInsu(com.BA.Ais.ContractCo	com.BA.Ais.ContractConclude.ejb.
4932	quotation(com.BA.Ais.ContractCon	com.BA.Ais.ContractConclude.ejb.
4971	compProd(com.BA.Ais.ContractC	com.BA.Ais.ContractConclude.ejb.
6193	insert()	com.BA.Ais.ContractConclude.ejb.
6230	process(InterfaceNo,DataContain	com.BA.Ais.ContractConclude.ejb.

Measured Value

Soft4Soft

SW Quality – Package Design

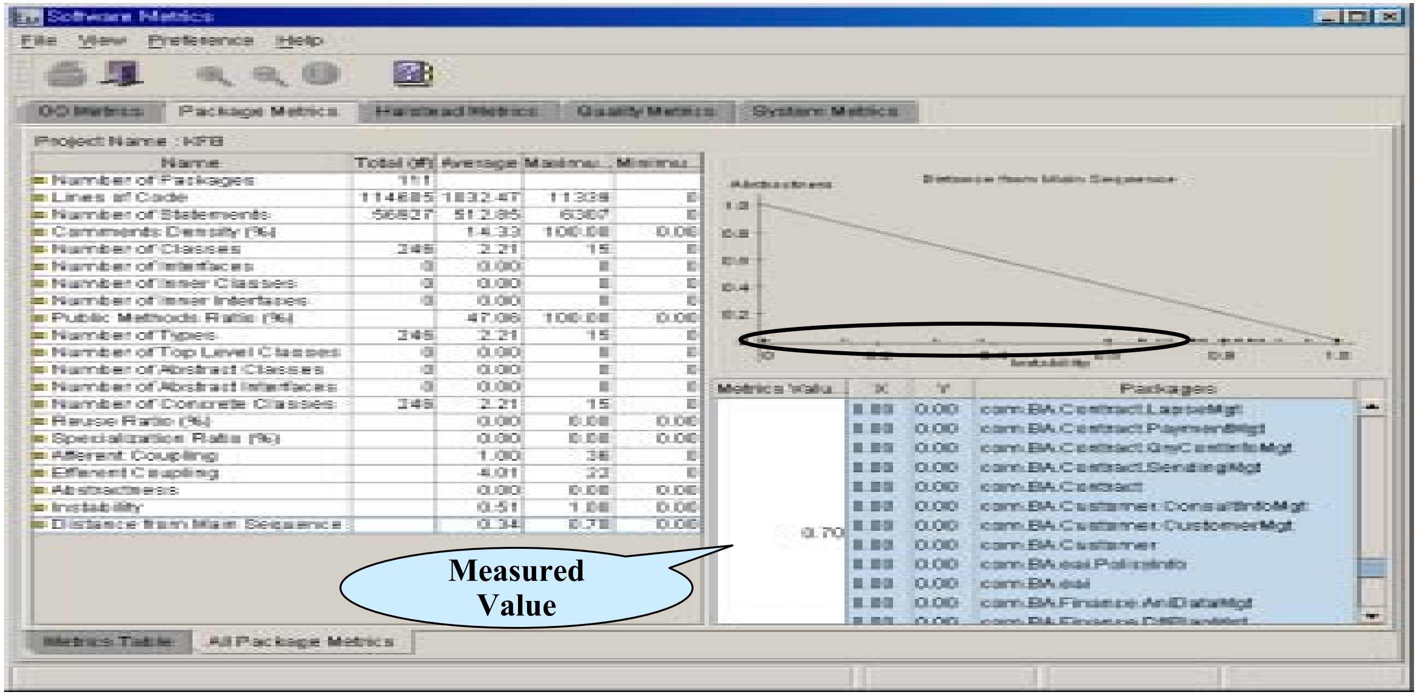
DIST(Distance from Main Sequence)

(Measure) measuring the degrees of abstraction and stability of a package

(Analysis) System Extensibility, System Changeability

(Evaluation) 66% Bad Quality – Re-design

# of Package	Recommended Value	Measure Value	# of Violation
111	0 ≤ NOUM ≤ 0.4	0 ≤ NOUM ≤ 0.7	74



Project Name : KFB

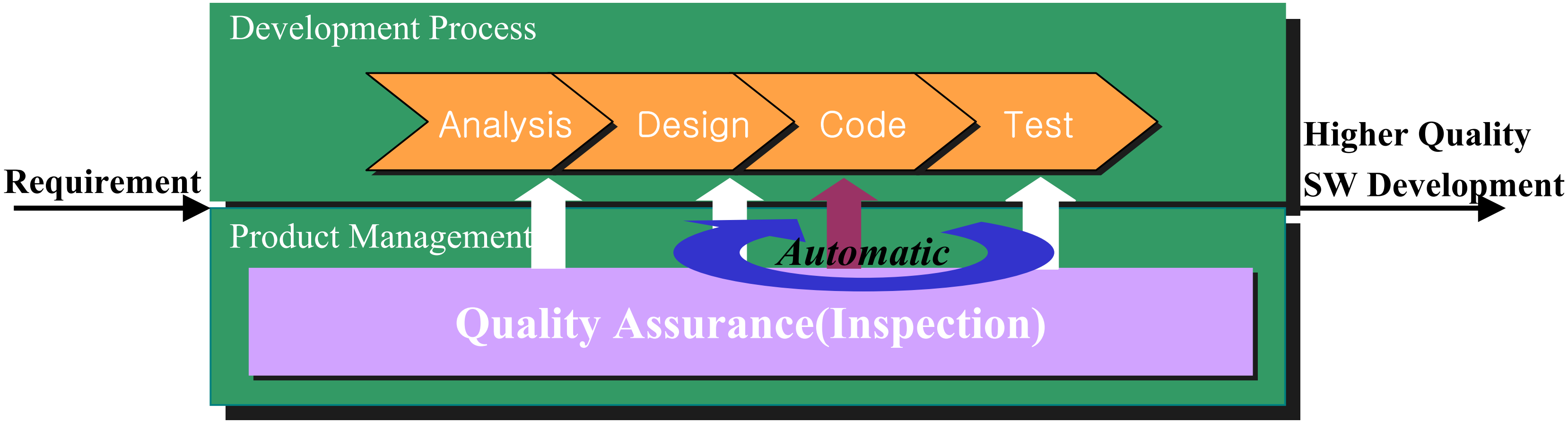
Name	Total (#)	Average	Maximu...	Minimu...
Number of Packages	111			
Lines of Code	114685	1832.47	11339	0
Number of Statements	56827	512.85	6307	0
Comments Density (%)		14.33	100.00	0.00
Number of Classes	245	2.21	15	0
Number of Interfaces	0	0.00	0	0
Number of Inner Classes	0	0.00	0	0
Number of Inner Interfaces	0	0.00	0	0
Public Methods Ratio (%)		47.06	100.00	0.00
Number of Types	245	2.21	15	0
Number of Top Level Classes	0	0.00	0	0
Number of Abstract Classes	0	0.00	0	0
Number of Abstract Interfaces	0	0.00	0	0
Number of Concrete Classes	245	2.21	15	0
Reuse Ratio (%)		0.00	0.00	0.00
Specialization Ratio (%)		0.00	0.00	0.00
Afferent Coupling		1.00	36	0
Efferent Coupling		4.01	22	0
Abstractness		0.00	0.00	0.00
Instability		0.51	1.00	0.00
Distance from Main Sequence		0.34	0.70	0.00

Metrics Valu	X	Y	Packages
	0.83	0.00	com.BA.Contract.LapseMgt
	0.83	0.00	com.BA.Contract.PaymentMgt
	0.83	0.00	com.BA.Contract.QryContractsMgt
	0.83	0.00	com.BA.Contract.SendingMgt
	0.83	0.00	com.BA.Contract
	0.83	0.00	com.BA.Customer.ConsultInfoMgt
0.70	0.83	0.00	com.BA.Customer.CustomerMgt
	0.83	0.00	com.BA.Customer
	0.83	0.00	com.BA.eai.PolicyInfo
	0.83	0.00	com.BA.eai
	0.83	0.00	com.BA.Finance.AnlDataMgt
	0.83	0.00	com.BA.Finance.DiffPaidMgt

Measured Value

Code Checker Tool

- Code Error Detection
  - Readability
  - Performance
  - Run-time error(DB Interface) – Memory Leak
- Code Quality and SW Process Model Map
  - Identify source code problems early in the development cycle



Code Inspection Solution

- **Code Inspection and and Expected Effect**

Inspection characteristic	Inspection sub-characteristic	Effect
Readability	Layout Style -Naming Convention Rules, etc. -Indentation & Comments Rules, etc.	-Readability Improvement -Easy to Maintenance
Potential error and Performance	Performance and Memory Guideline -Variable, Control, Exception Statement Rules, etc. -Unused Variable & Method Rules -I/O Resources Release Rules -System Statements Rules -EJB Statements Rules	-Performance Increase -Dead Code Prevention -Memory Leak Prevention -System Load Prevention -Testing Cost-Saving
DB Interface	Performance and Memory Guideline -BC4J Resources Release Rules -JDBC Resources Release Rules -JDBC with Framework Resources Release Rules	-Performance Increase -Memory Leak Prevention -System Load Prevention -Data integrity

Code Checker Tool

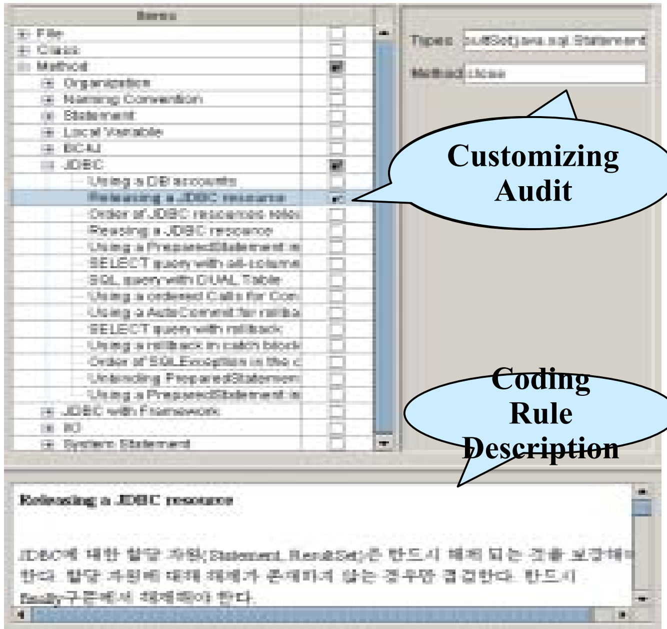
- Four Main Violation Types
  - These are essentially programming errors
  - Dead Code – Code Deletion
    - Unused Method
    - Unused Global Variable
    - Unused Local Variable
  - Performance – Code Modification
    - Method Call/Declaration in Loop Conditions
    - String Assignment
    - Console Message
  - Potential Error – Code Insertion
    - Empty Block Body
    - Empty catch/finally Block
    - Non return in the finally Block
    - I/O Resource Release
  - Memory Leak – Code Modification
    - JDBC Resource Release
    - Order of JDBC resources release

Code Checker Tool

Code Inspection - Code Checker

- Code Inspection per File/Class/Method
  - .110+ Rules(Customizing Quality Goals)
  - . Reporting Violation code& Messages
- – customize and extend coding rules

- Coding Style Inspection
  - .Improving Readability & Maintainability
- Performance, Memory Leak Inspection
  - .Preventing Coding Errors

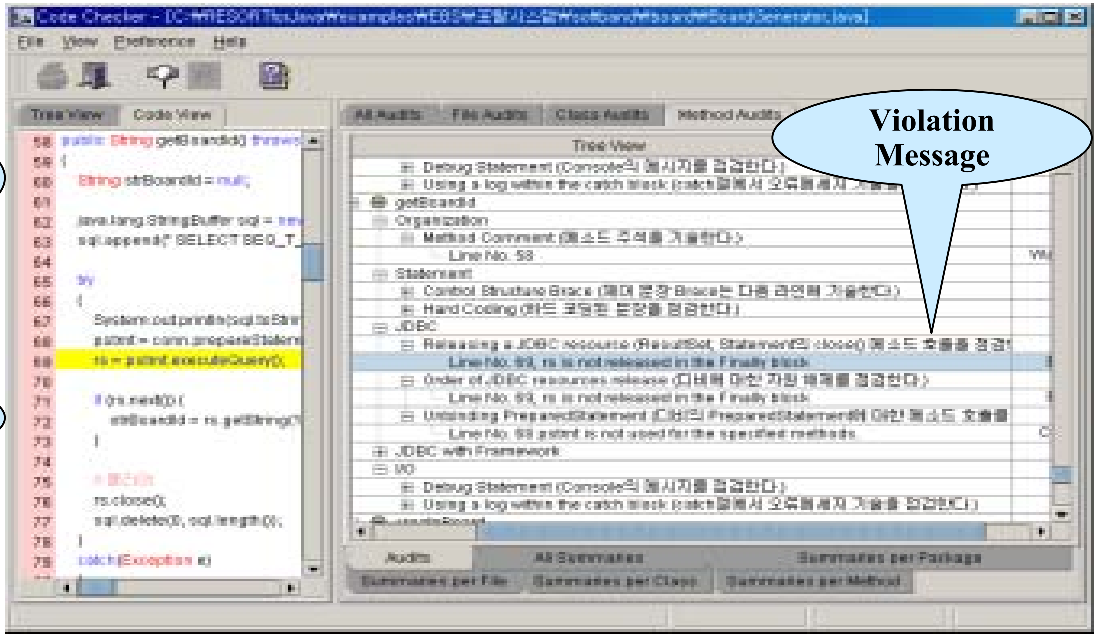

Items
File
Class
Method
Organization
Naming Convention
Statement
Local Variable
BCAJ
JDBC
Using a DB accounts
Releasing a JDBC resource
Order of JDBC resources relea
Reusing a JDBC resource
Using a PreparedStatement
SELECT query with all-column
SQL query with DUAL Table
Using a ordered Calls for Con
Using a AutoCommit for rollba
SELECT query with rollback
Using a rollback in catch block
Order of SQLException in the c
Unbinding PreparedStatement
Using a PreparedStatement
JDBC with Framework
IO
System Statement
Types: ultSet,java.sql.Statement
Method: close

Releasing a JDBC resource

JDBC에 대한 할당 자원(Statement, ResultSet)은 반드시 해제 되는 것을 보장해야 한다. 할당 자원에 대해 해제가 존재하지 않는 경우만 점검한다. 반드시 finally구문에서 해제해야 한다.

Customizing Audit

Coding Rule Description



```
public String getBoardId() throws
{
    String strBoardId = null;

    java.lang.StringBuffer sql = new
    sql.append(" SELECT SEQ_T_

    try
    {
        System.out.println(sql.toStrin
        pstmt = conn.prepareStateme
        rs = pstmt.executeQuery();

        if (rs.next()) {
            strBoardId = rs.getString(
        }

        // 종료처리
        rs.close();
        sql.delete(0, sql.length());
    }
    catch(Exception e)
```

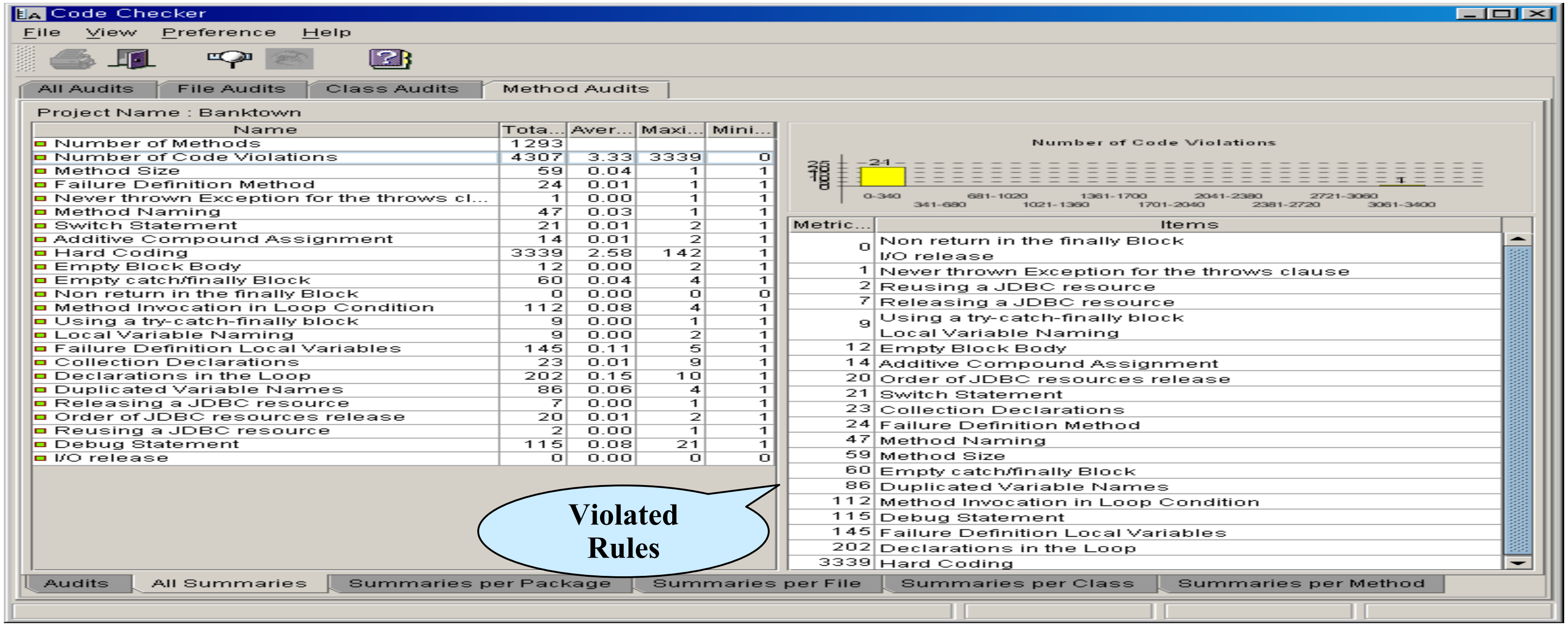
```
⊞ Debug Statement (Console의 메시지를 점검한다.)
⊞ Using a log within the catch block (catch절에서 오류메세지 기술...)
⊟ getBoardId
  ⊟ Organization
    ⊟ Method Comment (메소드 주석를 기술한다.)
        Line No. 58
  ⊟ Statement
    ⊞ Control Structure Brace (제어 문장 Brace는 다음 라인에 기술한다.)
    ⊞ Hard Coding (하드 코딩된 문장를 점검한다.)
  ⊟ JDBC
    ⊟ Releasing a JDBC resource (ResultSet, Statement의 close() 메소드 호출를 점검
        Line No. 69, rs is not released in the Finally block
    ⊟ Order of JDBC resources release (다비비 마친 자원 해제를 점검한다.)
        Line No. 69, rs is not released in the Finally block
    ⊟ Unbinding PreparedStatement (다른 PreparedStatement에 대한 메소드 호출를
        Line No. 68 pstmt is not used for the specified methods
  ⊞ JDBC with Framework
  ⊟ I/O
    ⊞ Debug Statement (Console의 메시지를 점검한다.)
    ⊞ Using a log within the catch block (catch절에서 오류메세지 기술을 점검한다.)
```

Violation Message

Code Checker Tool

Code Inspection - Code Checker

- Managing/Monitoring Code Quality per File/Class/Method
  .Rule Summary Report – Analysis for Violated Coding Rule
  .Each Rule Report – Analysis for Violated File/Class/Method per Rule



Project Name : Banktown

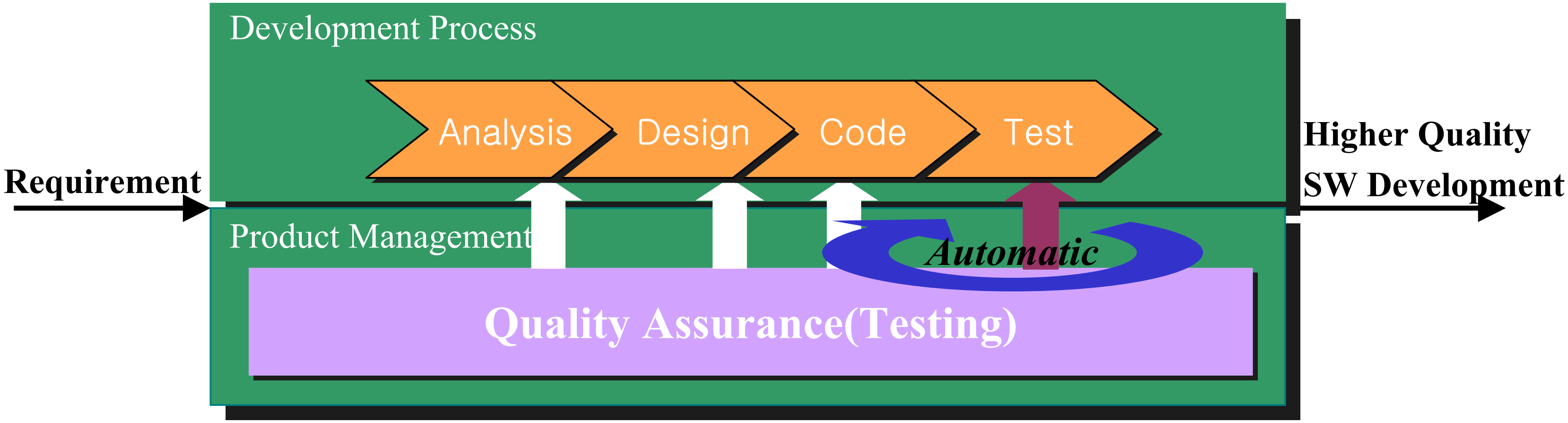
Name	Tota...	Aver...	Maxi...	Mini...
Number of Methods	1293			
Number of Code Violations	4307	3.33	3339	0
Method Size	59	0.04	1	1
Failure Definition Method	24	0.01	1	1
Never thrown Exception for the throws cl...	1	0.00	1	1
Method Naming	47	0.03	1	1
Switch Statement	21	0.01	2	1
Additive Compound Assignment	14	0.01	2	1
Hard Coding	3339	2.58	142	1
Empty Block Body	12	0.00	2	1
Empty catch/finally Block	60	0.04	4	1
Non return in the finally Block	0	0.00	0	0
Method Invocation in Loop Condition	112	0.08	4	1
Using a try-catch-finally block	9	0.00	1	1
Local Variable Naming	9	0.00	2	1
Failure Definition Local Variables	145	0.11	5	1
Collection Declarations	23	0.01	9	1
Declarations in the Loop	202	0.15	10	1
Duplicated Variable Names	86	0.06	4	1
Releasing a JDBC resource	7	0.00	1	1
Order of JDBC resources release	20	0.01	2	1
Reusing a JDBC resource	2	0.00	1	1
Debug Statement	115	0.08	21	1
I/O release	0	0.00	0	0

Metric...	Items
0	Non return in the finally Block I/O release
1	Never thrown Exception for the throws clause
2	Reusing a JDBC resource
7	Releasing a JDBC resource
9	Using a try-catch-finally block Local Variable Naming
12	Empty Block Body
14	Additive Compound Assignment
20	Order of JDBC resources release
21	Switch Statement
23	Collection Declarations
24	Failure Definition Method
47	Method Naming
59	Method Size
60	Empty catch/finally Block
86	Duplicated Variable Names
112	Method Invocation in Loop Condition
115	Debug Statement
145	Failure Definition Local Variables
202	Declarations in the Loop
3339	Hard Coding

Violated Rules

Test Tool

- Unit/Integration Testing
  - Try to detect problems with algorithms and/or logic (flow of control)
  - Try to detect problems with manipulation of data (data structures)
  - Try to detect method invocation problems
  - Try to detect method response time problems
- Test and SW Process Model Map
  - Identify Logic and Interface errors in the development cycle



Test Tool(Design)

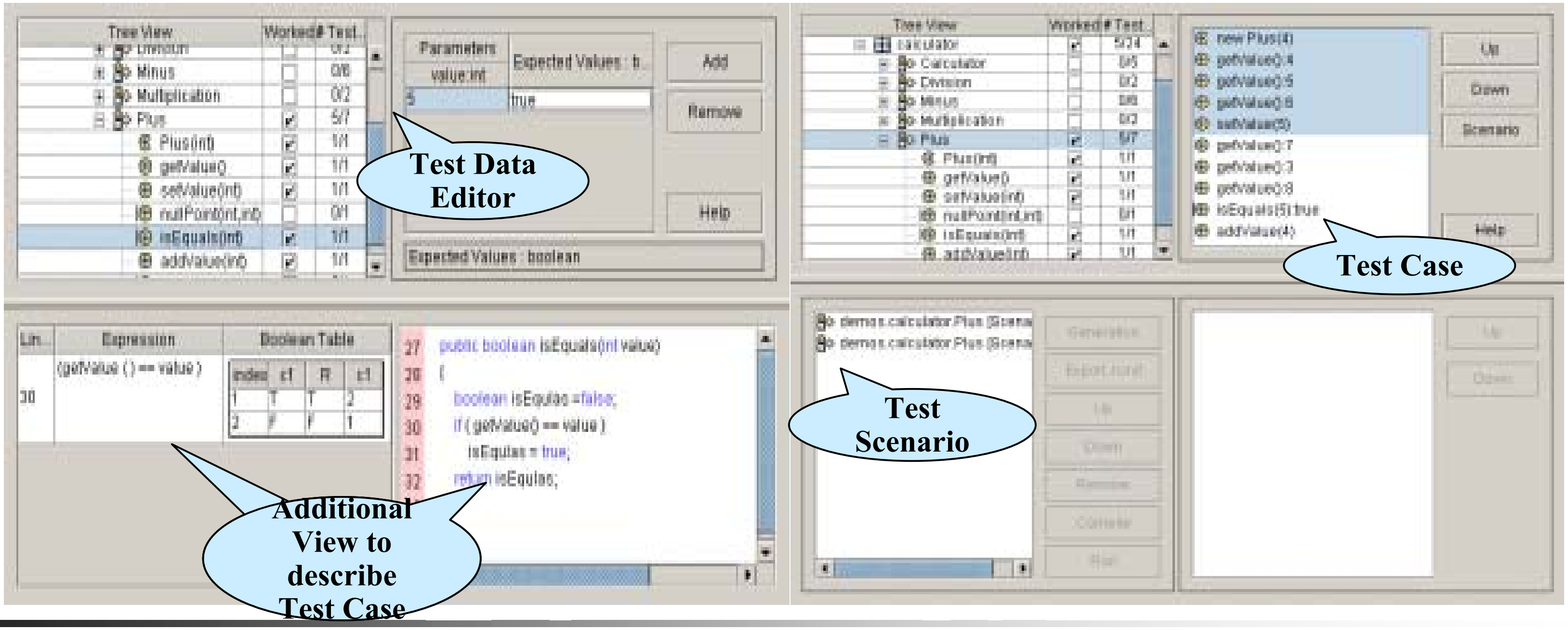
Unit/Integration Testing – Test Design

Test Case Design

- Boolean Table & Source Code
  : Supporting Optimized Test Case
- Input & Expected Value Audit

Test Scenario Design

- Scenario per Class/Package/Project
- Test Suite
- Transformation to JUnit Framework



Tree View	Worked	# Test
Division		0/2
Minus	☐	0/6
Multiplication	☐	0/2
Plus	☑	5/7
Plus(int)	☑	1/1
getValue()	☑	1/1
setValue(int)	☑	1/1
nullPoint(int,int)	☐	0/1
isEquals(int)	☑	1/1
addValue(int)	☑	1/1

Parameters value:int	Expected Values : b...
5	true

Expected Values : boolean

Test Data Editor

Lin...	Expression	Boolean Table
30	(getValue () == value)	index / c1 / R / c1 1 / T / T / 2 2 / F / F / 1

```
public boolean isEquals(int value)
{
   boolean isEqulas =false;
   if ( getValue() == value )
      isEqulas = true;
   return isEqulas;
```

Additional View to describe Test Case

Tree View	Worked	# Test
calculator	☑	5/24
Calculator	☐	0/5
Division	☐	0/2
Minus	☐	0/6
Multiplication	☐	0/2
Plus	☑	5/7
Plus(int)	☑	1/1
getValue()	☑	1/1
setValue(int)	☑	1/1
nullPoint(int,int)	☐	0/1
isEquals(int)	☑	1/1
addValue(int)	☑	1/1

new Plus(4)
getValue():4
getValue():5
getValue():6
setValue(5)
getValue():7
getValue():3
getValue():8
isEquals(5):true
addValue(4)

Test Case

demos.calculator.Plus [Scena
demos.calculator.Plus [Scena

Test Scenario

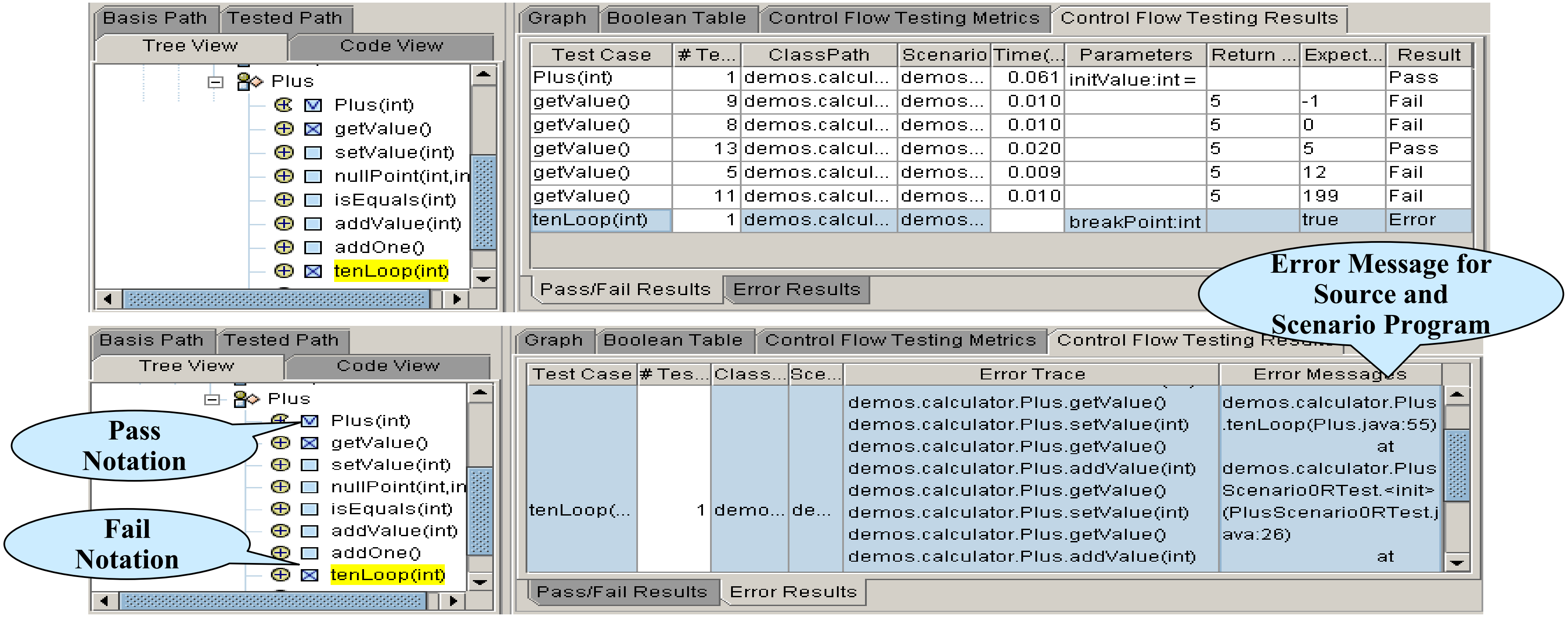
Test Tool(Evaluation)

Unit/Integration Testing – Test Result Analysis

Test Result Analysis per Test Case

- Test Case Result Analysis : Pass, Fail, or Error

-Run-time Error Analysis : Error Trace, Error Message



Plus
- ☑ Plus(int)
- ☒ getValue()
- □ setValue(int)
- □ nullPoint(int,in
- □ isEquals(int)
- □ addValue(int)
- □ addOne()
- ☒ tenLoop(int)

Test Case	# Te...	ClassPath	Scenario	Time(...	Parameters	Return ...	Expect...	Result
Plus(int)	1	demos.calcul...	demos...	0.061	initValue:int =			Pass
getValue()	9	demos.calcul...	demos...	0.010		5	-1	Fail
getValue()	8	demos.calcul...	demos...	0.010		5	0	Fail
getValue()	13	demos.calcul...	demos...	0.020		5	5	Pass
getValue()	5	demos.calcul...	demos...	0.009		5	12	Fail
getValue()	11	demos.calcul...	demos...	0.010		5	199	Fail
tenLoop(int)	1	demos.calcul...	demos...		breakPoint:int		true	Error

Error Message for Source and Scenario Program

Pass Notation

Fail Notation

Test Case	# Tes...	Class...	Sce...	Error Trace	Error Messages

tenLoop(...	1	demo...	de...	demos.calculator.Plus.getValue()
				demos.calculator.Plus.setValue(int)
				demos.calculator.Plus.getValue()
				demos.calculator.Plus.addValue(int)
				demos.calculator.Plus.getValue()
				demos.calculator.Plus.setValue(int)
				demos.calculator.Plus.getValue()
				demos.calculator.Plus.addValue(int)
					demos.calculator.Plus
					.tenLoop(Plus.java:55)
					at
					demos.calculator.Plus
					Scenario0RTest.<init>
					(PlusScenario0RTest.j
					ava:26)
					at

Soft4Soft

Test Tool(Evaluation)

Unit/Integration Testing – Coverage and Performance Analysis

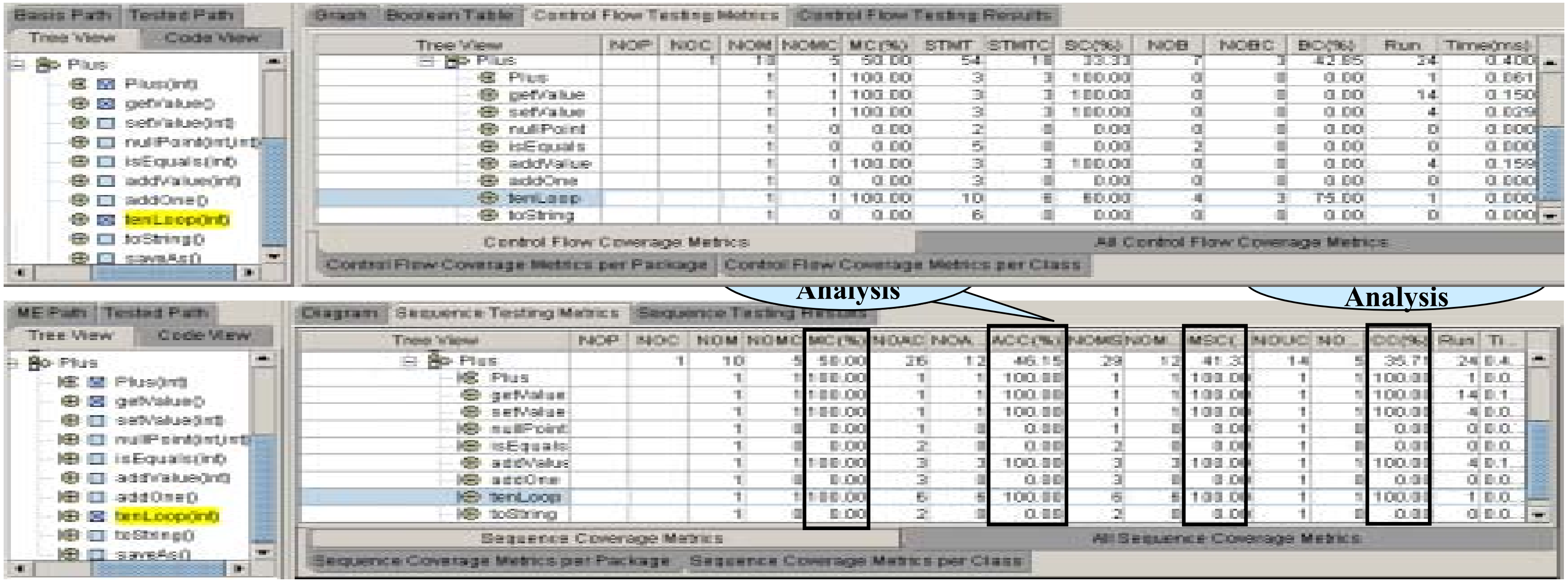
Coverage Analysis : 30+ Coverage Metrics

-Control Flow Coverage: Statement, Branch

- Data Flow Coverage: All-DU Path, All-C-Uses Path, All-P-Uses Path

-Sequence Coverage: Activation, Message-send, Class

Performance Analysis(Time/Run) : Bottlenecks Analysis



Basis Path | Tested Path

Tree View | Code View

- Plus
  - Plus(int)
  - getValue()
  - setValue(int)
  - nullPoint(int,int)
  - isEquals(int)
  - addValue(int)
  - addOne()
  - tenLoop(int)
  - toString()
  - saveAs()

Graph | Boolean Table | Control Flow Testing Metrics | Control Flow Testing Results

Tree View	NOP	NOC	NOM	NOMC	MC(%)	STMT	STMTC	SC(%)	NOB	NOBC	BC(%)	Run	Time(ms)
Plus		1	10	5	50.00	54	18	33.33	7	3	42.85	24	0.400
Plus			1	1	100.00	3	3	100.00	0	0	0.00	1	0.061
getValue			1	1	100.00	3	3	100.00	0	0	0.00	14	0.150
setValue			1	1	100.00	3	3	100.00	0	0	0.00	4	0.029
nullPoint			1	0	0.00	2	0	0.00	0	0	0.00	0	0.000
isEquals			1	0	0.00	5	0	0.00	2	0	0.00	0	0.000
addValue			1	1	100.00	3	3	100.00	0	0	0.00	4	0.159
addOne			1	0	0.00	3	0	0.00	0	0	0.00	0	0.000
tenLoop			1	1	100.00	10	6	60.00	4	3	75.00	1	0.000
toString			1	0	0.00	6	0	0.00	0	0	0.00	0	0.000

Control Flow Coverage Metrics | All Control Flow Coverage Metrics

Control Flow Coverage Metrics per Package | Control Flow Coverage Metrics per Class

ME Path | Tested Path

Tree View | Code View

- Plus
  - Plus(int)
  - getValue()
  - setValue(int)
  - nullPoint(int,int)
  - isEquals(int)
  - addValue(int)
  - addOne()
  - tenLoop(int)
  - toString()
  - saveAs()

Diagram | Sequence Testing Metrics | Sequence Testing Results

Tree View	NOP	NOC	NOM	NOMC	MC(%)	NOAC	NOA	ACC(%)	NOMS	NOM	MSC(	NOUC	NO..	CC(%)	Run	Ti..
Plus		1	10	5	50.00	26	12	46.15	29	12	41.37	14	5	35.71	24	0.4..
Plus			1	1	100.00	1	1	100.00	1	1	100.00	1	1	100.00	1	0.0..
getValue			1	1	100.00	1	1	100.00	1	1	100.00	1	1	100.00	14	0.1..
setValue			1	1	100.00	1	1	100.00	1	1	100.00	1	1	100.00	4	0.0..
nullPoint			1	0	0.00	1	0	0.00	1	0	0.00	1	0	0.00	0	0.0..
isEquals			1	0	0.00	2	0	0.00	2	0	0.00	1	0	0.00	0	0.0..
addValue			1	1	100.00	3	3	100.00	3	3	100.00	1	1	100.00	4	0.1..
addOne			1	0	0.00	3	0	0.00	3	0	0.00	1	0	0.00	0	0.0..
tenLoop			1	1	100.00	6	6	100.00	6	6	100.00	1	1	100.00	1	0.0..
toString			1	0	0.00	2	0	0.00	2	0	0.00	1	0	0.00	0	0.0..

Sequence Coverage Metrics | All Sequence Coverage Metrics

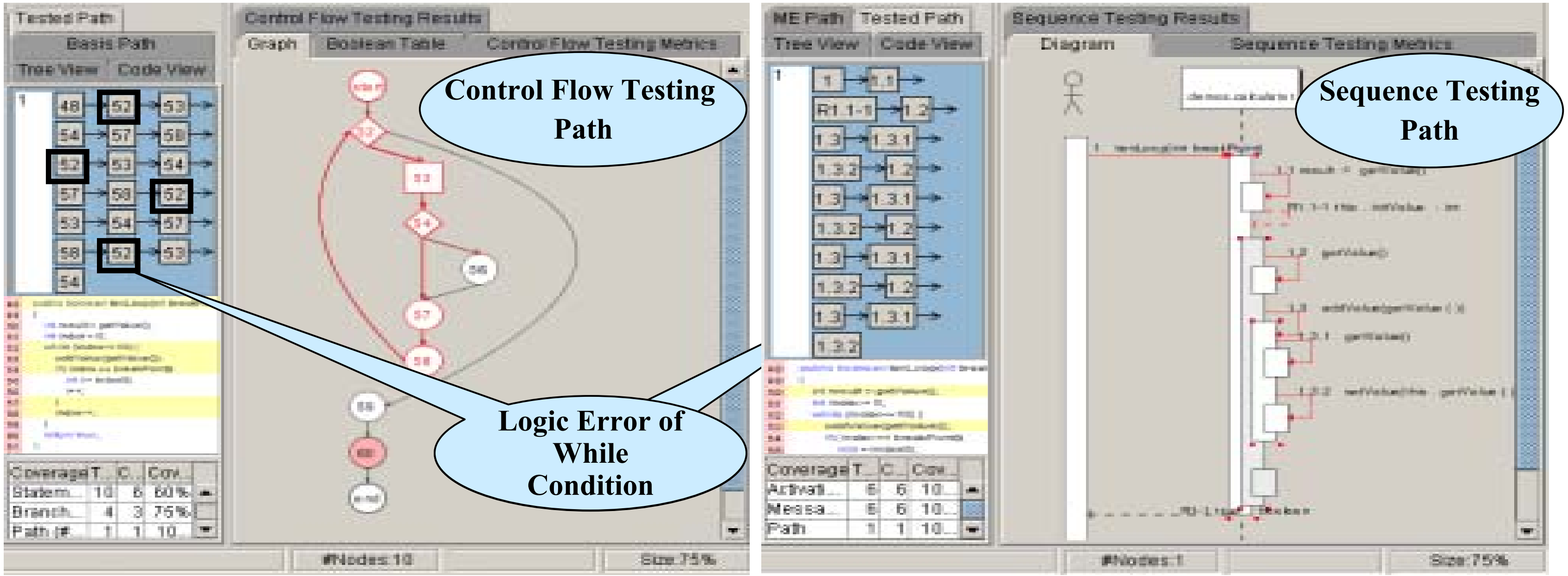
Sequence Coverage Metrics per Package | Sequence Coverage Metrics per Class

Test Tool(Evaluation)

Unit/Integration Testing – Historical Path Monitor

Executed Paths Analysis per Test Case
−Unit Testing(Control/Data Flow) Analysis
  .Executed Code and Path
  .Control/Data Flow Coverage
  .Logic Error

−Integration Testing(Sequence) Analysis
  . Executed Message and Path
  .OO Coverage
  .Interface Error



CoverageT...	C...	Cov...
Statem... 10	6	60%
Branch... 4	3	75%
Path (#... 1	1	10...

CoverageT...	C...	Cov...
Activati... 6	6	10...
Messa... 6	6	10...
Path 1	1	10...

Soft4Soft

Soft4Soft

T205, ICU VBI Center, 103-6, Munji-Dong,
Yousung-Gu, Daejon, 305-714

Tel : +82-42-866-6632~3
Fax: +82-42-866-6626

Sale Supports : sales@soft4soft.com
Technical Supports : info@soft4soft.com

www.soft4soft.com