Empowered by Innovation NEC

Proposal of “Ask Why” Framework to Analyze Defect Root Causes

November, 2, 2011

NEC

Takeshi Mori, Ryou Kurashita, Naomi Honda

Outline

1.Introduction

Business Domains of NEC

IT/Network Solutions Business

System Integration Services

Computer Platforms

Network System

Social Infrastructure

Mobile/Personal Solutions Business

Personal Solutions

Mobile Solutions

Semiconductor Solutions

Electronic Components and others

Electron Devices Business

Software products in the Computer Platforms domain

System departments

My responsibility is the QA in middleware domain

General-purpose software

Hardware departments

Application software
Middleware **Server Manager, Thin client Manager, Storage Manager, Network Monitoring..**
OS SX, ACOS, UNIX, Windows, Linux
Server Storage network

Preventive action for released products

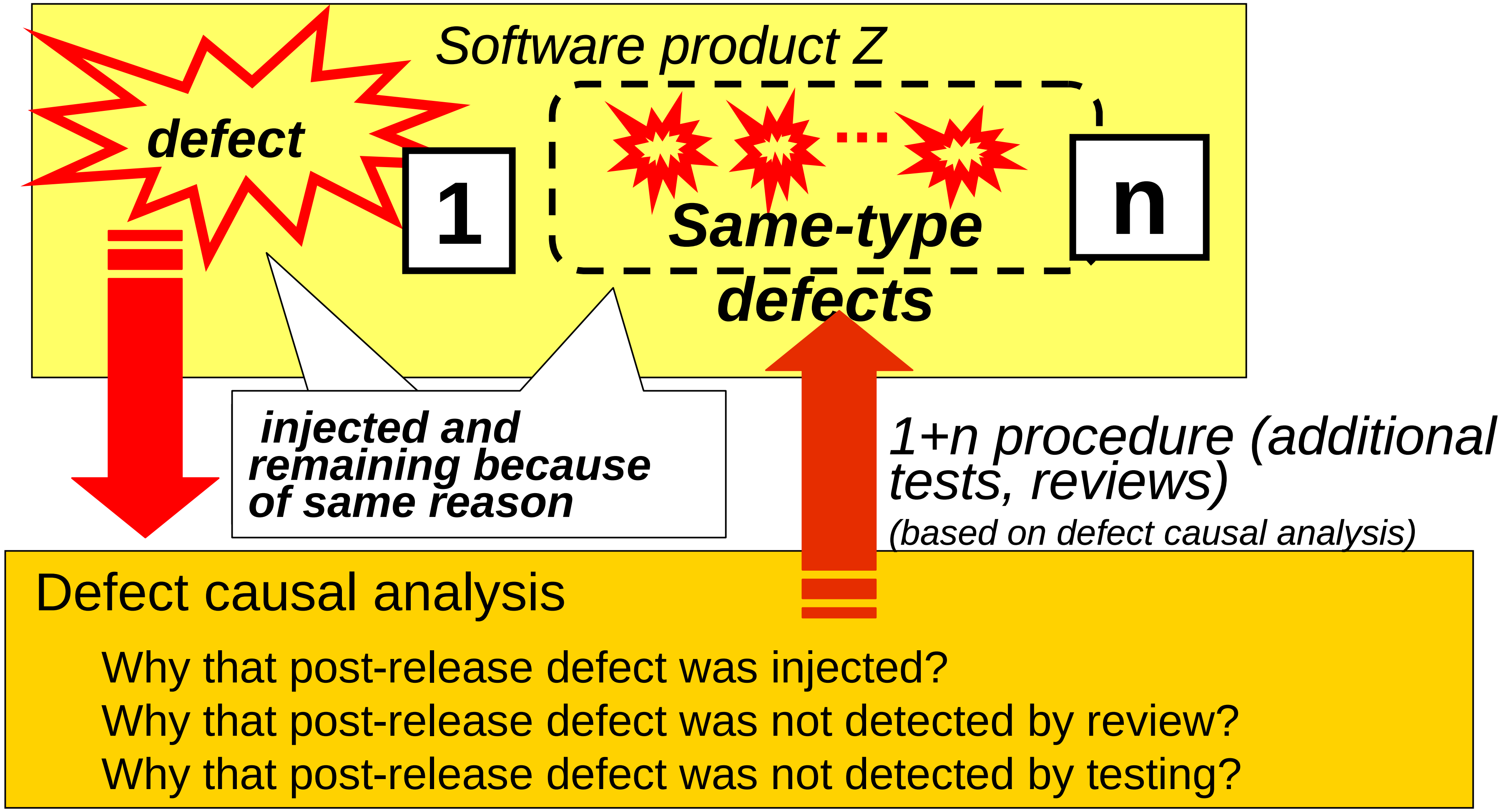
<Purpose>

Detect any remaining same-type defects from the released products when one field defect occurs

Same-type defects

Remaining defects which cause is the same with the field defect

1+n procedure


Software product Z
defect
1
…
Same-type defects
n
injected and remaining because of same reason
1+n procedure (additional tests, reviews)
(based on defect causal analysis)

Defect causal analysis

Why that post-release defect was injected?
Why that post-release defect was not detected by review?
Why that post-release defect was not detected by testing?

2. Our Problem

We use ”Ask Why” analysis method, when we analyze the cause

”Ask Why” analysis method

-> Analyze the root cause by asking why repeatedly

Our Problem

Cause analysis depends on individual skills

-> There are many cases in which the root cause cannot be found

Typical failure example of the "Ask Why" analysis

It is misunderstand that the defect was injected in the coding phase although it was injected in the design phase.

: because the code is eventually modified

We get a wrong viewpoint and a wrong object

Example: Root cause is to mistake in detailed design phase

⇒ Object of addional review

× Source Code

○ Detailed design specification document

We can not detect same-type defects

To solve our problem

The standard analysis framework could be defined
if the purpose of analysis was limited to 1+n procedure

Experience reviewing many analysis reports

Systematization

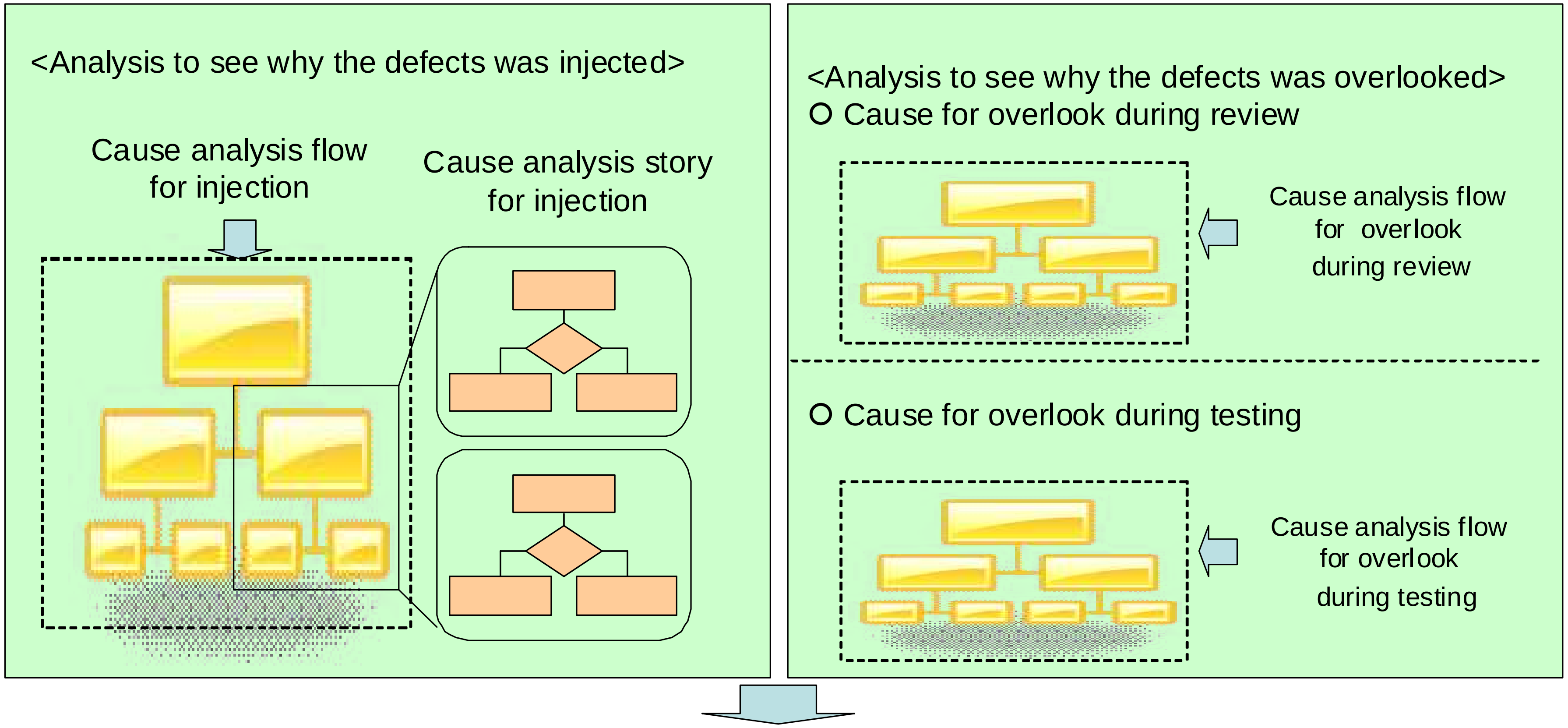
Classification

Defining a standard analysis framework

Principle of the framework

- Consider "Why was the defect injected?" and "Why was it not detected by review or testing?" separately
- We should pay attention to the process in which defects were injected rather than human errors as much as possible
- The framework provides decision branches to lead the analysis

3. "Ask Why" framework



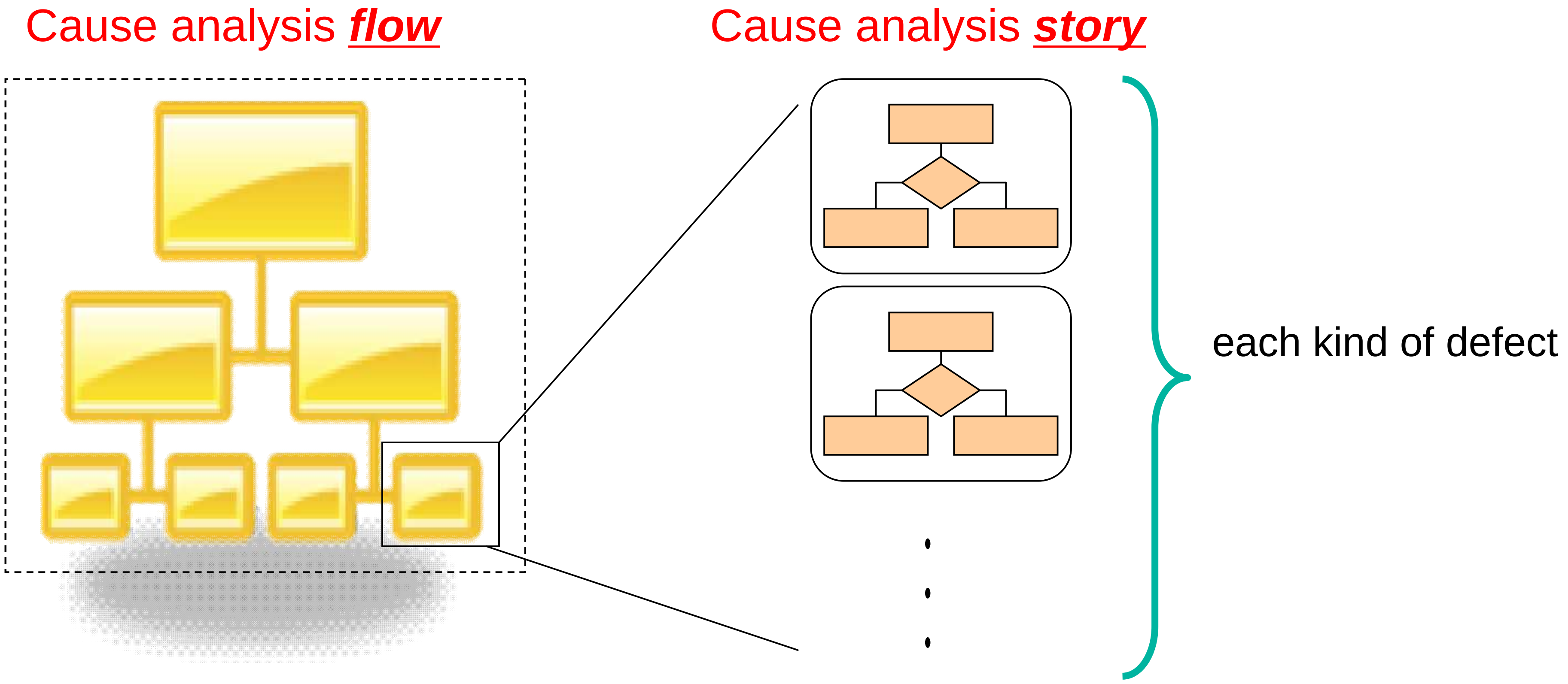
<Analysis to see why the defects was injected>
Cause analysis flow for injection
Cause analysis story for injection
<Analysis to see why the defects was overlooked>
O Cause for overlook during review
Cause analysis flow for overlook during review
O Cause for overlook during testing
Cause analysis flow for overlook during testing

1+n Procedure (Detect the same types of defects in released products)

Feedback to processes (Improvement for next development)

Cause analysis ***story*** for injection

- A par of "Cause analysis ***flow***"
- Defining the analysis process of each kind of defect ("the flow" does not depend on the type of defects)



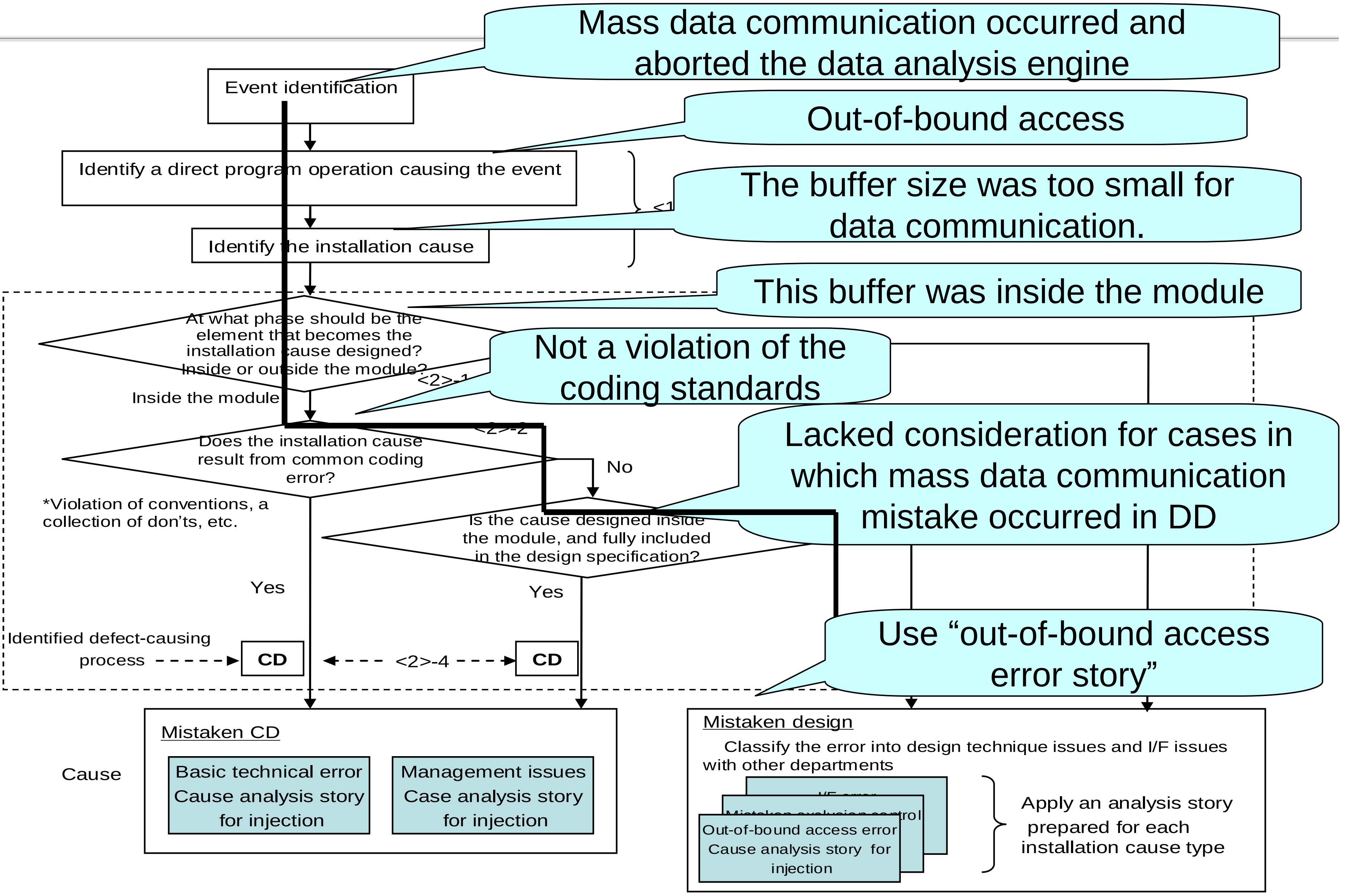
Case Study

Specific case

Mass data communication occurred
and aborted the data analysis engine

The engine must not abort even if Mass data communication occurred.

3.1 Cause analysis flow for injection

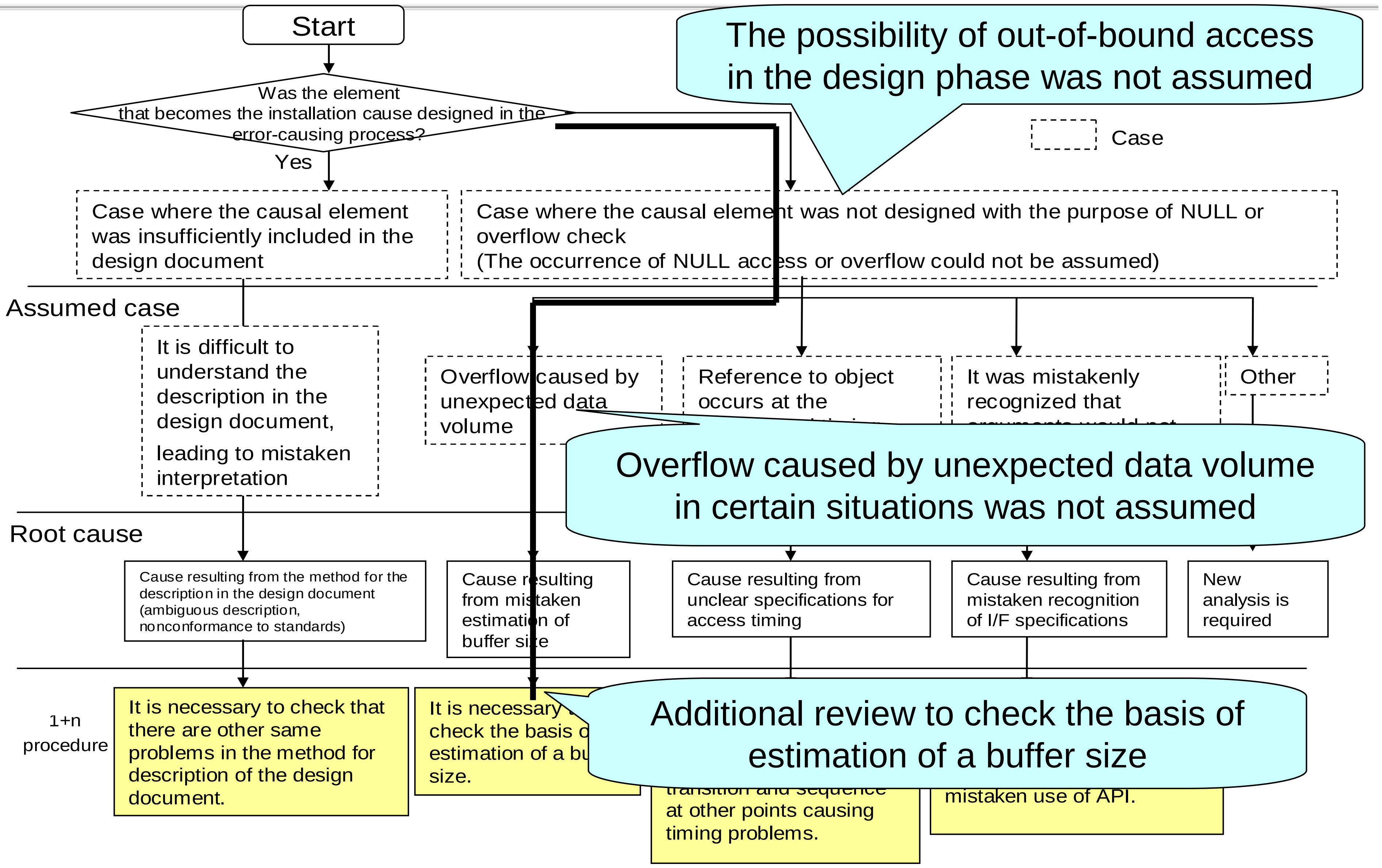

Mass data communication occurred and aborted the data analysis engine
Out-of-bound access
The buffer size was too small for data communication.
This buffer was inside the module
Not a violation of the coding standards
Lacked consideration for cases in which mass data communication mistake occurred in DD
Use “out-of-bound access error story”
Event identification
Identify a direct program operation causing the event
<1
Identify the installation cause
At what phase should the element that becomes the installation cause designed? Inside or outside the module?
<2>-1
Inside the module
<2>-2
Does the installation cause result from common coding error?
No
*Violation of conventions, a collection of don’ts, etc.
Is the cause designed inside the module, and fully included in the design specification?
Yes
Yes
Identified defect-causing process
CD
<2>-4
CD
Cause
Mistaken CD
Basic technical error Cause analysis story for injection
Management issues Case analysis story for injection
Mistaken design
Classify the error into design technique issues and I/F issues with other departments
I/F error
Mistaken exclusion control
Out-of-bound access error Cause analysis story for injection
Apply an analysis story prepared for each installation cause type

Cause analysis Story for injection

Error Type	Description (Installation cause)
Out-of-bound access error	Access to areas beyond buffer space or NULL access
I/F error	Recognition error of argument, return value and function specifications about function I/F
Exclusion control error	Resource exclusion control error in multi-process (thread) environment
Mistaken error processing	Mistaken error processing, omission of error processing
Threshold value/boundary value error	Mistaken processing of threshold value/boundary value
Startup/termination error	Mistaken processing of startup/termination at startup/termination of AP and server
String operation error	Mistaken processing of special character or Japanese character codes
Resource release error	Omission of resource release
Mistaken processing of unexpected cases	Mistaken processing of cases where unexpected message/object arrives

（In our organization, we found these 9 types of stories covered almost 75 % of the analyses）

Cause analysis story for injection of out-of-bound access error

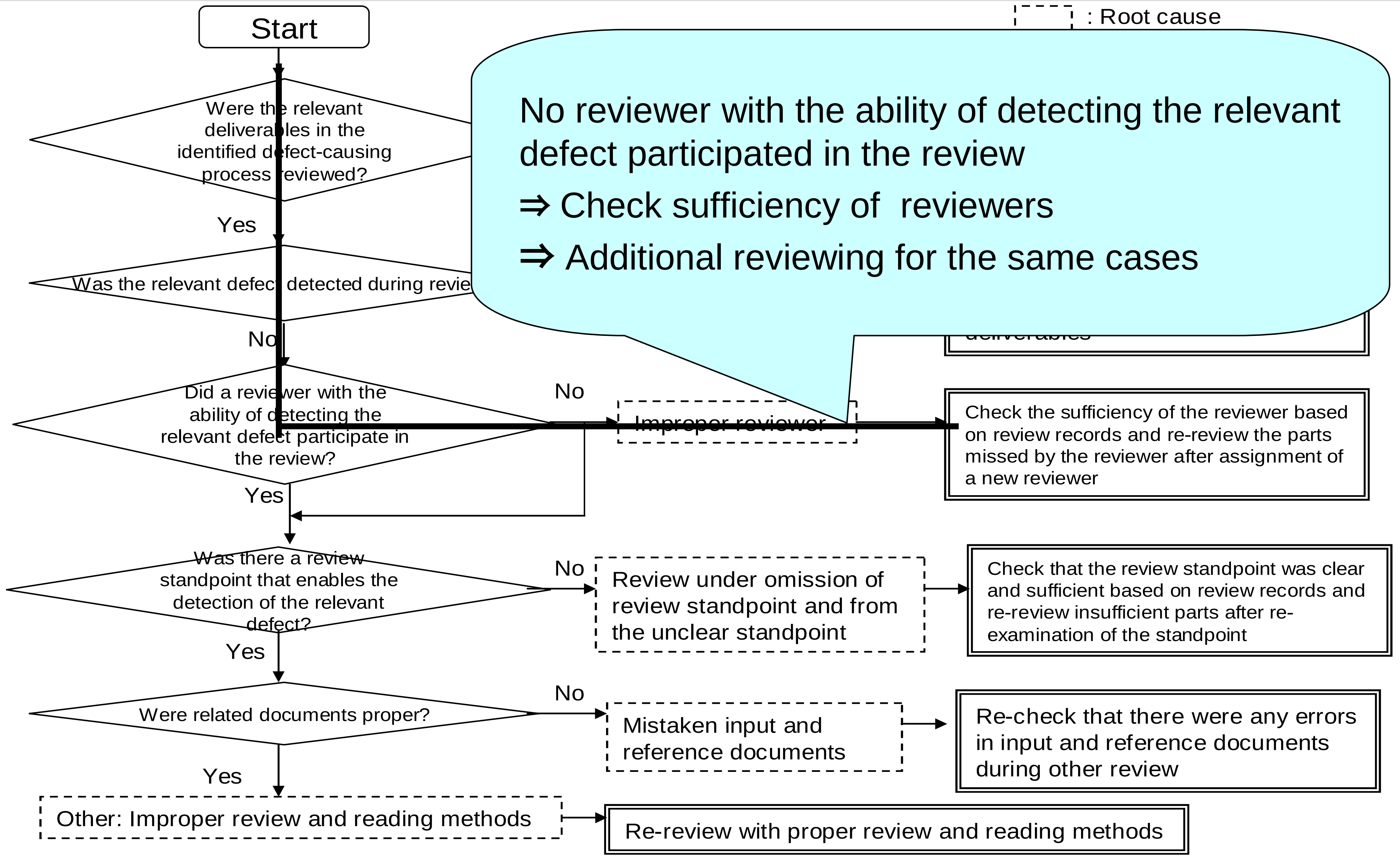


Start
Was the element that becomes the installation cause designed in the error-causing process?
Yes
The possibility of out-of-bound access in the design phase was not assumed
Case
Case where the causal element was insufficiently included in the design document
Case where the causal element was not designed with the purpose of NULL or overflow check (The occurrence of NULL access or overflow could not be assumed)
Assumed case
It is difficult to understand the description in the design document, leading to mistaken interpretation
Overflow caused by unexpected data volume
Reference to object occurs at the
It was mistakenly recognized that
Other
Overflow caused by unexpected data volume in certain situations was not assumed
Root cause
Cause resulting from the method for the description in the design document (ambiguous description, nonconformance to standards)
Cause resulting from mistaken estimation of buffer size
Cause resulting from unclear specifications for access timing
Cause resulting from mistaken recognition of I/F specifications
New analysis is required
1+n procedure
It is necessary to check that there are other same problems in the method for description of the design document.
It is necessary to check the basis of estimation of a bu size.
Additional review to check the basis of estimation of a buffer size
transition and sequence at other points causing timing problems.
mistaken use of API.

3.2 Cause analysis flow for overlook

We apply the flow to analyze the cause in which they were not able to detect the injected defects during testing and review.

Cause analysis flow for outlook during review



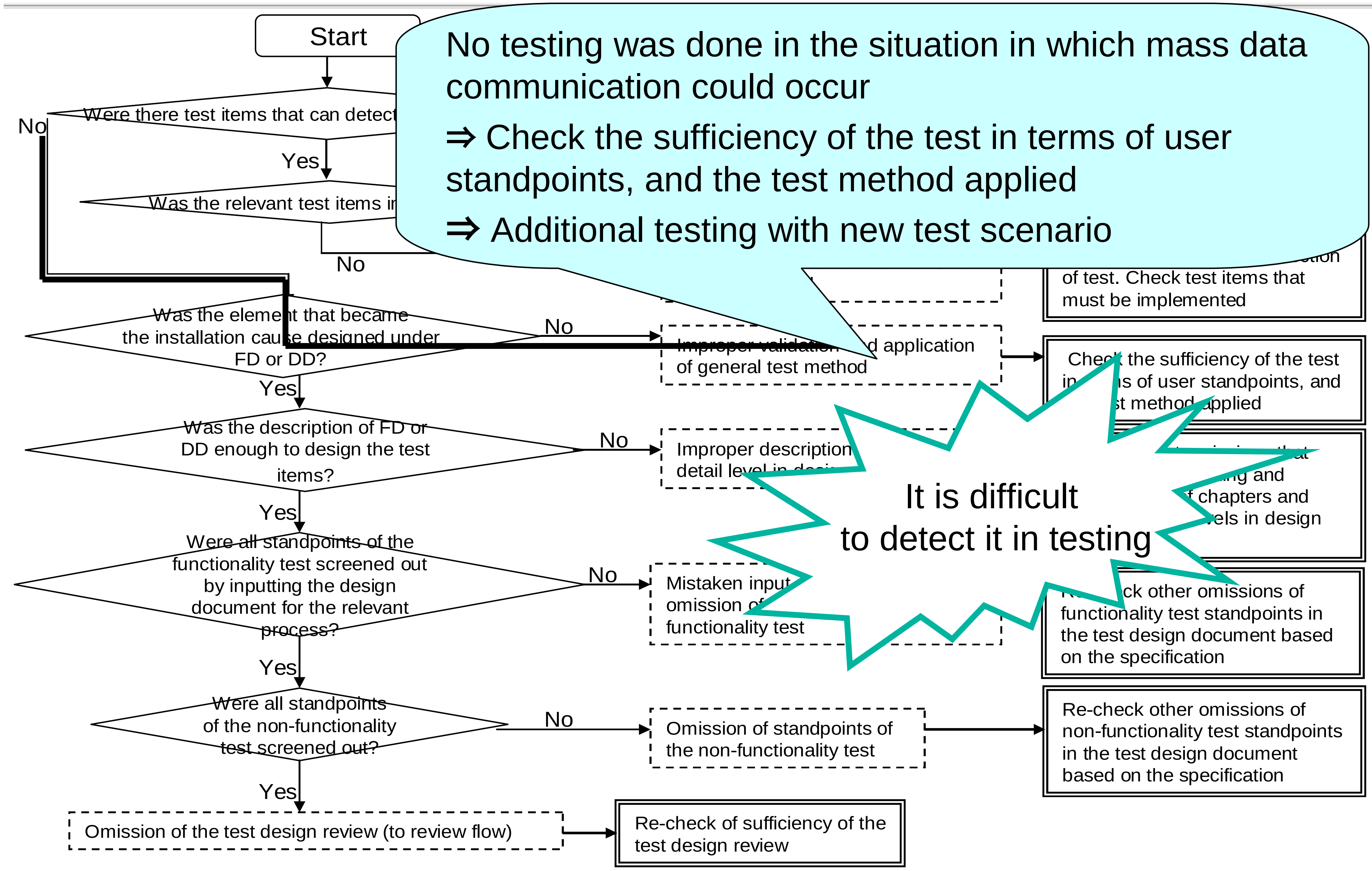
: Root cause
Start
Were the relevant deliverables in the identified defect-causing process reviewed?
Yes
Was the relevant defect detected during revie
No
Did a reviewer with the ability of detecting the relevant defect participate in the review?
No
Improper reviewer
Check the sufficiency of the reviewer based on review records and re-review the parts missed by the reviewer after assignment of a new reviewer
Yes
Was there a review standpoint that enables the detection of the relevant defect?
No
Review under omission of review standpoint and from the unclear standpoint
Check that the review standpoint was clear and sufficient based on review records and re-review insufficient parts after re-examination of the standpoint
Yes
Were related documents proper?
No
Mistaken input and reference documents
Re-check that there were any errors in input and reference documents during other review
Yes
Other: Improper review and reading methods
Re-review with proper review and reading methods

No reviewer with the ability of detecting the relevant defect participated in the review

⇒ Check sufficiency of reviewers

⇒ Additional reviewing for the same cases

Cause analysis flow for outlook during testing


Start
Were there test items that can detect
No
Yes
Was the relevant test items i
No
Was the element that became the installation cause designed under FD or DD?
No
Improper validation ... d application of general test method
Check the sufficiency of the test in ... s of user standpoints, and ... t method applied
of test. Check test items that must be implemented
Yes
Was the description of FD or DD enough to design the test items?
No
Improper description detail level in desig
Yes
Were all standpoints of the functionality test screened out by inputting the design document for the relevant process?
No
Mistaken input omission of functionality test
Re-check other omissions of functionality test standpoints in the test design document based on the specification
Yes
Were all standpoints of the non-functionality test screened out?
No
Omission of standpoints of the non-functionality test
Re-check other omissions of non-functionality test standpoints in the test design document based on the specification
Yes
Omission of the test design review (to review flow)
Re-check of sufficiency of the test design review
It is difficult to detect it in testing

No testing was done in the situation in which mass data communication could occur

⇒ Check the sufficiency of the test in terms of user standpoints, and the test method applied

⇒ Additional testing with new test scenario

Root cause and 1+n procedure for the case

Case: Mass data communication occurred and aborted the data analysis engine

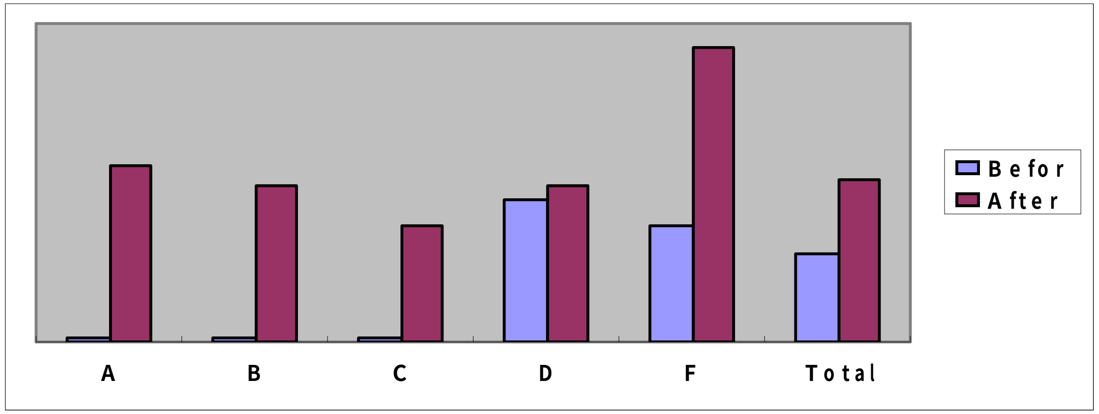
	Root cause	1+n procedure
Injection	Overflow caused by unexpected data volume in the certain situation was not assumed	Additional review to check the basis of estimation of a buffer size
Overlook During review	No reviewer with the ability of detecting the relevant defect participated in the review	Check the sufficiency of a reviewer ⇒ Additional reviewing for the same cases
Overlook during testing	No testing was done in the situation in which mass data communication could occur	Check the sufficiency of the test in terms of user standpoints ⇒ Additional testing with new test scenario

We actually performed the 1+n procedure and succeeded to detect 3 defects.

4. Conclusion

Effectiveness

The rate of detecting of similar defects improved more than 20% and it improved in almost all groups



Future challenges

It is necessary for us to enhance the flows and the stories

4. Conclusion

It is possible to define the analysis method as a framework if limiting the purpose of analysis.

Empowered by Innovation

NEC