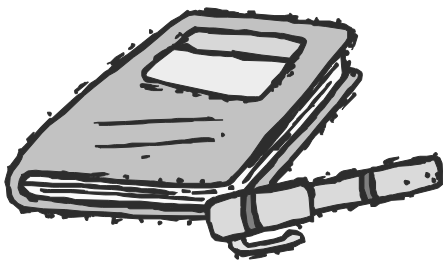


AsiaSTAR 2001

KyouGyou KeiKaku

(An Industry Project)



Dr Clive Boughton
Thom Larner
Annette Vincent
Masayoshi Kuroda

Toyo Technika & Software Improvements

INSPECTIONS

DEFINITION

An inspection is a

- * **formal**
- * **rigorous**
- * **in-depth**

technical review intended to discover defects as soon as possible after they are introduced (Rakitin - 1997).

Similar definitions are available from Gilb - 1988 and Sommerville - 1989.

Toyo Technika & Software Improvements

INSPECTIONS



DEFINITION

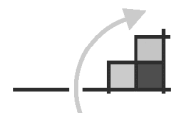
An error is a problem that is discovered during the phase in which it originates.

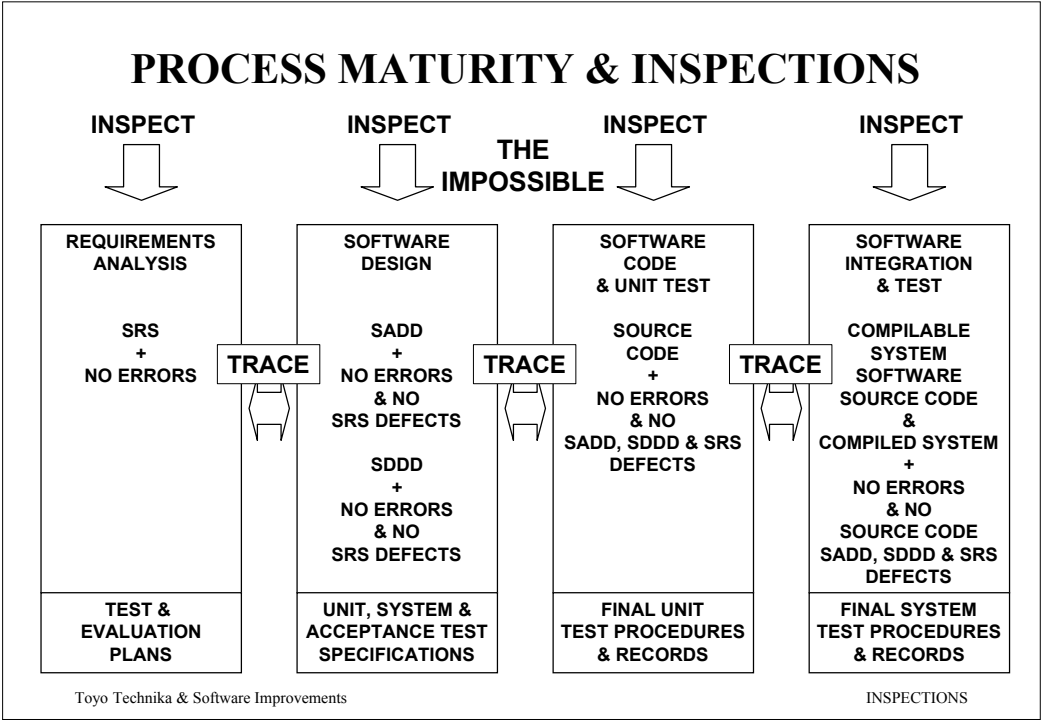
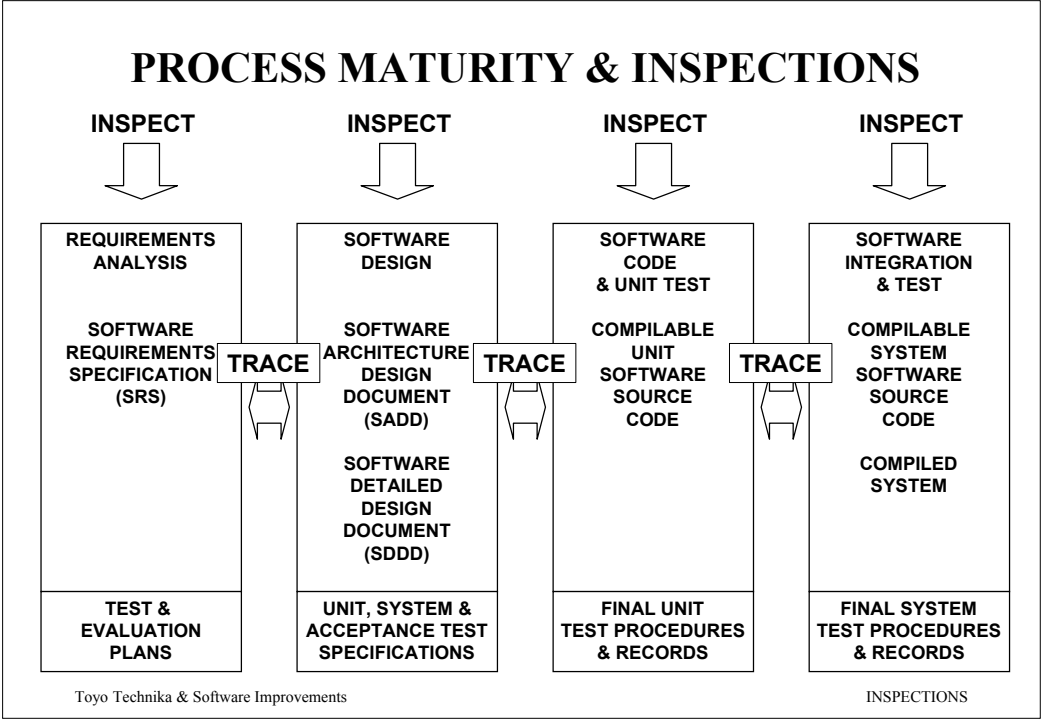
A defect is a problem that is discovered beyond the phase in which it originates.

It is better to eliminate errors before they become defects.

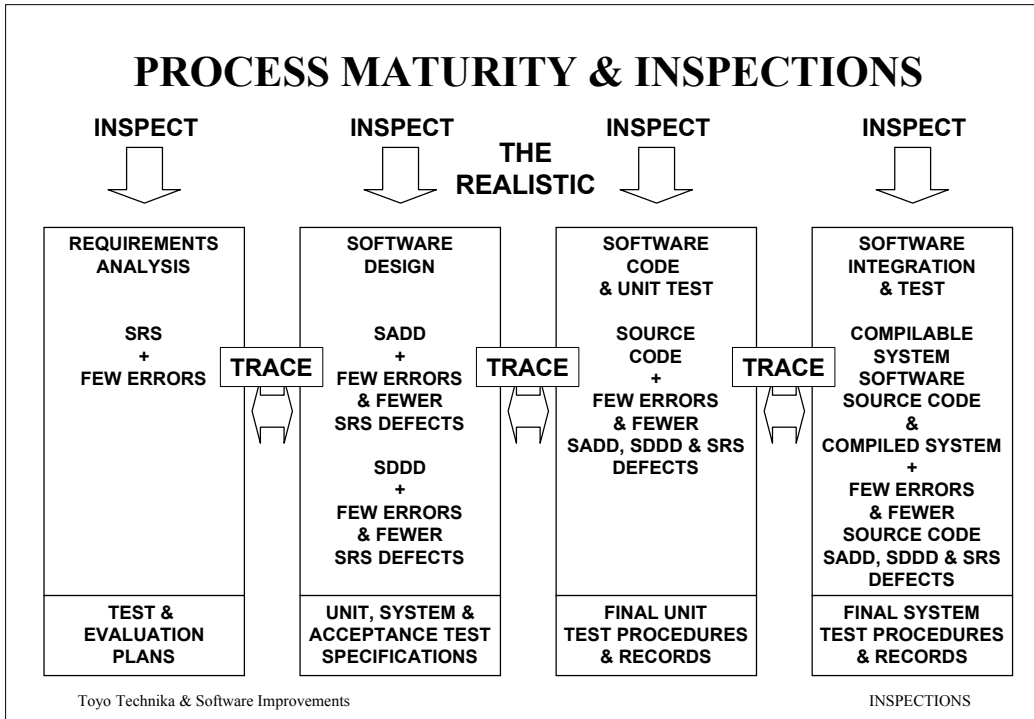
OBJECTIVES

- Find *defects* as *early* as possible in the software development process.
- Reach *agreement* on *extent of rework* needed to redress defects.
- Verify that *completed rework* meets predefined *criteria*.
- Provide *data* on product *quality and process effectiveness*.
- Increase *technical knowledge* of software teams.
- Improve *effectiveness* of software *validation*.
- Raise *standards* of software engineering.
- Discover *properties* of software product and process that can be used to *predict level of product quality*.

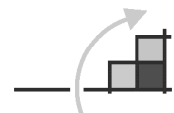
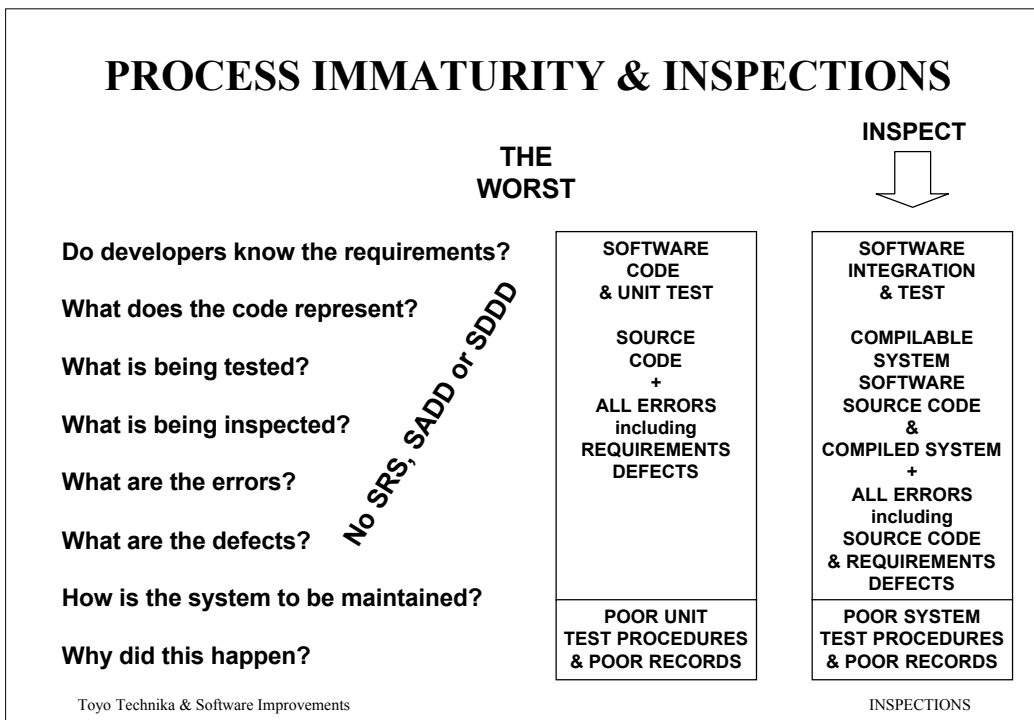




PROCESS MATURITY & INSPECTIONS



PROCESS IMMATURITY & INSPECTIONS



INSPECTION REALISM (Fagan, Gilb)

- **60% of defects exist before coding commences - they are introduced easily but removed with difficulty.**
- **Inspection is about 80% effective in removing existing defects.**
- **Testing is 50% - 55% (max.) effective in identifying & removing defects for a single test process.**
- **Inspection is but one method to control quality - it needs to be used in conjunction with other static and also dynamic techniques.**

Toyo Technika & Software Improvements

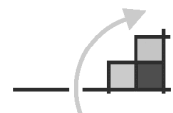
INSPECTIONS

BENEFITS OF INSPECTIONS (Gilb)

- **Earlier delivery (25% - 35%).**
- **Reduced development costs (25% - 35%).**
- **Reduced maintenance costs (10 - 30 times less).**
- **Much improved quality (10 - 100 times fewer defects).**
- **Increased productivity (obvious from figures above).**
- **Reduced testing & machine time (85% improvement).**

Toyo Technika & Software Improvements

INSPECTIONS



INSPECTION CHECKLISTS

Example Source Code Inspection Checklist (see Rakitin)

- * **Has design been implemented completely & correctly?**
- * **Are there missing or extraneous functions?**
- * ***Is each loop executed the correct number of times?***
- * ***Will each loop terminate?***
- * ***Are all possible loop exits correct?***
- * ***Will the program terminate without the need to abort?***
- * ***Are all CASE statements evaluated as expected?***
- * ***Is there any unreachable code?***
- * ***Are there any off-by-one iteration errors?***
- * ***Are there any dangling else clauses?***
- * **Is pointer addressing used appropriately & correctly?**
- * **Are pointer parameters used as values & vice versa?**
- * **Are boundary conditions considered (low level languages)?**

Toyo Technika & Software Improvements

INSPECTIONS

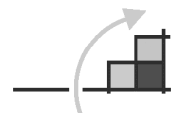
INSPECTION CHECKLISTS

Example Source Code Inspection Checklist cont'd

- * ***Does number of input parameters match the number of arguments?***
- * ***Do parameter & argument attributes match?***
- * ***Do the units of parameters & arguments match?***
- * **Are any input arguments altered?**
- * **Are global variable definitions consistent across modules?**
- * **Are any constants passed as arguments?**
- * ***Are any functions called from which there is no return?***
- * **Are returned VOID values used?**
- * **Are all interfaces correctly used as defined in the SADD/SDDD?**
- * **..... And more!!**

Toyo Technika & Software Improvements

INSPECTIONS



JAVA CODE INSPECTIONS (Consortium)

- We did not know the requirements!
- We did not know the design!
- We did not know test results!
- Traditional inspection - inappropriate!
- Not difficult to find simple coding errors!
- Difficult to find requirements defects!
- WHAT TO DO?

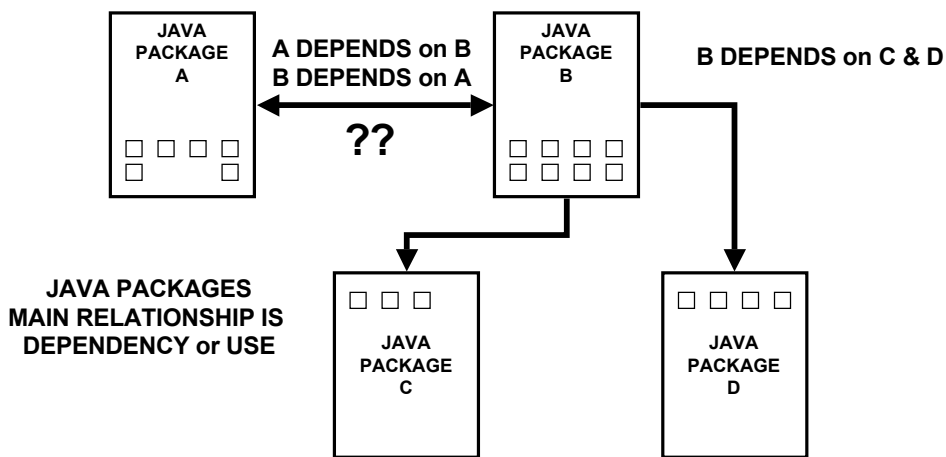
*No SRS, SADD or SDDD,
Test descriptions, Test results*

INSPECT



- DELIVERABLE SOFTWARE
- COMPILABLE SYSTEM SOFTWARE SOURCE CODE
- + ALL ERRORS & DEFECTS

ANATOMY OF Object-Oriented S/W SYSTEMS

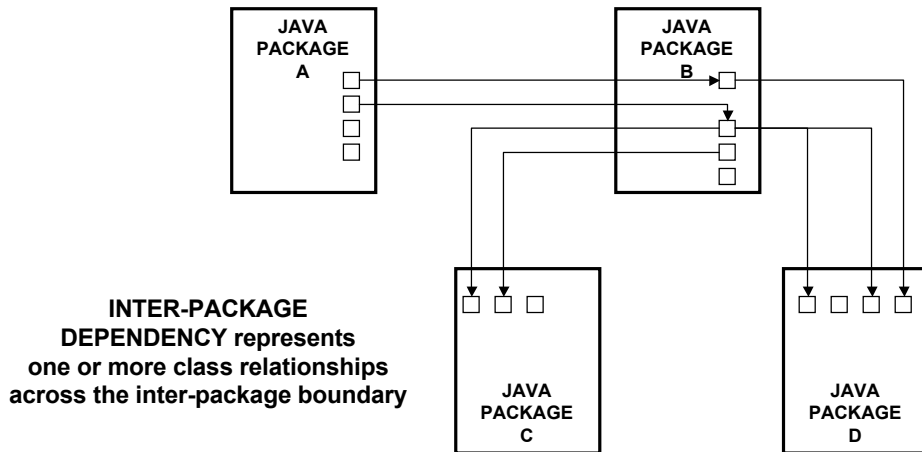


JAVA PACKAGES
MAIN RELATIONSHIP IS
DEPENDENCY or USE

JAVA PACKAGES typically contain many inter-related CLASSES



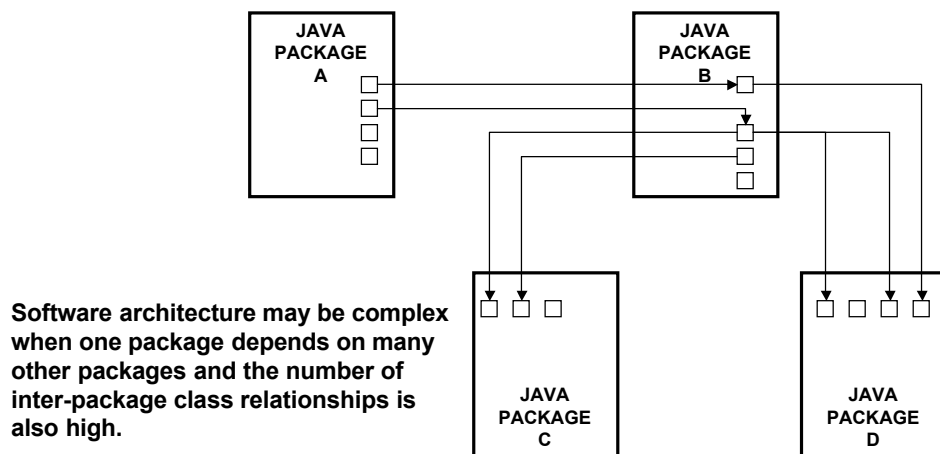
ANATOMY OF Object-Oriented S/W SYSTEMS



Toyo Technika & Software Improvements

INSPECTIONS

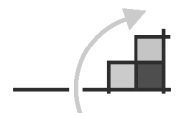
ANATOMY OF Object-Oriented S/W SYSTEMS



HIGH BANDWIDTH IS AN INDICATOR OF POOR STRUCTURE (from work done in Ada)

Toyo Technika & Software Improvements

INSPECTIONS



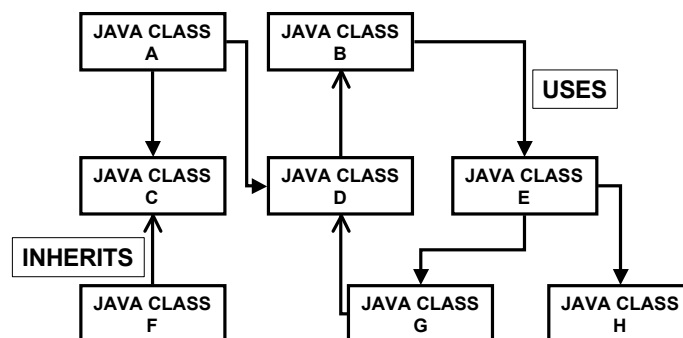
ANATOMY OF Object-Oriented S/W SYSTEMS PACKAGES with HIGH BANDWIDTH

- * are (usually) internally complex
- * are typically difficult to change/maintain
- * typically do not match the design
- * are difficult to test adequately
- * contain many undiscovered errors/defects
- * often express a problem with the design

Toyo Technika & Software Improvements

INSPECTIONS

ANATOMY OF Object-Oriented S/W SYSTEMS

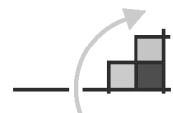


Intra-package inter-class relationships in Java systems represent a more detailed view of a package.

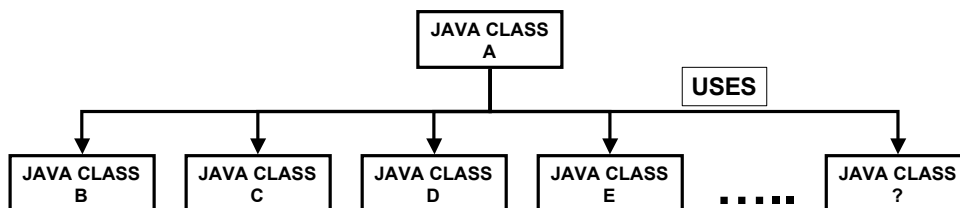
These inter-class relationships are also dependencies.

Toyo Technika & Software Improvements

INSPECTIONS



ANATOMY OF Object-Oriented S/W SYSTEMS



When a Java class depends on too many other classes then it is a clear indicator of class structural complexity.

Dependency also includes inheritance - but the single inheritance properties of Java reduces the possibility of complex inheritance hierarchies.

Toyo Technika & Software Improvements

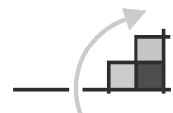
INSPECTIONS

ANATOMY OF Object-Oriented S/W SYSTEMS CLASSES with MANY RELATIONSHIPS

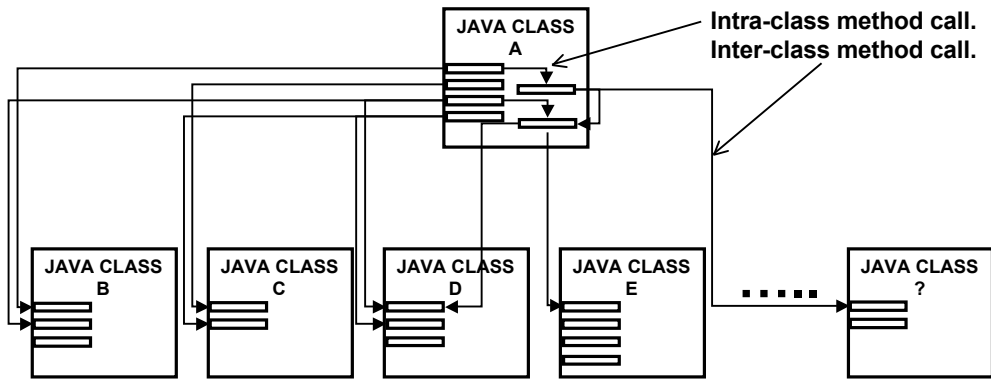
- * **are (usually) internally complex**
- * **can be difficult to change/maintain**
- * **typically do not match the design**
- * **are (usually) difficult to test adequately**
- * **can contain many undiscovered errors/defects**
- * **often express a problem with the detailed design**

Toyo Technika & Software Improvements

INSPECTIONS



ANATOMY OF Object-Oriented S/W SYSTEMS



Inter-class relationships in Java systems are exemplified as method calls.

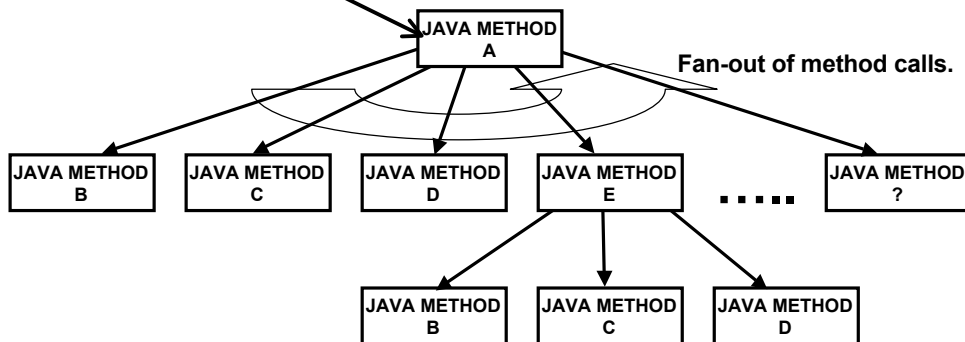
Intra-class method calls are also possible.

Toyo Technika & Software Improvements

INSPECTIONS

ANATOMY OF Object-Oriented S/W SYSTEMS

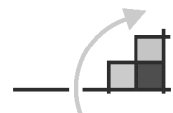
Internal procedure of method may be complex.



When a Java class-method depends on too many other methods then it is a clear indicator of class-method procedural complexity.

Toyo Technika & Software Improvements

INSPECTIONS



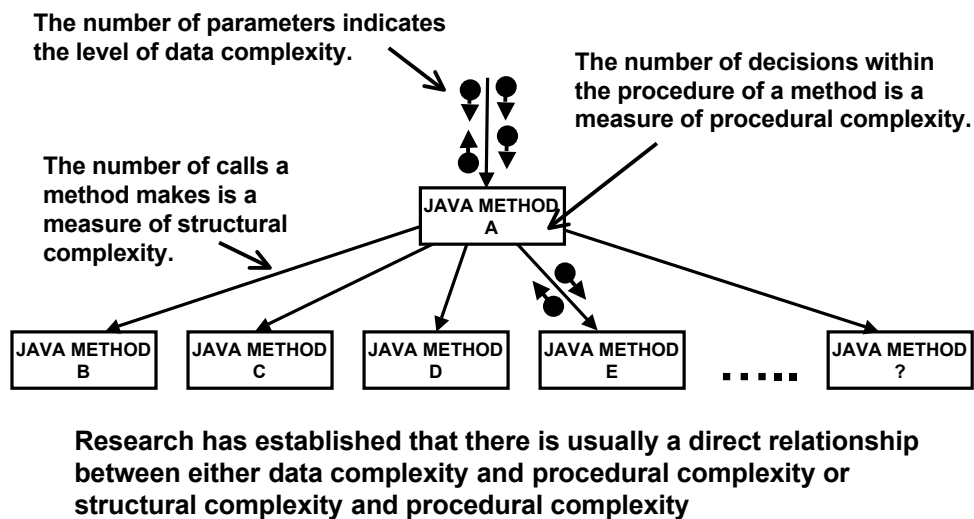
ANATOMY OF Object-Oriented S/W SYSTEMS METHODS with HIGH FANOUT

- * are (usually) internally complex
- * can be difficult to change/maintain
- * typically do not match the design well
- * are (usually) difficult to test adequately
- * can contain many undiscovered errors/defects
- * usually indicate a problem with the detailed design

Toyo Technika & Software Improvements

INSPECTIONS

ANATOMY OF Object-Oriented S/W SYSTEMS



Toyo Technika & Software Improvements

INSPECTIONS



THE NATURE OF JAVA SYSTEMS

Is there a relationship between number of DEFECTS and

- * data complexity?
- * procedural complexity?
- * structural complexity?

The hand inspection of Java code systems done by Software Improvements was aimed at determining such relationships.

Toyo Technika & Software Improvements

INSPECTIONS

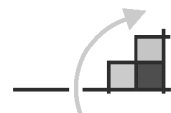
PASS RULES

#	Rule Title	Description	Impacts
1	Missing constructor	Class does not contain any constructor methods.	Maintainability Functionality Portability Reliability Usability
2	Poor Re-use of Inherited Code	Class B extends class A, but does not call any constructor within A.	Maintainability Functionality
3	Non-private Instance Variables	Class A contains instance variables that are not declared "private".	Maintainability Reliability Usability
4	Unnecessary Class Importation	Class A imports class B, but does not use class B.	Maintainability Usability
5	Wasted Wrapper	Class A wraps a primitive type in an object, but does not do anything	Efficiency
6	Boolean Assignment in Test Condition	Code uses "=" instead of "==".	
7	Exit From non-main Method	A method other than main() calls the exit() method.	

Program Analysis and Style Systems from SQI web-site (under Project Tools)

Toyo Technika & Software Improvements

INSPECTIONS



PASS Rules

#	Rule Title	Description	Impacts
8	Embedded Assignment	Assignment occurs within a boolean expression.	
9	Compulsory "default" case for "switch" statements	All "switch" statements must have a "default" case.	
10	Inconsistent "switch" handling	There must be either: - a "return" from each "case", or - a "return" only at the bottom of the "switch" code	Maintainability
11	Variable Hiding (between classes)	Class B extends class A, and both define a class level variable with the same name.	Efficiency Readability Maintainability
12	Variable Hiding (one class)	A method defines a variable with the same name as a class level variable.	Efficiency Readability Maintainability
13	Case-distinguished same named methods	Two methods use the same name but different capitalisation.	Readability Maintainability
14	Case-distinguished same named variables	Two variables use the same name but different capitalisation.	Readability Maintainability

Program Analysis and Style Systems from SQI web-site (under Project Tools)

Toyo Technika & Software Improvements

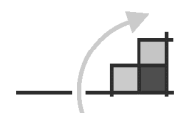
INSPECTIONS

HEURISTICS

#	Heuristic Title	Description	Impacts
1	Large number of "public" variables	A class has many variables declared as "public".	Reliability Maintainability
2	Absence of "get" and "set" methods for "private" variables	A class does not provide "get" and "set" access functions for variables declared as "private".	Maintainability
3	Poor code re-use or unnecessary code replication	A class, or classes, contain code that is repeated unnecessarily.	Maintainability Understandability
4	Absence of adequate or meaningful comments	A class does not contain sufficient comments to explain its purpose and any relevant implementation decisions.	Maintainability Readability Understandability
5	Use of the "goto" statement	A class uses the "goto" statement.	Readability Understandability Maintainability
6	Poor code layout	Improper or inconsistent indentation or insufficient blank lines to space the code.	Readability Maintainability
7	Hard-coded constants in multiple locations	The code contains constants that are hard-coded and are used in more than one location.	Maintainability
8	Other anomalies	Reserved for any poor coding habits observed which do not fit into the above categories.	

Toyo Technika & Software Improvements

INSPECTIONS



OUR INSPECTIONS

Data gathered during high-level inspections:

- Package name
- Class name
- File name
- Libraries imported
- Super classes extended
- Interfaces implemented
- PASS Rule & Heuristic violations
- File length (lines)
- Lines of code (non-comment, non-blank)
- Total lines of comments
- Actual lines of comments

Data gathered during detailed inspections:

- Package name
- Class name
- File name
- Libraries imported
- Super classes extended
- Interfaces implemented
- PASS Rule & Heuristic violations
- File length (lines)
- Lines of code (non-comment, non-blank)
- Total lines of comments
- Actual lines of comments
- Number of variables (including modifiers)
- Methods defined
- Method length (lines)
- Method lines of code (non-comment, non-blank)
- Method comments
- Method actual comments
- Number of parameters to method
- Number of decisions made in method
- Number of variables in method
- Methods called by method

Toyo Technika & Software Improvements

INSPECTIONS

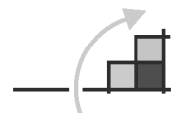
HIGH-LEVEL INSPECTION SAMPLE DATA

Table of statistics gathered during high-level inspections:

File Name	Lines	Source Code (non comment, non blank)	Total Comments	Actual Comments
CylcommonDesignGroupModel.java	1205	906	192	94
ZaiGenshiReferConditions.java	89	17	57	21
UkiMonyOutInputController.java	191	115	50	23
UKpBuyTotalView.java	344	184	136	76
SyuOuterShipScheduleView.java	153	119	12	5
StpPrCsLossView.java	369	201	132	75
SeiListBagData.java	119	95	27	19
NyuOuterInputController.java	239	158	71	22
MttBillDetailValues.java	61	32	22	12
MasCalendarData.java	98	63	18	8
LgnLoginConstants.java	84	16	55	21
JyuToriMenuView.java	273	138	112	56
JisActInNukiModel.java	199	92	85	41
InkNewInkNyuShukkoController.java	362	241	101	51
HinMenuController.java	204	136	48	22
HatGensInputSupplierGrid.java	66	60	15	15
MasGeneral.java	31	5	23	11
HTMLFilterReader.java	100	73	18	8
ErrorManager.java	289	168	101	50
TOTALS	4476	2819	1275	630
AVERAGES	235.6	148.4	67.1	33.2

Toyo Technika & Software Improvements

INSPECTIONS



HIGH-LEVEL INSPECTION SAMPLE DATA

Table of PASS Rule violations gathered during high-level inspections:

File Name	PASS Rules													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
CylcommonDesignGroupModel.java	-	-	X	-	-	-	-	-	-	-	-	-	-	-
ZaiGenshiReferConditions.java	-	-	X	-	-	-	-	-	-	-	-	-	-	-
UkiMonyOutInputController.java	-	X	X	-	-	-	-	-	-	-	-	-	-	-
UKpBuyTotalView.java	-	-	X	-	-	-	-	-	-	-	-	-	-	-
SyuOuterShipScheduleView.java	-	-	X	-	-	-	-	-	-	-	-	-	-	-
StpPrclsLossView.java	-	-	X	-	-	-	-	-	-	-	-	-	-	-
SeilListBagData.java	-	-	X	-	-	-	-	-	-	-	-	-	-	-
NyuOuterInputController.java	-	X	X	-	-	-	-	-	-	-	-	-	-	-
MttBillDetailValues.java	-	-	X	-	-	-	-	-	-	-	-	-	-	-
MasCalendarData.java	-	-	X	-	-	-	-	-	-	-	-	-	-	-
LgnLoginConstants.java	-	-	X	-	-	-	-	-	-	-	-	-	-	-
JyuToriiMenuView.java	-	-	X	-	-	-	-	-	-	-	-	-	-	-
JisActInNukiModel.java	-	-	X	-	-	-	-	-	-	-	-	-	-	-
InkNewInkNyuShukkoController.java	-	-	-	-	-	-	-	-	-	-	-	-	-	-
HinMenuController.java	-	X	X	-	-	-	-	-	-	-	-	-	-	-
HatGensInputSupplierGrid.java	-	-	-	-	-	-	-	-	-	X	-	-	-	-
CharTypeRestrictor.java	-	-	X	-	-	-	-	-	-	X	-	-	-	-
LWComponentButton.java	-	X	X	-	-	-	-	X	-	-	-	X	-	-
Local2Unicode.java	X	-	-	-	-	-	-	-	-	-	-	-	-	-
Selecion.java	-	X	X	-	-	-	-	-	-	-	-	X	-	-
ByteToCharHex.java	X	-	-	-	-	-	-	-	-	-	-	-	-	-
TOTAL	2	5	17					1		1		2		
Estimated number of files that will violate this PASS Rule*	83	291	913					41	41	83		83		

* Figures based on more files that shown here.

Toyo Technika & Software Improvements

INSPECTIONS

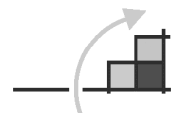
SAMPLE DETAILED INSPECTION DATA

Example project statistics derived from detailed inspections:

Total number of classes examined	156
Total lines	22901
Total source lines of code (non comment non blank)	18364
Total comments	1880
Total actual comments (excluding blank comments, html only comments and commented out code)	1129
Total number of methods (signature present)	1453
Average number of source lines per class	117.7
Average number of methods per class	9.3
Average number of source lines per method	10.9

Toyo Technika & Software Improvements

INSPECTIONS



SAMPLE DETAILED INSPECTION DATA

Example of detailed class data from project database:

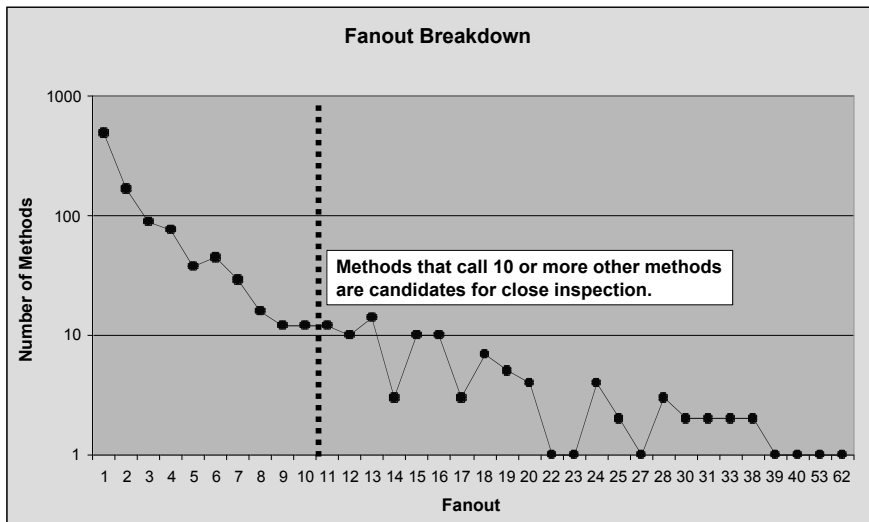
Package Name	Class Name	Length	Source Lines of Code	Comments	Actual Comments	Variables			Methods		
						Public	Protect	Private	Public	Protect	Private
jp.co.hinet.rad.dbtest	Column	162	90	51	17	0	0	1	17	0	1
	DataSet	63	51	0	0	0	2	2	9	0	0
	DeleteStatement	55	42	2	0	0	0	0	4	0	0
	IDObject	41	21	6	0	0	3	0	5	0	0
	IDObjectRef	43	28	6	0	0	3	0	5	0	0
	IMGregorianCalendar	90	76	1	0	0	0	1	9	0	1
	IMIDObject	37	29	0	0	0	0	1	6	0	0
	IMIDObjectRef	37	29	0	0	0	0	1	6	0	0
	IMInt	47	36	0	0	0	0	1	9	0	0
	IMResolver	21	14	0	0	0	1	0	5	0	0
	IMString	39	30	0	0	0	0	1	7	0	0
	InsertStatement	68	0	1	0	0	0	3	5	0	0
	PersistenceException	12	9	0	0	0	0	0	2	0	0
	PSet	162	124	16	5	0	7	0	16	0	0
	SelectStatement	84	60	1	0	0	0	5	6	0	0
	Transaction	66	52	2	2	0	0	3	6	0	0
	UpdateStatement	89	67	1	0	0	0	4	6	0	0
TOTAL		1116	758	87	24	0	16	23	123	0	2

Toyo Technika & Software Improvements

INSPECTIONS

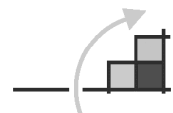
SAMPLE DETAILED INSPECTION DATA

Example fan-out data derived from detailed project database:



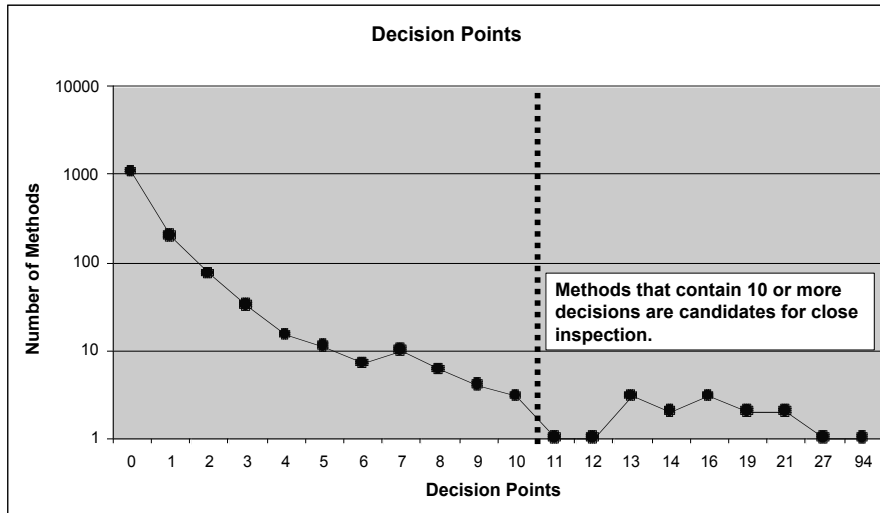
Toyo Technika & Software Improvements

INSPECTIONS



SAMPLE DETAILED INSPECTION DATA

Decision point data derived from detailed project database:

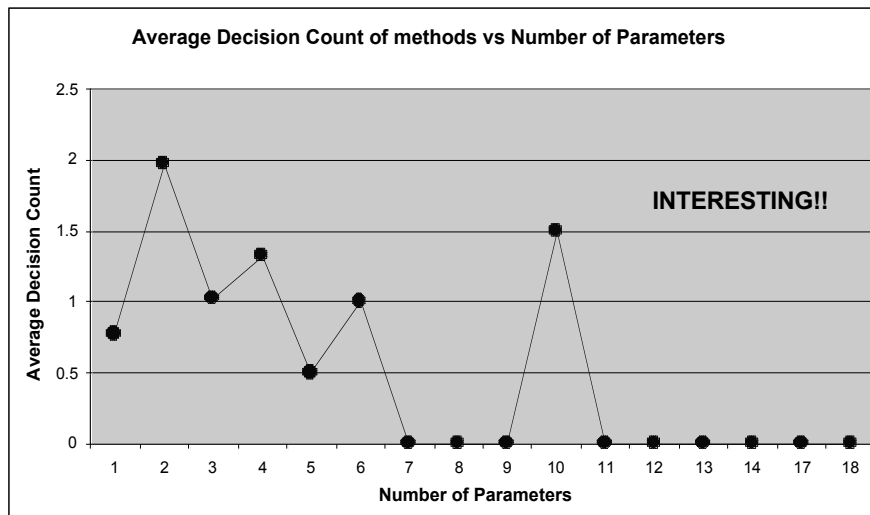


Toyo Technika & Software Improvements

INSPECTIONS

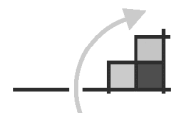
SAMPLE DETAILED INSPECTION DATA

Decision count data derived from detailed project database:



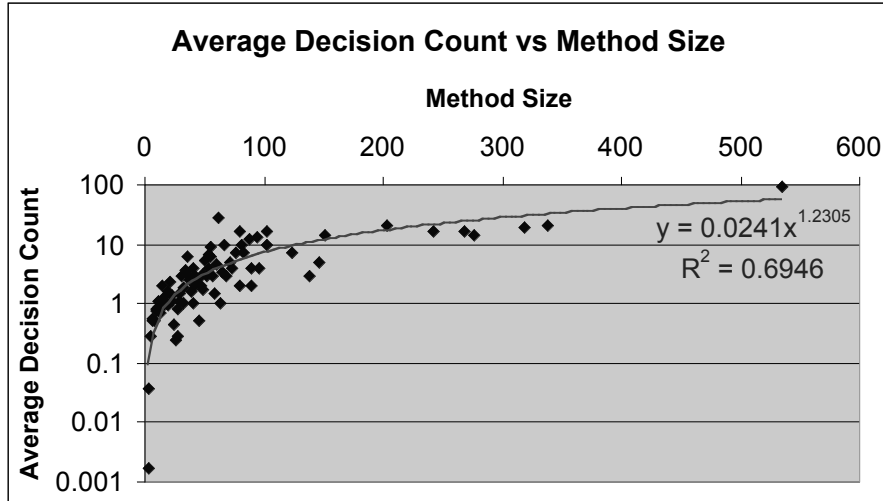
Toyo Technika & Software Improvements

INSPECTIONS



SAMPLE DETAILED INSPECTION DATA

Decision count vs Method Size data derived from detailed project database:

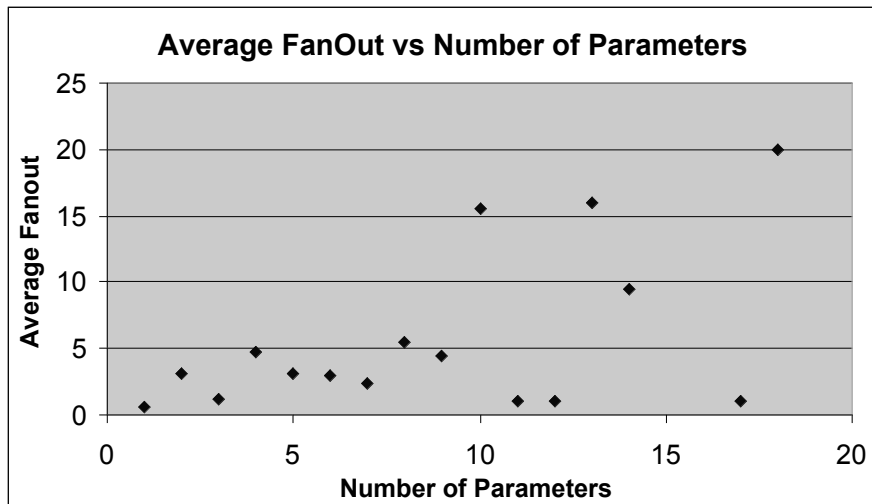


Toyo Technika & Software Improvements

INSPECTIONS

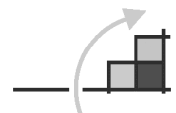
SAMPLE DETAILED INSPECTION DATA

Fanout vs Parameters derived from detailed project database:



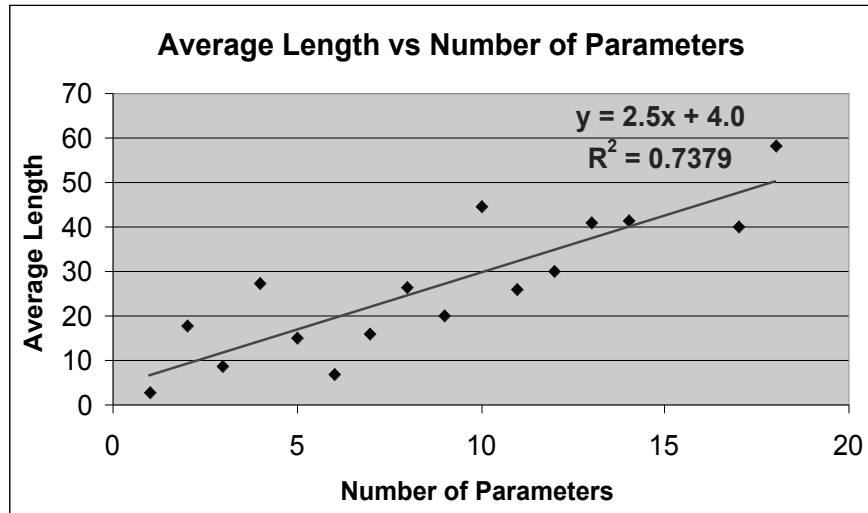
Toyo Technika & Software Improvements

INSPECTIONS



SAMPLE DETAILED INSPECTION DATA

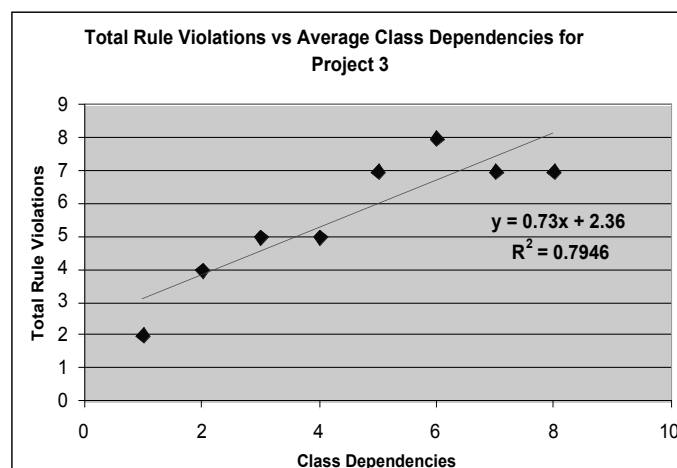
Length vs Parameters derived from detailed project database:



Toyo Technika & Software Improvements

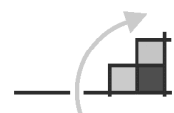
INSPECTIONS

TOTAL RULE VIOLATIONS vs CLASS DEPENDENCIES



Toyo Technika & Software Improvements

INSPECTIONS



Number of Parameters vs Number of Rule Violations

Params	Rule Violations	Params	Rule Violations
2.4	1	2.6	1
2.3	2	1.8	2
2.7	3	1.5	3
3.6	4	1.3	4
5.9	5		
2	2		
1.3	1		
3.0	2		
2.3	3		
4.2	4		

Toyo Technika & Software Improvements

INSPECTIONS

Fanout vs Number of Rule Violations

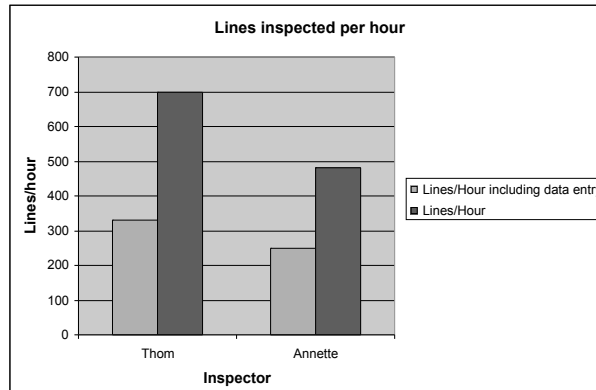
Fanout	Rule Violations	Fanout	Rule Violations
23	1	23	1
21	2	67	2
32	3	93	3
41	4	91	4
53	5		
30	2		
24	4		
21	1		
22	2		
26	3		
29	4		

Toyo Technika & Software Improvements

INSPECTIONS



INSPECTION RATE DATA

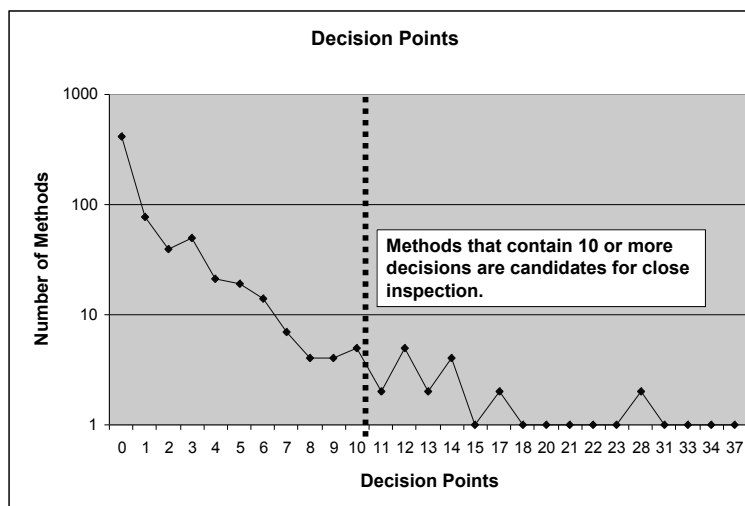


Toyo Technika & Software Improvements

INSPECTIONS

SAMPLE DETAILED INSPECTION DATA

Decision point data derived from detailed project database P0009:



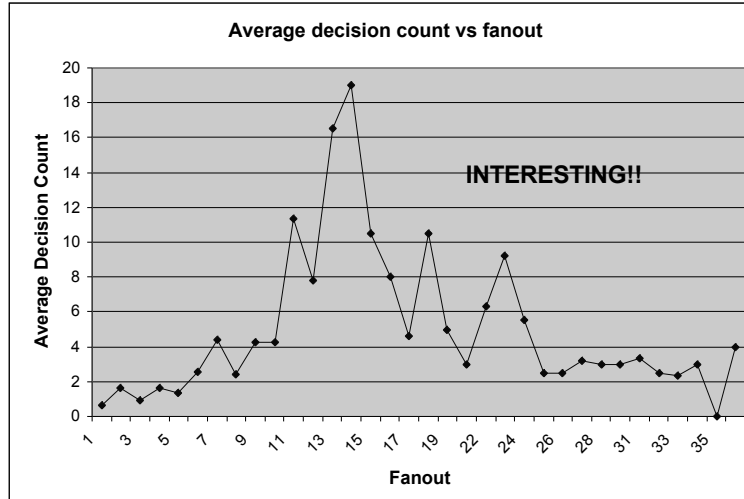
Toyo Technika & Software Improvements

INSPECTIONS



SAMPLE DETAILED INSPECTION DATA

Decision count data derived from detailed project database P0009:

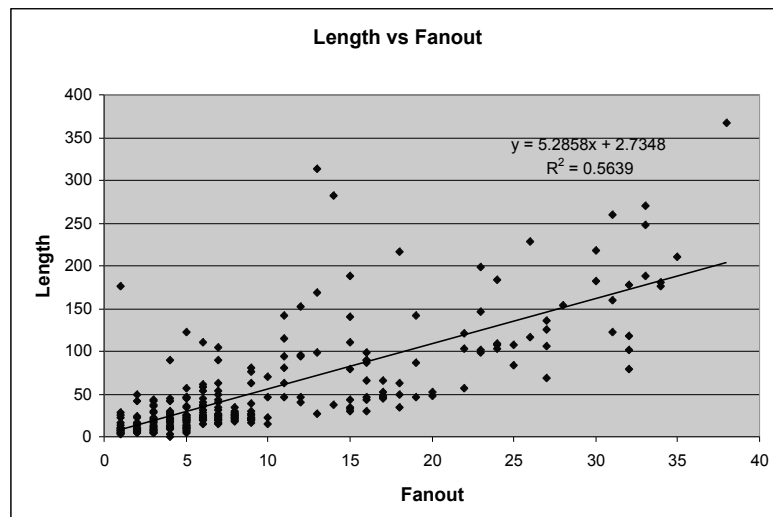


Toyo Technika & Software Improvements

INSPECTIONS

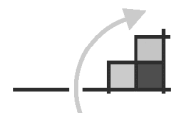
SAMPLE DETAILED INSPECTION DATA

Length vs Fanout derived from detailed project database P0009:



Toyo Technika & Software Improvements

INSPECTIONS



REFERENCES

- **Software Verification and Validation - A Practitioner's Guide:** Steven Rakitin (Artech House 1997).
- **Static Inspection - Tapping the Wheels of Software:** Les Hatton (IEEE Software 1995).
- **Principles of Software Engineering Management:** Tom Gilb (Adison-Wesley 1988).
- **Handbook of Walkthroughs, Inspections and Technical Reviews:** Daniel Freedman and Gerald Weinberg (Dorset House 1990).
- **Measuring Software Design Quality:** D.N.Card and R.L.Glass (Prentice-Hall 1990)
- **Software Engineering:** Ian Sommerville (Adison-Wesley 1989).

