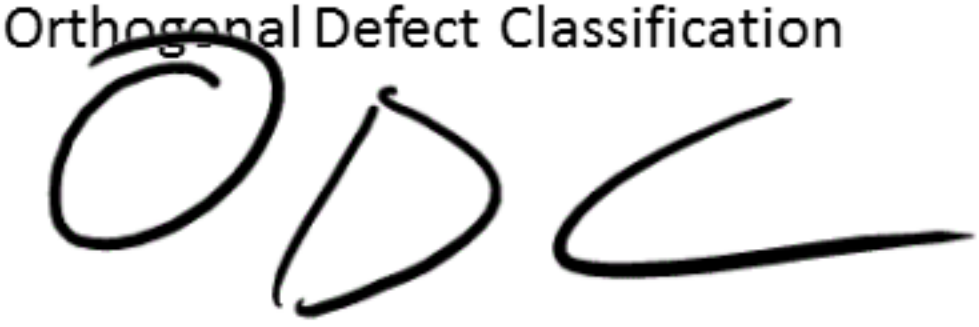
SE4831 Software Quality Assurance
Orthogonal Defect Classification

- Objectives
  - Explain the concept of Orthogonal Defect Classification
  - Explain the relationship between mistakes, faults, and failures
  - Classify a defect's open section based on a verbal description of the defect
  - Interpret defect curves based on an analysis of defects using Orthogonal Defect Classification



List 3 items in software development that impact quality.

- testing, requirements, software design & process
- "RequirementsTeam Test PlanDevelopment Cycle"
- "Design Reviews Tests"
- Time constraints, Budget Constraints, Coding Practices
- "1. Testing2. Reqs Management 3. Peer Review"
- "Defects Experience Planning"
- "Testing plan Time Experience"
- "Design Time Background"
- "Time constraints Code reviews Bad coding standards"
- "Testing Knowledge of the subject matter Planning"
- "Development practicesTestingPeer Review"
- "ExperienceFamiliarity with languageCompleteness of RequirementsTime spent in designCode reviews (code review checklists)etc..."
- "WalkthroughsInspectionsReviews"
- "DefectsArchitectureOptimization"
- "Team communicationCommunication with stakeholdersConstant requirements changing (""feature creep"")"
- "1. Peer reviews (More eyes = generally better code)2. Having a proper architecture for the system3. Thorough testing"
- "TestingPeer ReviewPersonel Skills"

What did you do on your SDL projects to ensure that the final delivered product was of an appropriate level of quality?

- "group reviews for specific & specialized tasksJUnit & visual testingfunctional and design testing"
- "Meetings with stakeholdersDedicated Sprint to testing and verfication"
- "Make sure it passed automated tests and peer reviews.Develop our processes through reviews and retrospectives.Documentation.Really mostly process improvements."
- Peer reviews
- We did a lot of testing and code review. When applicable we had unit tests written, and before someone closed their task it had to be code reviewed by at least two people.
- Proper testing proceedure, diligent requirements gathering
- Tested code by hand and peer reviewed
- Swarmed PBIs to make sure things got done
- "Formal code reviewsChecked eachother's code constantly"
- "Code reviews before deploymentAttempted to assign tasks to people with experiece in what was neededDiscussed how to solve the problem if necessary"
- We did peer code reviews before submitting a code change to

What did you do on your SDL projects to ensure that the final delivered product was of an appropriate level of quality?

- "Pray it worked...We had some tests surrounding the output of our parser/compiler to ensure it was returning the expected results after we modified it. Due to the nature of our project most everything was visual testing and it changed frequently as the visuals for the program constantly took different forms. We did reviews, but none ever found a bug so those were pretty useless. One major thing we did was have everyone present when we merged the different parts together so we had as many eyes on the code looking for problematic errors as possible."
- Standardized documentation to ensure familiraity, regular code reviews, making sure people don't get sucked into only one aspect/technology so that everybody knows what is going on from a high level throughout all parts of the project.
- "Communication with stakeholder to make sure it met requirements Demonstrations of software Lots of documentation"

For your senior design project (or SDL project if you are not in senior design), list one quality goal for your project.

- Downloading OBJ files from our server should be encrypted according to the industry standards using HTTPS. - Security Analysis Testing
- Scalability - we want our web application to be able to support a small amount of users initially, but eventually grow to support more.
- We have a system for code reviews where, for each sprint, a different person is assigned to reviewing one member's work for that sprint.
- Code and design reviews
- It has to be reliable and have 99.9% uptime
- Maintainability
- Over 60% code coverage of unit tests
- For our senior design project we are looking for realiablility as a quality goal because a big part of our app is regular google map updates. We want the app to rarely crash when the user is using the app, and if it does we want it to have a safe crash.
- No critical defects
- The site must be able to handle multiple concurrent users
- To make sure data is transferred between merchant and

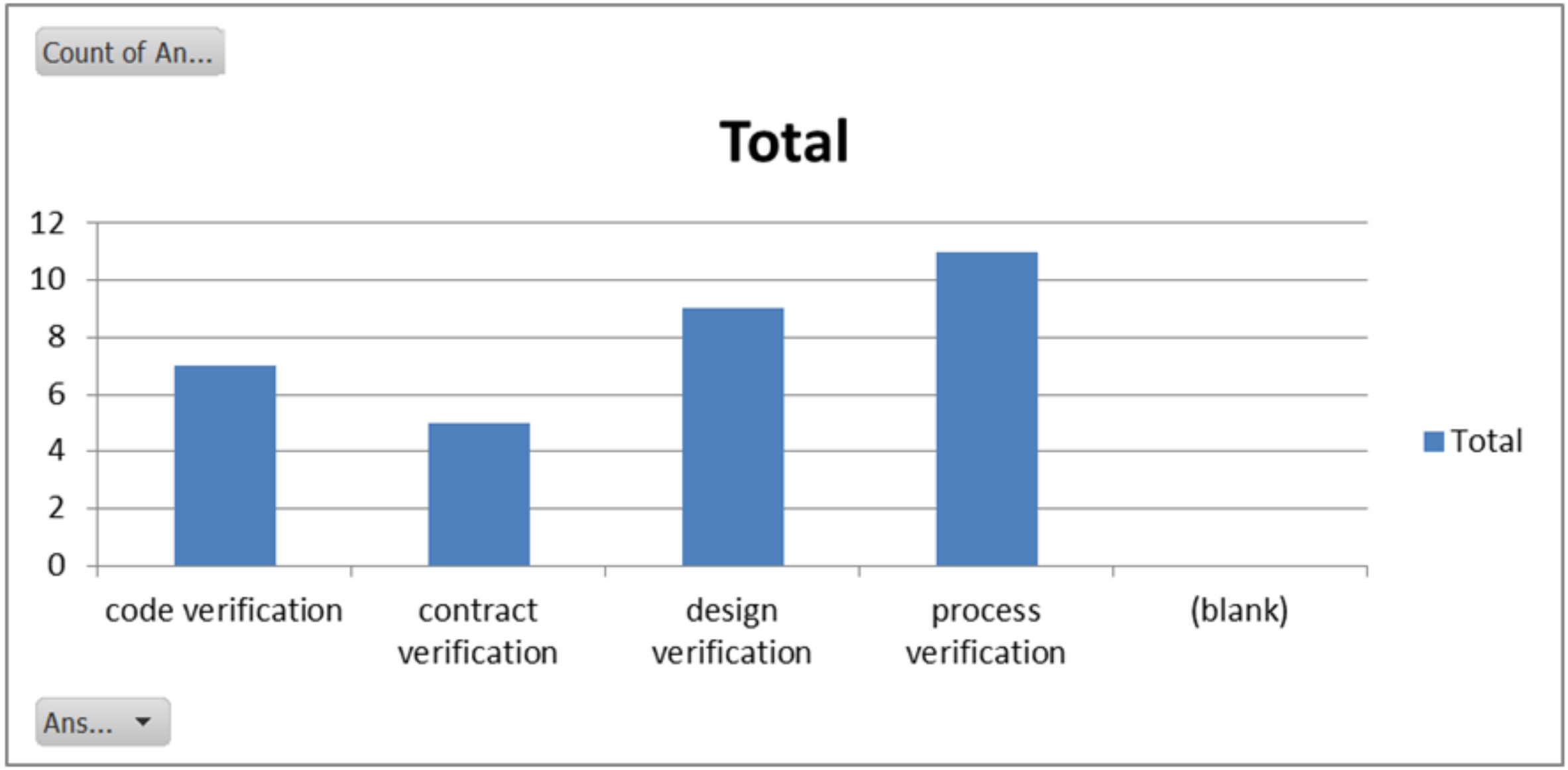
1	XI C	SOFTWARE QUALITY Methods for SQA and V&V

While developing a software product, a development team performs verification of the following:

- Proper sequences of events, inputs, outputs, interfaces, and logic flow
- Allocation of timing and sizing budgets
- Error definition, isolation, and recovery
- Proper treatment of safety, security, and other critical requirements

Question Which of the following types of activity were they performing?

a) Design verification
b) Contract verification
c) Code verification
d) Process verification


Count of An...
Total
12
10
8
6
4
2
0
code verification
contract verification
design verification
process verification
(blank)
Total
Ans...

Answer: a

Explanation

Section 6.4.2.4 of ISO/IEC 12207.0 describes design verification as being defined by the following criteria:

a. The design is correct and consistent with and traceable to requirements.

b. The design implements proper sequence of events, inputs, outputs, interfaces, logic flow, allocation of timing and sizing budgets, and error definition, isolation, and recovery.

c. Selected design can be derived from requirements.

d. The design implements safety, security, and other critical requirements correctly, as shown by suitably rigorous methods.

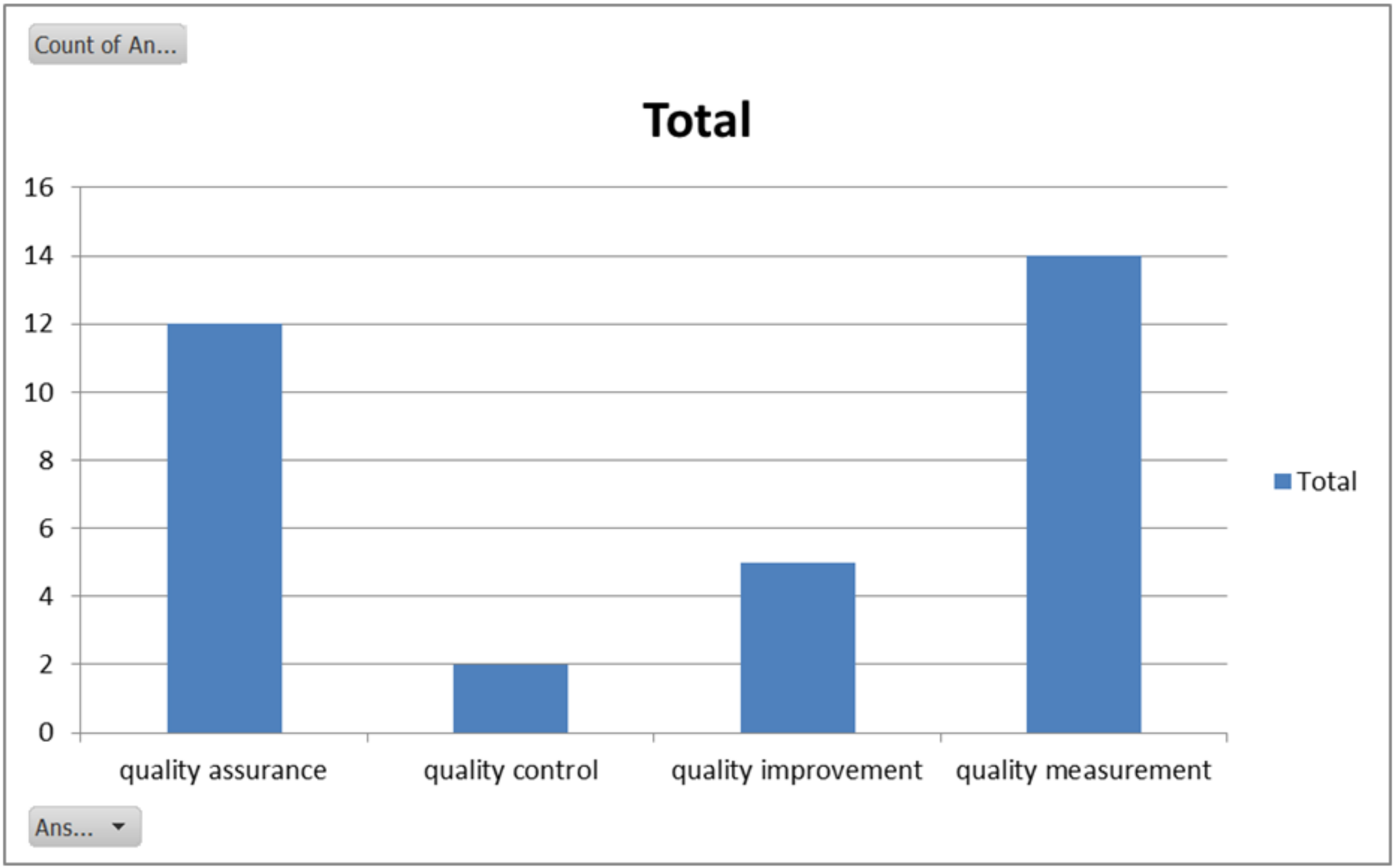
References

1. *Industry Implementation of International Standard ISO/IEC 12207: 1995 Standard for Information Technology — Software Lifecycle Processes*, IEEE Press, 1996.

2	XI A	Software Quality Software quality concepts

Question An engineer has been tasked to evaluate the peer review process being used to support the development of the company's new software product. The task would be considered

a) Quality control
b) Quality assurance
c) Quality measurement
d) Quality improvement


Count of An...
Total
16
14
12
10
8
6
4
2
0
quality assurance
quality control
quality improvement
quality measurement
Total
Ans...

Answer: b

Explanation

The IEEE Standard for Software Engineering Terminology defines quality control as (1) "a set of activities designed to evaluate the quality of developed or manufactured products" and (2) "the process of verifying one's own work or that of a co-worker." This standard also defines quality assurance as (1) "a planned and systematic pattern of all actions necessary to provide adequate confidence that an item or product conforms to established technical requirements", and (2) "a set of activities designed to evaluate the process by which products are developed or manufactured."

To summarize, quality assurance is a set of activities designed to evaluate a process by which products are developed, whereas quality control is defined as a set of activities designed to evaluate the quality of a developed product.

Reference

1. *IEEE Standard 610.12, IEEE Standard for Software Engineering Terminology*, IEEE Press, 1990 (R2002).

4	XI A	SOFTWARE QUALITY Software quality concepts

Question Quality assurance may be applied to:

I. Requirements

II. Design

III. Code

IV. Testing

a) I and II only
b) I, II, and III only
c) I, II, III, and IV
d) IV only

Answer: c

Explanation

Section 6.3 of ISO/IEC 12207.0 describes quality assurance as follows:

"The Quality Assurance Process is a process for providing adequate assurance that the software products and processes in the project life cycle conform to their specified requirements and adhere to their established plans. To be unbiased, quality assurance needs to have organizational freedom and authority from persons directly responsible for developing the software product or executing the process in the project. Quality assurance may be internal or external depending on whether evidence of product or process quality is demonstrated to the management of the supplier or the acquirer. Quality assurance may make use of the results of other supporting processes, such as Verification, Validation, Joint Reviews, Audits, and Problem Resolution."

The phrase "software products and processes in the project life cycle conform to their specified requirements and adhere to their established plans" is an inclusive statement identifying that all aspects of the project lifecycle are candidates for the application of quality assurance.

Defect Modeling

Statistical Defect Models

- abstract
- quantitative
- distant from the programmer
- low cost
- capable of automation
- restricted to a few domains

Root Cause Analysis

- down to earth
- qualitative
- programmer's perspective
- high cost
- human intensive
- wide range of domains

what went wrong

Multi-dimensional classification

A defect is classified across multiple dimensions

	Causal attributes	*Sub-population attributes*	*Effect attributes*
	Defect type	***Source***	***Impact***
	provides feedback on the development process	type of code that is corrected - new, old, reused, vendored, re-fixed, ...	the resultant effect on the customer - capability, usability, ...
	Trigger	**Phase found**	**Severity**
	provides feedback on the verification processes - testing, review, beta test, ...	defined on the development process activities - design, review, test, ...	IBM uses 1-4, where 1 is the highest signifying major outage, while 4 could be an annoyance

Two touchpoints for a defect

- Opening
- Closing

Info we discover when a defect is found

Know what happens when a defect is found, what was done, etc

Sort of use knowledge gained by closing a defect.

Error, Fault and Failure Definition

Source	Model
ANSI / IEEE 729-1983	error → fault → failure
Fenton	error → fault → failure
Shooman	fault → error → failure
IEC 1508	fault → error → failure
Hatton	error → fault or defect or bug → failure

- Error
  - A Human Action that produces an incorrect result
- Fault
  - An incorrect step, process, or data definition in a computer program
  - Injected by a software engineer during development.
  - Static Property of the artifact
- Failure
  - An unexpected departure of the software package from expected operational characteristics.

Section Opening

- Activity:
  - Activity that was being performed at the time the defect was discovered.
- Trigger
  - The environment or condition that had to exist for the defect to surface.
- Impact:
  - the impact which you judge the defect would have had upon the customer if it had escaped to the field.

Opener Section

Opener Section

(These attributes are usually available when the defect is opened.)

Defect Removal Activities	Triggers	Impact
Design Rev Code Inspection Unit test Function Test System Test	1.Design Conformance 2.Logic/ Flow 3.Backward Compatibility 4.Lateral Compatibility 5.Concurrency 6.Internal Document 7.Language Dependency 8.Side Effect 9.Rare Situations 10.Simple Path 11.Complex Path 12.Coverage 13.Variation 14.Sequencing 15.Interaction 16.Workload/Stress 17.Recovery/Exception 18.Startup/Restart 19.Hardware Configuration 20.Software Configuration 21.Blocked Test (previously Normal Mode)	1.Installability 2.Serviceability 3.Standards 4.Integrity/Security 5.Migration 6.Reliability 7.Performance 8.Documentation 9.Requirements 10.Maintenance 11.Usability 12.Accessibility 13.Capability

Example

- When testing a given program, the program crashed. The program crashed because the user entered a 0 in a dialog box. When the program crashed, it gave the error message "Division by 0."
  - How would this defect be classified?

Example Classification

Opener Section

(These attributes are usually available when the defect is opened.)

Defect Removal Activities	Triggers	Impact
Design Rev Code Inspection Unit test Function Test System Test	1. Design Conformance 2. Logic/ Flow 3. Backward Compatibility 4. Lateral Compatibility 5. Concurrency 6. Internal Document 7. Language Dependency 8. Side Effect 9. Rare Situations 10. Simple Path 11. Complex Path 12. Coverage 13. Variation 14. Sequencing 15. Interaction 16. Workload/Stress 17. Recovery/Exception 18. Startup/Restart 19. Hardware Configuration 20. Software Configuration 21. Blocked Test (previously Normal Mode)	1. Installability 2. Serviceability 3. Standards 4. Integrity/Security 5. Migration 6. Reliability 7. Performance 8. Documentation 9. Requirements 10. Maintenance 11. Usability 12. Accessibility 13. Capability

Closer Section

- Closer Section
  - Attributes you can classify when you know how the defect was fixed.
- Target
  - High level identity of the entity that was fixed.
- Defect Type
  - Represents the nature of the actual correction that was made.
- Qualifier:
  - Captures the element of either a nonexistent or wrong or irrelevant implementation.
- Source
  - Identifies the origin of the Target
- Age
  - Identifies the history of the

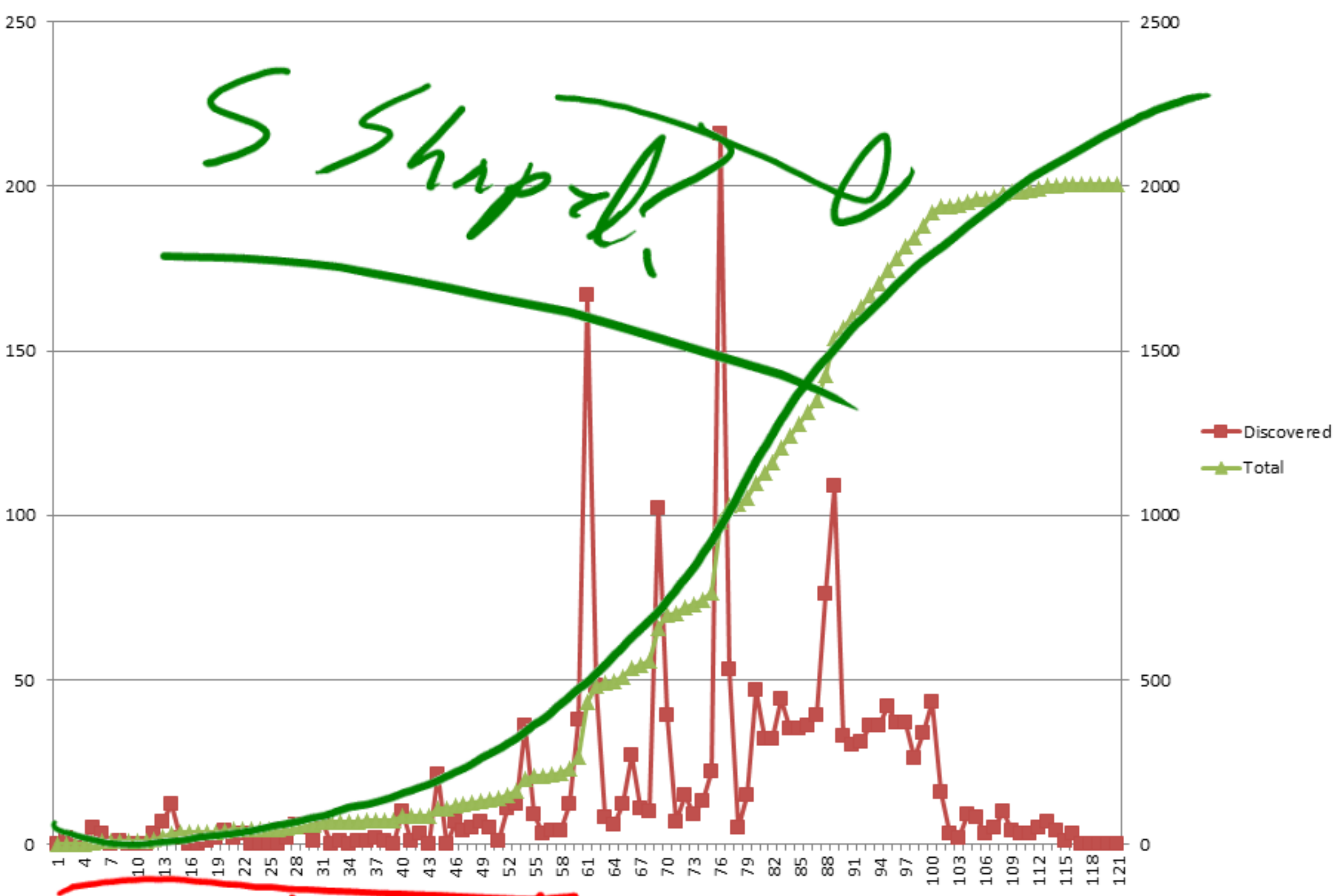
Closer Section

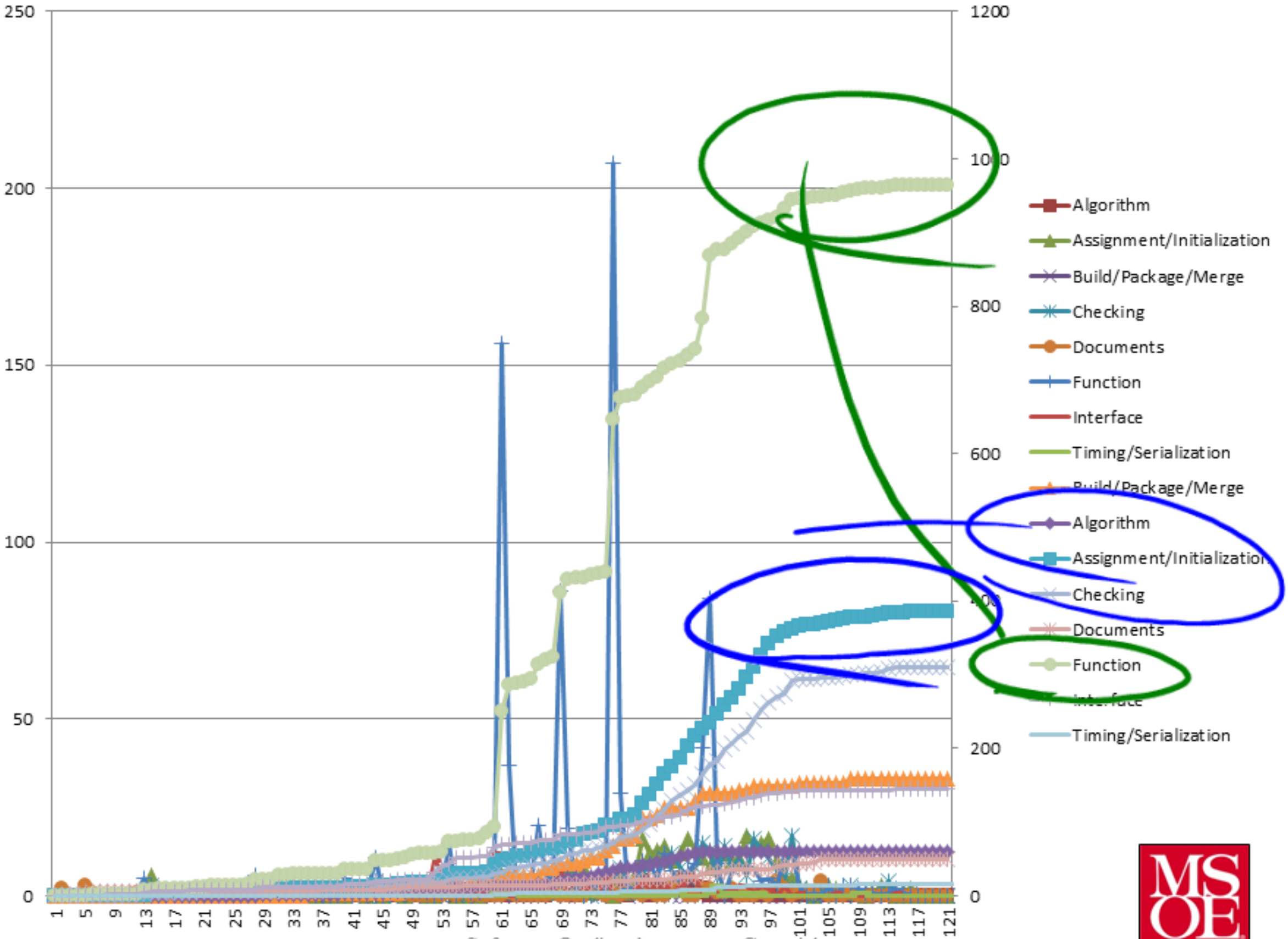
(These attributes are usually available when the defect is fixed.)

Target	Defect Type	Qualifer	Age	Source
Design/Code	1.Assign/Init 2.Checking 3.Alg/Method 4.Func/Class/Object 5.Timing/Serial 6.Interface/O-O Messages 7.Relationship	1.Missing 2.Incorrect 3.Extraneous	1.Base 2.New 3.Rewritten 4.ReFixed	1.Developed In-House 2.Reused FromLibrary 3.Outsourced 4.Ported

IBM Project Analysis

- Project Conducted at IBM
  - 2008 filed defects


250
200
150
100
50
0
2500
2000
1500
1000
500
0
Discovered
Total
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100 103 106 109 112 115 118 121
S Shipped
weeks

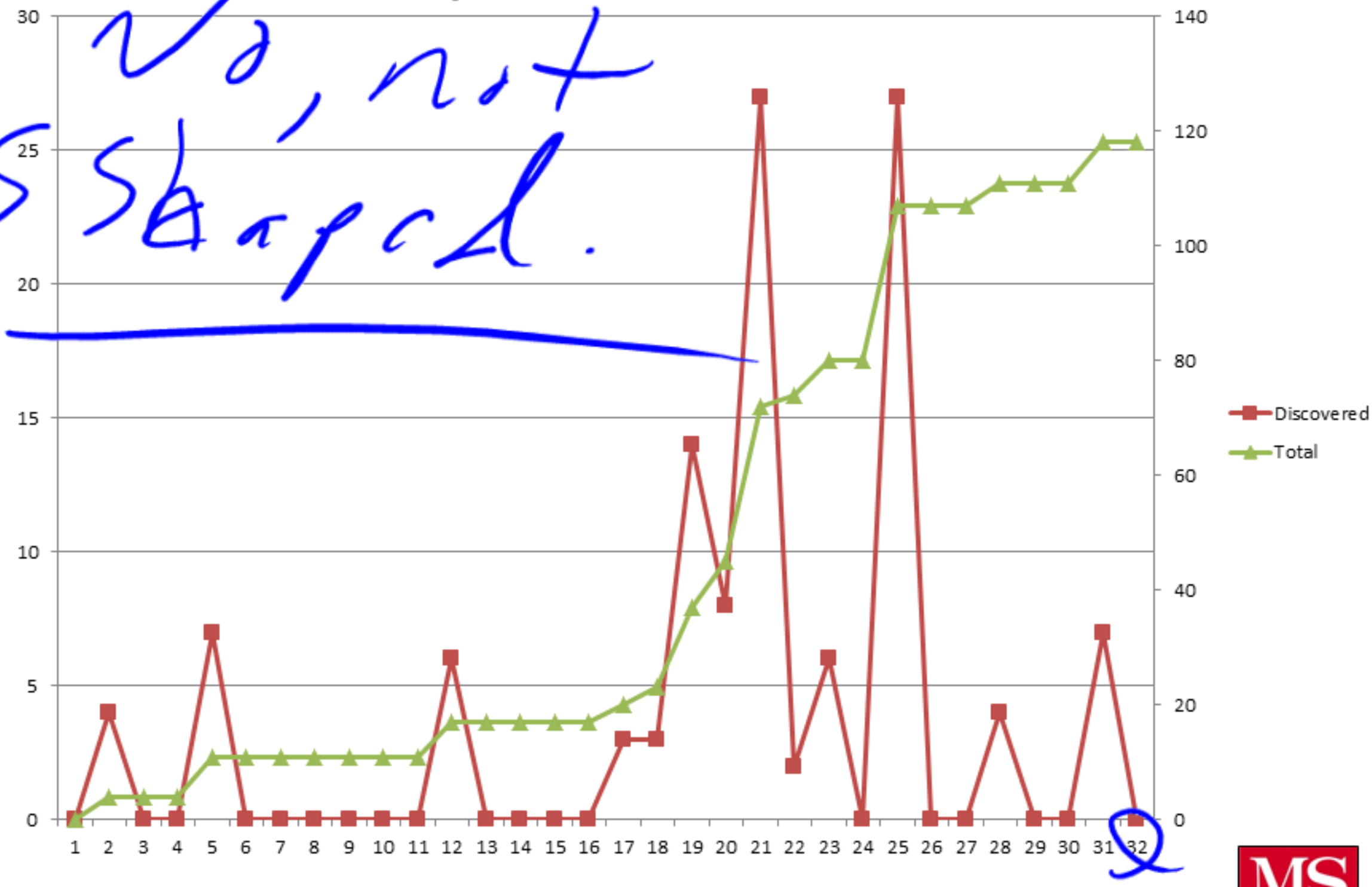

250
200
150
100
50
0
1200
1000
800
600
400
200
0
1
5
9
13
17
21
25
29
33
37
41
45
49
53
57
61
65
69
73
77
81
85
89
93
97
101
105
109
113
117
121
Algorithm
Assignment/Initialization
Build/Package/Merge
Checking
Documents
Function
Interface
Timing/Serialization
Build/Package/Merge
Algorithm
Assignment/Initialization
Checking
Documents
Function
Timing/Serialization

Another Example

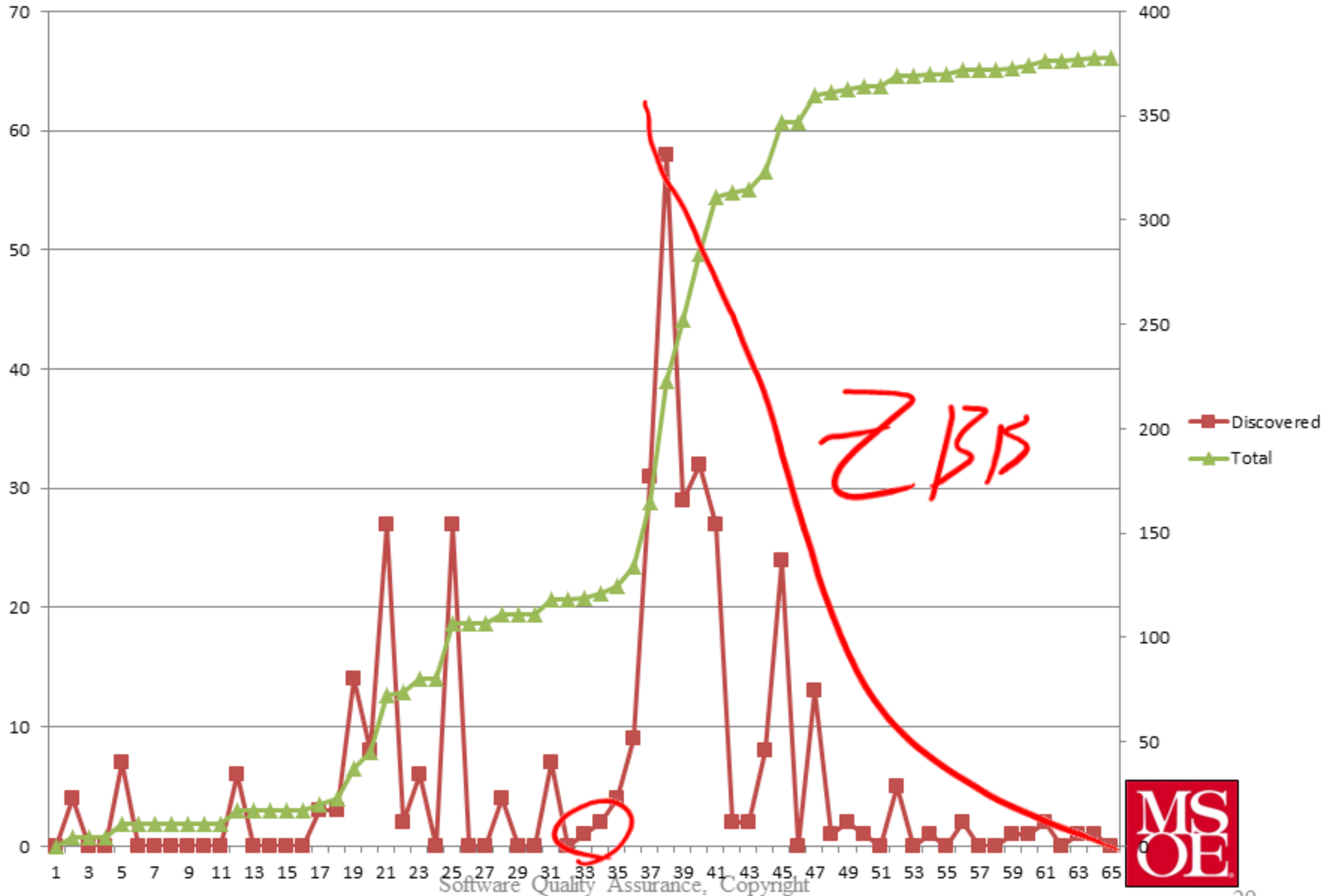
- Software Developed at IBM
  - I won't tell you the total number of defects yet

- Ready for Release?

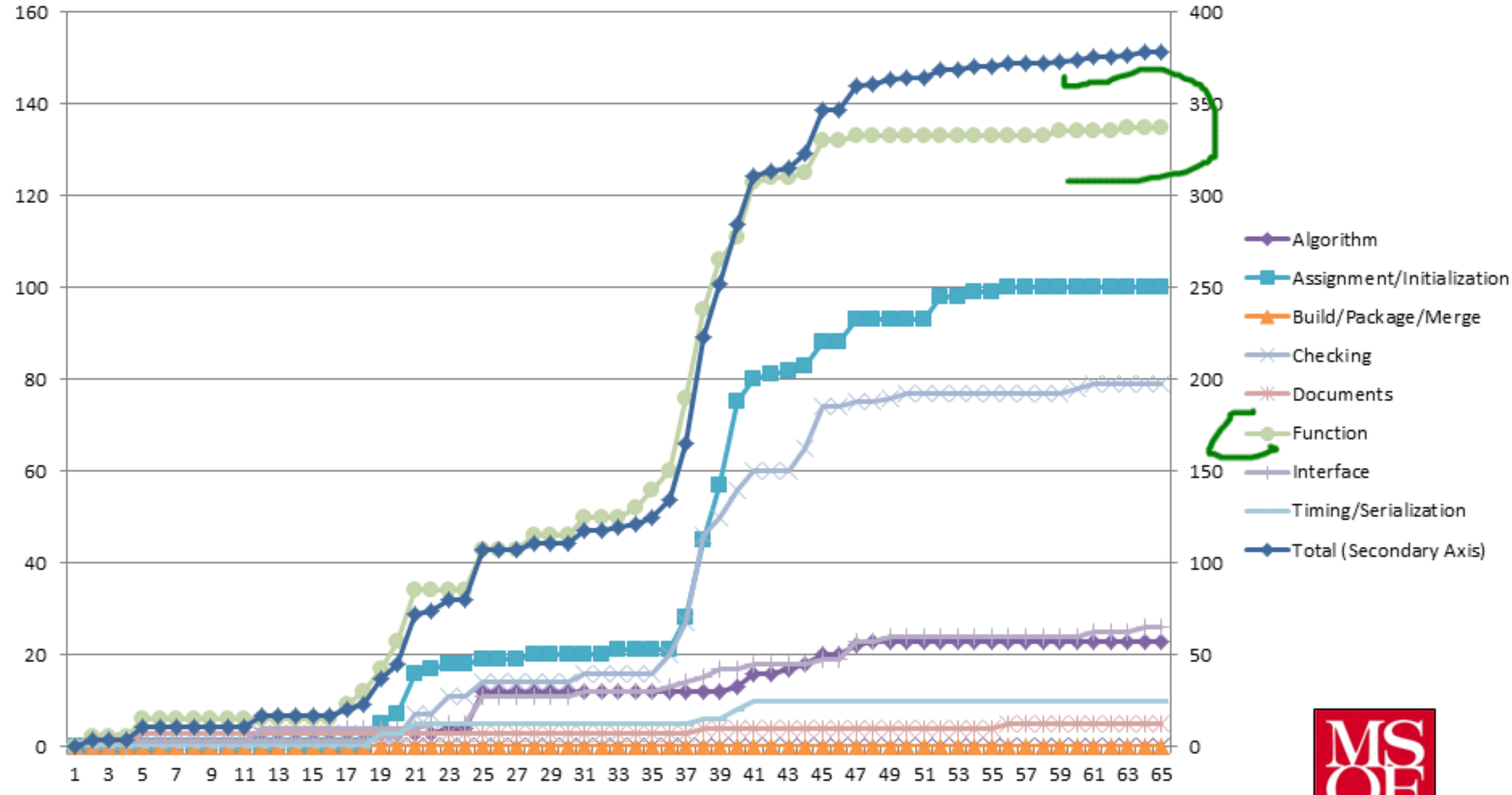
No, not shaped.



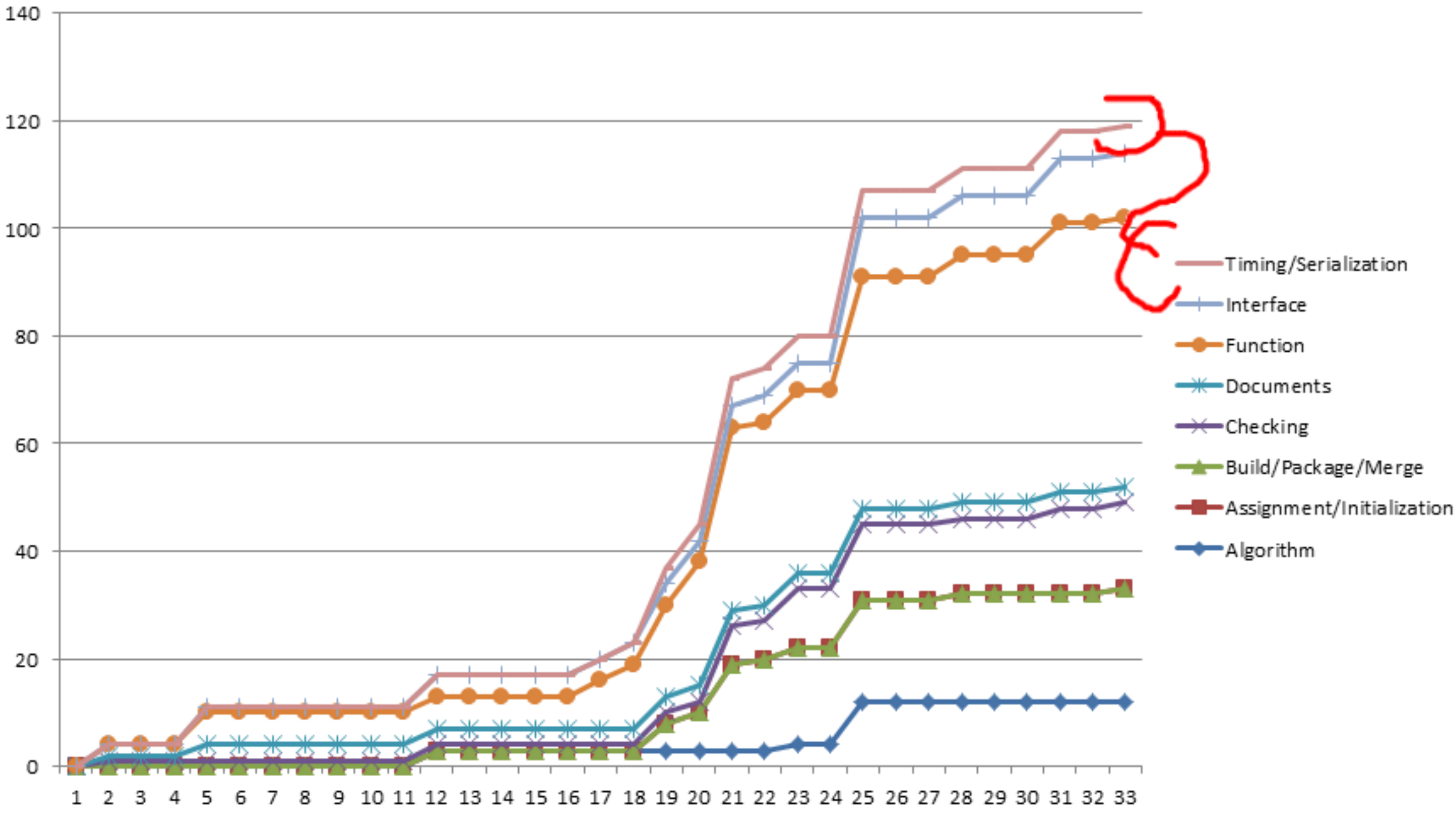
- The whole picture



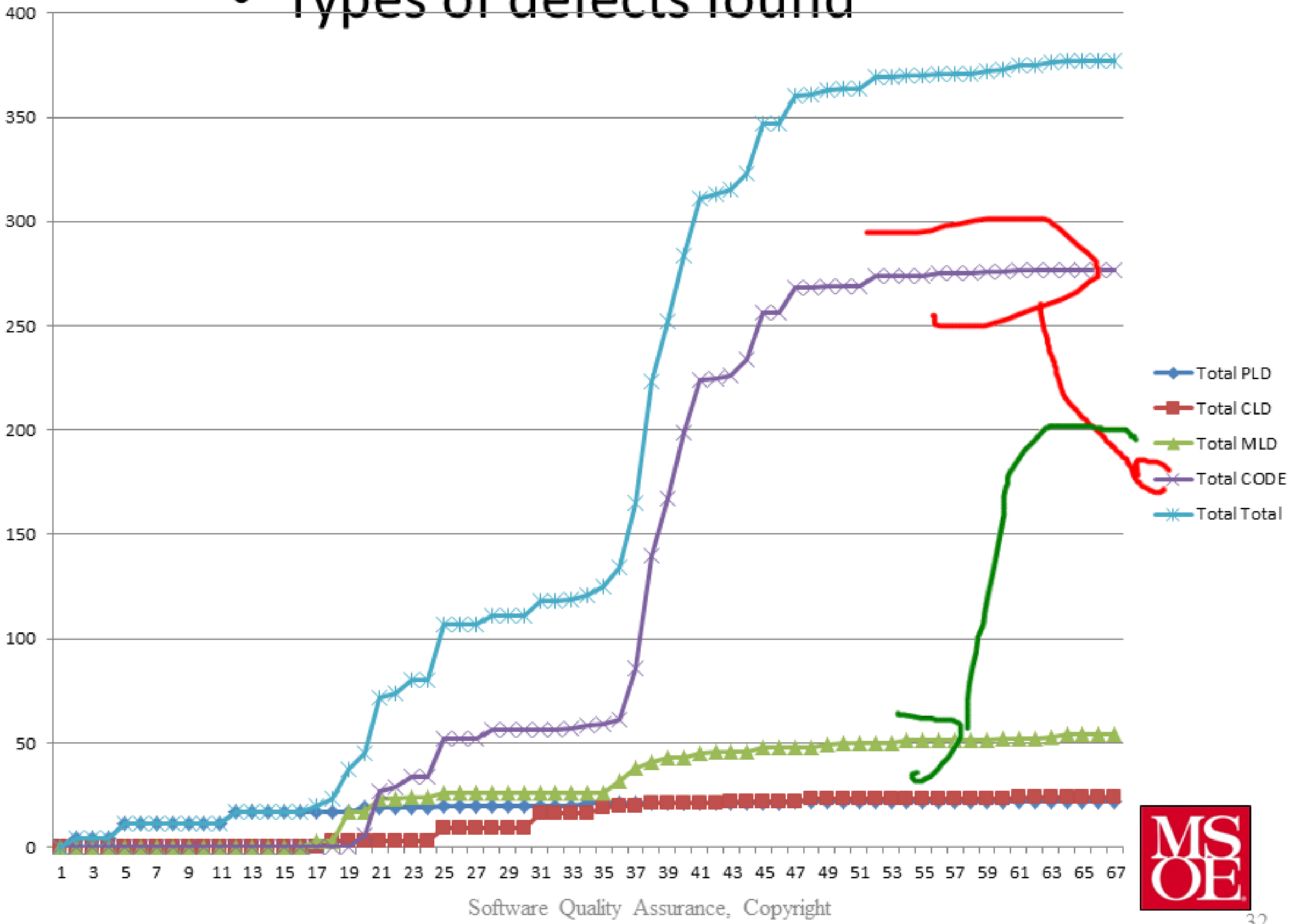
- Found Defect Types



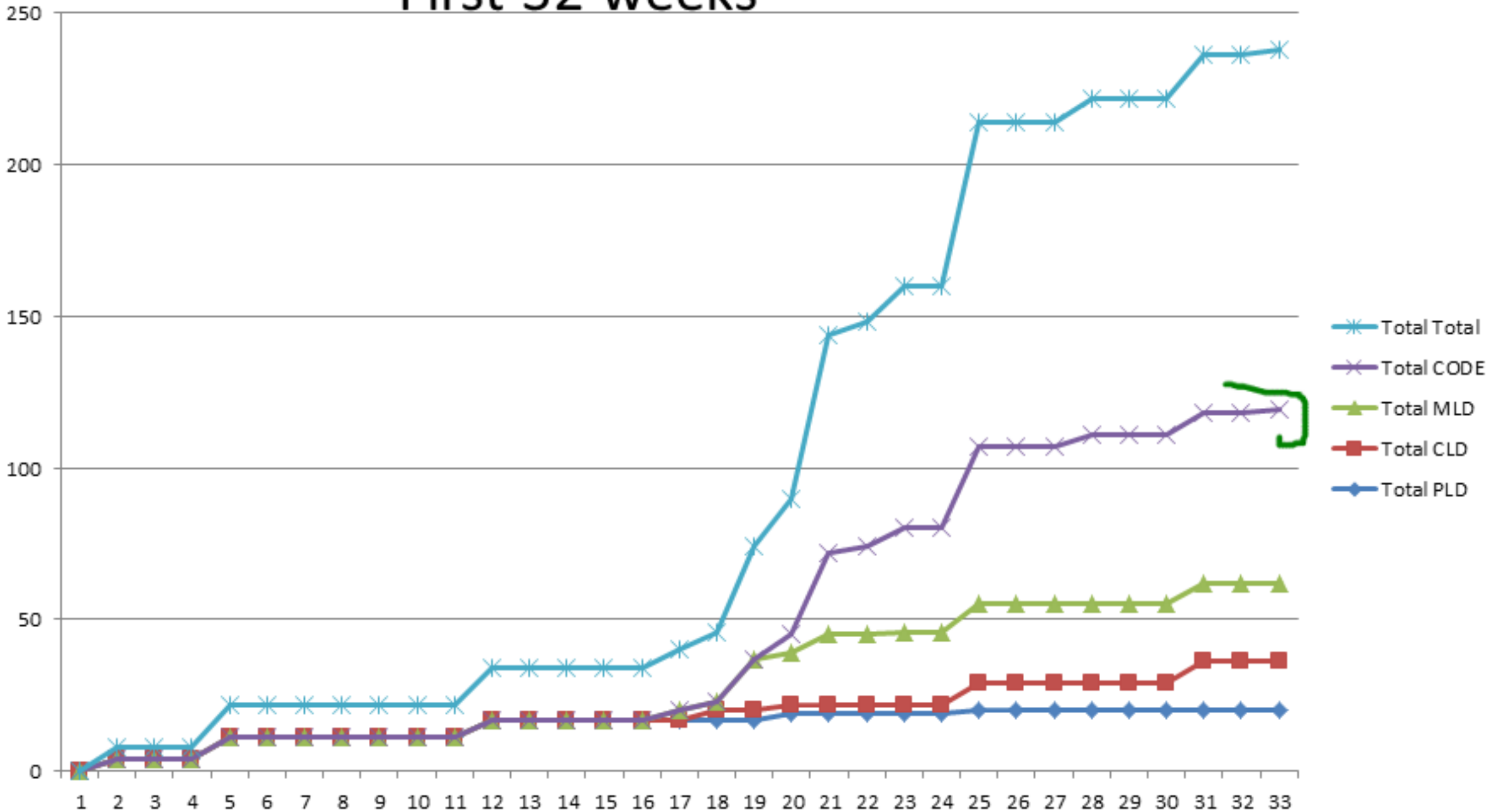
• Found Defect Types, first 32 weeks



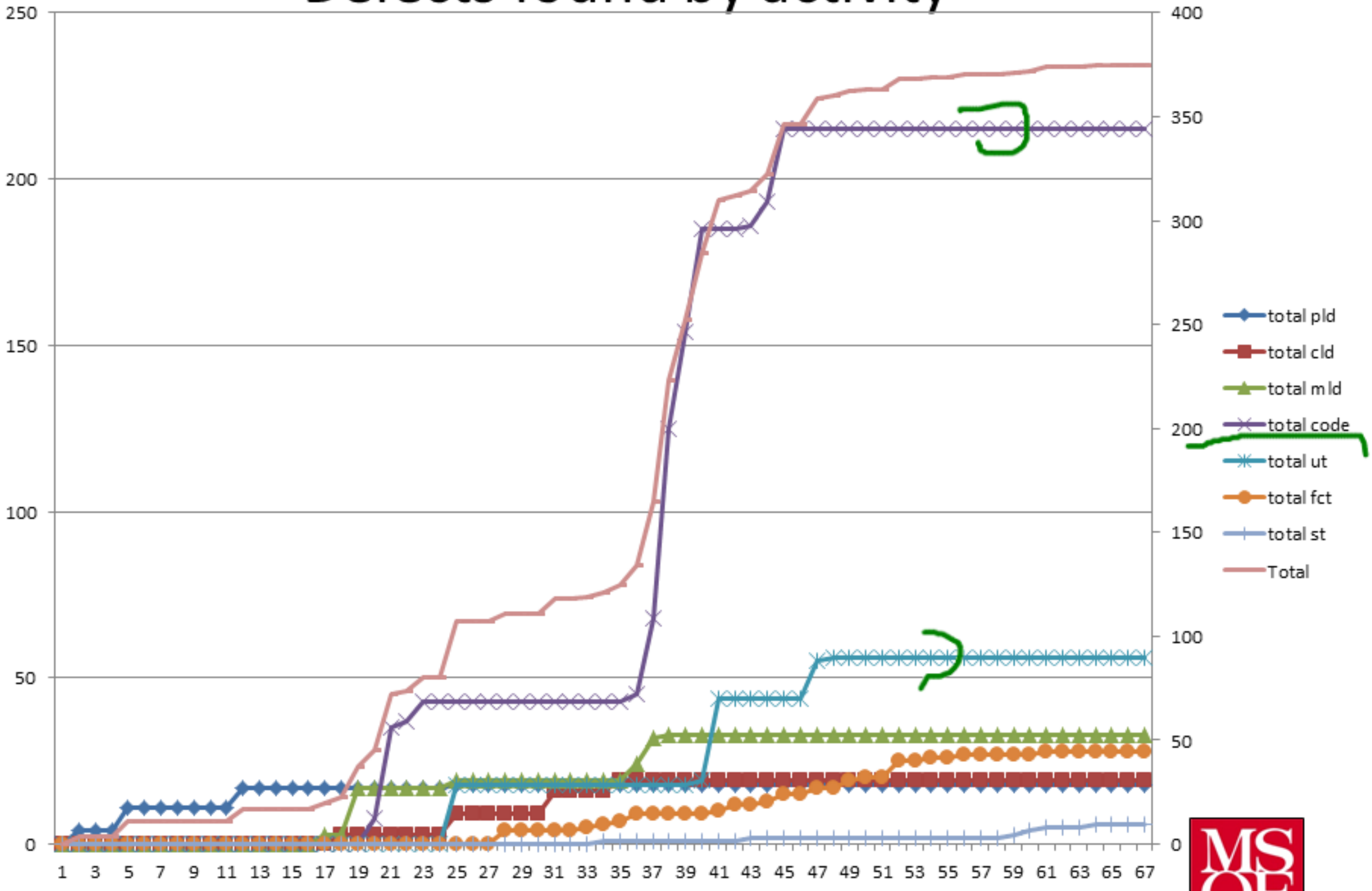
• Types of defects found



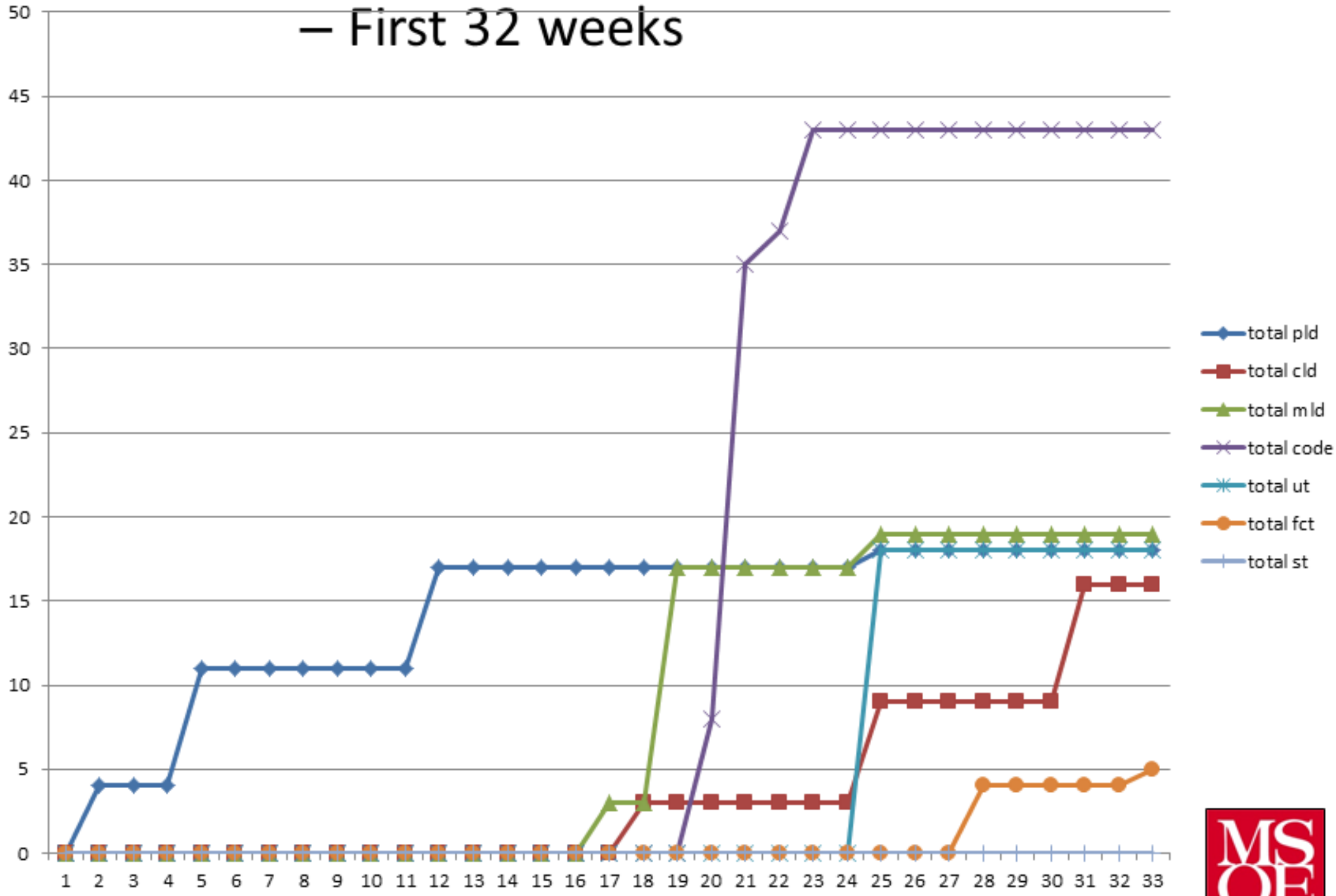
- Types of defects found
  - First 32 weeks

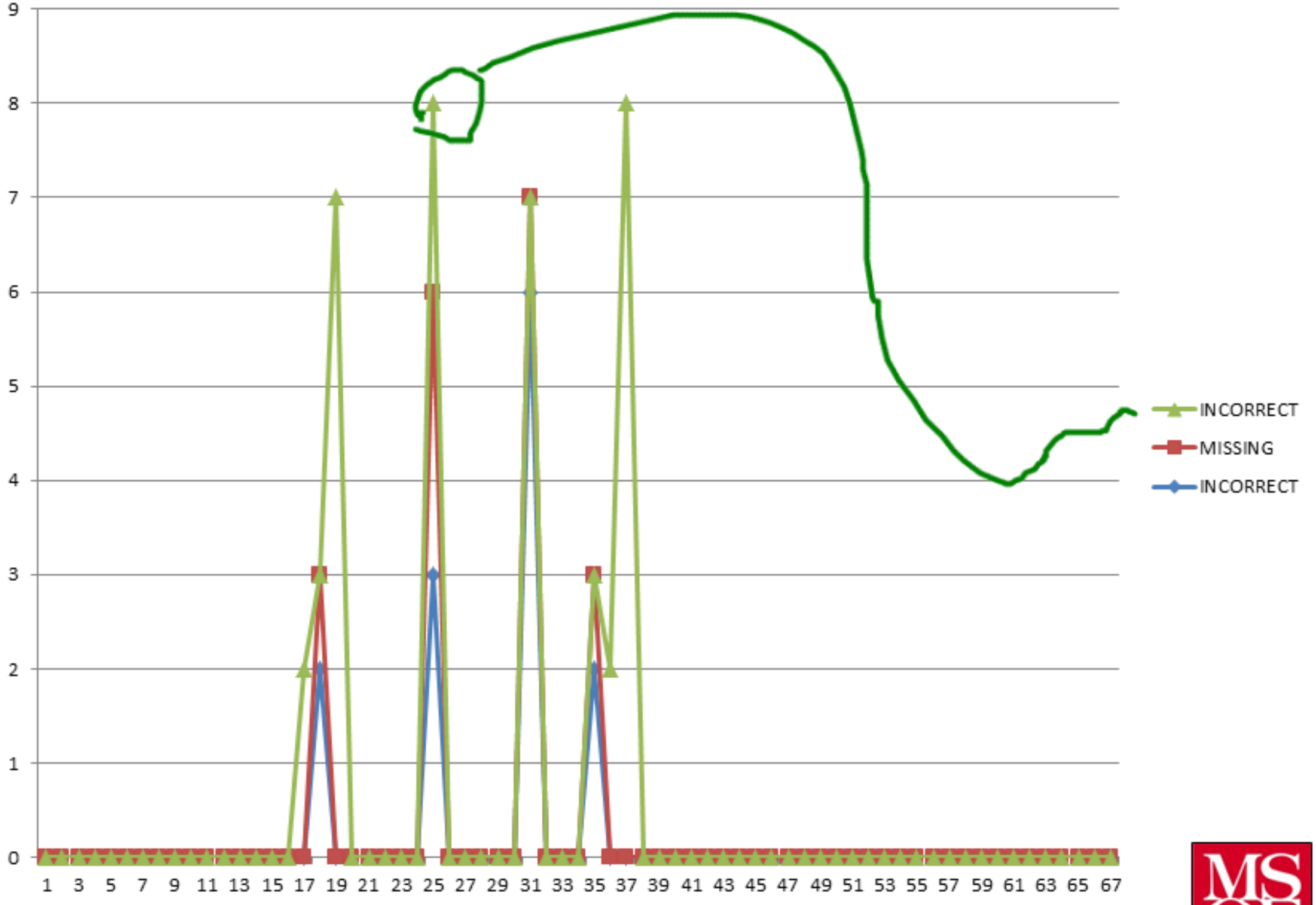


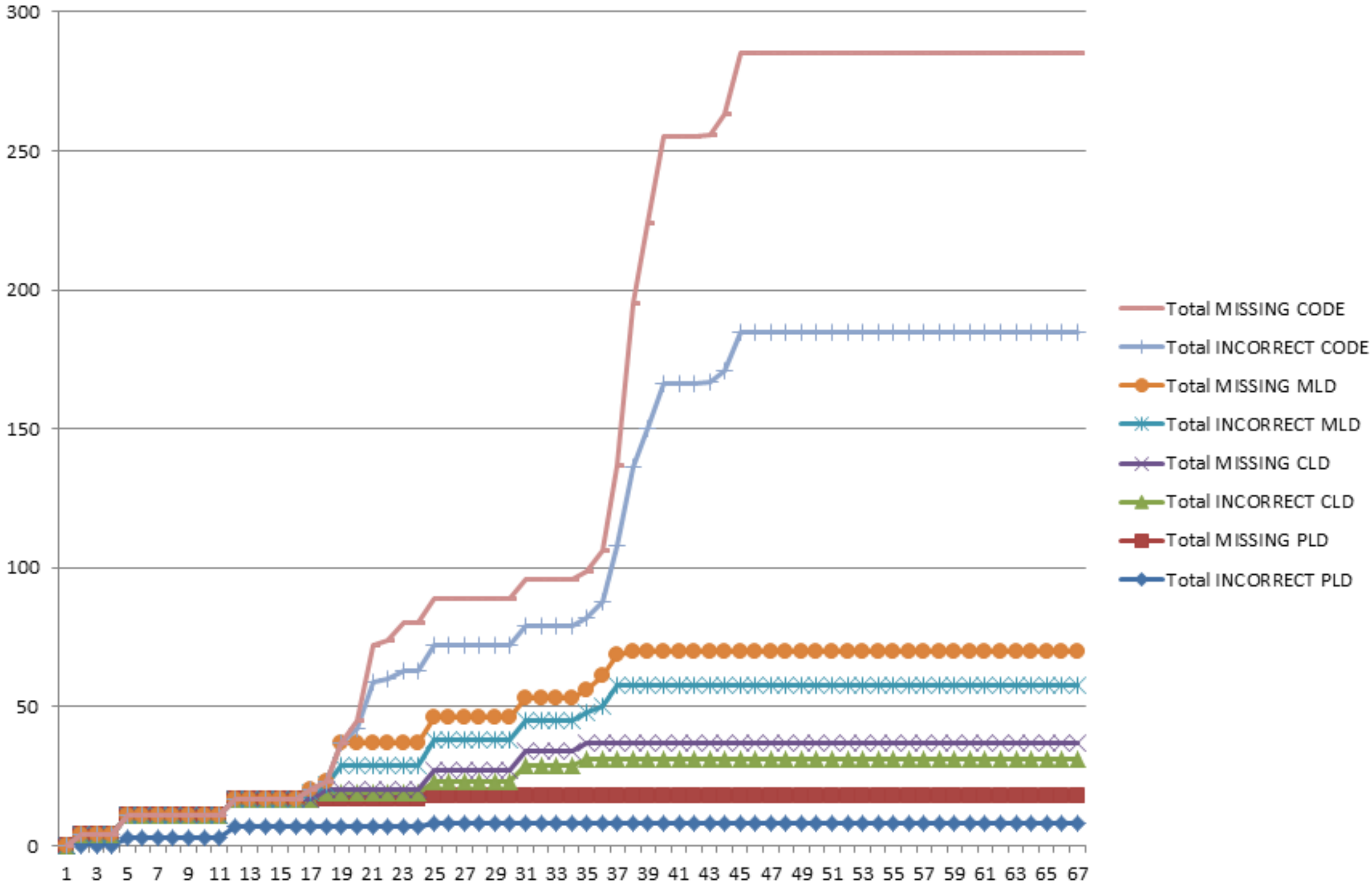
- Defects found by activity

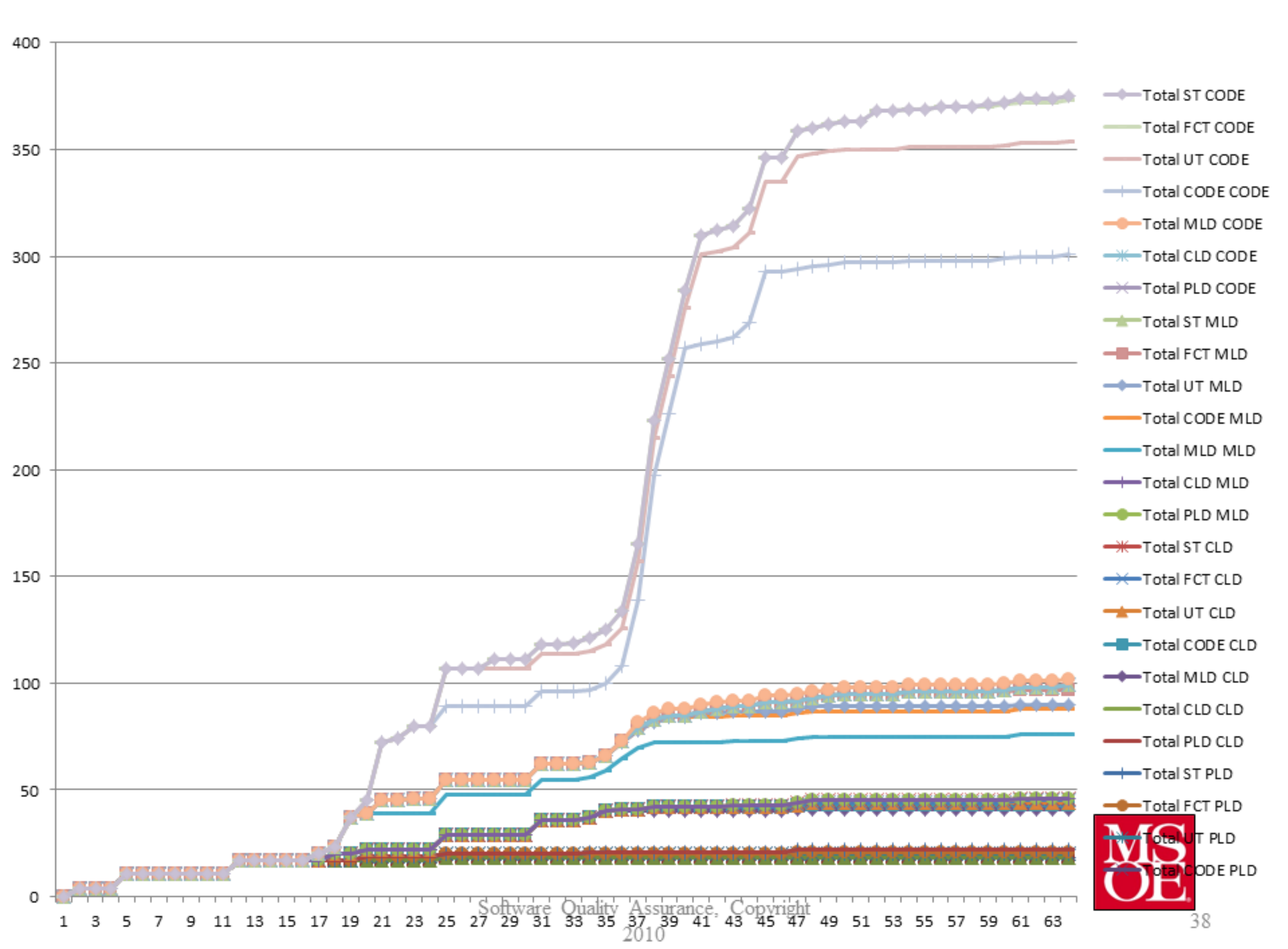


- Defects found by activity
  - First 32 weeks




9
8
7
6
5
4
3
2
1
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67
INCORRECT
MISSING
INCORRECT


300
250
200
150
100
50
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67
Total MISSING CODE
Total INCORRECT CODE
Total MISSING MLD
Total INCORRECT MLD
Total MISSING CLD
Total INCORRECT CLD
Total MISSING PLD
Total INCORRECT PLD


400
350
300
250
200
150
100
50
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63
Total ST CODE
Total FCT CODE
Total UT CODE
Total CODE CODE
Total MLD CODE
Total CLD CODE
Total PLD CODE
Total ST MLD
Total FCT MLD
Total UT MLD
Total CODE MLD
Total MLD MLD
Total CLD MLD
Total PLD MLD
Total ST CLD
Total FCT CLD
Total UT CLD
Total CODE CLD
Total MLD CLD
Total CLD CLD
Total PLD CLD
Total ST PLD
Total FCT PLD
Total UT PLD
Total CODE PLD