Software Testing and Software Log Analysis: When Will They Meet?

Prof. Mika Mäntylä,
University of Helsinki, Finland

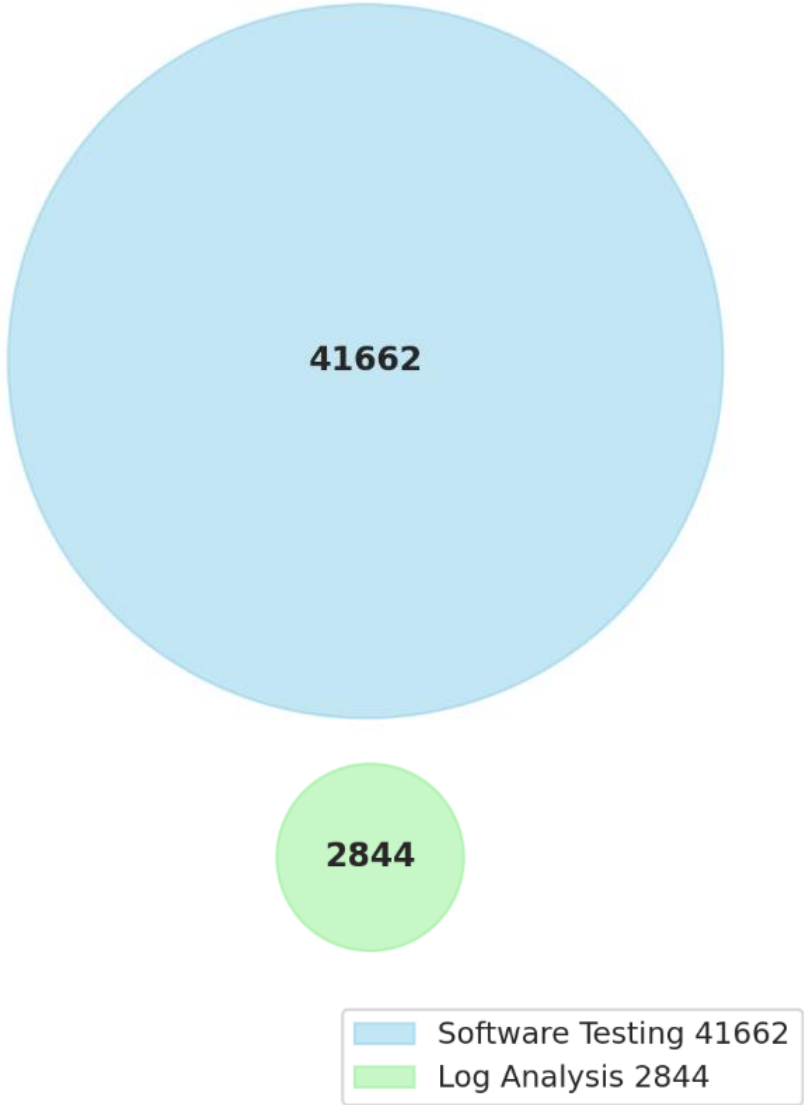
SUOMEN AKATEMIA

Outline

- Size of Body of Knowledge
- Software Testing
- Log Analysis
- Intersection
  - Logs as Coverage targets
  - Logs as Oracles

Both Software Testing and Log Analysis are big fields

- Software Testing 41,662
  - `TITLE-ABS-KEY ( "software testing" ) AND ( LIMIT-TO ( SUBJAREA , "COMP" ) )`
- Log Analysis 2,844
  - `( TITLE-ABS-KEY ( "log analysis" ) OR TITLE-ABS-KEY ( "log anomaly detection" ) OR TITLE-ABS-KEY ( "log file analysis" ) OR TITLE-ABS-KEY ( "log file anomaly detection" ) OR TITLE-ABS-KEY ( "software log" ) OR TITLE-ABS-KEY ( "software execution log" ) ) AND ( LIMIT-TO ( SUBJAREA , "COMP" ) )`

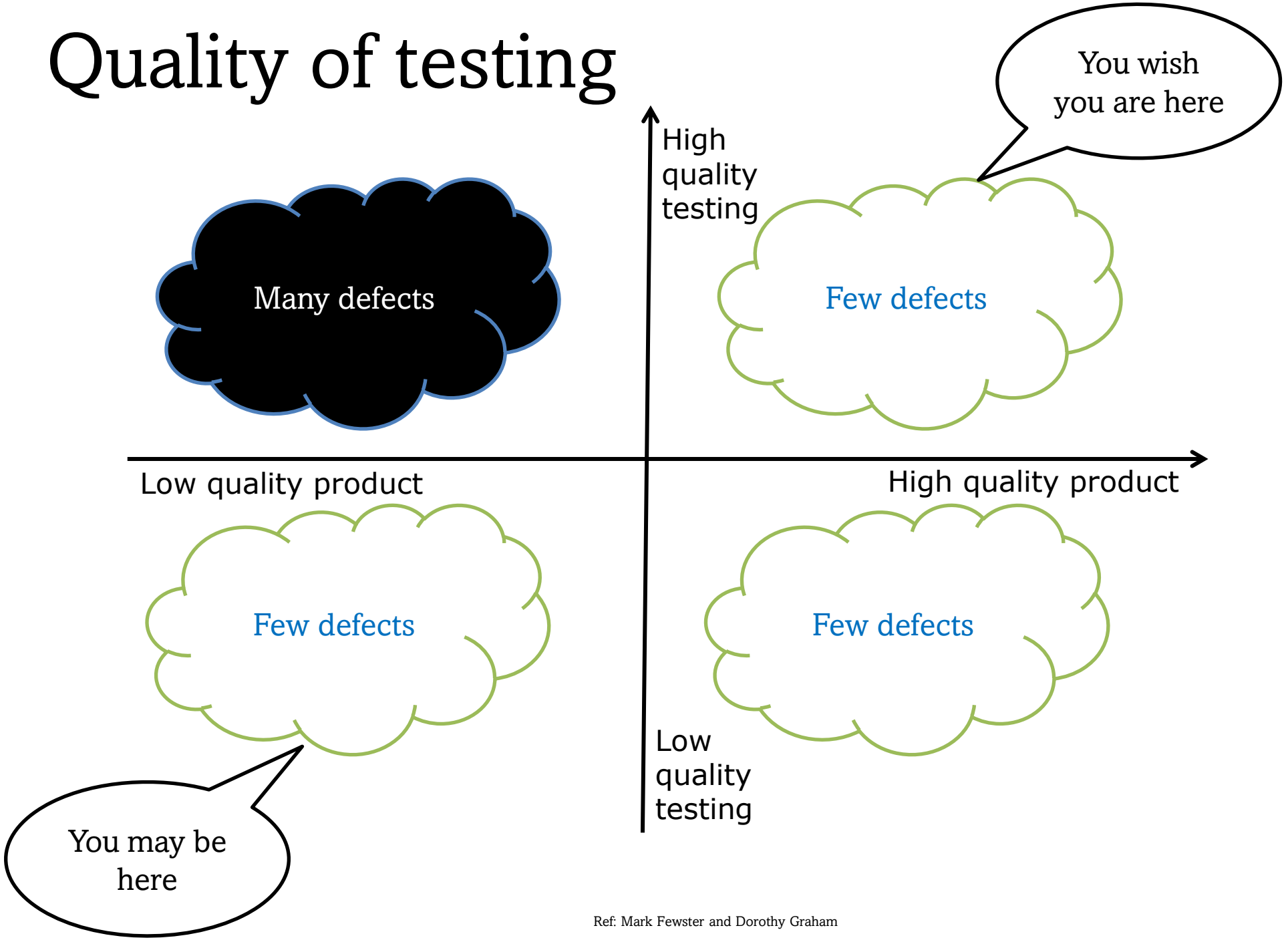


SOFTWARE TESTING

Software Testing is Information Seeking

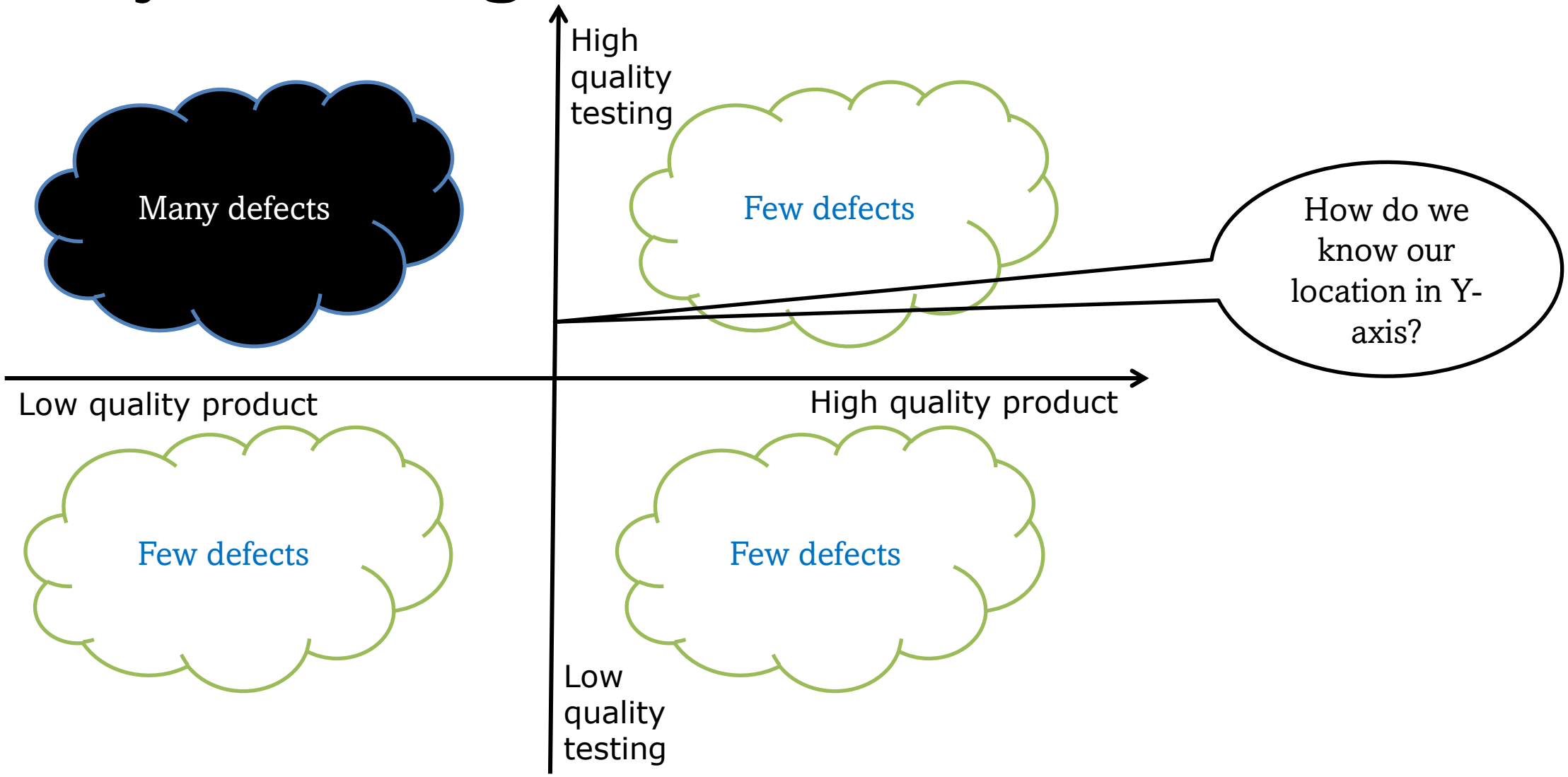
- Many definitions exist
- Perspective:
  - Verification: Are we building the product right?
  - Validation: Are we building the right product?
  - Regressions: Does the product still work?
- Type of information:
  - Functionality
  - Performance
  - Security

Quality of testing



Ref: Mark Fewster and Dorothy Graham

Quality of testing

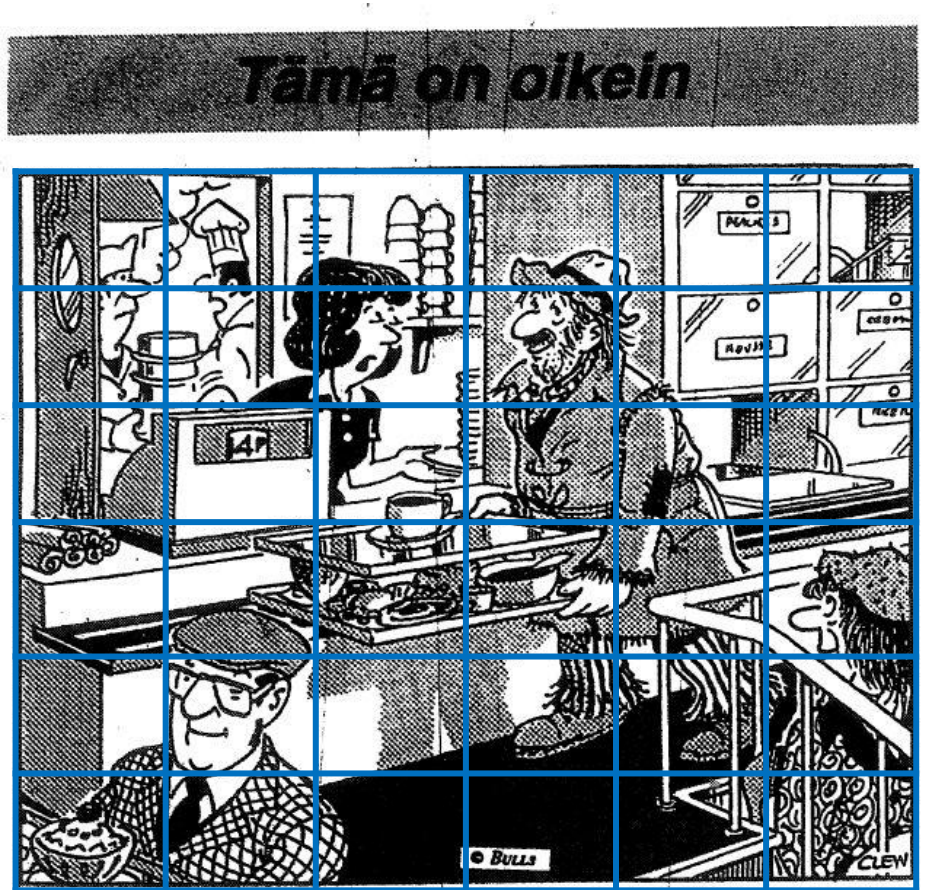


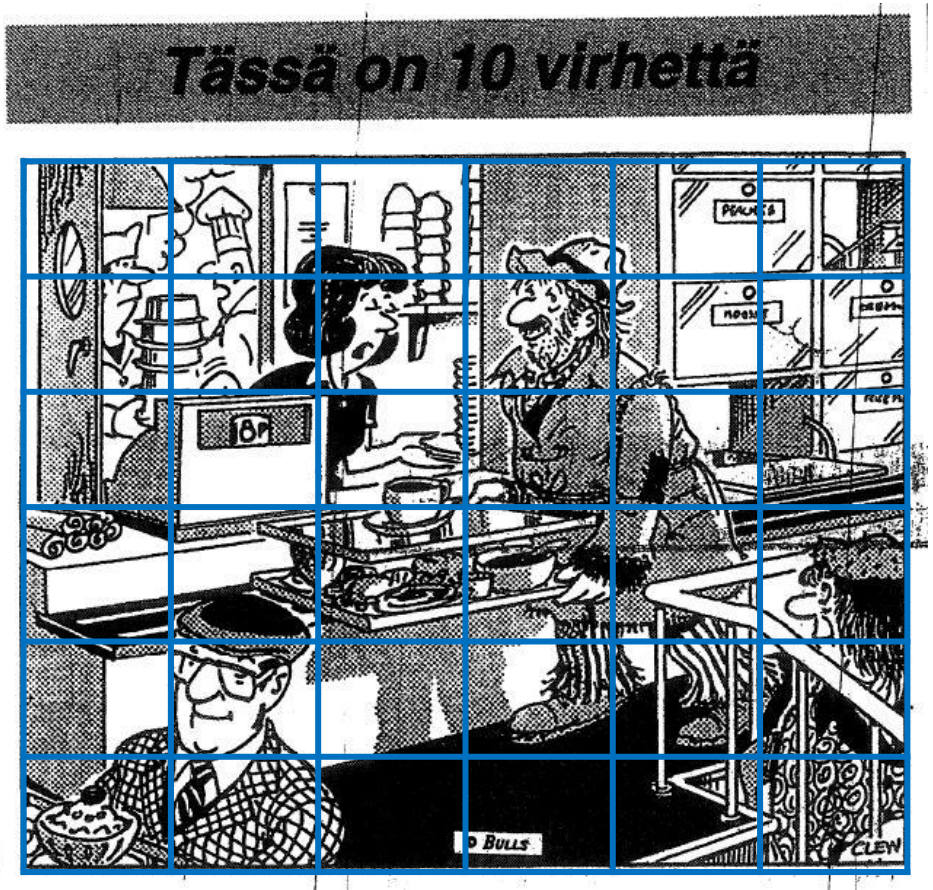
Ref: Mark Fewster and Dorothy Graham

- There are 10 defects in the figure on the right
- QA task : Find defects

- How do you know you have done a good job in finding defects in the figure?
- Quality of testing?

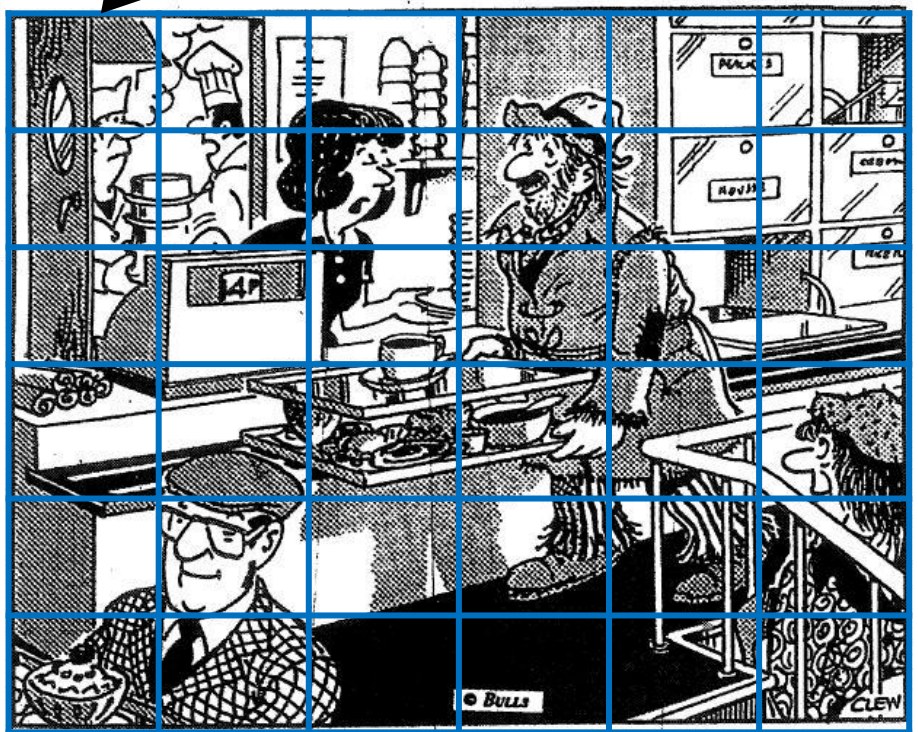
Coverage grid!
100% Coverage -> Good testing



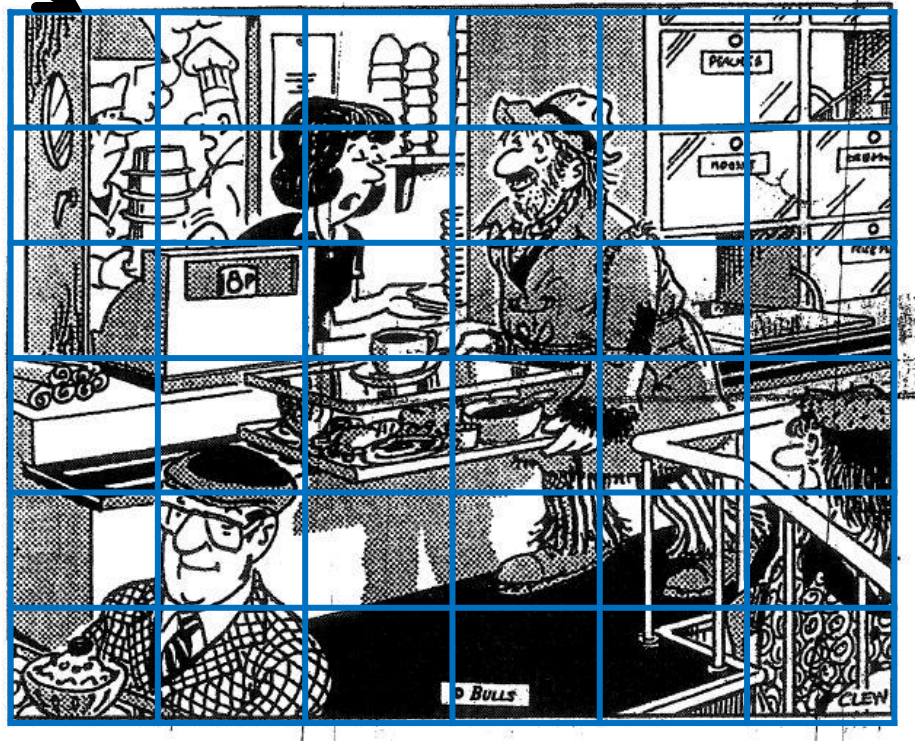


?

Tämä on oikein

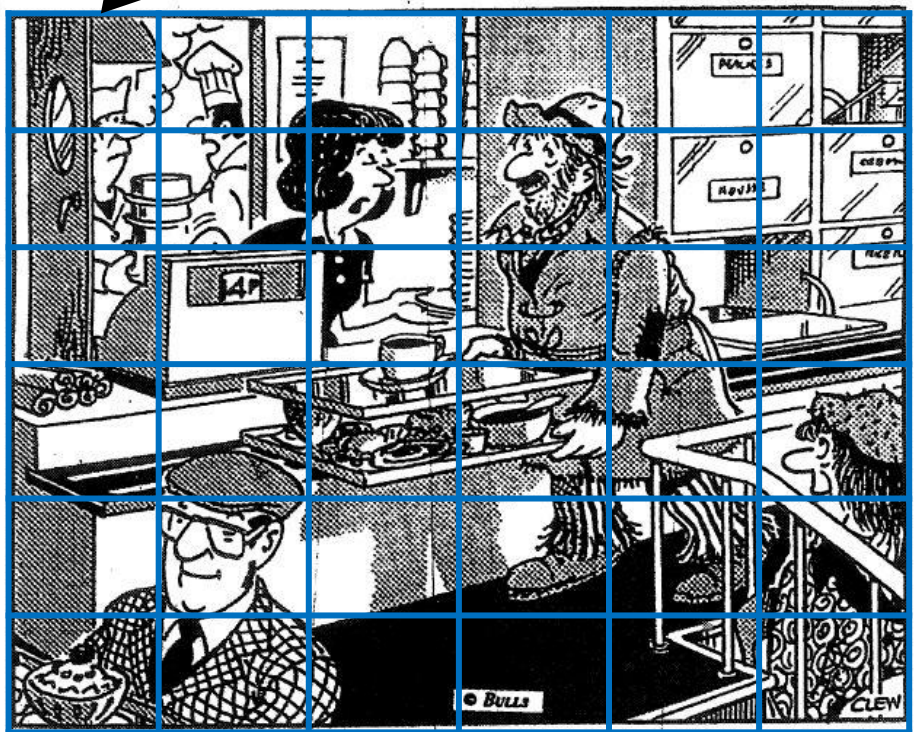


Tässä on 10 virhettä

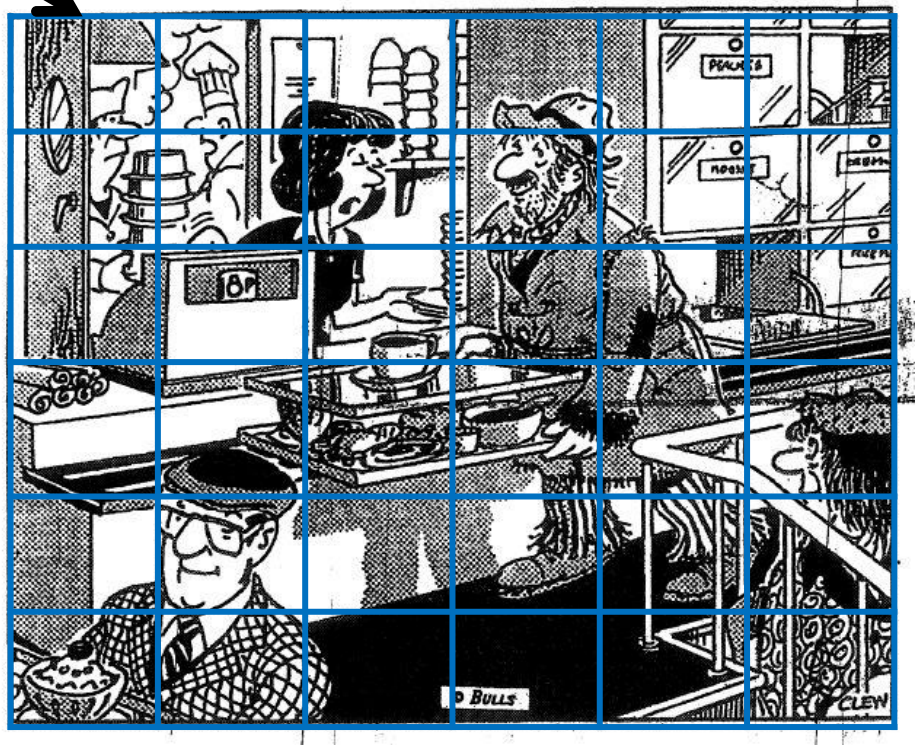


Pass

Tämä on oikein

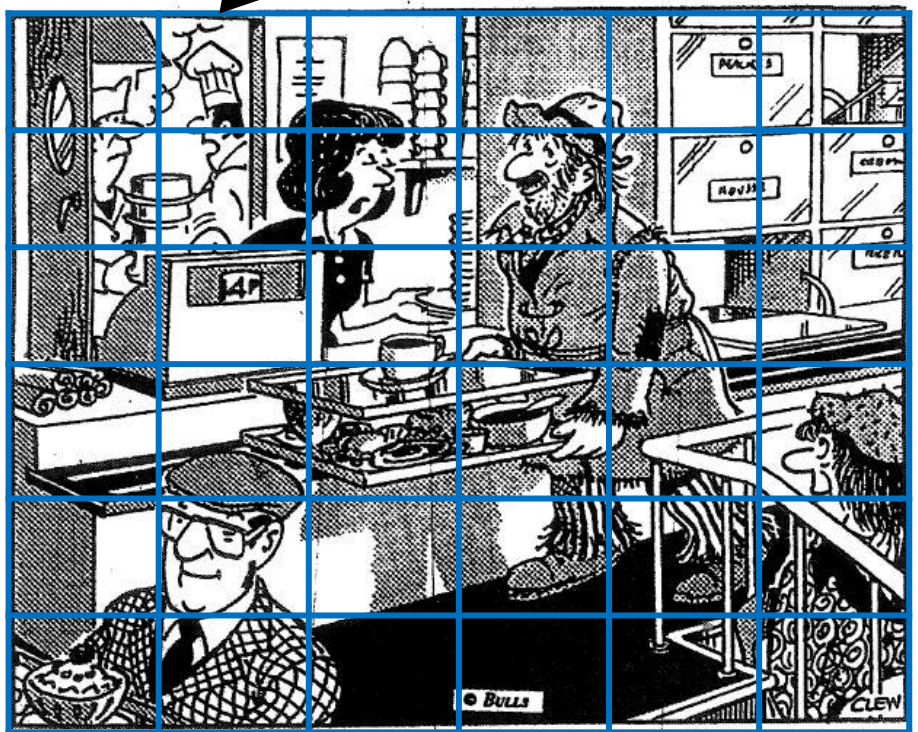


Tässä on 10 virhettä

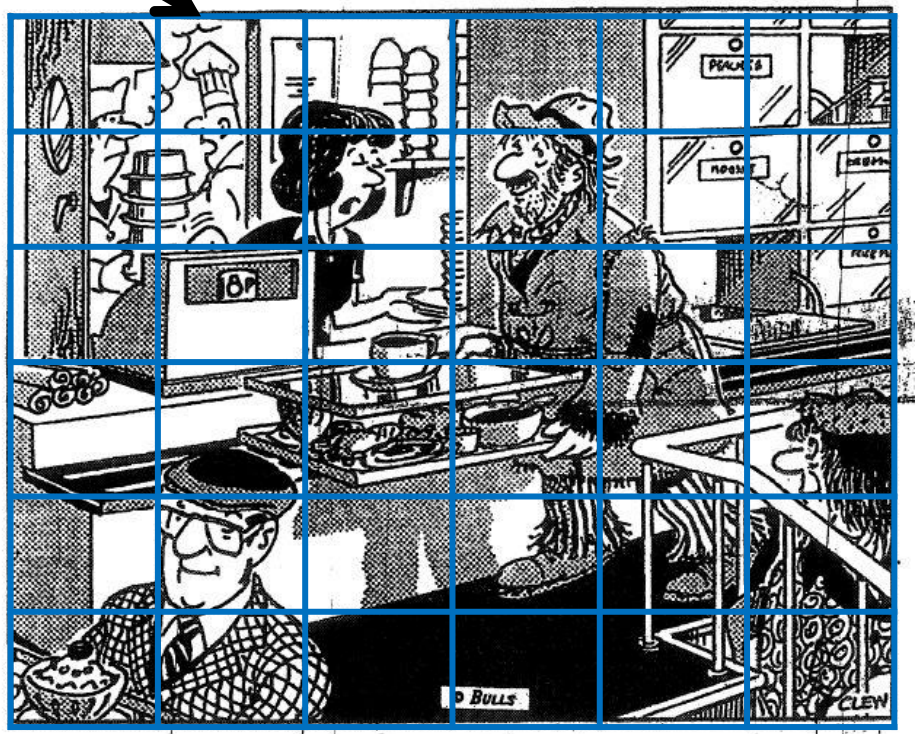


Failure

Tämä on oikein

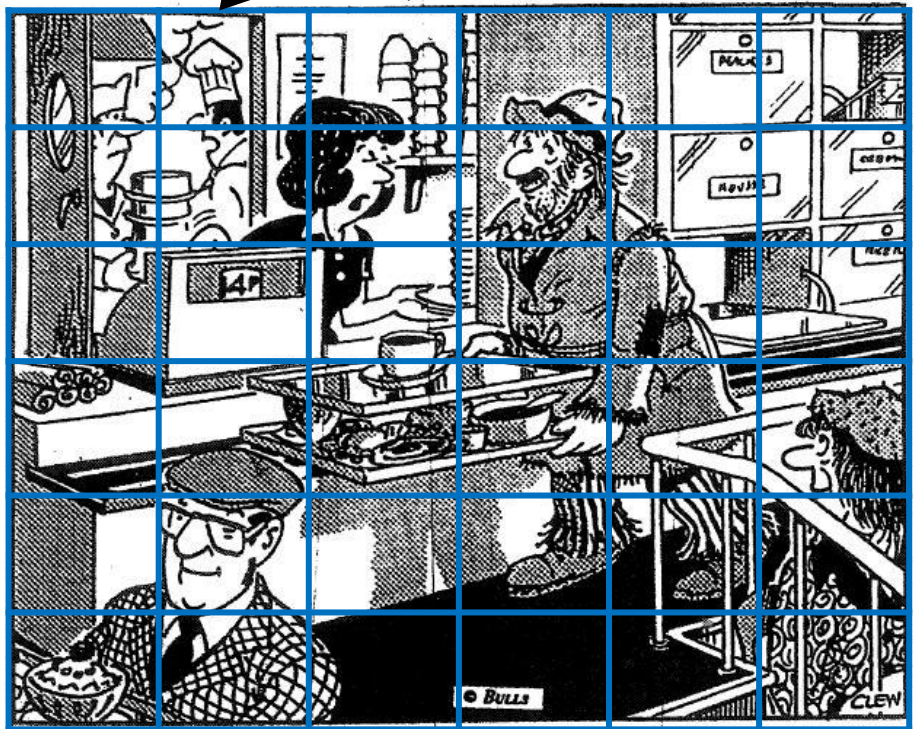


Tässä on 10 virhettä

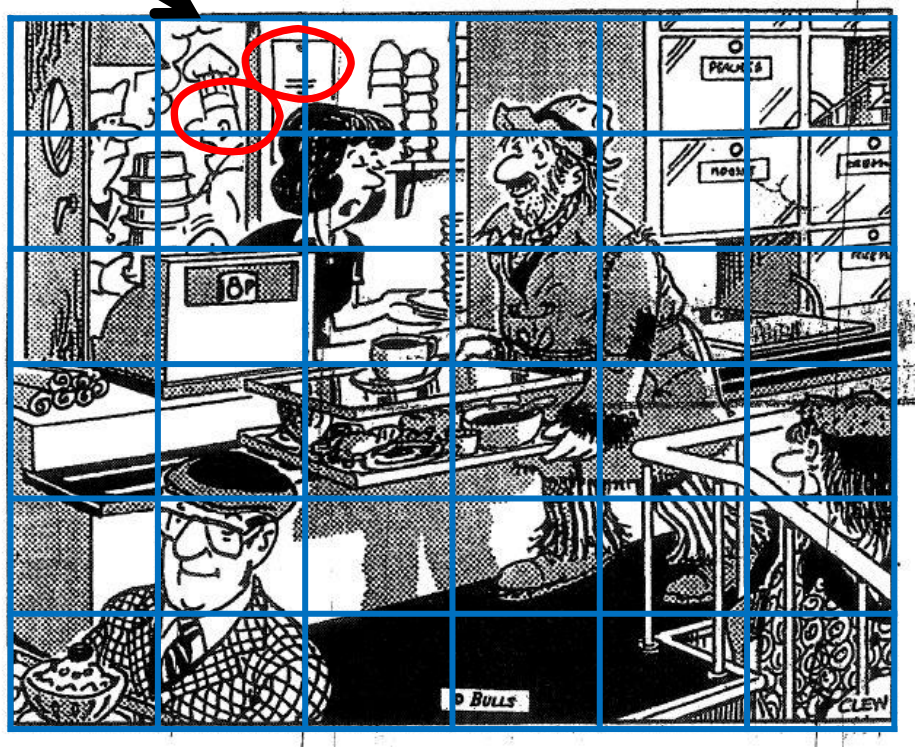


Failure

Tämä on oikein

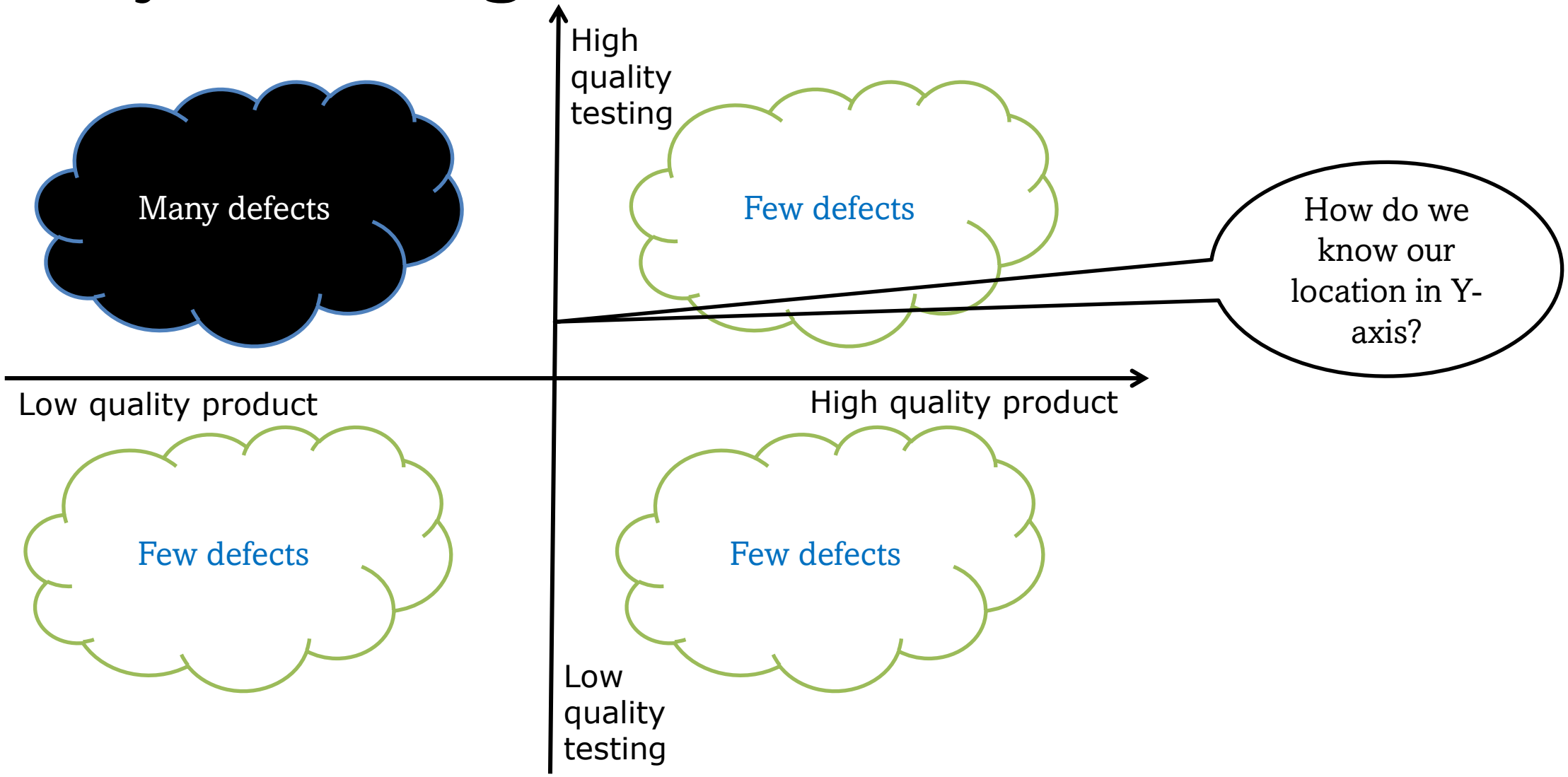


Tässä on 10 virhettä



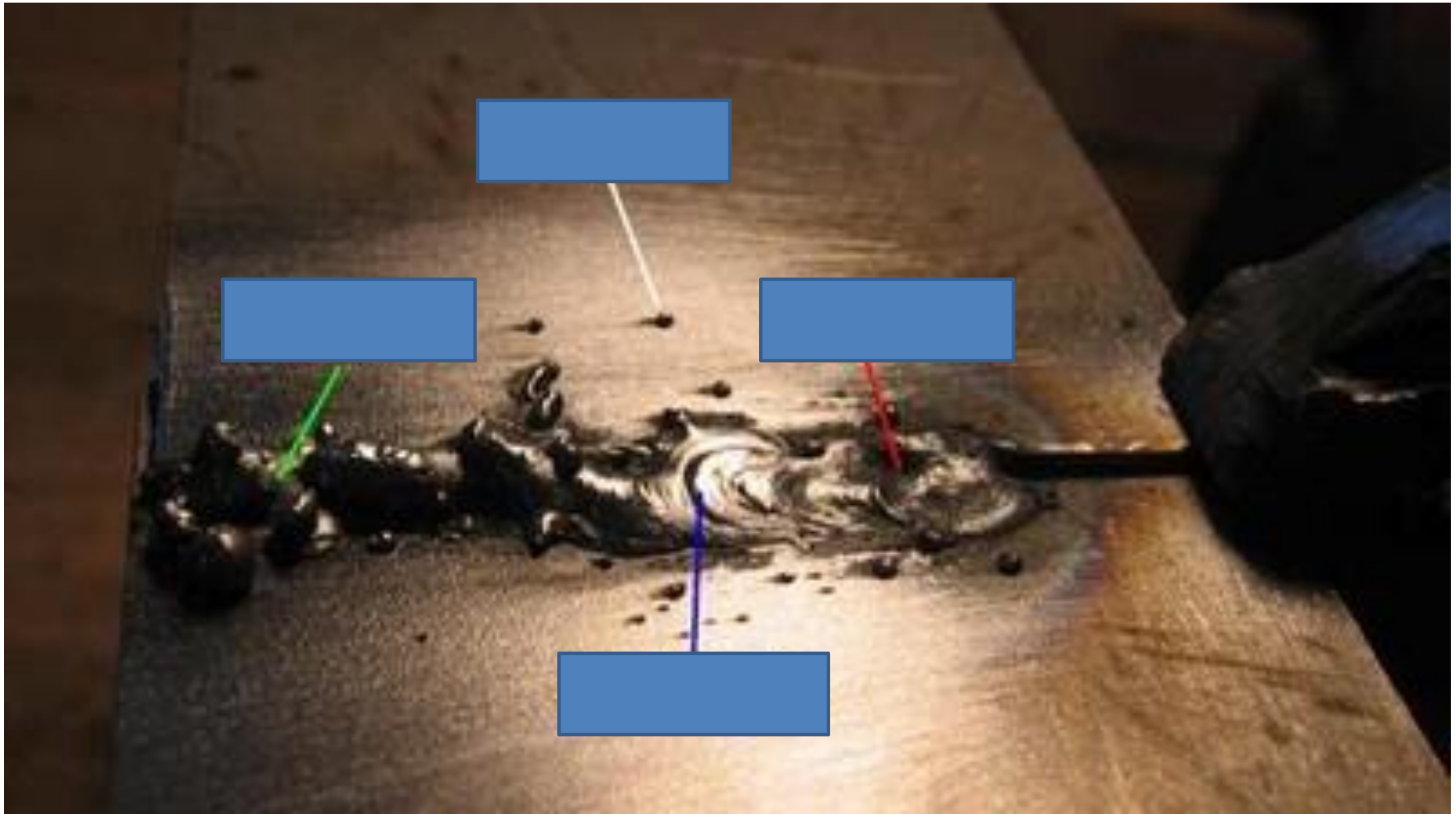
- Quality of testing?
- Is only about the coverage in this task

Quality of testing

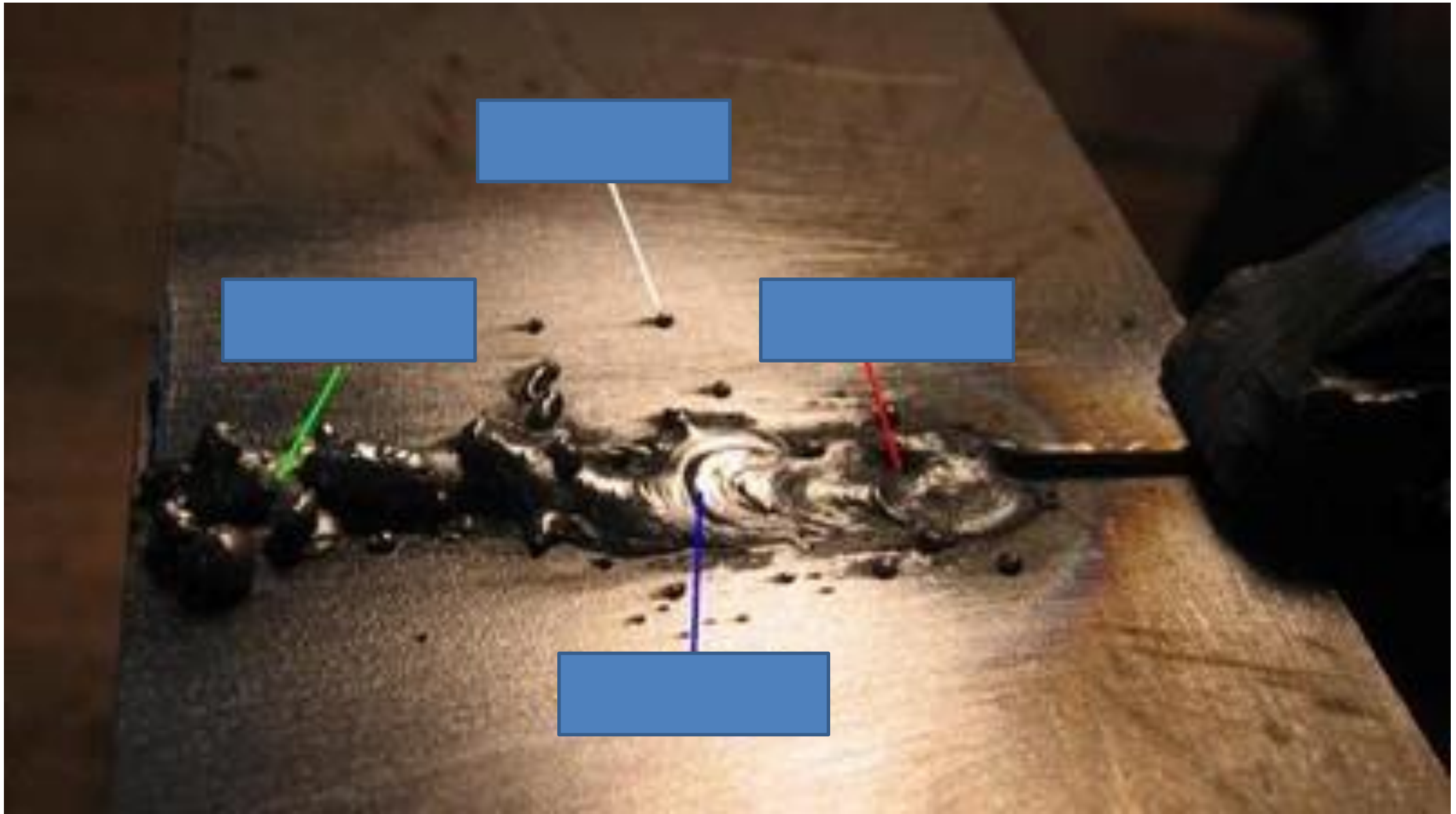


Ref: Mark Fewster and Dorothy Graham

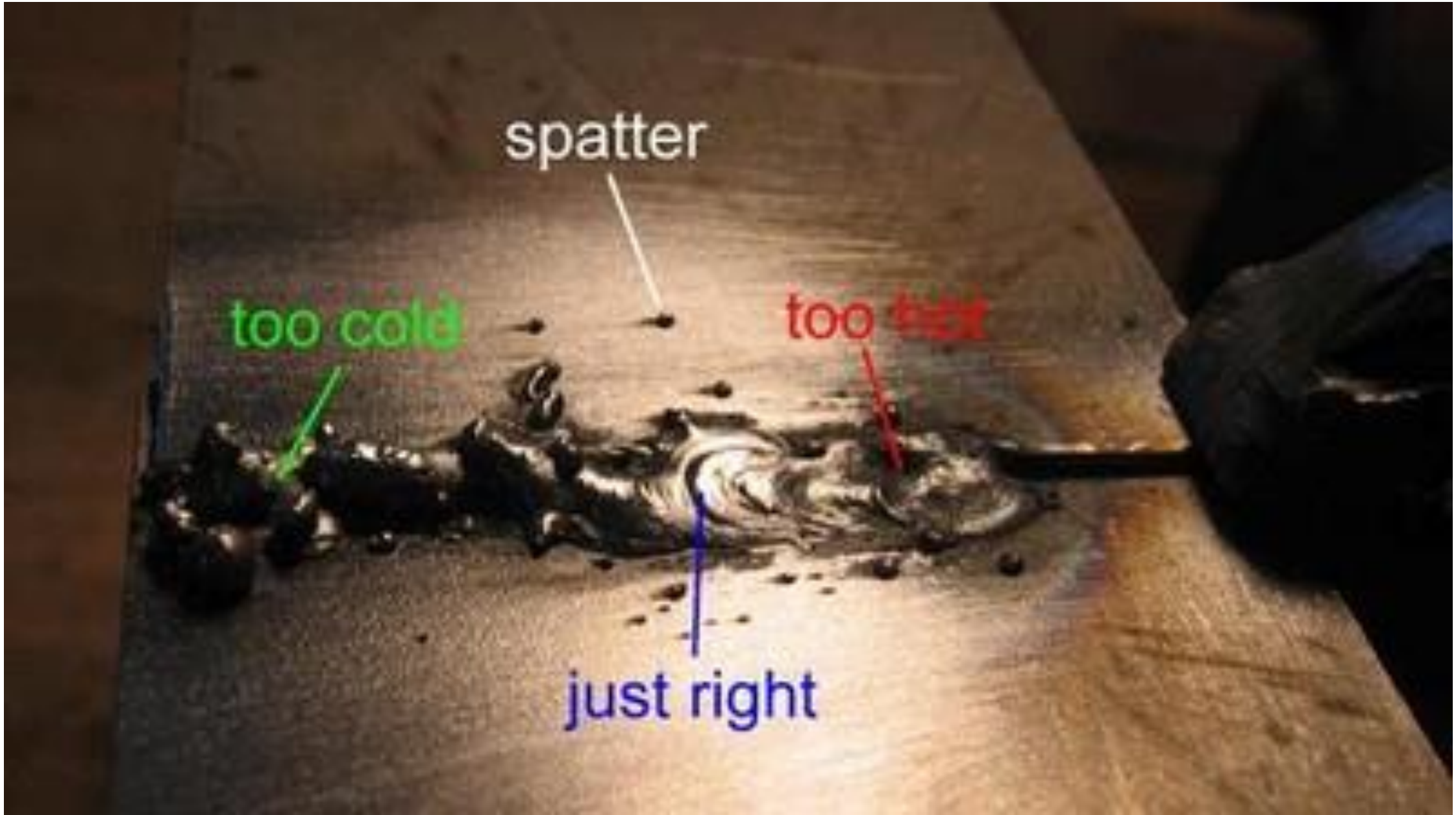
Coverage (defect locations) are given but “testing” job is not easy



What about right answer? Oracle?

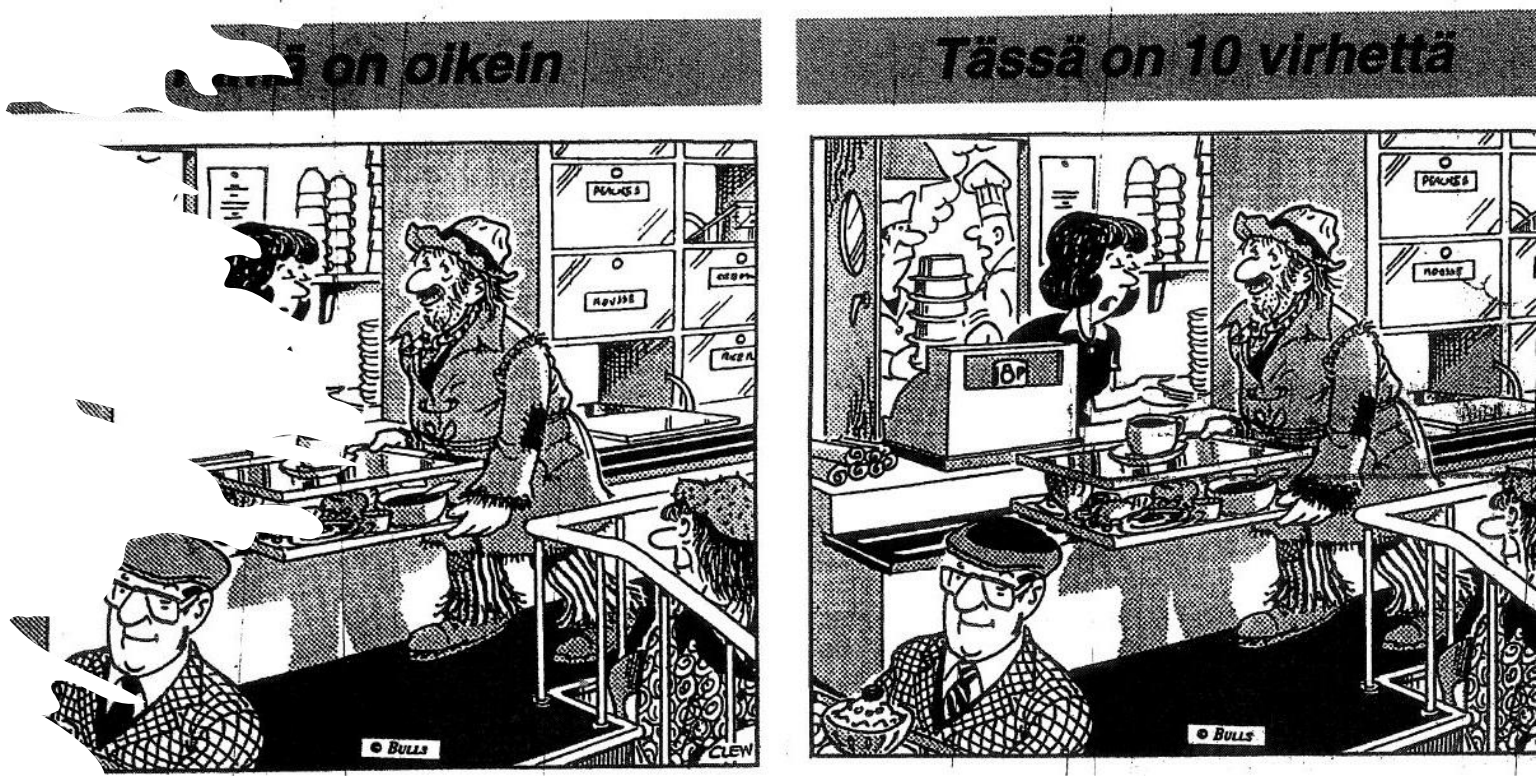


Welding Oracle

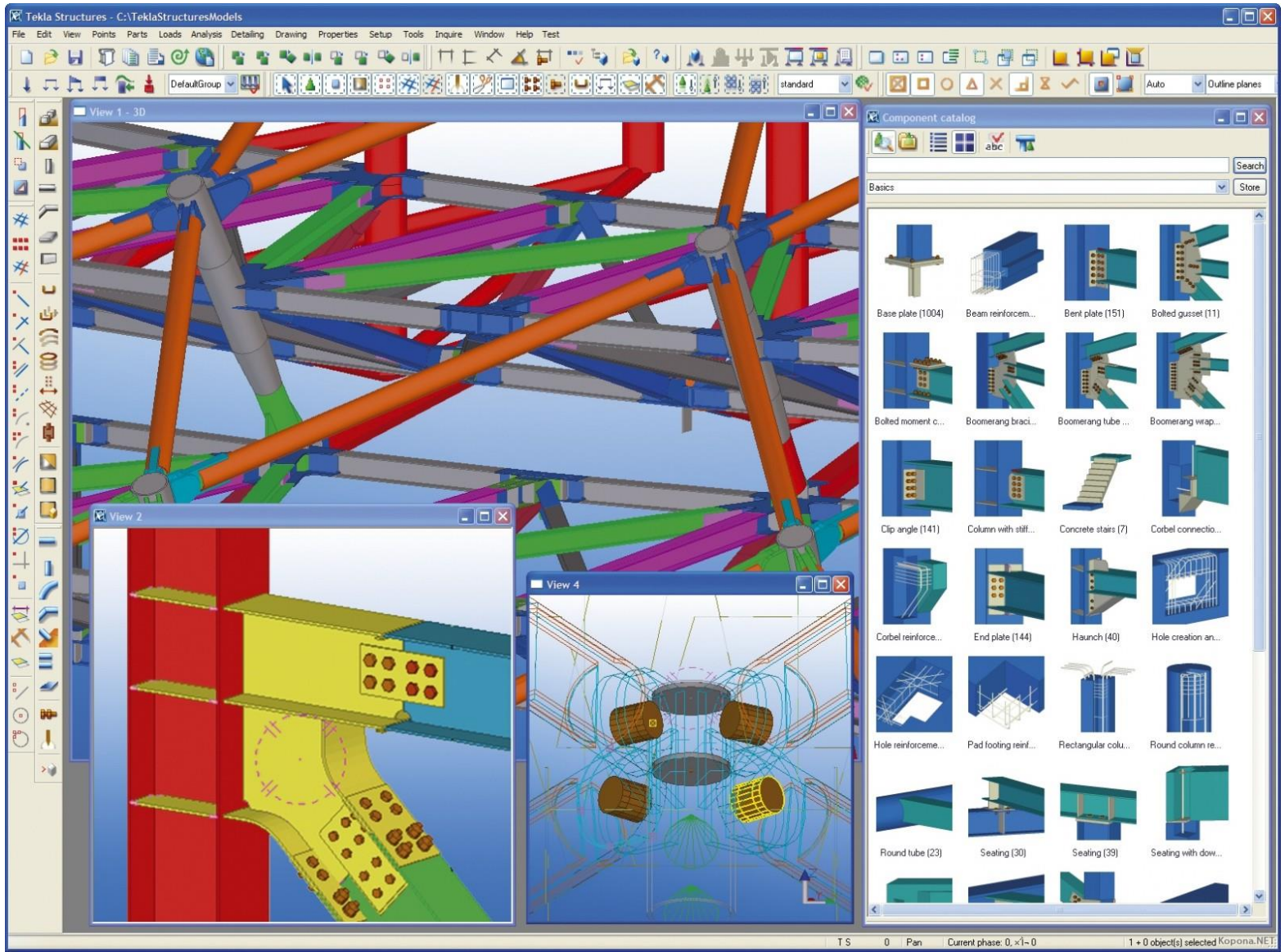


Coverage & Oracle

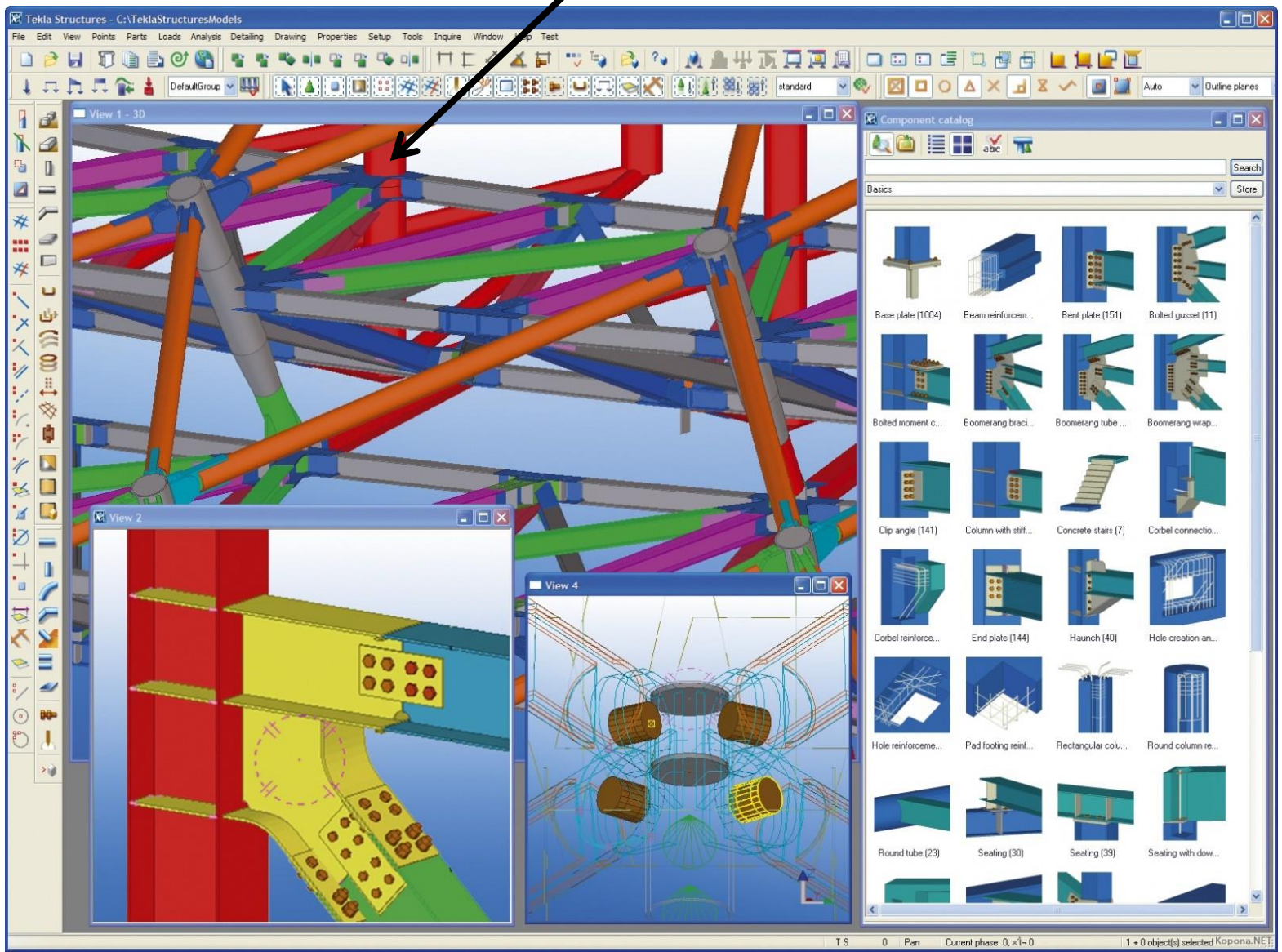
- In left we have the perfect oracle
  - i.e. the correct answer
- In this first task, detection was dependent on the coverage
- In the second task, detection was dependent on the Oracle



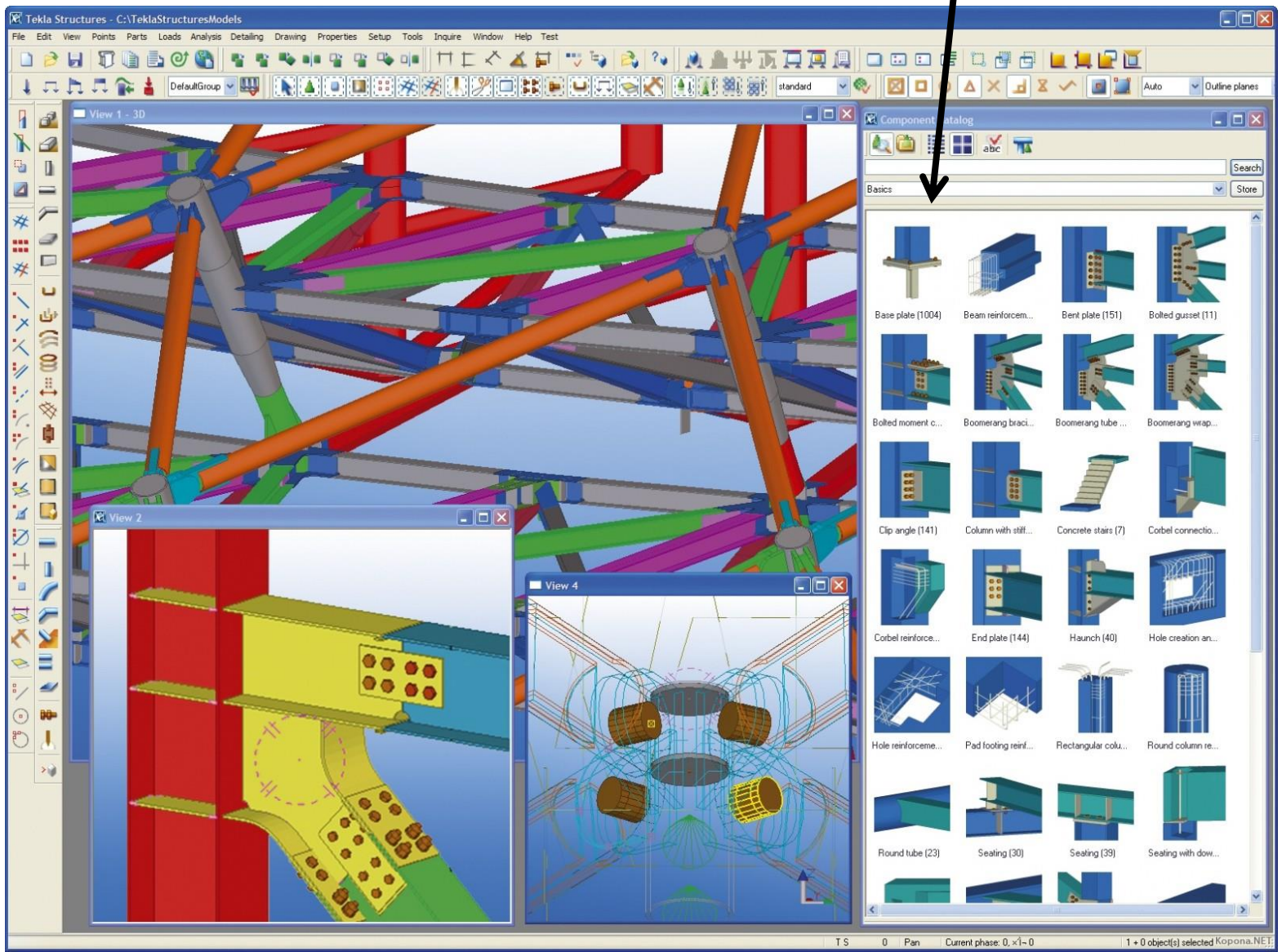
Software example of oracle problems



Oracle - Are the colors correct in the image? What is the meaning of the colors

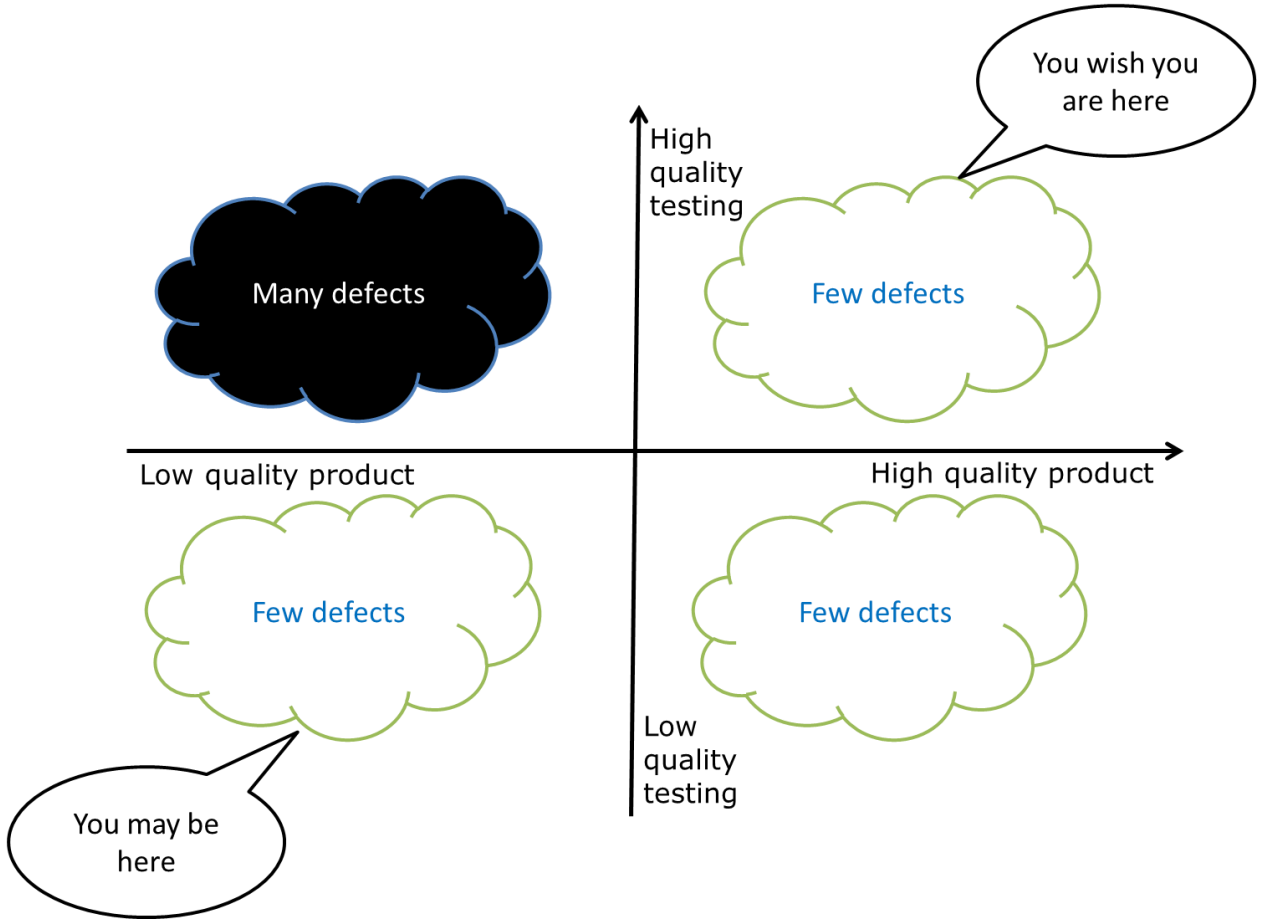


Oracle - Are all the components available?



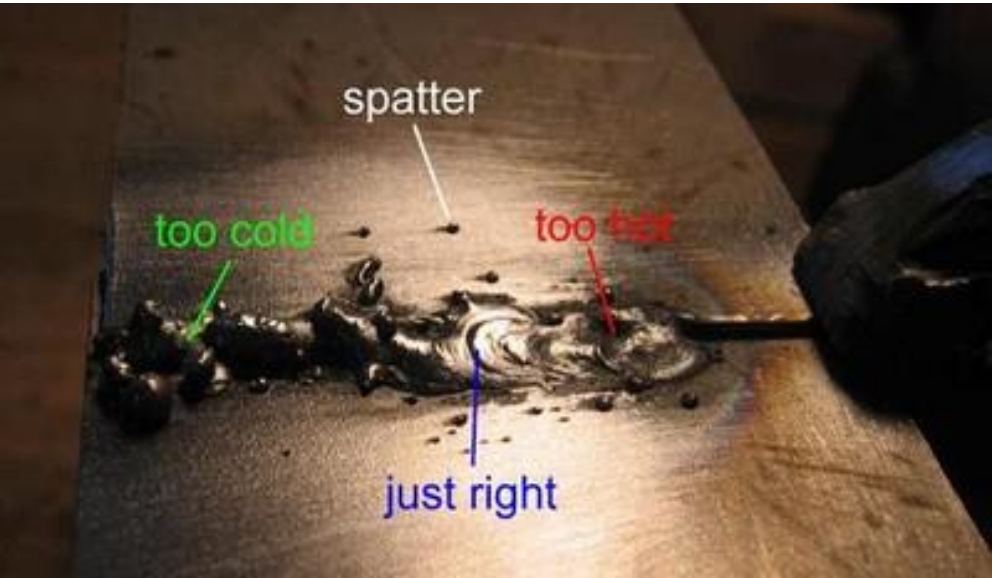
Quality of testing – Two dimensions

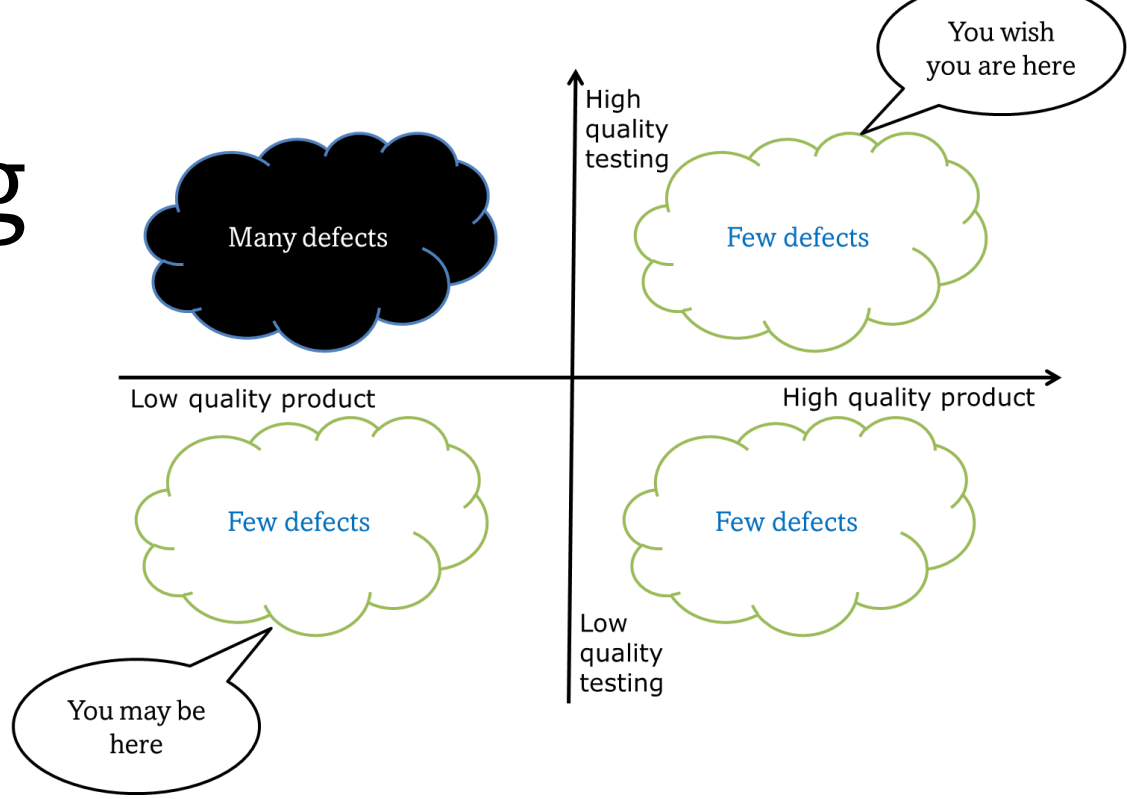
- Coverage – What areas have been tested
- Oracle – How good is the detection of defects (of the areas that have been covered)

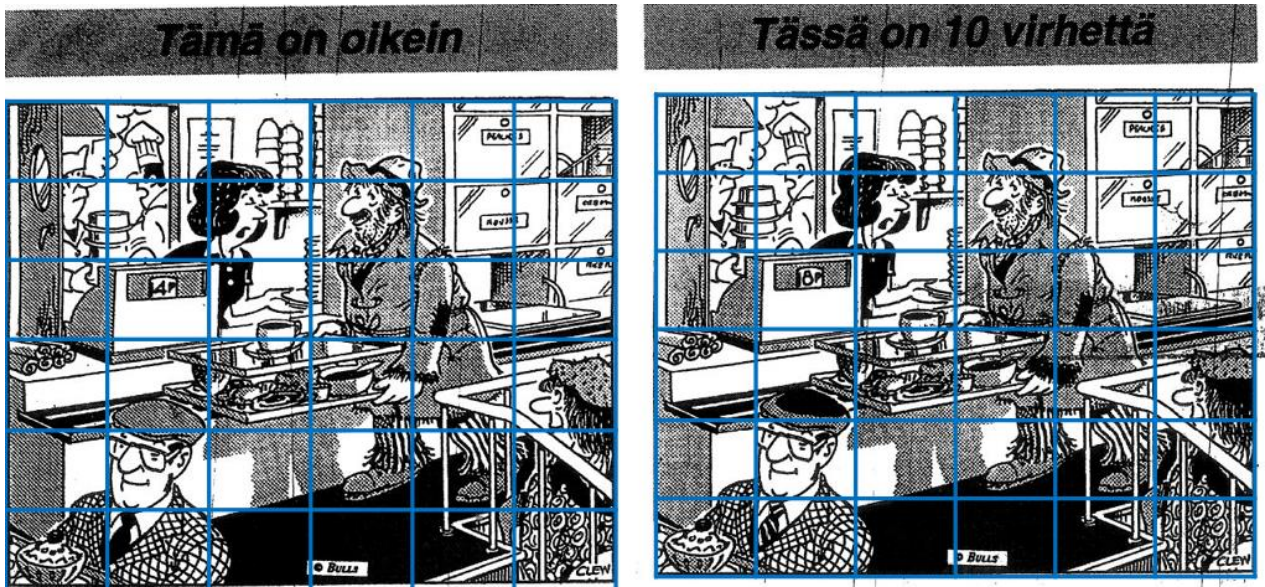


Testing

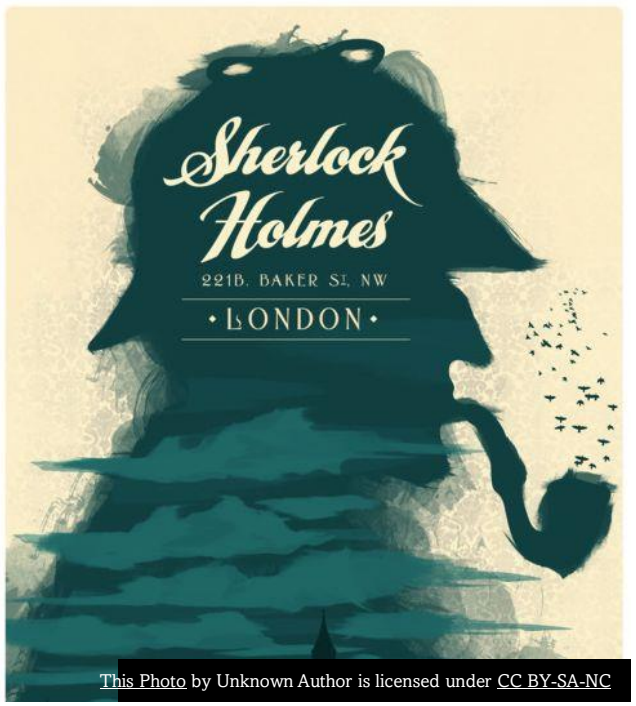
- Information seeking
- Max: Coverage
- Max: Oracle accuracy
- Minimize: Cost

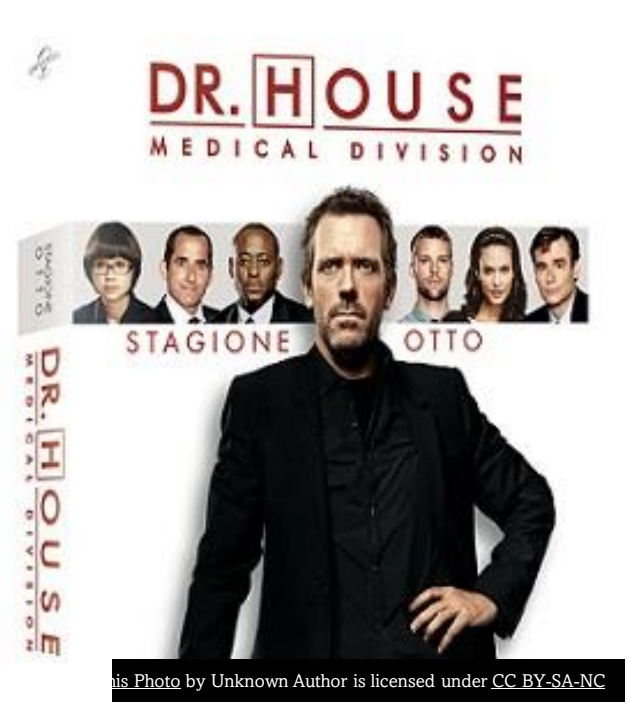


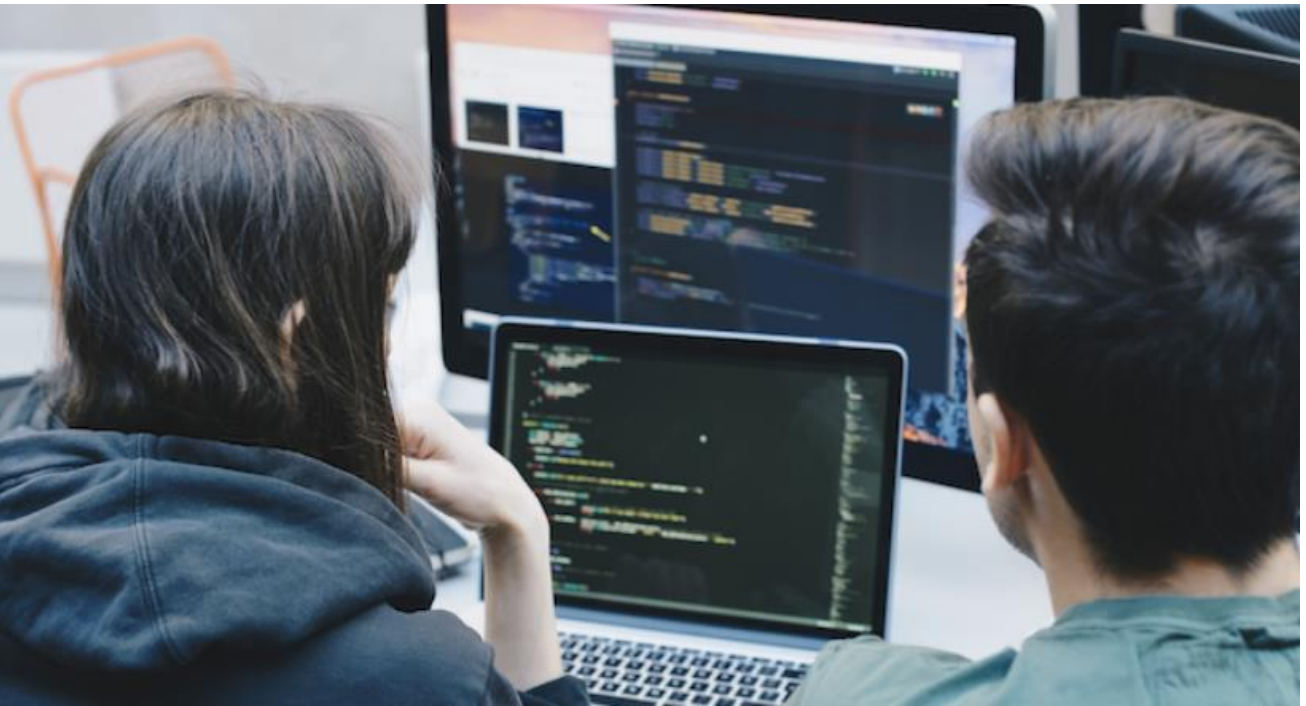




LOG ANALYSIS







Log Analysis Investigation of Software Behavior

- A software engineer investigating software behavior is akin to…
  - a medical doctor investigating a patient
  - a detective investigating a crime
- Software behavior = What happens in Testing or Operations

Types of Log data

- Execution logs
  - Textual
  - Whatever the developer happened to log
  - Series of events
- Metrics (CPU, Memory etc)
  - Series continuous values
- Traces
  - Tree of services of request and messages
  - Microservices: Requests and messages sent between microservices as they fulfill a user request
  - Programs: Record of the execution of a program captured by a debugger or a profiling tool.

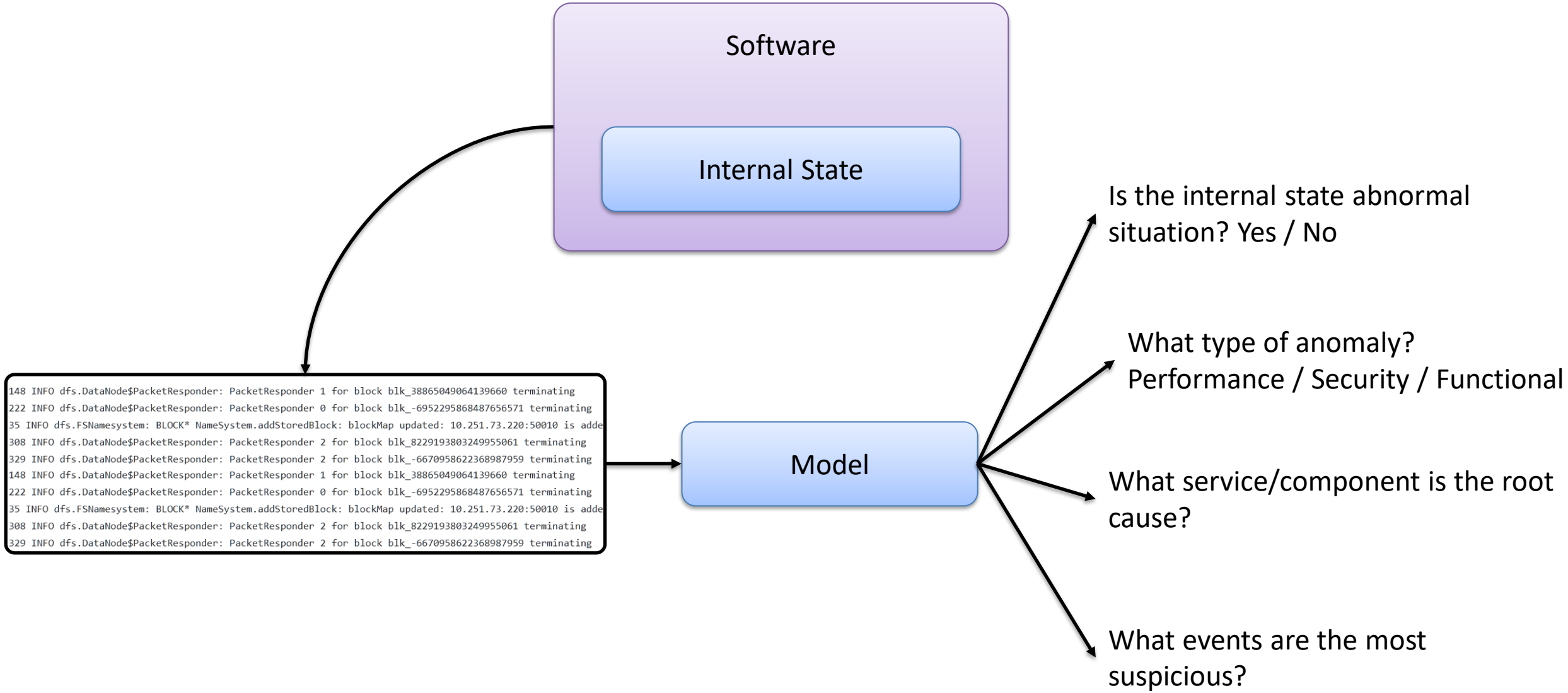
Execution Logs in code

```python
import logging

# Create a logger for the current module
logger = logging.getLogger(__name__)
# Configure the logger
logger.setLevel(logging.DEBUG)  # Set the minimum log level
# Create a console handler
console_handler = logging.StreamHandler()
# Set level and format for the handler
formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s',
console_handler.setFormatter(formatter)
# Add the handler to the logger
logger.addHandler(console_handler)

# Example usage of the logger
logger.debug("This is a debug message")
logger.info("This is an info message")
logger.warning("This is a warning message")
logger.error("This is an error message")
logger.critical("This is a critical message")
```

Software Log Analysis - Objectives

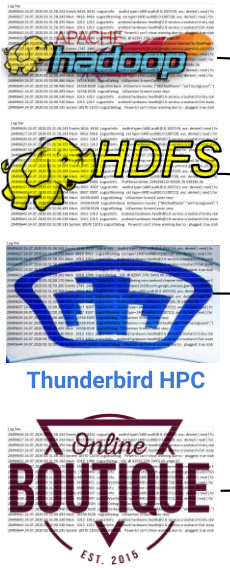


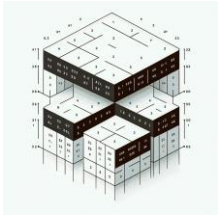
Log Processing Pipeline

Load and process to common format

Add log representations

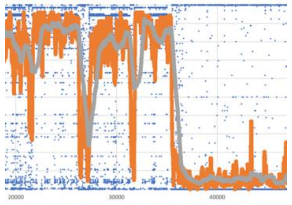
Detect anomalies





Time	Level	Message
2024-03-07 9:56:28	Info	Connection opened to 192.168.0.1
2024-03-07 9:57:28	Info	Reading data from 192.168.0.2

Char-3-grams	Cluster
[Con onn nne … 8.0 .0. 0.1]	E1
[Rea ead adi … 8.0 .0. 0.2]	E2

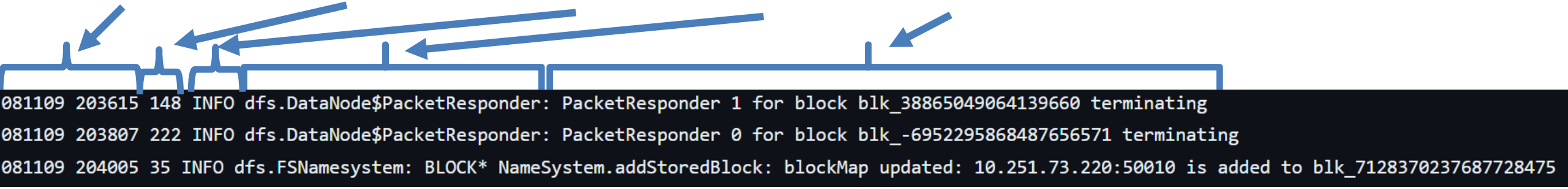


Anomaly	Ano score
0	0.02
1	0.95

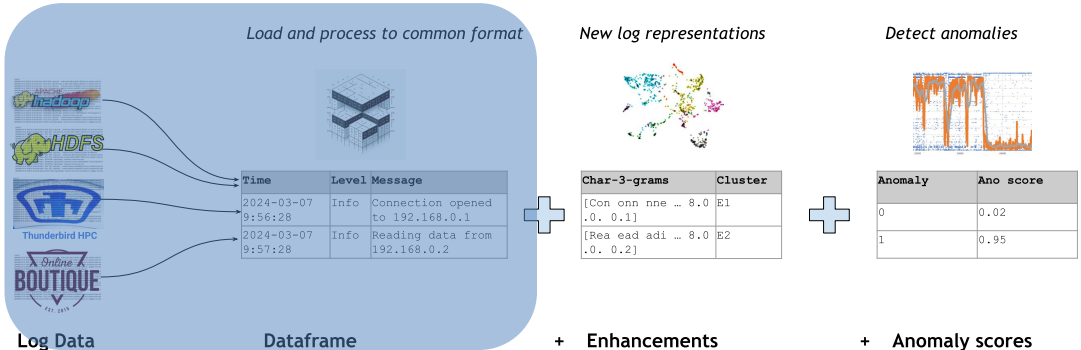
Log Data **Dataframe** **+ Enhancements** **+ Anomaly scores**

Load & Process

- Slice different parts to different variables
  - Time stamp, thread id, log level, component, log message



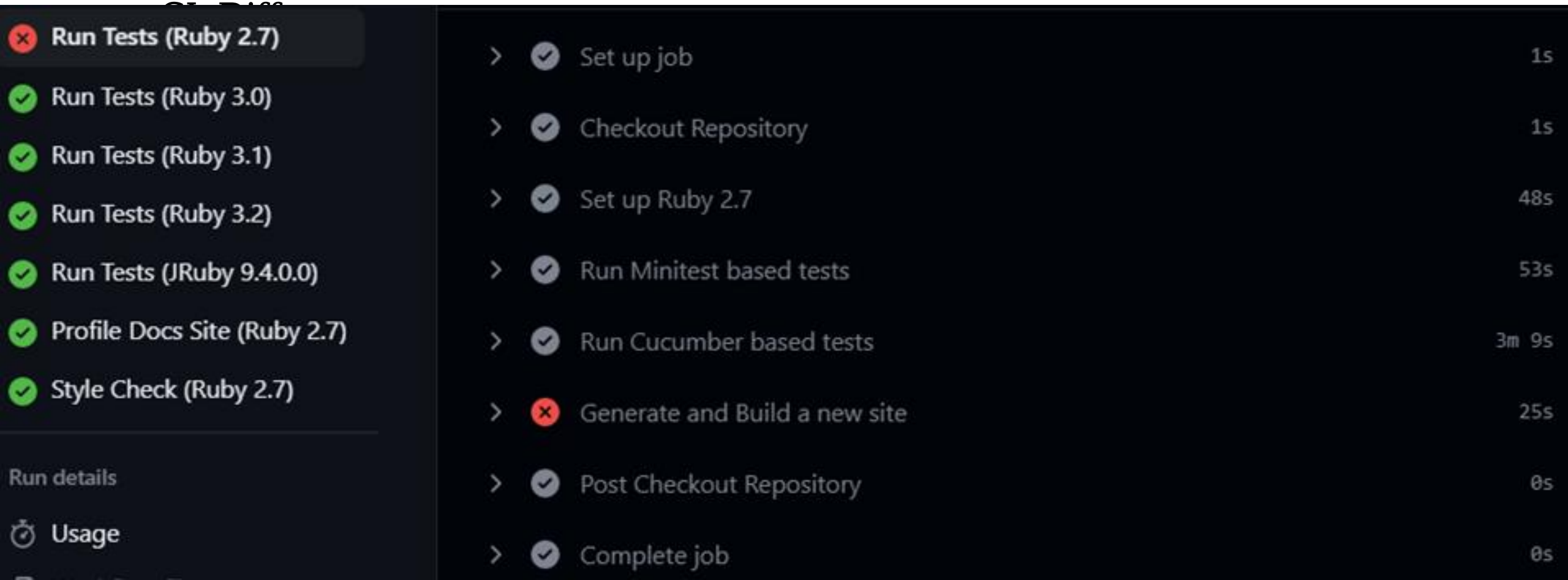
```
081109 203615 148 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_38865049064139660 terminating
081109 203807 222 INFO dfs.DataNode$PacketResponder: PacketResponder 0 for block blk_-6952295868487656571 terminating
081109 204005 35 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.73.220:50010 is added to blk_7128370237687728475
```



Load & Process

- Separate logs to correct sequences
  - Log has phases: separate them -> different model for each step

https://github.com/jekyll/jekyll/actions/runs/4915665388/jobs/8778424596



Load & Process

- Separate logs to correct sequences
  - Log has phases: separate them
    - CI: Different steps
  - Multiple threads push to single log file: separate them
    - HDFS log data: Block ID
    - HDFS log data : 10M log lines and 500k sequences

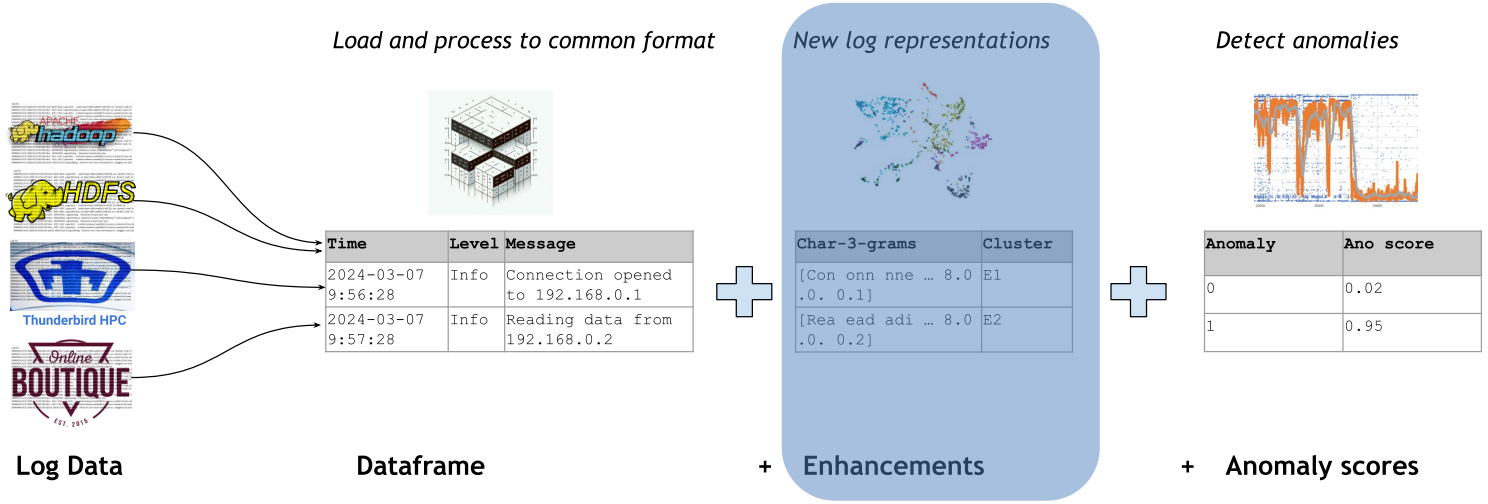
```
081109 203615 148 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_38865049064139660 terminating
081109 203807 222 INFO dfs.DataNode$PacketResponder: PacketResponder 0 for block blk_-6952295868487656571 terminating
081109 204005 35 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.73.220:50010 is added to blk_7128370237687728475
```

Load & Process

- Separate logs to correct sequences
  - Log has phases: separate them
    - CI: Different steps
  - Multiple threads push to single log file: separate them
    - HDFS log data: Block ID
  - Log has different tasks: separate them
    - Test automation: Test cases

Enhance Logs

- Choose appropriate Log representation, e.g.
  - Message length
  - Sequence duration
  - Character 3grams
  - Regex
    - E.g., Normalize log message




```
081109 203615 148 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_38865049064139660 terminating
081109 203807 222 INFO dfs.DataNode$PacketResponder: PacketResponder 0 for block blk_-6952295868487656571 terminating
081109 204005 35 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.73.220:50010 is added to blk_7128370237687728475
```

```
081109 203615 148 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block <BLK> terminating
081109 203807 222 INFO dfs.DataNode$PacketResponder: PacketResponder 0 for block <BLK> terminating
081109 204005 35 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: <IP> is added to <BLK>
```

Enhance Logs: Log Parsing or Log Clustering

- Many (~20) log parsers exist
  - Research field in itself

Tools and Benchmarks for Automated Log Parsing

Jieming Zhu[¶], Shilin He[†], Jinyang Liu[‡], Pinjia He[§], Qi Xie[‖], Zibin Zheng[‡], Michael R. Lyu[†]

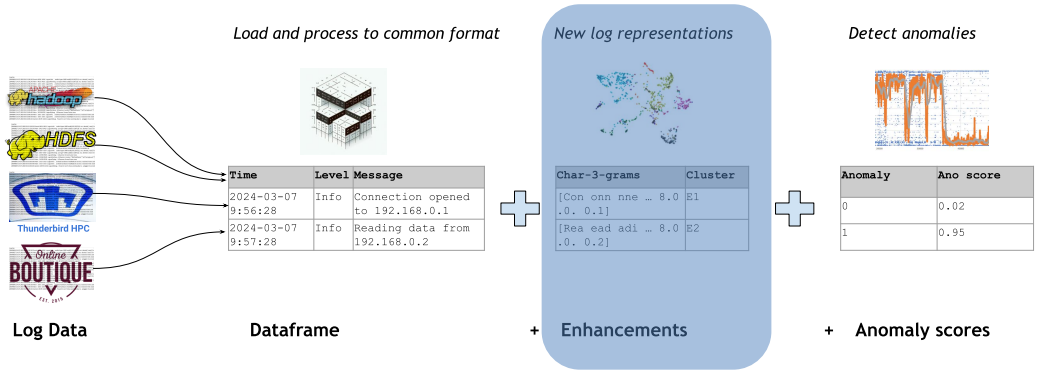
[¶]Huawei Noah's Ark Lab, Shenzhen, China
[†]Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong
[‡]School of Data and Computer Science, Sun Yat-Sen University, Guangzhou, China
[§]Department of Computer Science, ETH Zurich, Switzerland
[‖]School of Computer Science and Technology, Southwest Minzu University, Chengdu, China

jmzhu@ieee.org, slhe@cse.cuhk.edu.hk, liujy@logpai.com, pinjiahe@gmail.com
qi.xie.swun@gmail.com, zhzibin@mail.sysu.edu.cn, lyu@cse.cuhk.edu.hk

Log Parser	Year	Technique	Mode	Efficiency	C
SLCT	2003	Frequent pattern mining	Offline	High	
AEL	2008	Heuristics	Offline	High	
IPLoM	2012	Iterative partitioning	Offline	High	
LKE	2009	Clustering	Offline	Low	
LFA	2010	Frequent pattern mining	Offline	High	
LogSig	2011	Clustering	Offline	Medium	
SHISO	2013	Clustering	Online	High	
LogCluster	2015	Frequent pattern mining	Offline	High	
LenMa	2016	Clustering	Online	Medium	
LogMine	2016	Clustering	Offline	Medium	
Spell	2016	Longest common subsequence	Online	High	
Drain	2017	Parsing tree	Online	High	
MoLFI	2018	Evolutionary algorithms	Offline	Low	

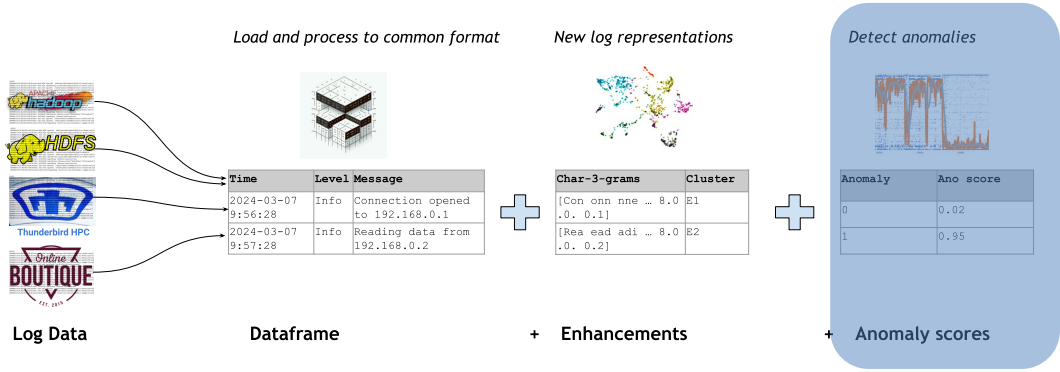
Enhance Logs : Log Parsing or Log Clustering

- Separate fixed part (template) from variable part (parameter)
- `Connection opened to 192.168.0.1`
  - Fixed (template) part: `Connection opened to <*>`
  - Variable (parameter): `192.168.0.1`
- Turns stream of messages to stream events
- Benefits
  - Simplifies analysis
  - Enables next event prediction, state machines, look-ahead pairs
    - NEP: E1 E2 E4 -> ?
- Drawbacks
  - Takes times
  - Can reduce anomaly prediction accuracy
  - Parameters get lost



Log Message	Cluster
Connection opened to 192.168.0.1	E1
Reading data from 192.168.0.1	E2
Connection opened to 192.168.15.1	E1
Connection closed 192.168.0.1	E3

Anomaly Detection

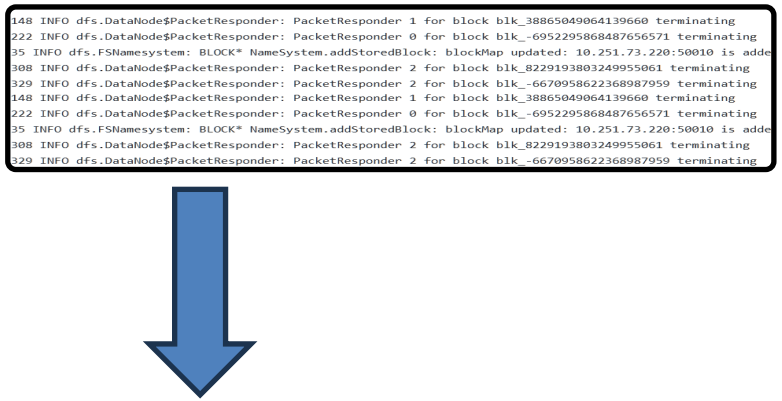


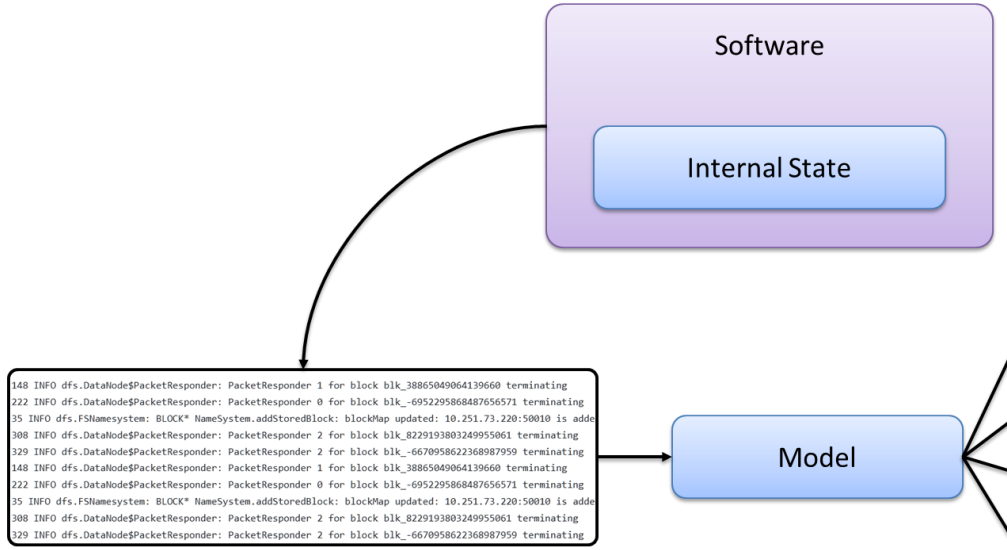
- Columns: Log Representations
- Rows: ML algos:
  - DT – Decision Tree, SVM – Support Vector Machine, LR – Logistic Regression, RF – Random Forrest, XGB – Extrement Gradient Booosting

ANOMALY DETECTION F1-BINARY TRAINED ON 0.5% SUBSET OF HDFS DATA.

	Words	Drain	Lenma	Spell	Bert	Average
DT	0.9719	0.9816	0.9803	0.9828	0.9301	0.9693
SVM	0.9568	0.9591	0.9605	0.9559	0.8569	0.9378
LR	0.9476	0.8879	0.8900	0.9233	0.5841	0.8466
RF	0.9717	0.9749	0.9668	0.9809	0.9382	0.9665
XGB	0.9721	0.9482	0.9492	0.9535	0.9408	0.9528
Average	0.9640	0.9503	0.9494	0.9593	0.8500	

Overview

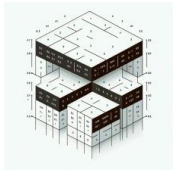




Load and process to common format

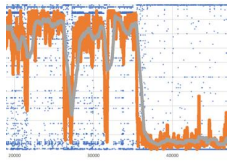
Add log representations

Detect anomalies



Time	Level	Message
2024-03-07 9:56:28	Info	Connection opened to 192.168.0.1
2024-03-07 9:57:28	Info	Reading data from 192.168.0.2

Char-3-grams	Cluster
[Con onn nne … 8.0 .0. 0.1]	E1
[Rea ead adi … 8.0 .0. 0.2]	E2



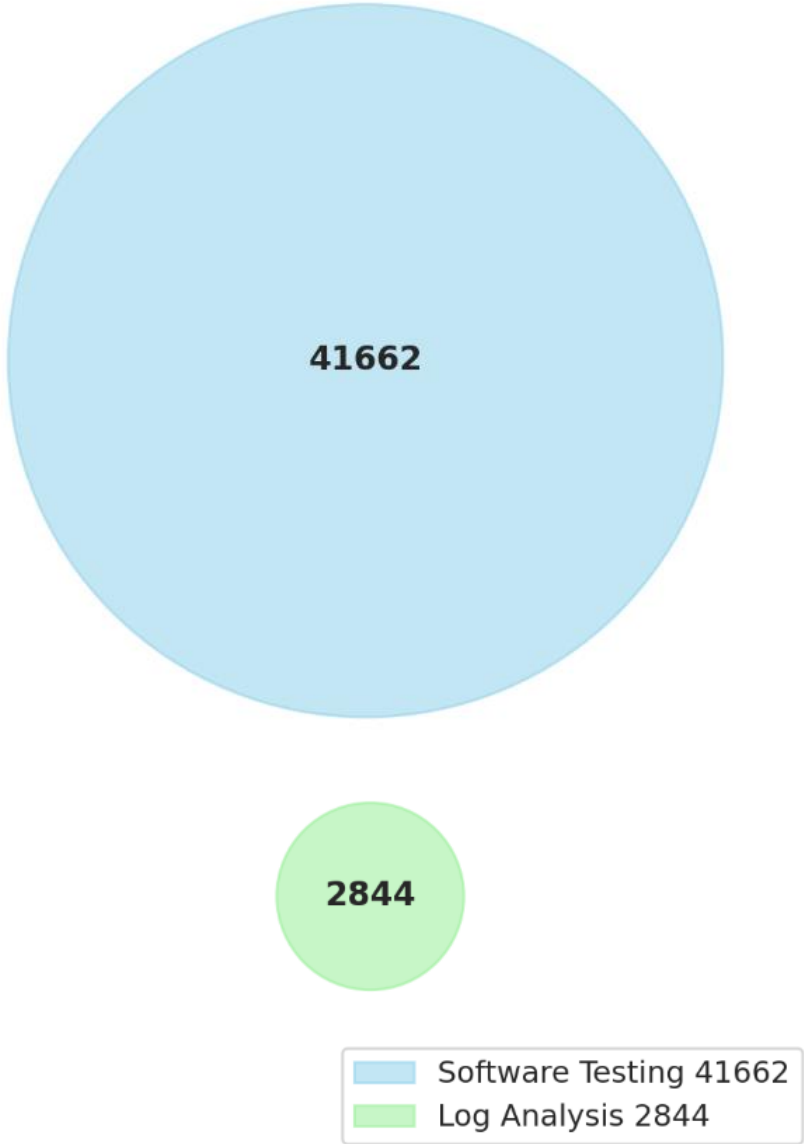
Anomaly	Ano score
0	0.02
1	0.95

Log Data **Dataframe** **+ Enhancements** **+ Anomaly scores**

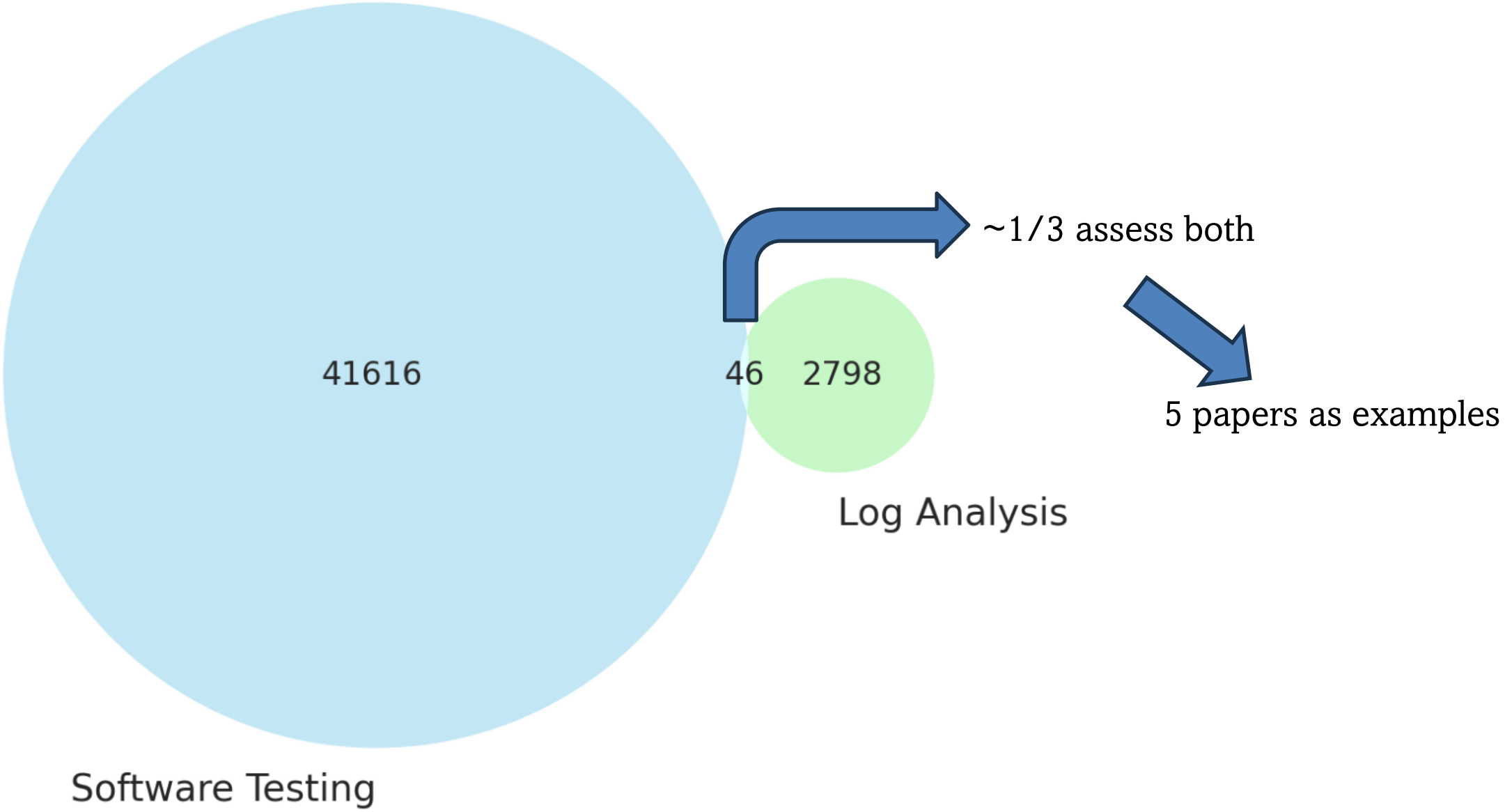
INTERSECTION BETWEEN SOFTWARE TESTING AND LOG ANALYSIS

Both Software Testing and Log Analysis are big fields

- Software Testing 41,662
  - `TITLE-ABS-KEY ( "software testing" ) AND ( LIMIT-TO ( SUBJAREA , "COMP" ) )`
- Log Analysis 2,844
  - `( TITLE-ABS-KEY ( "log analysis" ) OR TITLE-ABS-KEY ( "log anomaly detection" ) OR TITLE-ABS-KEY ( "log file analysis" ) OR TITLE-ABS-KEY ( "log file anomaly detection" ) OR TITLE-ABS-KEY ( "software log" ) OR TITLE-ABS-KEY ( "software execution log" ) ) AND ( LIMIT-TO ( SUBJAREA , "COMP" ) )`

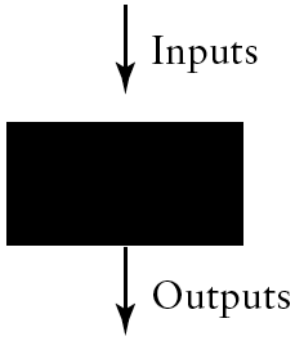
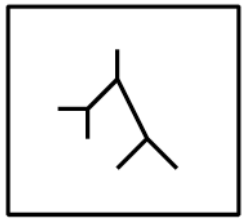


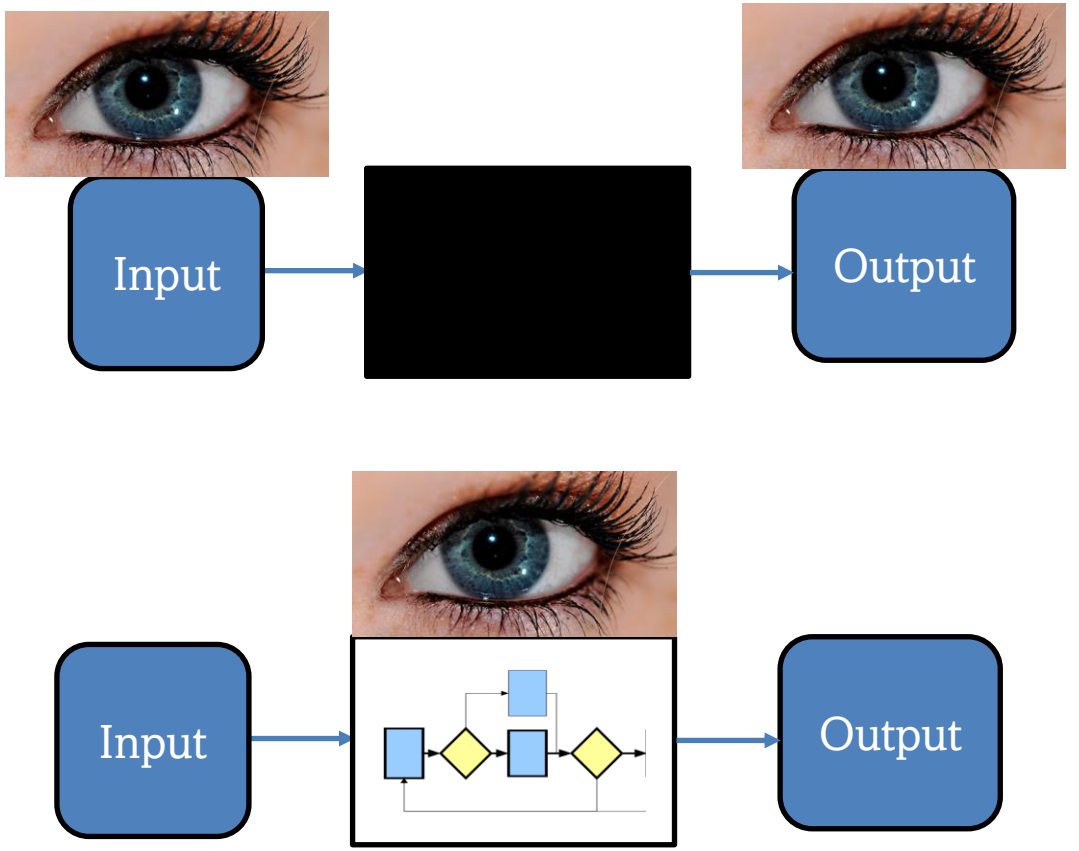
Intersection – Software Testing and Log Analysis



LOG ANALYSIS – COVERAGE

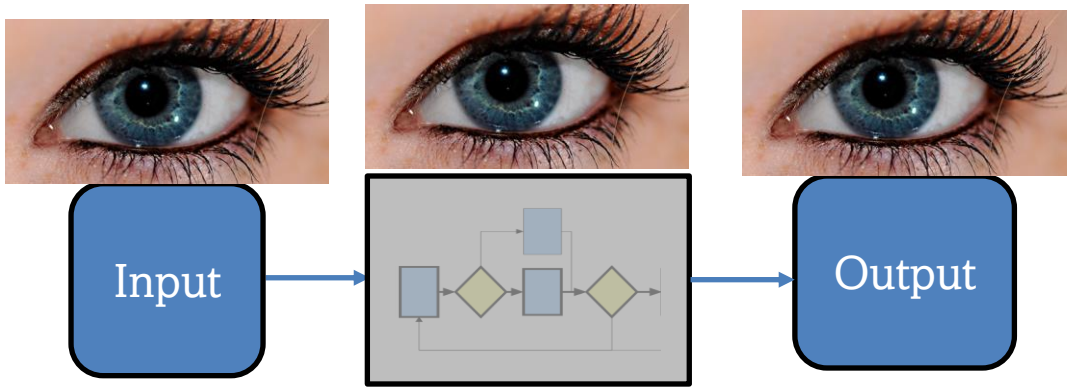
Knowledge sources and methods for testing

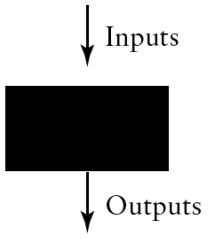
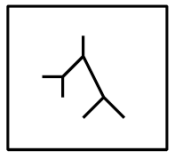
Test Strategy	Tester's View	Knowledge Sources	Methods
Black box		Requirements document Specifications Domain knowledge Defect analysis data	Equivalence class partitioning Boundary value analysis State transition testing Cause and effect graphing Error guessing
White box		High-level design Detailed design Control flow graphs Cyclomatic complexity	Statement testing Branch testing Path testing Data flow testing Mutation testing Loop testing

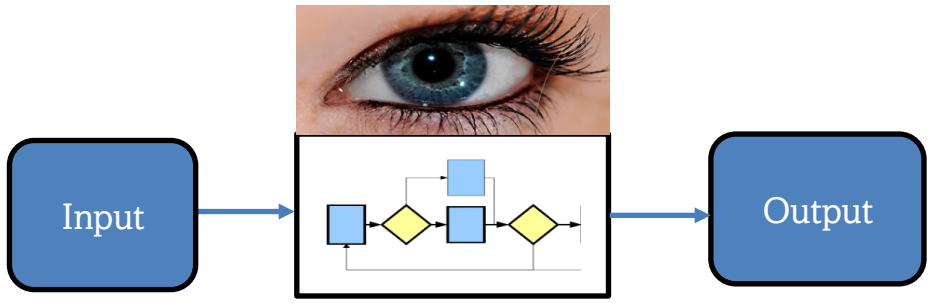


Log analysis: From black to grey box testing

Knowledge source
Execution logs
Traces
Performance metrics



Test Strategy	Tester's View	Knowledge Sources	Methods
Black box		Requirements document Specifications Domain knowledge Defect analysis data	Equivalence class partitioning Boundary value analysis State transition testing Cause and effect graphing Error guessing
White box		High-level design Detailed design Control flow graphs Cyclomatic complexity	Statement testing Branch testing Path testing Data flow testing Mutation testing Loop testing



Logs as Test Coverage Target

- Search-based testing
  - Objectives: max coverage, minimize execution time, generate crash
  - Log objective:
    - Max: unique log statements (=coverage),
    - Min: count of log messages (=cost)
- Assess test suite realism
  - Log objective: Max log message similarity between production and test
  - Reliability and Load testing [1,2]
- Test case prioritization
  - Past work: Diversity of test cases leads to better prioritization
  - Log objective: Max diversity of test logs [3]

[1] Tian X, Li H, Liu F. **Web service reliability test method based on log analysis**. In2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C) 2017 Jul 25 (pp. 195-199). IEEE.
[2] Chen J, Shang W, Hassan AE, Wang Y, Lin J. **An experience report of generating load tests using log-recovered workloads at varying granularities of user behaviour**. In2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE) 2019 Nov 11 (pp. 669-681). IEEE.
[3] Chen Z, Chen J, Wang W, Zhou J, Wang M, Chen X, Zhou S, Wang J**. Exploring better black-Box test case prioritization via log analysis**. ACM Transactions on Software Engineering and Methodology. 2023 Apr 26;32(3):1-32.

LOG ANALYSIS – ORACLE

The Oracle Problem in Software Testing: A Survey

Earl T. Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo

Abstract—Testing involves examining the behaviour of a system in order to discover potential faults. Given an input for a system, the challenge of distinguishing the corresponding desired, correct behaviour from potentially incorrect behavior is called the "test oracle problem". Test oracle automation is important to remove a current bottleneck that inhibits greater overall test automation. Without test oracle automation, the human has to determine whether observed behaviour is correct. The literature on test oracles has introduced techniques for oracle automation, including modelling, specifications, contract-driven development and metamorphic testing. When none of these is completely adequate, the final source of test oracle information remains the human, who may be aware of informal specifications, expectations, norms and domain specific information that provide informal oracle guidance. All forms of test oracles, even the humble human, involve challenges of reducing cost and increasing benefit. This paper provides a comprehensive survey of current approaches to the test oracle problem and an analysis of trends in this important area of software testing research and practice.

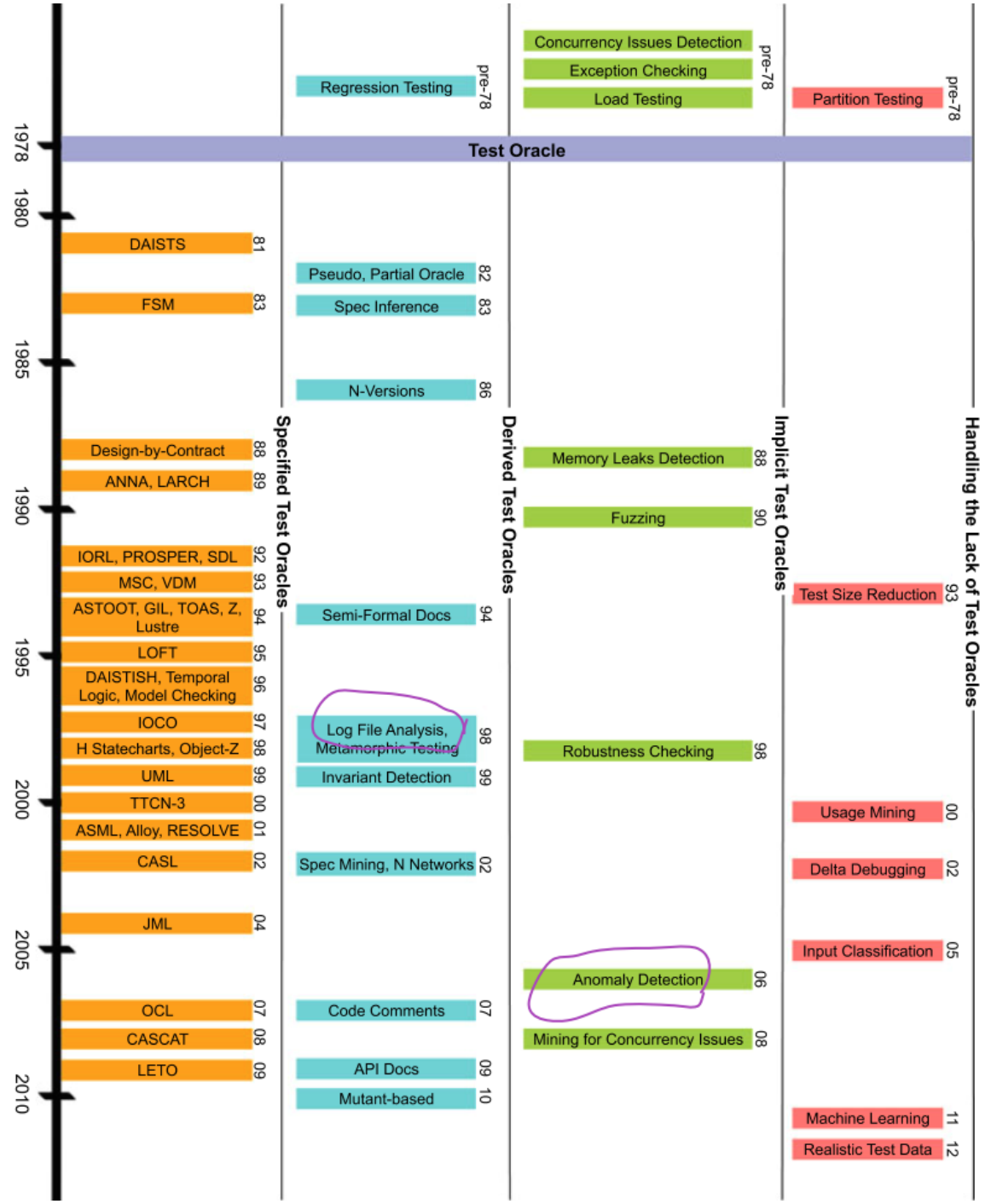
Index Terms—Test oracle, automatic testing, testing formalism

1 Introduction

Much work on software testing seeks to automate as much of the test process as practical and desirable, to make testing faster, cheaper, and more reliable. To this end, we need a test oracle, a procedure that distinguishes between the correct and incorrect behaviors of the System Under Test (SUT).

However, compared to many aspects of test automation, the problem of automating the test oracle has received significantly less attention, and remains comparatively less well-might be a detailed, and possibly formal, specification of intended behaviour. One might also hope that the code itself contains pre- and post- conditions that implement well-understood contract-driven development approaches [135]. In these situations, the test oracle cost problem is ameliorated by the presence of an automatable test oracle to which a testing tool can refer to check outputs, free from the need for costly human intervention.

Where no full specification of the properties of the SUT



Analysis of Test Automation Results -> Better Oracle Granularity

TABLE 21 Approaches to analyse test automation results

Approaches	Description
Interpret and classify test automation results	Analyse failed tests to find the root reason for failure and classify the results to prevent potential incidents.
More than 'pass' or 'fail'	Store and review the artefacts (logs, screenshots, comparisons, or video recordings of test runs, and others generated from executing automated tests) to get complement information for debugging and fixing issues.
Notifications	Set the notifications (on test execution tools) to alarm the failures of critical automated tests, so that the priority can be given to analyse and solve the failures of critical automated tests when receiving the notifications.
Tool support	Use test tools that can give a clear overview of each step of the test flow so that failures can be quickly identified.
Smoke tests	Run smoke tests on automated test suites incrementally to expose reasons for failures.
Big picture	In addition to analyse a single test run results, it is essential to combine test automation results collected from different sources (e.g., across multiple test tools, test runs, configurations, integration builds, and milestones) into a big picture view of outcomes.
Keep history	Store test automation results for a period of time to enable progress tracking, regression identification, and flaky tests identification.

Received: 6 July 2021 | Revised: 22 November 2021 | Accepted: 29 November 2021
DOI: 10.1002/stvr.1804

SURVEY ARTICLE

WILEY

Improving test automation maturity: A multivocal literature review

Yuqing Wang | Mika V. Mäntylä | Zihao Liu | Jouni Markkula | Päivi Raulamo-jurvanen

M3S research unit, University of Oulu, Pentti Kaiteran katu 1, Oulu, Finland

Correspondence
Yuqing Wang, M3S research unit, University of Oulu, Pentti Kaiteran katu 1, Oulu 90014, Finland.
Email: yuqing.wang@oulu.fi

Funding information
TEA3; Tauno Tönningin Säätiö; Business Finland, Grant/Award Number: 3192/31/2017; Tauno Tönning, Grant/Award Number: 20210086

Abstract
Mature test automation is key for achieving software quality at speed. In this paper, we present a multivocal literature review with the objective to survey and synthesize the guidelines given in the literature for improving test automation maturity. We selected and reviewed 81 primary studies, consisting of 26 academic literature and 55 grey literature sources. From primary studies, we extracted 26 test automation best practices (e.g., Define an effective test automation strategy, Set up good test environments, and Develop high-quality test scripts) and collected many pieces of advice (e.g., in forms of implementation/improvement approaches, technical techniques, concepts, and experience-based heuristics) on how to conduct these best practices. We made main observations: (1) There are only six best practices whose positive effect on maturity improvement have been evaluated by academic studies using formal empirical methods; (2) several technical related best practices in this MLR were not presented in test maturity models; (3) some best practices can be linked to success factors and maturity impediments proposed by other scholars; (4) most pieces of advice on how to conduct proposed best practices were identified from experience studies and their effectiveness need to be further evaluated with cross-site empirical evidence using formal empirical methods; (5) in the literature, some advice on how to conduct certain best practices are conflicting, and some advice on how to conduct certain best practices still need further qualitative analysis.

KEYWORDS
improvement, maturity, practice, software, systematic literature review, test automation

Review 81 sources (26 academic, 55 grey sources)

Logs as partial Oracles or more granular Oracles

- Partial Oracle - State machine for logs [1]
  - Developers explicitly include commands to log events of interest.
  - Test oracles simulate execution of each individual state machine reacting to only the logged events relevant to that state machine
- Granular Oracle - Next event prediction on Logs [2]
  - Existing oracle tells that a long reliability test run failed
    - 40-80k log lines
  - Use next event prediction (from passing testing runs) to score log lines of anomalousness

[1] Andrews JH. **Testing using log file analysis: tools, methods, and issues**. InProceedings 13th IEEE International Conference on Automated Software Engineering (Cat. No. 98EX239) 1998 Oct 13 (pp. 157-166). IEEE.
[2] Mäntylä M, Varela M, Hashemi S. **Pinpointing anomaly events in logs from stability testing – n-grams vs. deep-learning**. In2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW) 2022 Apr 4 (pp. 285-292). IEEE.

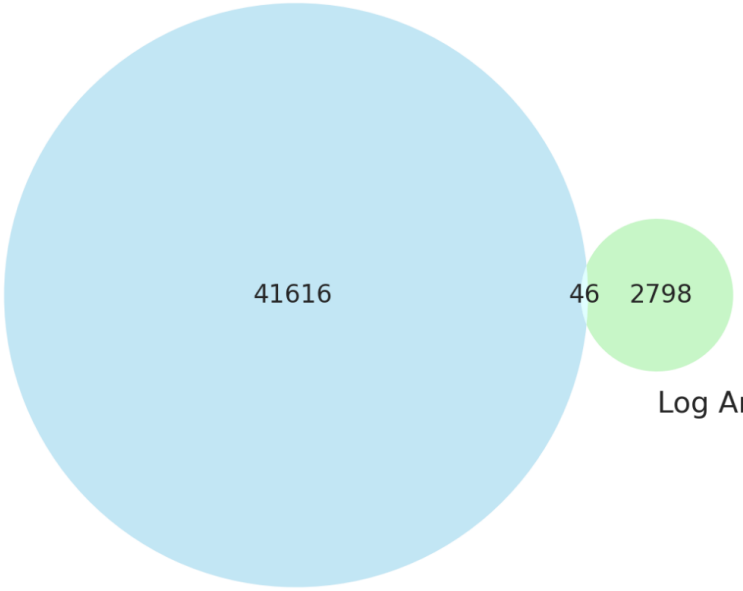
LOG ANALYSIS – TESTING CONCLUSION

Other Future Work Ideas

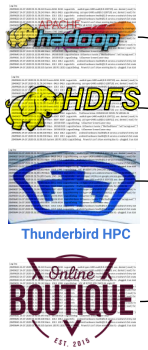
- CI Logs - Failure prediction and analysis
  - Unfortunately, raw CI data of TravisTorrent [1] no longer available
    - Aggregates are not as useful
    - Service providers have limited similar data collection efforts.
  - CI service provides no longer allow data harvesting data
- What about LLMs
  - LLMs are expensive and logs are massive -> Multi-level system
    - LLMs top level lower level classical computing and MLs
- Microservice-based systems / Serverless
  - Microservices offer logs but also traces and metrics collection

[1] Beller, M., Gousios, G., & Zaidman, A. (2017). TravisTorrent: Synthesizing Travis CI and GitHub for FullStack Research on Continuous Integration. In Proceedings - 2017 IEEE/ACM 14th International Conference on Mining Software Repositories, MSR 2017 (pp. 447--450). IEEE . https://doi.org/10.1109/MSR.2017.24

Prof. Mika Mäntylä mika.mantyla@helsinki
University of Helsinki, Finland
https://github.com/EvoTestOps/LogLead



Log Analysis



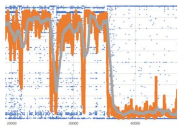
Load and process to common format | *Add log representations* | *Detect anomalies*

Time	Level	Message
2024-03-07 9:56:28	Info	Connection opened to 192.168.0.1
2024-03-07 9:57:28	Info	Reading data from 192.168.0.2

+

Char-3-grams	Cluster
[Con onn nne … 8.0 .0. 0.1]	E1
[Rea ead adi … 8.0 .0. 0.2]	E2

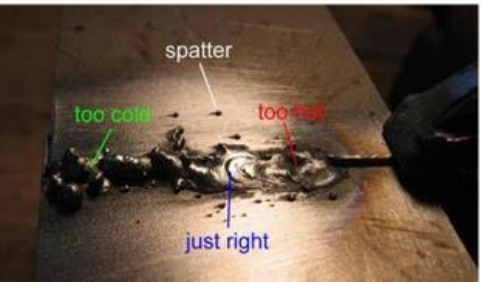
+

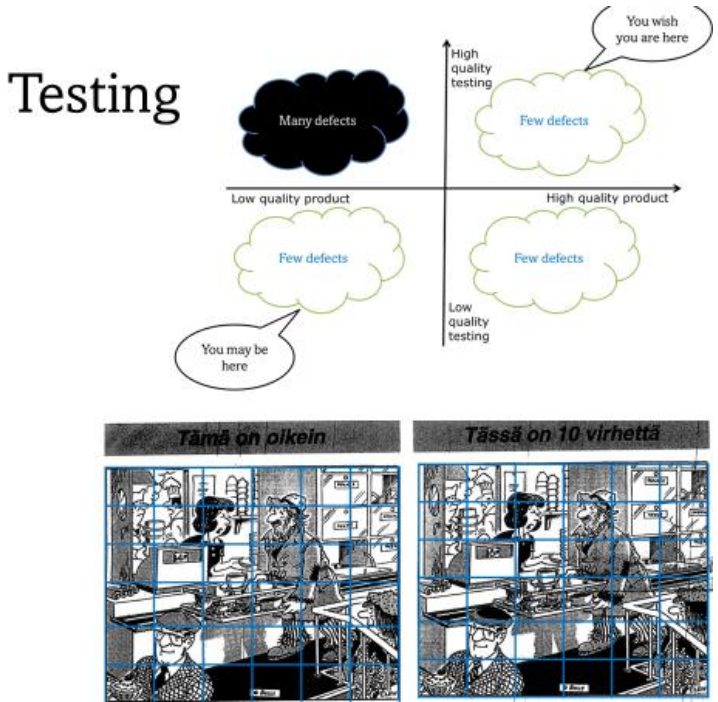


Anomaly	Ano score
0	0.02
1	0.95

Log Data **Dataframe** + **Enhancements** + **Anomaly scores**

- Information seeking
- Max: Coverage
- Max: Oracle accuracy
- Minimize: Cost





Ideas on using Logs for Coverage and Oracles

- Coverage:
  - Search-based testing
    - Log objective: Max unique log statements (coverage), Min count of log message (cost)
  - Assess Test Suite Realism
    - Log objective: Max log message similarity between production and test
  - Test case prioritization
    - Log objective: Max diversity of test logs
- Oracle:
  - Partial Oracle: State machine for logs
  - Oracle Granularity improvement: Next event prediction on Logs