An Approach to Quality Progress in Large-Scale, Short-Term Software Development

2017/3/20-22 Ver. 1.0

Naoki Toko

株式会社　日立製作所
システム＆サービスビジネス統括本部
品質保証本部

Contents

HITACHI have quality management method.
Based on Japanese way, Using metrics.

+QM3S

Quality Mind and Skill Scoring System

Problem

Software : Large-Scale, Short-Term

Risk : Poor-Quality, Big-Delay, Great-Cost

QM3S

Point 1 No additional cost

Point 2 Easy detection for miss

Point 3 Improvement of quality mind

Point 4 Visualization of working quality

Upgrade HITACHI way

ISO9000®	CMMI®	Japanese ways Metrics etc.	HITACHI experience

2. Introduction

HITACHI
Inspire the Next

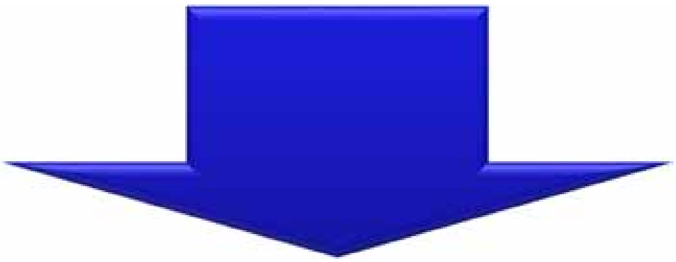
We examined the method for measuring and improving the "Quality Progress" in software development

1. Analysis of processes that have a greater effect on product quality
2. Quantification of the work quality of developer

In acceptance test,
If the software's quality is poor

Delivery
delays

Cost
overruns



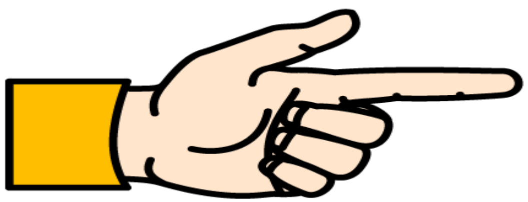
It is important to predict product quality before testing begins

4. Approach to Quality Progress

Targeting the same software

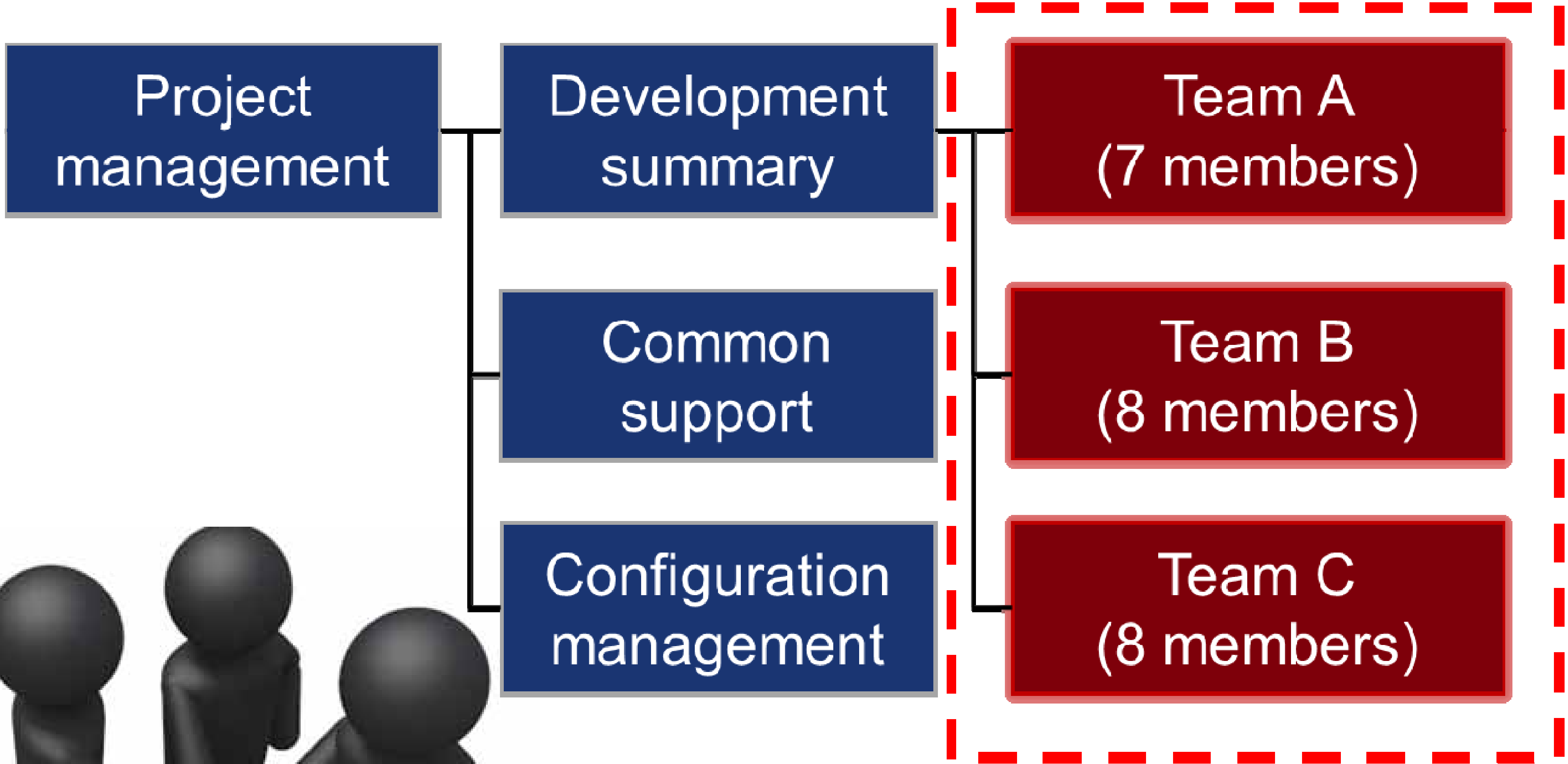
Multiple teams tests

Differences arose in the test results



We found out
which process
was the primary cause

4.1.1 Analysis target (test structure)



Project structure

4.1.2 Analysis target (measurement data)

Quality metrics

the number of bugs, the number of checklist items,etc.

Progress metrics

the number of days of delay, delay staff-hours

Work processes

the count and time of reviews, the count and time of meetings,etc.

Differences of each team

4.1.3 Process analysis results

"good process" that produces a good quality products will meet the following criteria

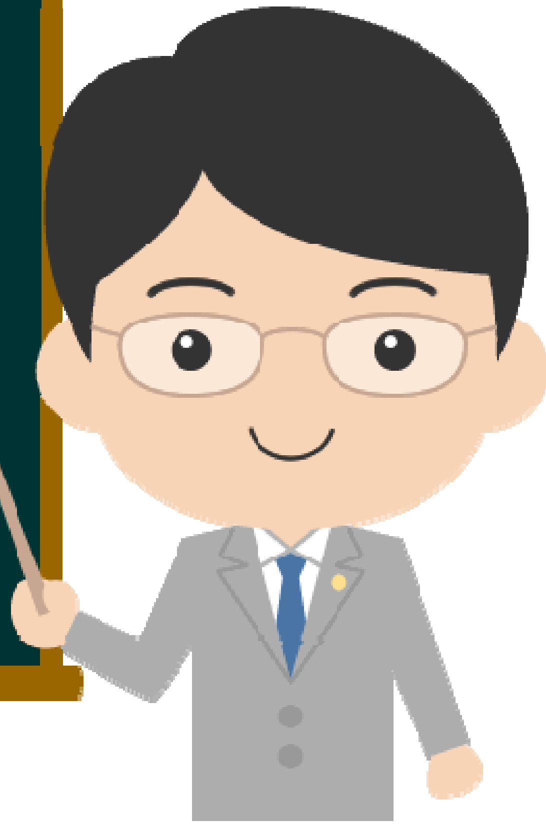
1. Appropriately use of a self-checklist
2. Implements appropriate reviews
3. Sharing of quality-related information

HITACHI
Inspire the Next

To improve quality before testing

"Quality Progress"

Quantifying the process implementation status



QM3S

Quality Mind and Skill Scoring System

■HITACHI way

Process	Self-check	Review	Audit
	Developer	Development leader	Quality Assurance
Event	Execute checklist	Confirm check results	Check for completed checklist

4.2.1 QM3S procedure overview

Before

Process	Self-check	Review	Audit
	Developer	Development leader	Quality Assurance
Event	Execute checklist	Confirm check results	Check for completed checklist

New procedure

Process	Self-check	Review	Audit
	Developer	Development leader	Quality Assurance
Event	Execute **new** checklist	Confirm check results Provide **feedback**	Analyze **quality progress**

Point 1 **Point 2** **Point 3**

HITACHI
Inspire the Next

■Before

No.	Content of check	Developer	Development leader
		Name	Name
1	For IF statement branches, ▲▲▲ must be observed.		☑
2	For character code conversions, use ○○.		☑
n	■■■….		☑

■New checklist

No.	Content of check	Developer	Development leader
		Name	Name
1	For IF statement branches, ▲▲▲ must be observed.	10	0
2	For character code conversions, use ○○.	2	4
n	Quality Progress	XXX points	

Point 1

Point 2

Point 3

4.2.1 QM3S procedure overview

■Execute new checklist

No.	Content of check	Developer Name	
1	For IF statement branches, ▲▲▲ must be observed.	10	
2	For character code conversions, use ◯◯.	2	4
n	Quality Progress	XXX points	

Point 1

- **Impossible to enter by copying and pasting**
- **No additional cost**

HITACHI
Inspire the Next

■Provide feedback

No.	Content of check	Develop...	Development leader
		Name	Name
1	For IF statement branches, ▲▲▲ must be observed.	10	0
2	For character code conversions, use 〇〇.	2	4
n	Quality Progress	XXX points	

Point 2

- Easy detection for miss
- We can see how much feedback leader gave

HITACHI
Inspire the Next

■Analyze Quality Progress

No.	Content of check	Developer	Development leader
		Name	Name
1	For IF statement	10	0
2	For character coc	2	4
n	Quality Progress	XXX points	

Point 3

- Improvement of quality mind
- Visualization of working quality

4.2.2 QM3S application results

Software : Large-Scale, Short-Term
Risk : Poor-Quality, Big-Delay, Great-Cost
Developers : China (International Procurement)
Target phase：Detailed design ~ Unit testing

QM3S	Applied	Did not apply
System scale	1,096 kS	202 kS
Number of team	11	2
Number of Developer	60	11

kS:1,000 step(Lines of code)

HITACHI
Inspire the Next

Results of bug density measurements

Team	Self-check	Review	Test
Applied	2.8bugs/kS	0.5bugs/kS	4.3bugs/kS
Did not apply	0.9bugs/kS	0.4bugs/kS	7.4bugs/kS

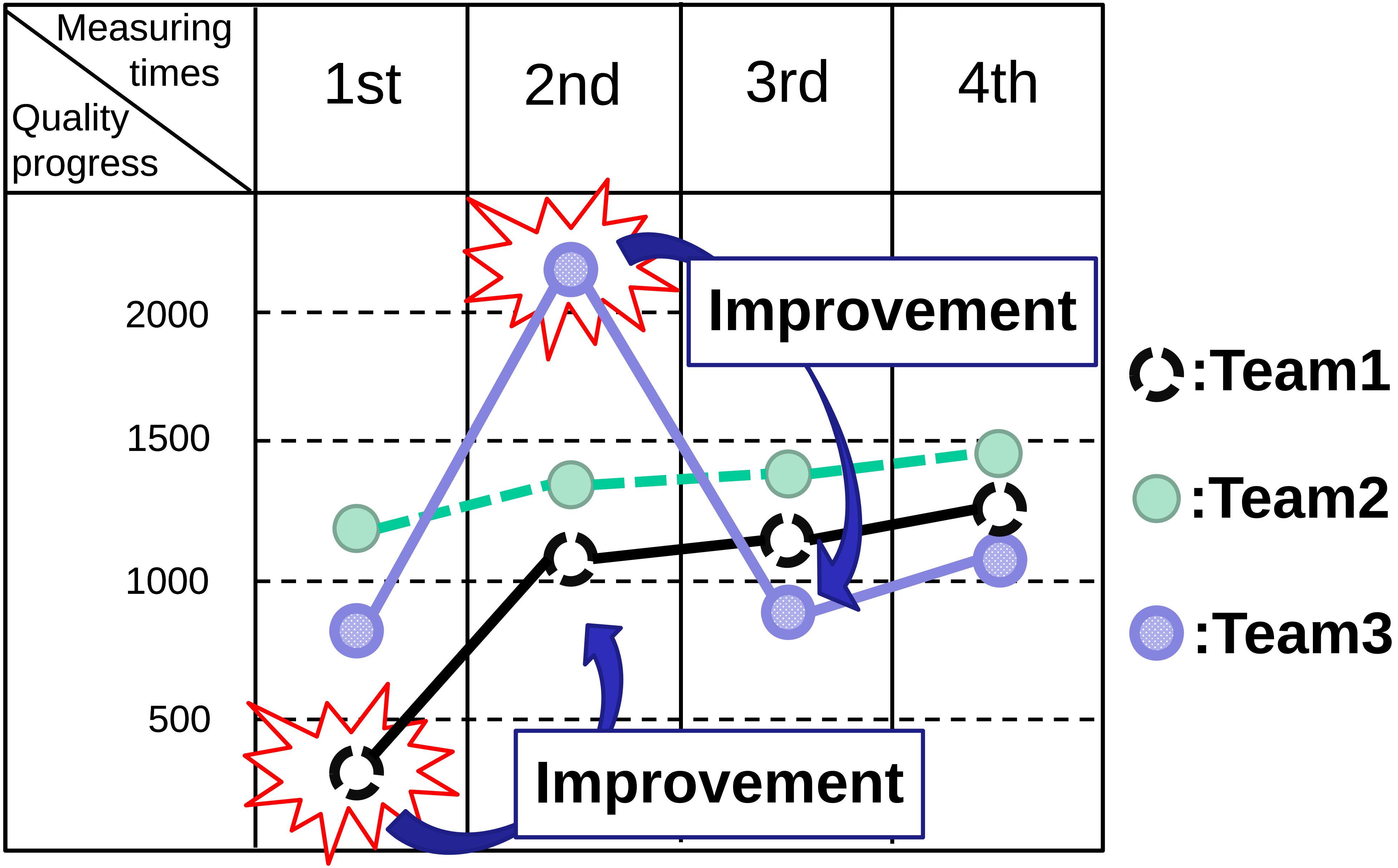
Value planned	Undefined		4.3bugs/kS

Bugs/kS: bugs in 1,000 Lines of code

More bugs can be discovered.
Approach the value planned.

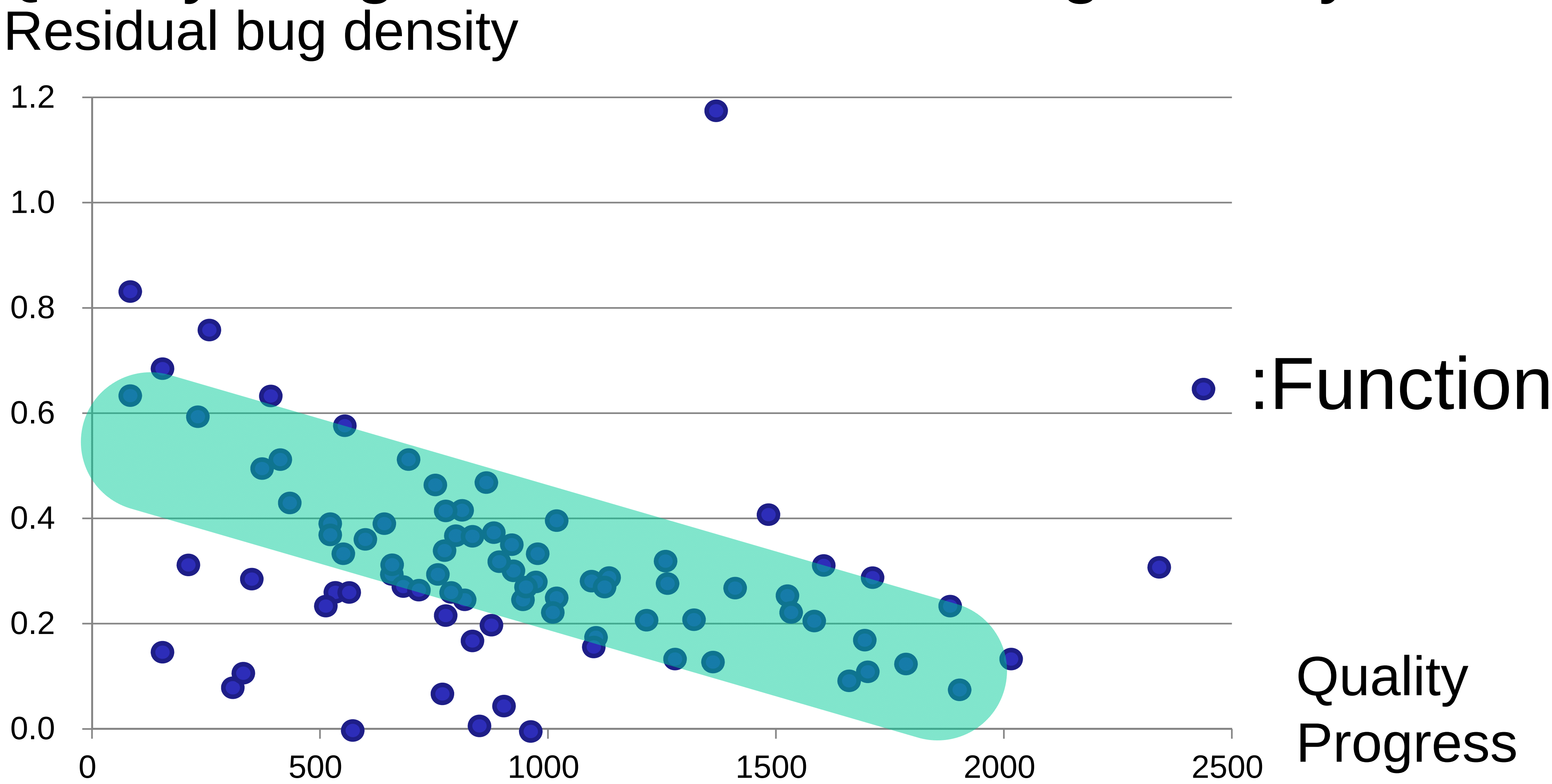
HITACHI
Inspire the Next

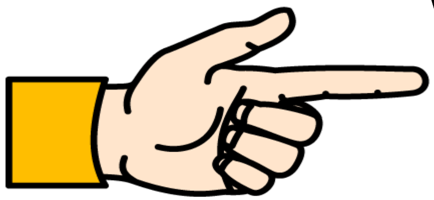
Results of Quality Progress



HITACHI
Inspire the Next

Quality Progress and residual bug density





When quality progress is large,
Residual bug density is low.

4.2.3 To predict product quality from work quality

- New checklist format
- Quality Progress
- We can predict the quality of software

PATENT PENDING

HITACHI
Inspire the Next

We can

QM3S

Quality Mind and Skill Scoring System

1. measure Quality Progress in each phase of software development.
2. measure Quality Progress before testing.
3. predict and improve the quality of outsourced software.
4. reduce the risk of delivery delays and cost overruns.

HITACHI
Inspire the Next

We are still examining:

- The **appropriate** number of checklist items
- **Priority** of checklist items
- **Frequency** of checklist item re-examination

END

An Approach to Quality Progress in Large-Scale, Short-Term Software Development

Hitachi, Ltd.
Information & Communication Technology Business Division
Quality Assurance Division

HITACHI
Inspire the Next

7. appendix

HITACHI
Inspire the Next

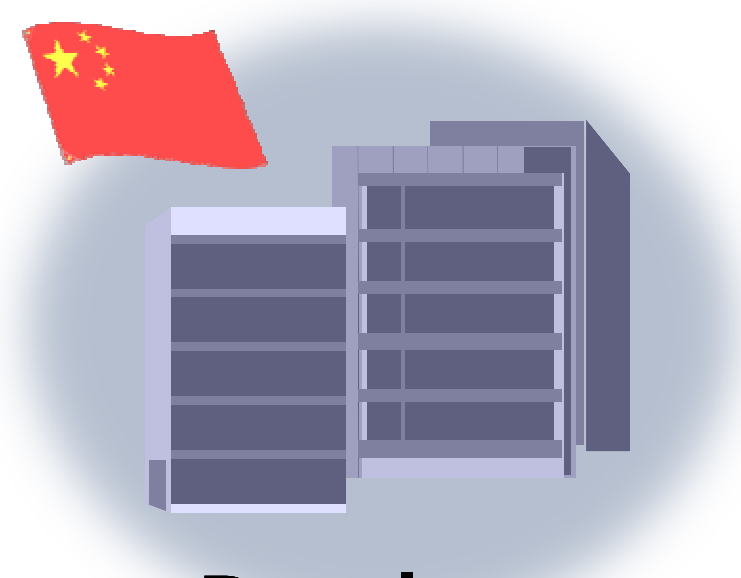
No.	Main category	Data	Source
1	Quality metrics	Number of undetected bugs	Bug report
2		Number of undetected bug criticality (A + B) events	Bug report
3		Bug density	Bug report
4		Number of bug criticality (A) events	Bug report
5		Checklist density	Checklist
6		Number of checklist classifications (abnormal + limit)	Checklist
7	Progress metrics	Delay days	Progress report
8		Delay staff-hours	Progress report
9	Work processes	Number of checklist reviews	Review minutes
10		Time of checklist review	Review report
11		Ratio of noted items concerning checklist classifications (abnormal, limit)	Review minutes
12		Existence of detailed schedule	Interview
13		Number of morning and evening meetings	Interview
14		Time of morning and evening meetings	Interview
15		Number of pending issues	Pending issue management ledger
16		Average delay (days) in tackling pending issues	Pending issue management ledger
17		Opportunity to share information on bugs	Interview
18		Overall ratio of similarity review target	Bug report

7. appendix

No.	Classification	metrics	Work process
1	Quality	Bug density/number of bug criticality (A) events	**Using a self-checklist, review** the checklist and check for omissions on an item-by-item basis.
2			Provide opportunity to **share information** on bugs. (Example: Hold morning and evening meetings.)
3		Ratio of checklist classifications (abnormal + limit)	**Using a self-checklist, review** the checklist and check for omissions on an item-by-item basis.
4			When each Developer completes the first one, **review** the checklist. From the second one and thereafter, reflect noted items.
5			During the checklist **review**, confirm whether the standard values of checklist classifications (normal: 60%; abnormal: 20%; limit: 20%) are met.
6	Progress	Delay days/delay staff-hours	Manage work such that the team can **share information** on bugs discovered early.
7			Set pending issue deadlines after clarifying the basis for said deadlines. (Indicate the priority of pending issue measures, and then tackle the work.)

Lines of code (LOC) is used as the metrics for expressing the size of a program in HITACHI.

Coding rules are provided for each program language.
These rules are designed to reduce inconsistencies in code that is produced by different developers.
Because of these rules, we can assume that the same number of lines of code (LOC) will be produced from the same design document even if the developer is different.
In Japan, LOC is used as an empirical metrics because it allows information to be obtained and shared easily.



Developers

QM3S is useful for Developers

- To review the project structure
- To improve quality mind of members
- To understand the work of the Developer

HITACHI
Inspire the Next

Software summary : Test administration system
System scale : 1,298 kS (Lines of code)
Development period : 1year8months (**QM3S**：5months)