Conquer CI Server!

Re-establishment of Order and Nurture of the Solid Organization by Project Metrics and Automation Techniques

Sep/1/2015
Hiroyuki Ito, Kazuhisa Naoi
Rakuten, Inc.
http://www.rakuten.co.jp/

Who are you? (1)

Hiroyuki Ito (The Hiro)

@hageyahhoo (Twitter)

hageyahhoo (Facebook)

Former Agile Coach from Test-Driven Development Group

Scrum Alliance CSM CERTIFIED

Scrum Alliance CSP CERTIFIED

Scrum Alliance CSPO CERTIFIED

A speaker of Agile2014

Technology–Driven Developmen
Using Automation and
Development Techniques
to Grow an Agile Culture

Jul/29/2014
Hiroyuki Ito
Development Process Optimization Department, Rakuten, Inc.
http://www.rakuten.co.jp/

楽天 Rakuten

Please check presentation and paper ☺

Who are you? (2)

Kazuhisa Naoi

Travel Extranet Group

Travel Product Development Section

Travel Service Development Department

Rakuten, Inc.

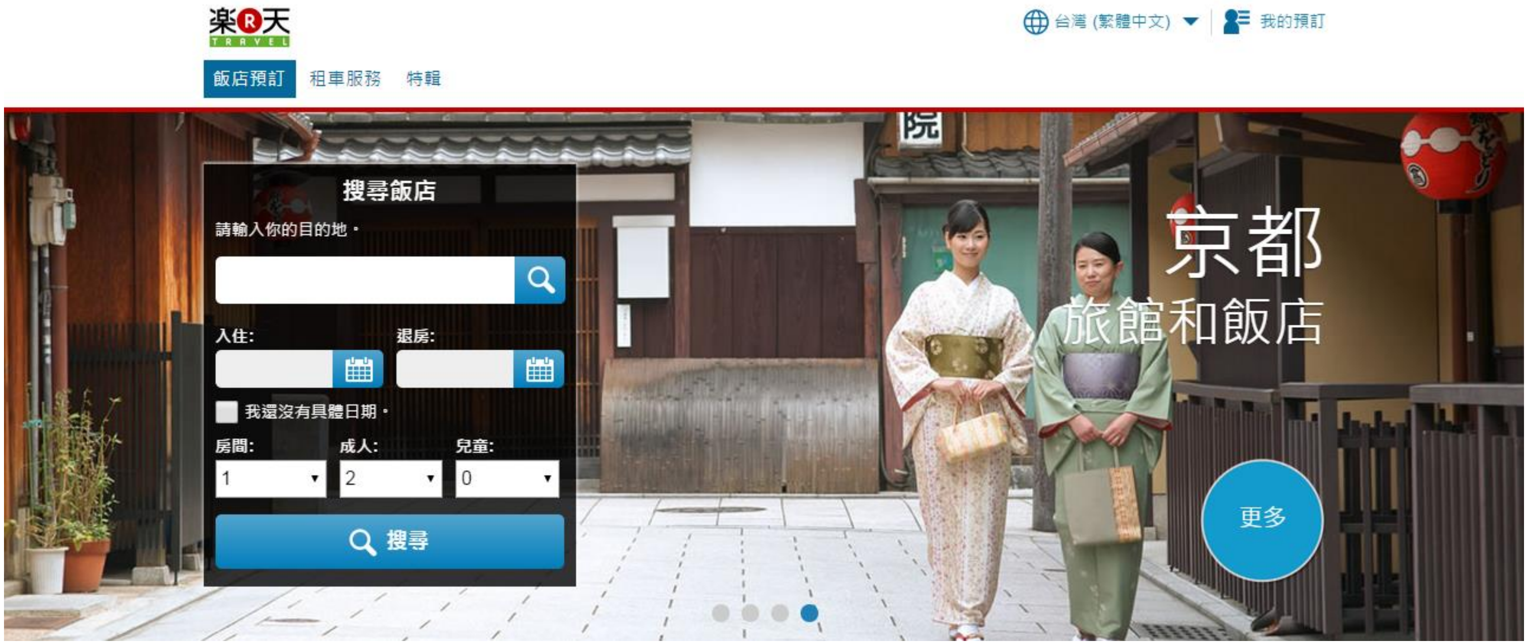
Group Manager

kazuhisa.naoi@rakuten.com

@naoinaoi (Twitter)

naoinaoi (Facebook)

Rakuten Travel has Taiwanese site



Please visit us!!

http://travel.rakuten.com.tw/

This is a real Kaizen story of Rakuten Travel in 2014.

Theme

1. Project Metrics

2. The importance of Strategy Formulation

3. Benefit of Infrastructure as Code

4. Technology-Driven Development

Main theme

1. Project Metrics

Agenda

1. Challenges

2. What should we s

The importance of Strategy Formulation

Project Metrics

3. How could we red

Benefit of Infrastructure as Code

Project Metrics

4. How could we nu

Technology-Driven Development

Project Metrics

5. Conclusion

Project Metrics

1. Challenges at that time

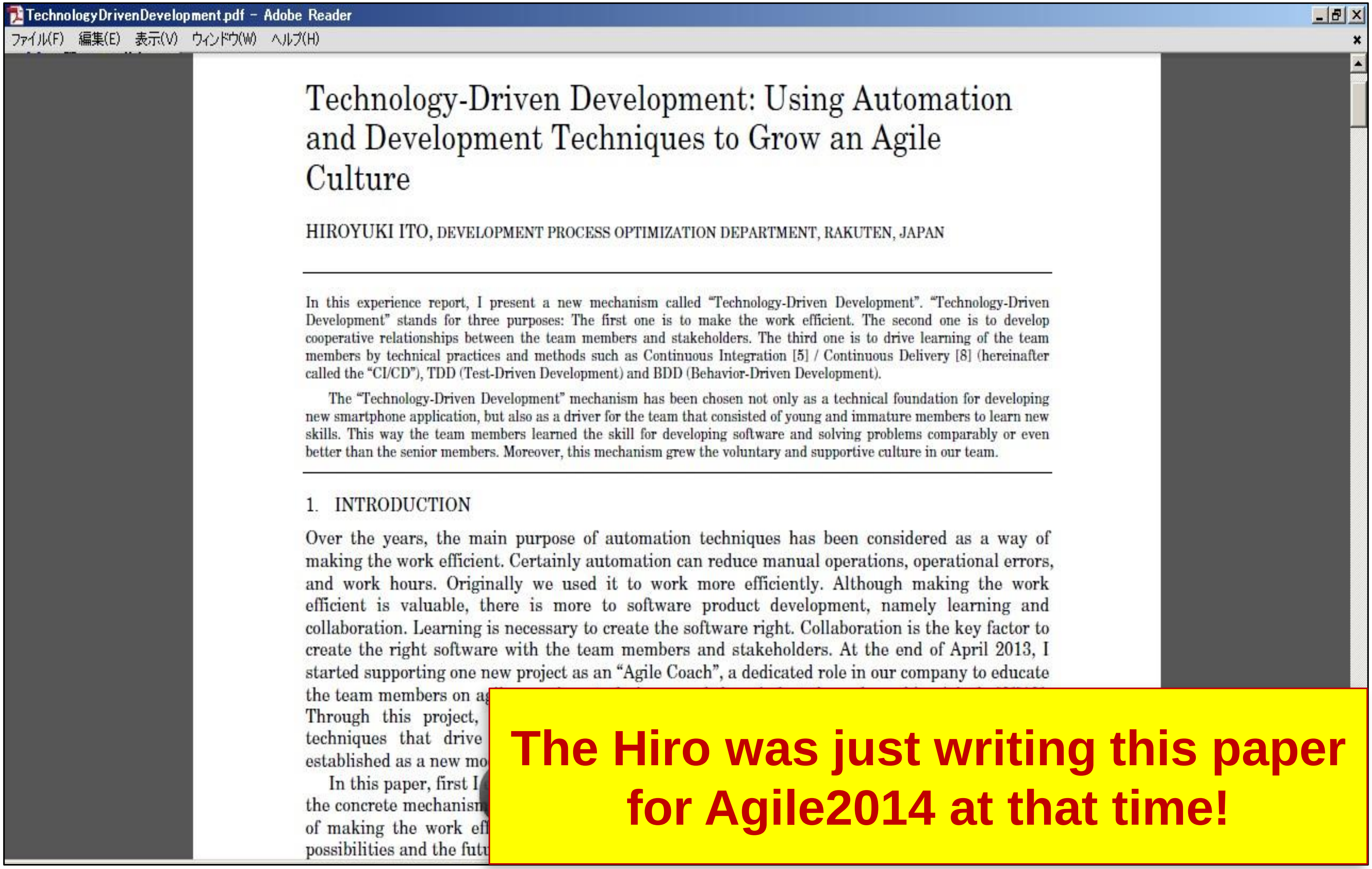
2. What should we show as accomplishments?

3. How could we reduce the load of CI server?

4. How could we nurture members and teams?

5. Conclusion

The end of June 2014


TechnologyDrivenDevelopment.pdf - Adobe Reader
ファイル(F) 編集(E) 表示(V) ウィンドウ(W) ヘルプ(H)

Technology-Driven Development: Using Automation and Development Techniques to Grow an Agile Culture

HIROYUKI ITO, DEVELOPMENT PROCESS OPTIMIZATION DEPARTMENT, RAKUTEN, JAPAN

In this experience report, I present a new mechanism called "Technology-Driven Development". "Technology-Driven Development" stands for three purposes: The first one is to make the work efficient. The second one is to develop cooperative relationships between the team members and stakeholders. The third one is to drive learning of the team members by technical practices and methods such as Continuous Integration [5] / Continuous Delivery [8] (hereinafter called the "CI/CD"), TDD (Test-Driven Development) and BDD (Behavior-Driven Development).

The "Technology-Driven Development" mechanism has been chosen not only as a technical foundation for developing new smartphone application, but also as a driver for the team that consisted of young and immature members to learn new skills. This way the team members learned the skill for developing software and solving problems comparably or even better than the senior members. Moreover, this mechanism grew the voluntary and supportive culture in our team.

1. INTRODUCTION

Over the years, the main purpose of automation techniques has been considered as a way of making the work efficient. Certainly automation can reduce manual operations, operational errors, and work hours. Originally we used it to work more efficiently. Although making the work efficient is valuable, there is more to software product development, namely learning and collaboration. Learning is necessary to create the software right. Collaboration is the key factor to create the right software with the team members and stakeholders. At the end of April 2013, I started supporting one new project as an "Agile Coach", a dedicated role in our company to educate the team members on a
Through this project,
techniques that drive
established as a new mo

In this paper, first I
the concrete mechanism
of making the work eff
possibilities and the futu

The Hiro was just writing this paper for Agile2014 at that time!

Troubles came here!

We **couldn't deliver products on time** due to the overload of CI server!

If we add jobs more, CI server will stop and **we cannot provide services**!

Our status at that time

Leading automation at one QA project

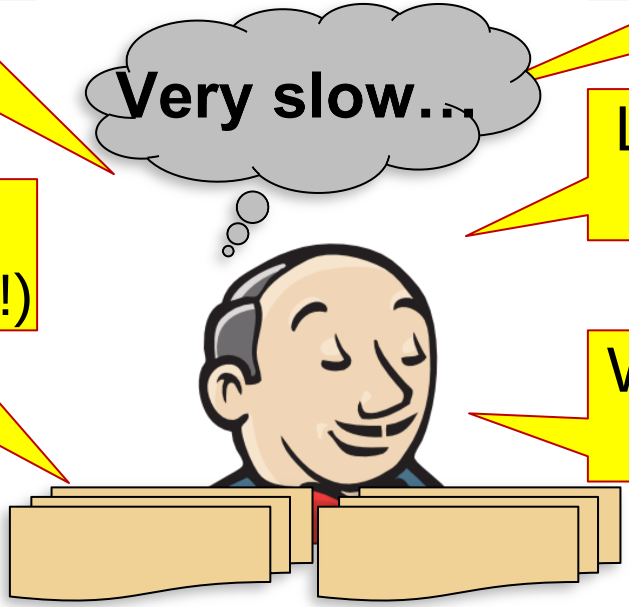
- Implementing Build Pipeline
- Realizing Release Automation
- Introducing Smoke Testing

**Leading a big project
as a manager
to make their systems multilingual**

**No responsibilities of CI server
to configure, monitor, and operate.**

We decided to solve these troubles!

Challenges at that time



Stand-alone server

Over 400 jobs
(would increase more!)

Nobody
monitored/maintained

Load average often became **over 10**

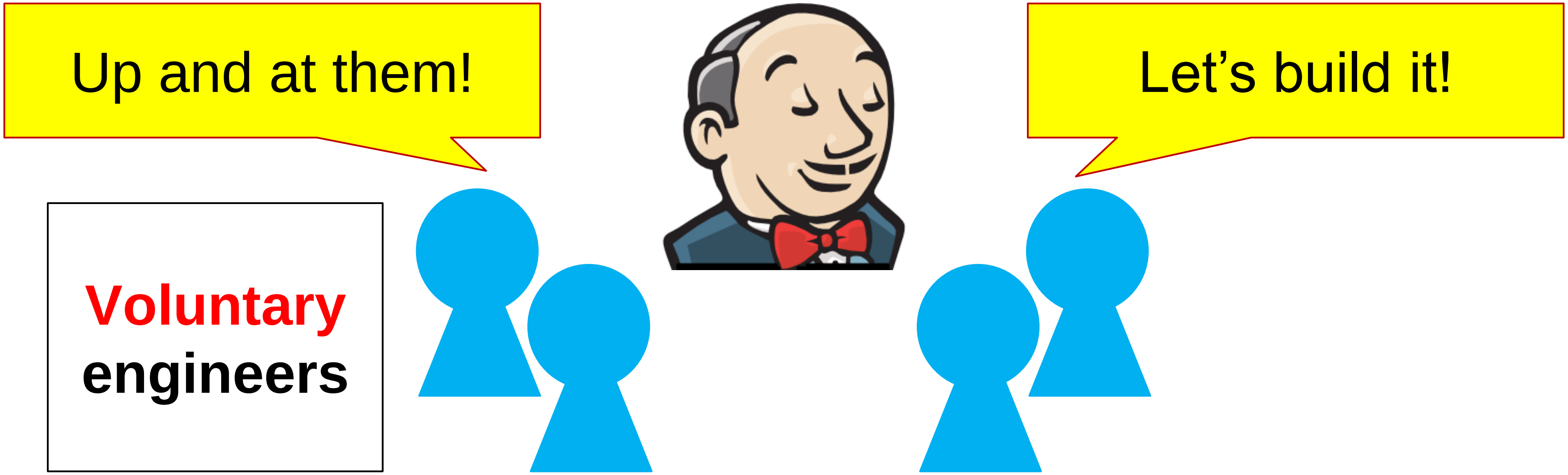
Waiting queue often became **over 10**

Often **couldn't open web pages** of CI server.

- Even trying "Script Console" was gamble ☹

Result of root cause analysis

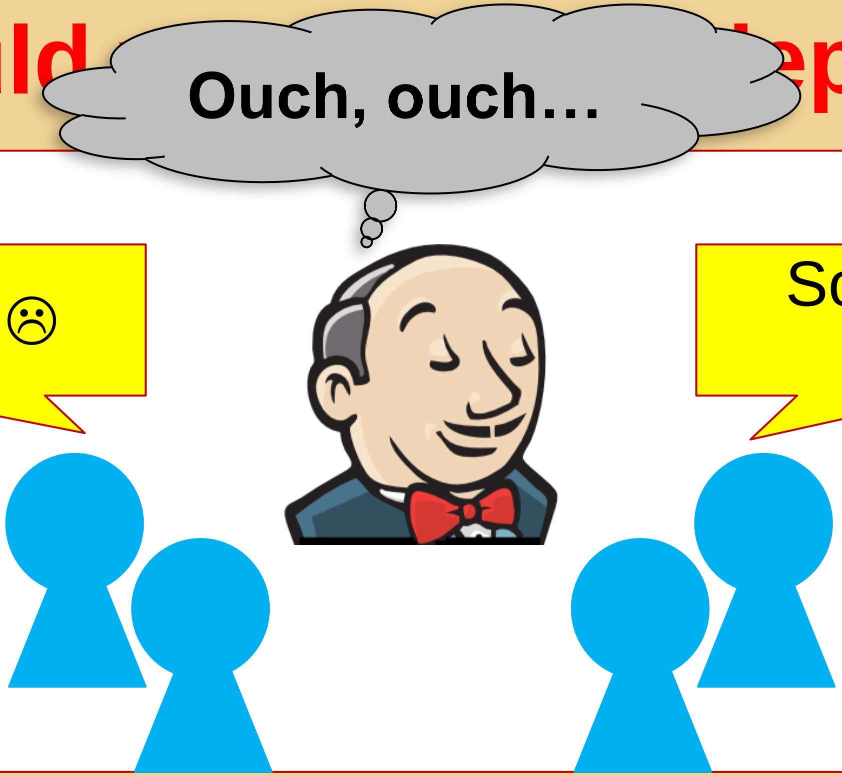
No dedicated team for CI server that could work for the dept totally



Built and operated **by voluntary engineers.**

Result of root cause analysis

No dedicated team for CI server
that could ept totally



Tons of work ☹

Somebody can solve it…!

Voluntary engineers

- Troubles **had been left** gradually.
- **No monitoring mechanism** for CI server.

The first action

Proposed to organize the dedicated and cross-functional team for CI server

Solves the overload with long-term and comprehensive view.

Has the responsibility to define vision and achieve it.

Dedicated team

To **avoid business loss** anymore,
it was required to **solve problems rationally**.

Jenkins Consolidation Team was organized.

This is an image.

1. Challenges

2. What should we show as accomplishments?

3. How could we reduce the load of CI server?

4. How could we nurture members and teams?

5. Conclusion

Problems found

What should we show **as accomplishments**?

Where should we **go**?

Are we **on the right path**?

Challenges at that time

As a business, we needed

- to **show** managers/stakeholders **progress**
- to **make** managers/stakeholders **happy**

However, what and how should we show **as a result of improving CI server** at all?

We needed to **inspect and adapt** continuously through these activities because they are **totally new**.

Project Metrics: as a solution

Find the numerical numbers which **can show managers/stakeholders progress easily**.

Especially look for the numerical numbers which you **can observe the transition**.

Track them periodically and focus on **the transition before/after actions**.

Always **review the effect** of them and **improve them if necessary**.

A lot of things to do!

How can we **reduce the load** of CI server?

How can we **scale up/out** CI server?

How should we **monitor** CI server?

How should we **nurture members**?

At first, clarified

- mission
- basic strategy
- roadmap

The point of this phase

Define them that **almost product members and stakeholders can agree on**.

- e.g.) Can infrastructure team agree on them?

Define them **not only from engineers' view**.

- It should cover the business value.

Make **measures sustainable**.

- Make the "Honeymoon Number" high.
- Include proper persons who can lead it.

1. Mission

Deliver products on time without any delay **in every way**.

- **It is based on the business value.**

Find infrastructure and mechanism for CI server to endure the increase of jobs.

- **It requires infrastructure team and others.**

Nurture new bloods in addition to find and solve problems.

- **It is for accomplishing sustainability.**

2. Basic strategy

1. Hasten to reduce the load of CI server

Purchase and set up additional servers to reduce loads while we are stopping overloaded jobs temporarily.

2. Solve the bottlenecks of CI servers

- Apply master/slave to balance loads easily.
- Refine CI servers by solving technical debts gradually.

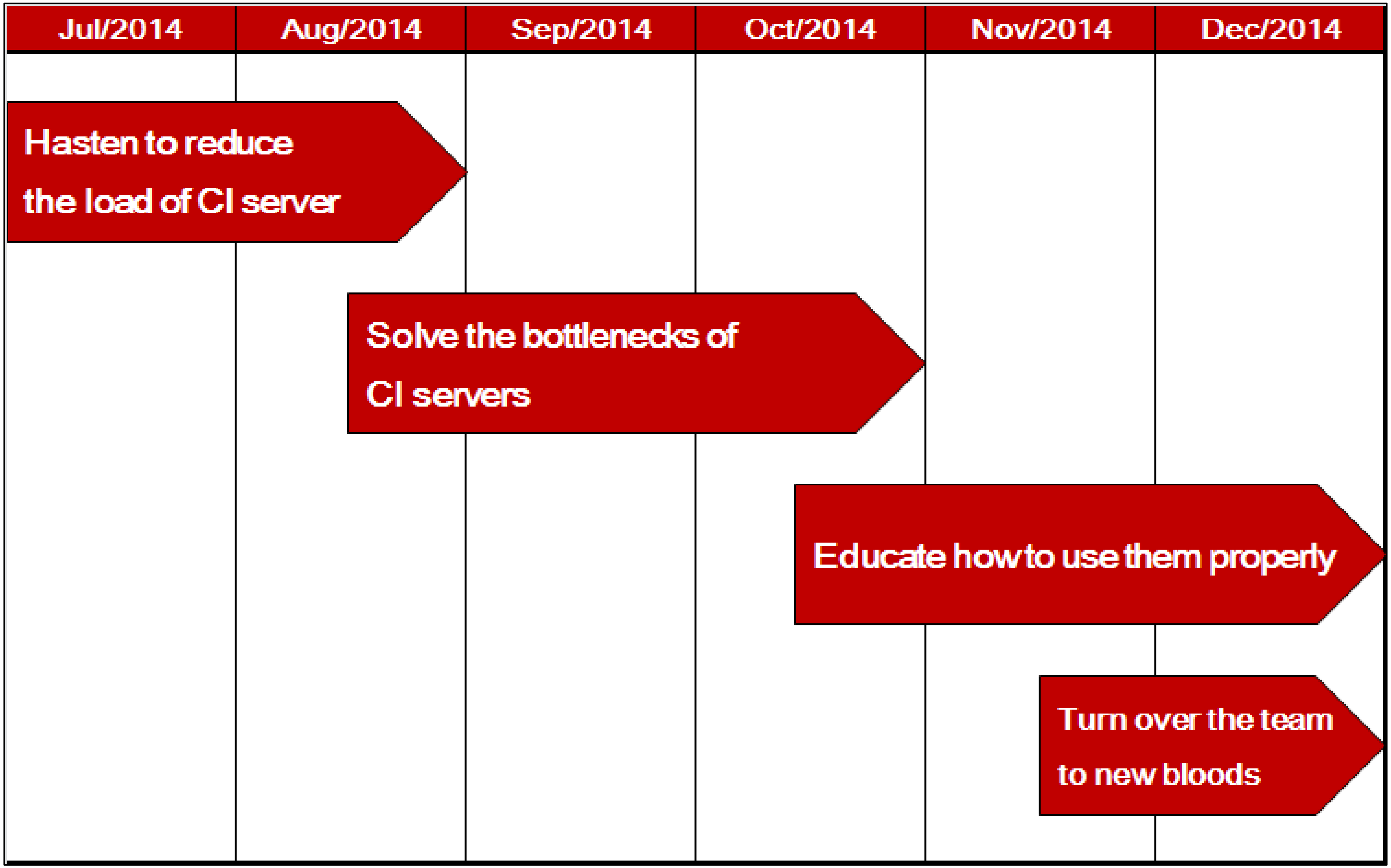
3. Educate how to use them properly

Nurture members/teams by teaching proper usage of CI servers and reduce technical debts simultaneously.

4. Turn over the team to new bloods

Accomplish sustainability of “Jenkins Consolidation” activities by nurturing new bloods.

3. Roadmap



Jul/2014 | Aug/2014 | Sep/2014 | Oct/2014 | Nov/2014 | Dec/2014

Hasten to reduce the load of CI server

Solve the bottlenecks of CI servers

Educate how to use them properly

Turn over the team to new bloods

Candidates of Project Metrics (1)

Reduction of system troubles	Transition and frequency of waiting/delaying delivery of the product
Status of infrastructure preparation	Difference between planned and actual to prepare servers • Measure waits and interruptions, too
	Difference between planned and actual to move all jobs on to each slave server
	Transition of mean time for server configuration and provisioning
Effect of master/slave adoption	Transition of migration ratio of jobs onto slaves
	Transition of “availability ratio” and “success ratio” of jobs on slaves
Status of Inquiries	Transition of inquiries created and resolved
	Transition of average mean time to solve

Candidates of Project Metrics (2)

Load information	Transition of the number of alert emails
	Transition of load on each server • Load average • CPU % • IO wait % • Disk usage
	Transition of “average execution time” and “average wait time” for each job
	Transition of waiting jobs to execute
	Transition and frequency of slow jobs
	How many persons do they feel that CI server becomes lighter and faster? (Gut feel)

Started the war for victory!

1. Challenges

2. What should we show as accomplishments?

3. How could we reduce the load of CI server?

4. How could we nurture members and teams?

5. Conclusion

Challenges at that time

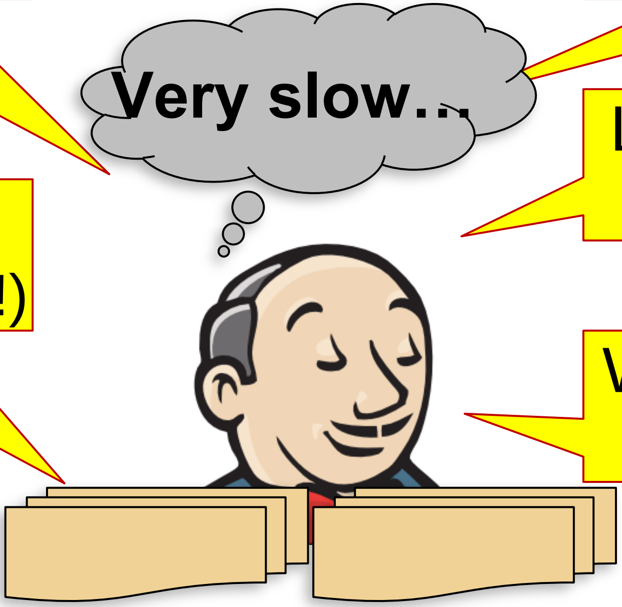
Stand-alone server

Over 400 jobs
(would increase more!)

Nobody
monitored/maintained

Load average often
became **over 10**

Waiting queue often
became **over 10**



Solve them at first!

Improved step-by-step

We only measured/show Project Metrics that were able to measure at that time.

Project Metrics measured at first (1)

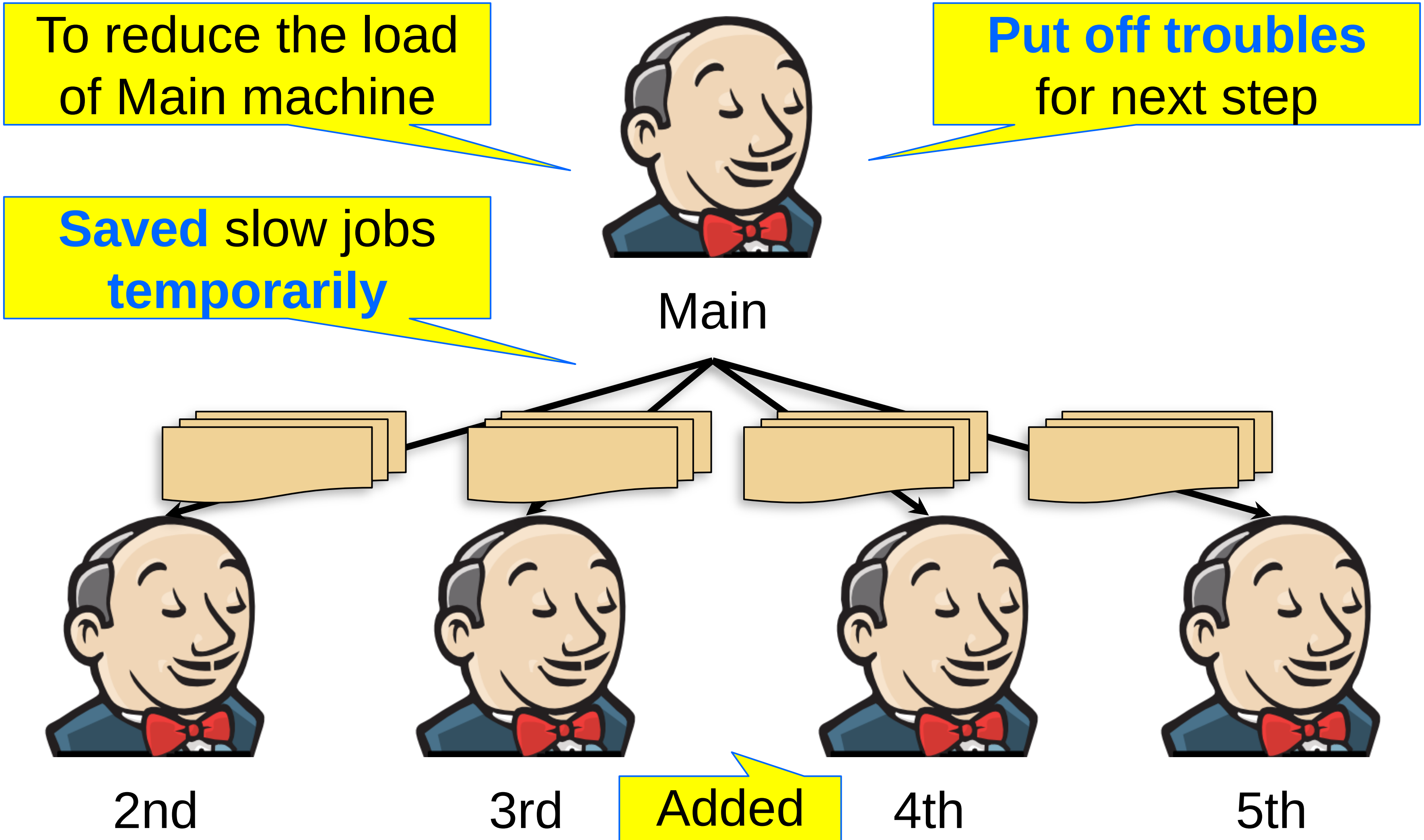
Reduction of system troubles	Transition and frequency of waiting/delaying delivery of the product
Status of infrastructure preparation	Difference between planned and actual to prepare servers • Measure waits and interruptions, too
	Difference between planned and actual to move all jobs on to each slave server
	Transition of mean time for server configuration and provisioning
Effect of master/slave adoption	Transition of migration ratio of jobs onto slaves
	Transition of “availability ratio” and “success ratio” of jobs on slaves
Status of Inquiries	Transition of inquiries created and resolved
	Transition of average mean time to solve

Project Metrics measured at first (2)

Load information	**Transition of the number of alert emails**
	Transition of load on each server • Load average • CPU % • IO wait % • Disk usage
	Transition of “average execution time” and “average wait time” for each job
	Transition of waiting jobs to execute
	Transition and frequency of slow jobs
	How many persons do they feel that CI server becomes lighter and faster? (Gut feel)

Measures

(1) Added servers simply



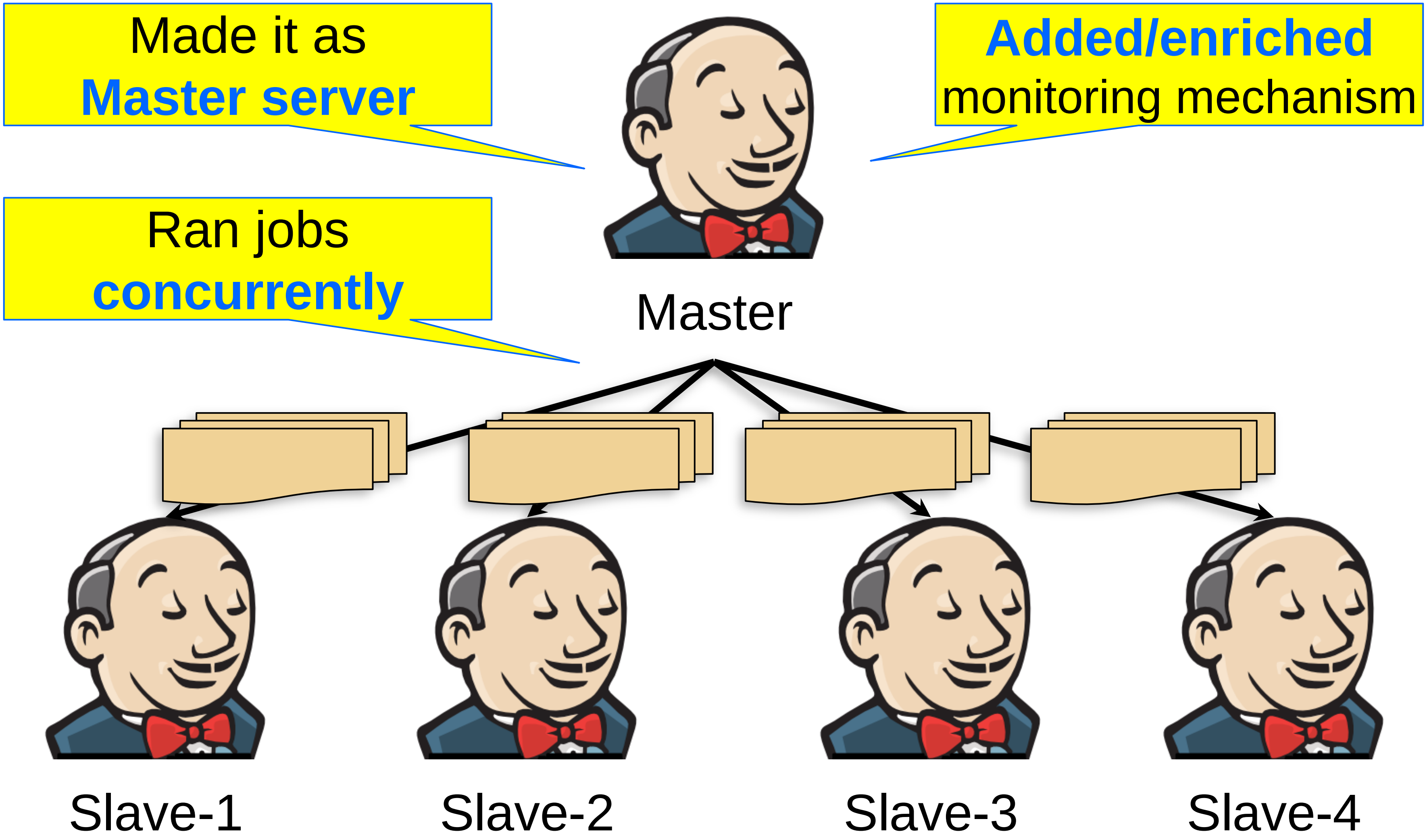
To reduce the load of Main machine

Put off troubles for next step

Saved slow jobs **temporarily**

Added 4 servers

(2) Adopted master/slave

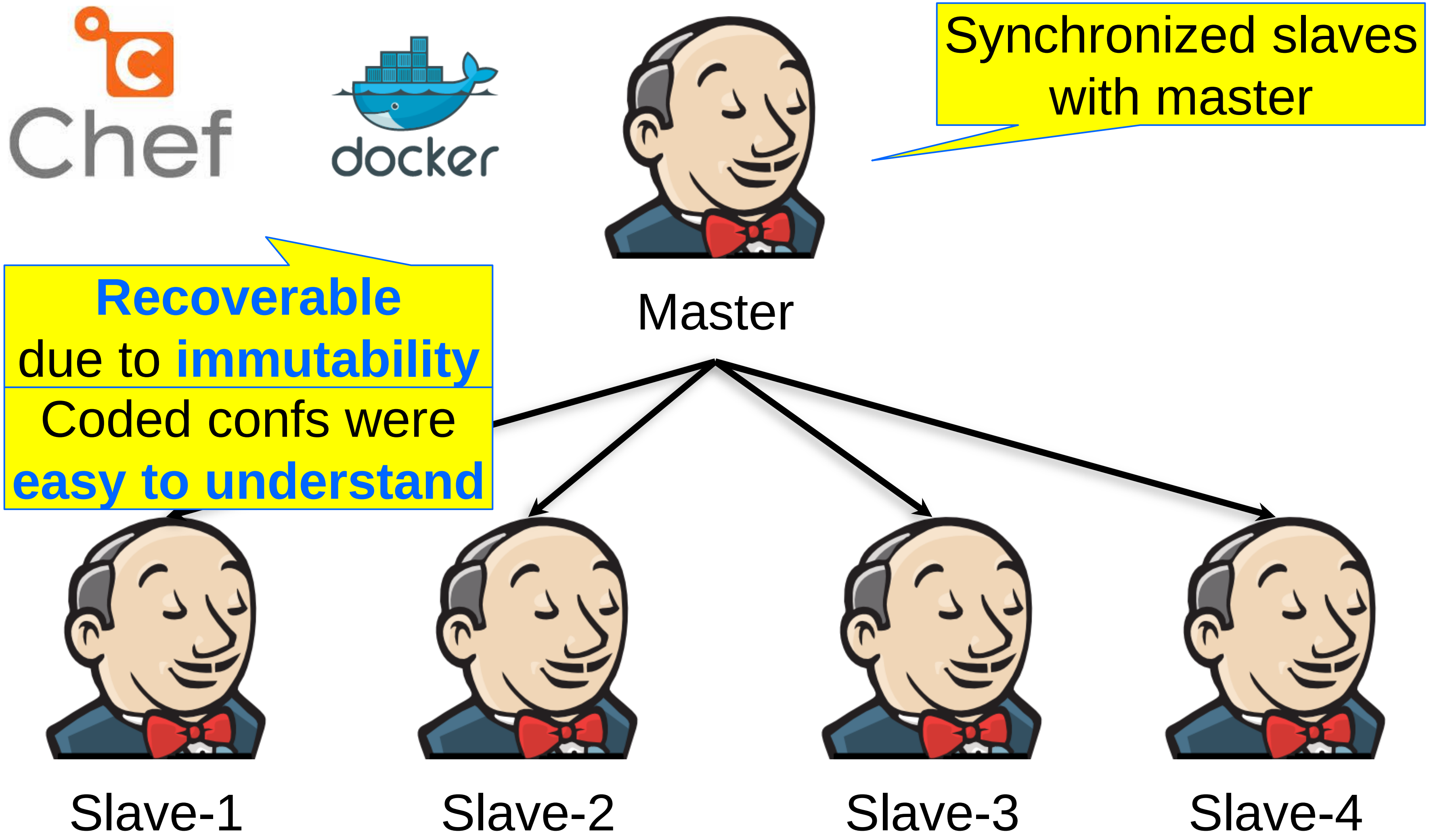


Made it as **Master server**

Added/enriched monitoring mechanism

Ran jobs **concurrently**

(3) Adopted Infrastructure as Code



Synchronized slaves with master

Recoverable
due to **immutability**

Coded confs were
easy to understand

Accomplishments

Result of preparing servers

Actions	Expected	Actual
Added servers simply	1 month	**1 week**
Adopted master/slave	1-2 month(s)	**1 week**
Adopted Infrastructure as Code	2 months	**1 month**

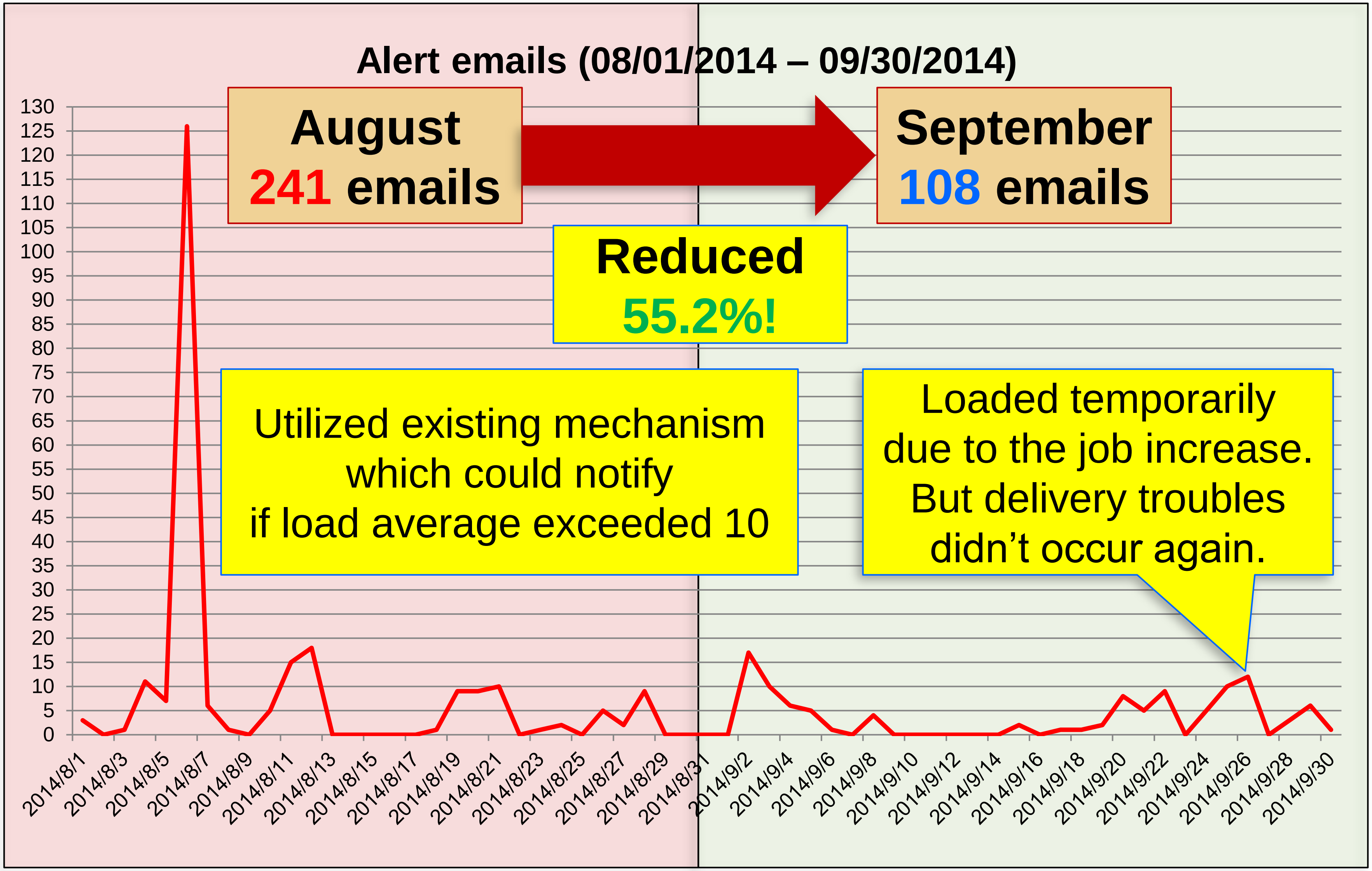
[Reasons]

- Prompt discussions with infrastructure team **led by Naoi-san**
- **Unexpected growth of young stars**

The point of negotiation and measures

- **We did just what we need** to achieve our target.
- We had already **prepared for next action** at that time!

Reduction of loads by the transition of alert emails



August
241 emails

September
108 emails

Reduced
55.2%!

Utilized existing mechanism which could notify if load average exceeded 10

Loaded temporarily due to the job increase. But delivery troubles didn't occur again.

Benefit of Infrastructure as Code (1)

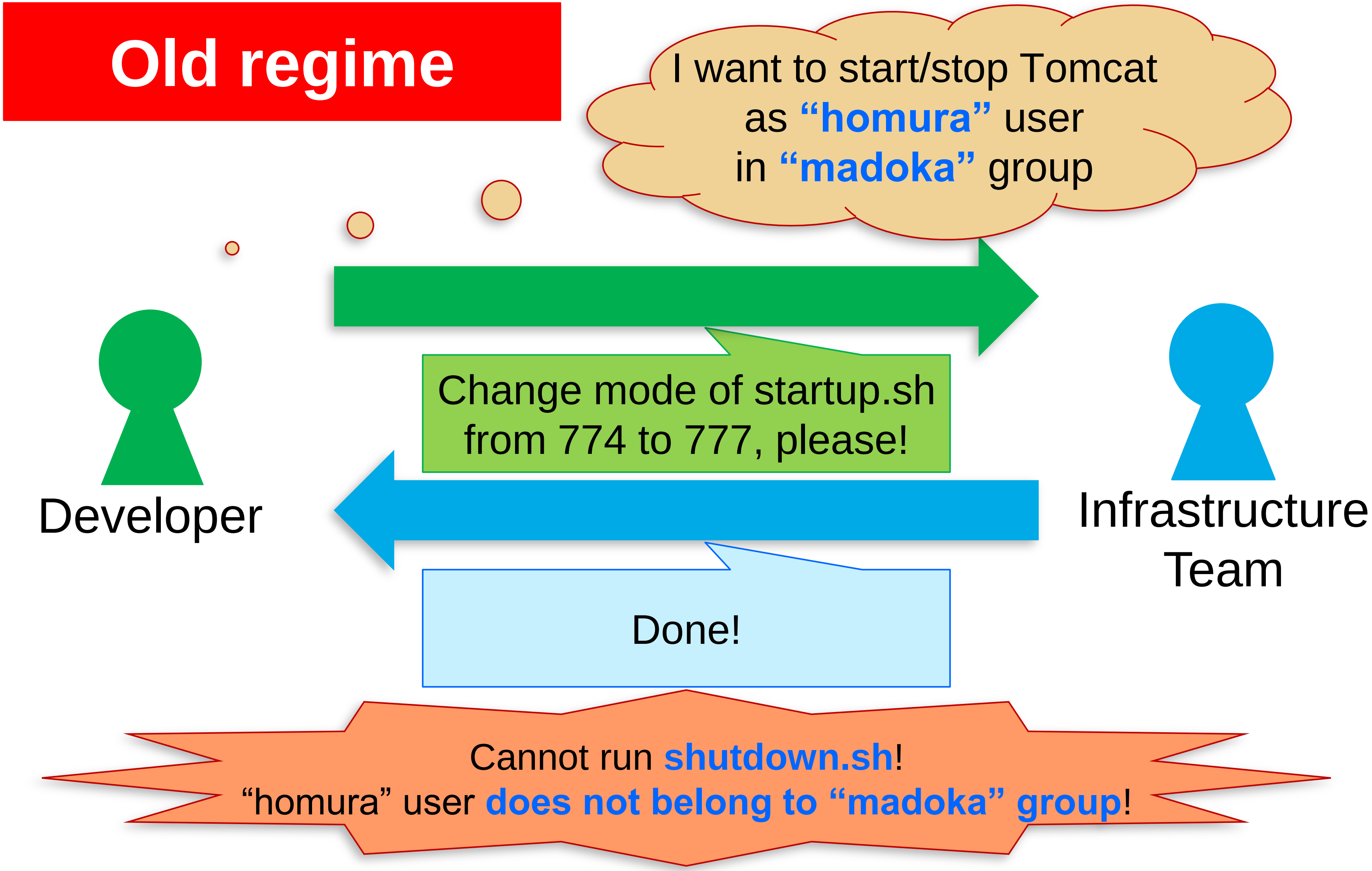
Before	After
1 week	30 minutes

[Reasons]

- **Eliminated miscommunications**
  by reducing exchanges
  between infrastructure team and developers.
- **Made works simpler and easier**
  because infrastructure team just have to run tests/recipes created/validated by engineers beforehand.
- **Could recover easily** if trouble happened
  due to Immutable Infrastructure.

Benefit of Infrastructure as Code (2)

Old regime



I want to start/stop Tomcat as **“homura”** user in **“madoka”** group

Developer

Change mode of startup.sh from 774 to 777, please!

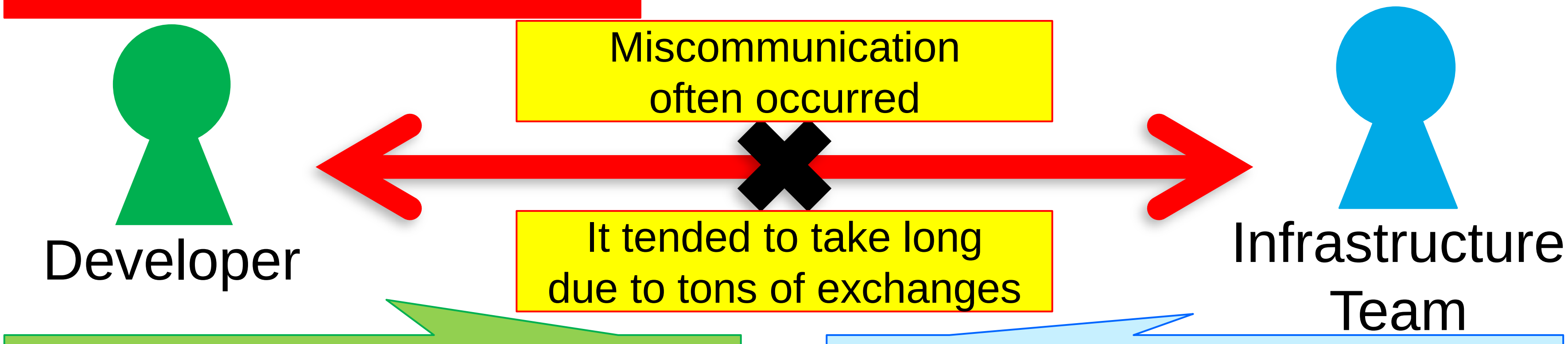
Infrastructure Team

Done!

Cannot run **shutdown.sh**!
“homura” user **does not belong to “madoka” group**!

Benefit of Infrastructure as Code (3)

Old regime


Miscommunication often occurred
Developer
Infrastructure Team
It tended to take long due to tons of exchanges

It was **difficult to check** all up-front due to **the differences of confs on each servers**.

They **couldn’t feel easy** after validation.

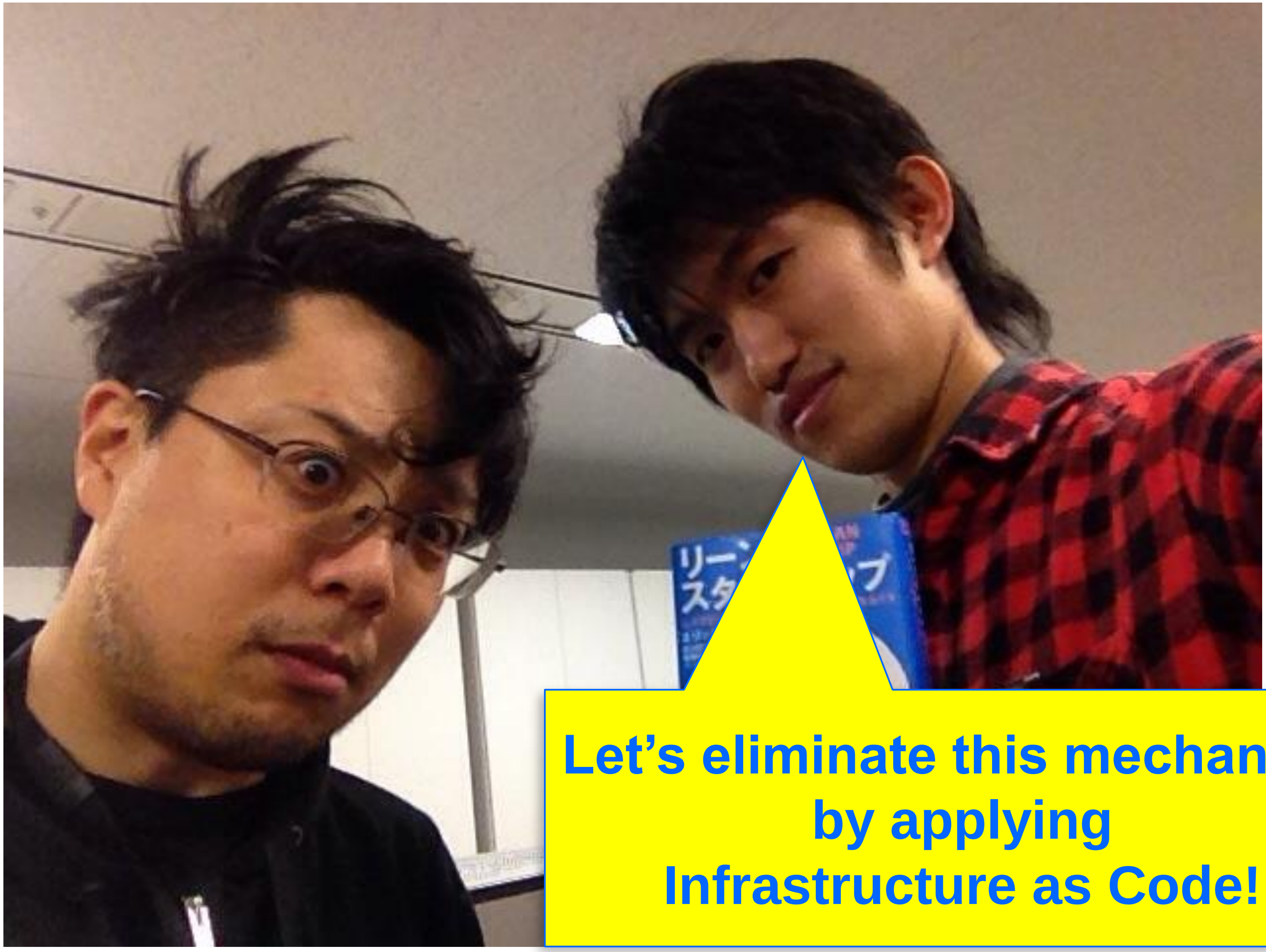
It **tended to include errors** on each request/configuration.

It was **difficult to know intentions/objectives** of these requests.

They **could configure** what they ordered **basically**.

They **tended to be late** due to **busyness**.

In fact, we faced with communication troubles…



Let's eliminate this mechanism by applying Infrastructure as Code!

Benefit of Infrastructure as Code (4)

New regime

Developer

Infrastructure Team

They can **try it easily** on their own PC **beforehand**.

They can **express** what they need as **tests and recipes**.

It can **reduce exchanges by Pull Request** of tests and recipes.

They just have to run tests and recipes!

It is **risk-free** because they can **recover easily**.

(Again) Result of preparing servers

Actions	Expected	Actual
Added servers simply	1 month	**1 week**
Adopted master/slave	1-2 month(s)	**1 week**
Adopted Infrastructure as Code	2 months	**1 month**

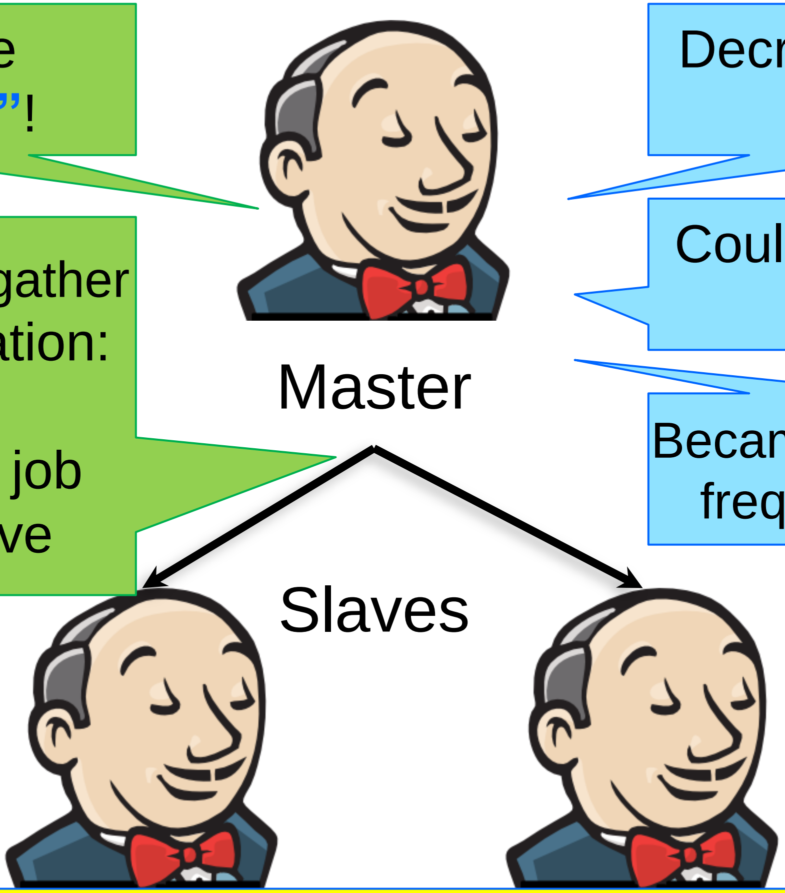
Additional improvements

Secondary effects of load reduction of Master server

Were able to use **"Script Console"**!

Could **automatically** gather the following information:

- List of jobs
- Exec time of each job
- Status of each slave



Decreased alert emails dramatically

Could afford to improve alert emails

Became easier to examine frequency of slow jobs

We were able to measure in more detail!

Project Metrics measured additionally (1)

Reduction of system troubles	Transition and frequency of waiting/delaying delivery of the product
Status of infrastructure preparation	Difference between planned and actual to prepare servers • Measure waits and interruptions, too
	Difference between planned and actual to move all jobs on to each slave server
	Transition of mean time for server configuration and provisioning
Effect of master/slave adoption	**Transition of migration ratio of jobs onto slaves**
	Transition of "availability ratio" and "success ratio" of jobs on slaves
Status of Inquiries	Transition of inquiries created and resolved
	Transition of average mean time to solve

Project Metrics measured additionally (2)

Load information	Transition of the number of alert emails
	Transition of load on each server • **Load average** • **CPU %** • **IO wait %** • **Disk usage**
	Transition of “average execution time” and “average wait time” for each job
	Transition of waiting jobs to execute
	Transition and frequency of slow jobs
	How many persons do they feel that CI server becomes lighter and faster? (Gut feel)

The importance of additional improvements

To continue improving steadily becomes the foundation of solid and high-performing teams and organization!

Alert emails (1) : Before improvement

```
09:40:04 up 154 days, 19:14,  2 users,  load average: 11.59,
11.32, 7.16

[Heavy Process Top 10] CPU(%), Proccess
165 xxxxx
```

Typo!

- Verbose information by running **“uptime”** command.
- Unable to identify the user of CPU-consuming processes.
  - Less information to identify the cause and to find the solution.
- Unable to identify the memory-consuming processes.
- **Unable to identify the job names that caused the overload.**

Alert emails (2) : First improvement

```
Load average: (1.03, 1.05, 1.23)
--- Top 10 Heavy CPU consumers ---
tomcat xxxxx
--- Top 10 Heavy Memory consumers ---
ito xxxxx
--- Jobs currently running ---
- Sayaka
```

- Focused only on **load average** as a result of “uptime” command.
- Started to gather/output the following information:
  - **the memory-consuming processes**
  - **user** of each process
  - **the job names** that caused the overload
    - **It was necessary to scroll down to the bottom of the notification because they were on the bottom.**

Alert emails (3) : Additional improvement

```
Load average: 10.0, 5.2, 2.4
--- Jobs currently running ---
- Sayaka
--- Top 10 Heavy CPU consumers ---
tomcat xxxxx
--- Top 10 Heavy Memory consumers ---
ito xxxxx
```

Output the job names **on the top of the notification**.

- Made identifying jobs very much easier.
  (We could identify jobs at one view!)
- Made identifying the cause and finding the solution easier.
  - **To be honest, we could have done it since then.**

Transition and frequency of slow jobs

Job name	Frequency
Madoka	32
Homura	17
Sayaka	15
Anko	13
Mami	13
Charlotte	4
QB	3
Hitomi	2
Uro-Buchi	2

Solve the jobs **from the top of the list preferentially**.

Notify it weekly to members/stakeholders to urge self-running actions.

Agile as learnings from failures

Gather/measure information **gradually**.

- You don’t need to measure everything.

All information is not necessarily useful.

-> Measure only useful one.

Need to **review/improve** Project Metrics **regularly/continuously**.

Unexpected things often occur.

-> Utilize them aggressively ☺

Retrospective of Project Metrics (1)

Reduction of system troubles	Transition and frequency of waiting/delaying delivery of the product
Status of infrastructure preparation	Difference between planned and actual to prepare servers • Measure waits and interruptions, too
	Difference between planned and actual to move all jobs on to each slave server
	Transition of mean time for server configuration and provisioning
Effect of master/slave adoption	Transition of migration ratio of jobs onto slaves
	Transition of “availability ratio” and “success ratio” of jobs on slaves
Status of Inquiries	Transition of inquiries created and resolved
	Transition of average mean time to solve

Retrospective of Project Metrics (1)

Reduction of system troubles	Transition and frequency of waiting/delaying delivery of the product
Status of infrastructure preparation	Difference between planned and actual to prepare servers for server configuration and provisioning
Effect of master/slave adoption	
Status of Inquiries	Transition of inquiries created and resolved Transition of average mean time to solve

Finished faster than we expected.

- Sufficient to show as accomplishments
- Already examined them for later actions
- **Work finished -> Stopped measuring**

- They were very useful to decide whether we need to solve problems right now or not when the overload was occurred.
- Sufficient to monitor **if necessary**.

Retrospective of Project Metrics (2)

Load information	Transition of the number of alert emails
	Transition of load on each server • Load average • CPU % • IO wait % • Disk usage
	Transition of "average execution time" and "average wait time" for each job
	Transition of waiting jobs to execute
	Transition and frequency of slow jobs
	How many persons do they feel that CI server becomes lighter and faster? (Gut feel)

Retrospective of Project Metrics (2)

Load information	Transition of the number of alert emails
	Transition of waiting jobs to execute
	Transition and frequency of slow jobs
	How many persons do they feel that CI server becomes lighter and faster? (Gut feel)

- We found it was **sufficient to gather/show only the following 2 numerical numbers**:
  - Transition of the number of alert emails
  - Transition and frequency of slow jobs
- It was sufficient to monitor

  if it was necessary to investigate in detail.

Succeeded dramatically!

1. Challenges

2. What should we show as accomplishments?

3. How could we reduce the load of CI server?

4. How could we nurture members and teams?

5. Conclusion

At the end of September 2014

We Rakuten hold Technology conference this time of year ☺

Additional task as a reward of accomplishment

The dept. set a goal to achieve
over 65% of UT line coverage
for all their products by the end of 2014

↓

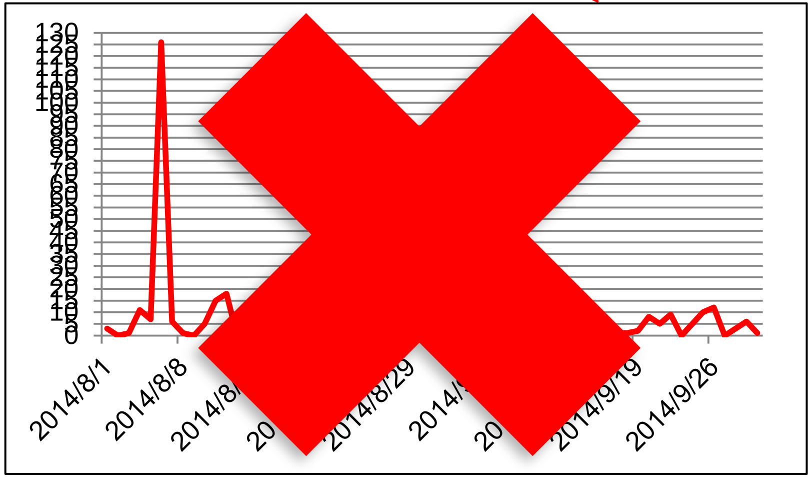
We decided to add it to our goal
because of load reduction of CI servers
and improvements faster than we expected!

The objective to set “UT Coverage” as a target

- **To make UT a habit** of our organization.
- **To motivate members** by numerical achievements.

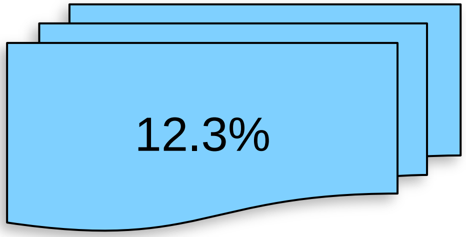
Challenges about UT coverage at that time

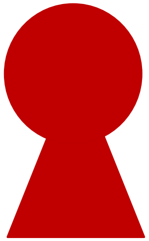
Lack of mechanisms to measure
UT coverage



Too many tools

- Cobertura
- Emma(JaCoCo)
- djUnit…





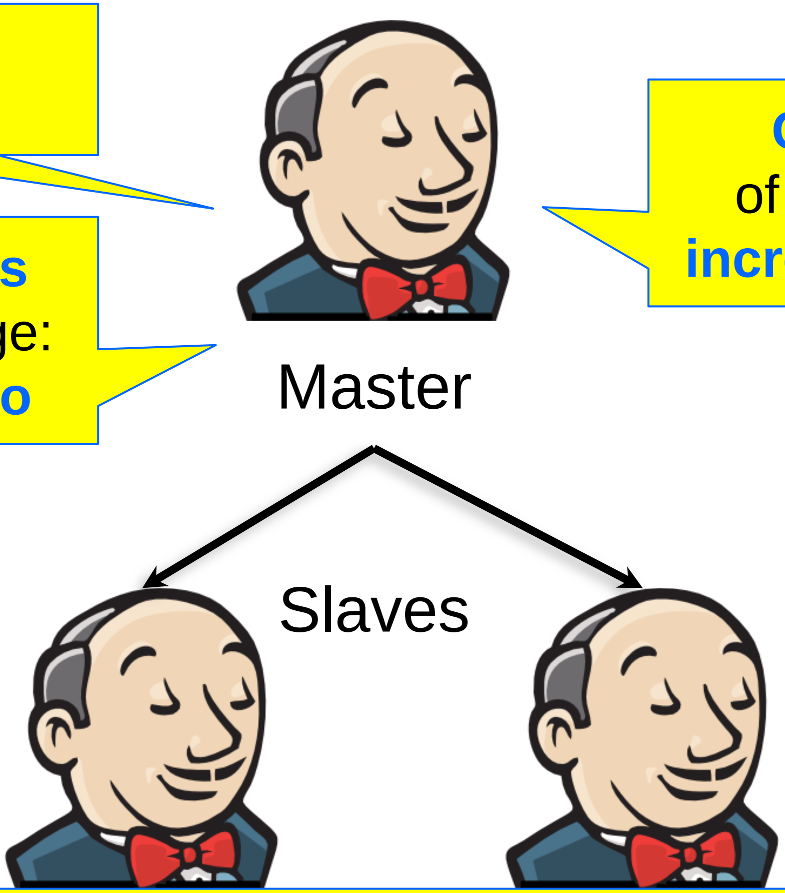
Lack of members
to lead this activity

Additional effects of load reduction of Master server

Were able to use **"View"**!

Chose **only 2 tools** to measure coverage: **Cobertura/JaCoCo**

Curious users of CI servers were **increasing gradually**.

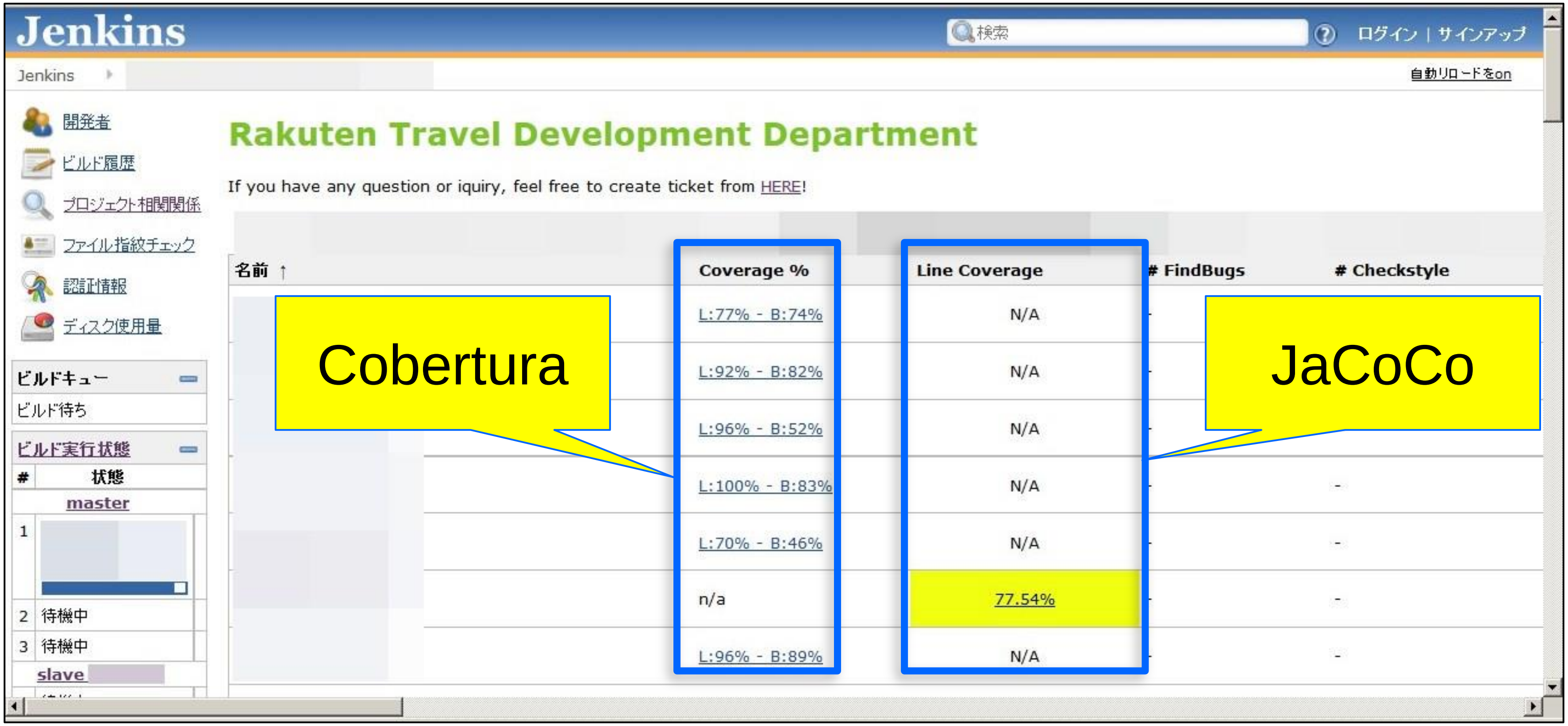


Master

Slaves

Go to the NEXTSTAGE!!

(1) Created “view” to show everyone current status



Rakuten Travel Development Department

If you have any question or iquiry, feel free to create ticket from HERE!

名前 ↑	Coverage %	Line Coverage	# FindBugs	# Checkstyle
	L:77% - B:74%	N/A	-	
	L:92% - B:82%	N/A	-	
	L:96% - B:52%	N/A	-	
	L:100% - B:83%	N/A	-	-
	L:70% - B:46%	N/A	-	-
	n/a	77.54%	-	-
	L:96% - B:89%	N/A	-	-

#	状態
1	
2	待機中
3	待機中

Cobertura

JaCoCo

(2) Recorded/reported the transition weekly

Total target jobs	74
Jobs which gather coverage	67
Jobs which achieved the target	35
Achievement ratio	47.3% (+18.9% WoW)

Emphasized results with weekly-basis comparisons.

(3) Organized the “65% Team” to achieve the goal

Target groups:
about 10

Assigned **curious users**
of CI servers
preferentially.

Aimed to achieve a goal
with them
collaboratively/effectively.

(3) Organized the “65% Team” to achieve the goal

Target groups:
about 10

We need to tell the truth…!

Assigned **curious users** of CI servers **preferentially**.

Aimed to achieve a goal with them **collaboratively/effectively**.

The true objective of “65% Team”

At first and officially, we organized it **to compensate for the lack of workers**.

The true objective was to turn over Jenkins Consolidation Team and its tasks!

- I needed to left the team at the end of the year.

The “65% Team” was **the target both to turn over tasks and to nurture**.

Technology-Driven Development: nurturing policy

Make the work efficient
by using Automation Techniques.

- e.g.) Infrastructure as Code

Drive learning by using
Automation Techniques/Project Metrics
with “inspect and adapt” style.

Develop cooperative relationships
by sharing goals/problems/progress
with Project Metrics/automatic notifications.

The early steering of the “65% Team”

Held meetings regularly/weekly
to share problems/knowledge face-to-face.

Told them metrics in detail/preferentially
to make them act faster.

Let them share problems/knowledge/metrics
to each team.

Let them talk to each other honestly
about problems.

The late steering of the “65% Team” (to nurture)

Let them focus on how to solve problems, rather than just reporting them.

Made them **accustom to do “inspect and adapt”** with the latest metrics.

Let them **hand off steering the team**.

- Urged their autonomy.

Let them **hand off our work gradually** based on Infrastructure as Code and OJT.

One day on November 2014

Should we do it, too?

I built this mechanism.
So please try it.

I’d like to help that team
to solve the problem.

Pay dirt…!

Growth of the “65% Team”

Improved UT coverage clearly!

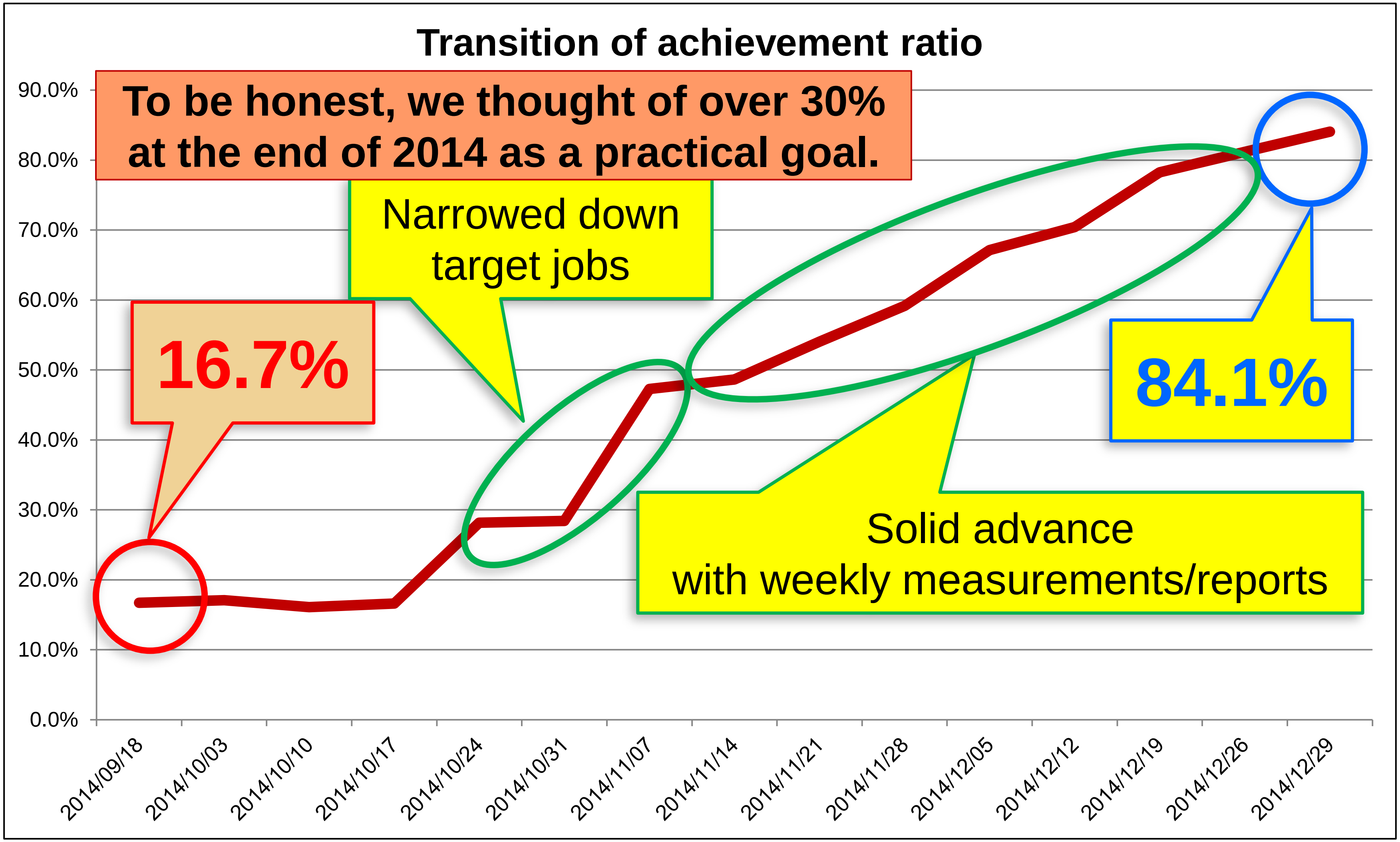
- Improvement of UT coverage generated additional motivation to improve it more!

Solved problems by themselves!

- With Infrastructure as Code and OJT
- Done **about 10 tasks** in December 2014

Started helping other teams voluntarily if they found problems.

Growth of the “65% Team” from Project Metrics



The impression of a series of improvements

Improvements never start without measuring current status and achievements numerically!

Total victory!

1. Challenges

2. What should we show as accomplishments?

3. How could we reduce the load of CI server?

4. How could we nurture members and teams?

5. Conclusion

Final accomplishments

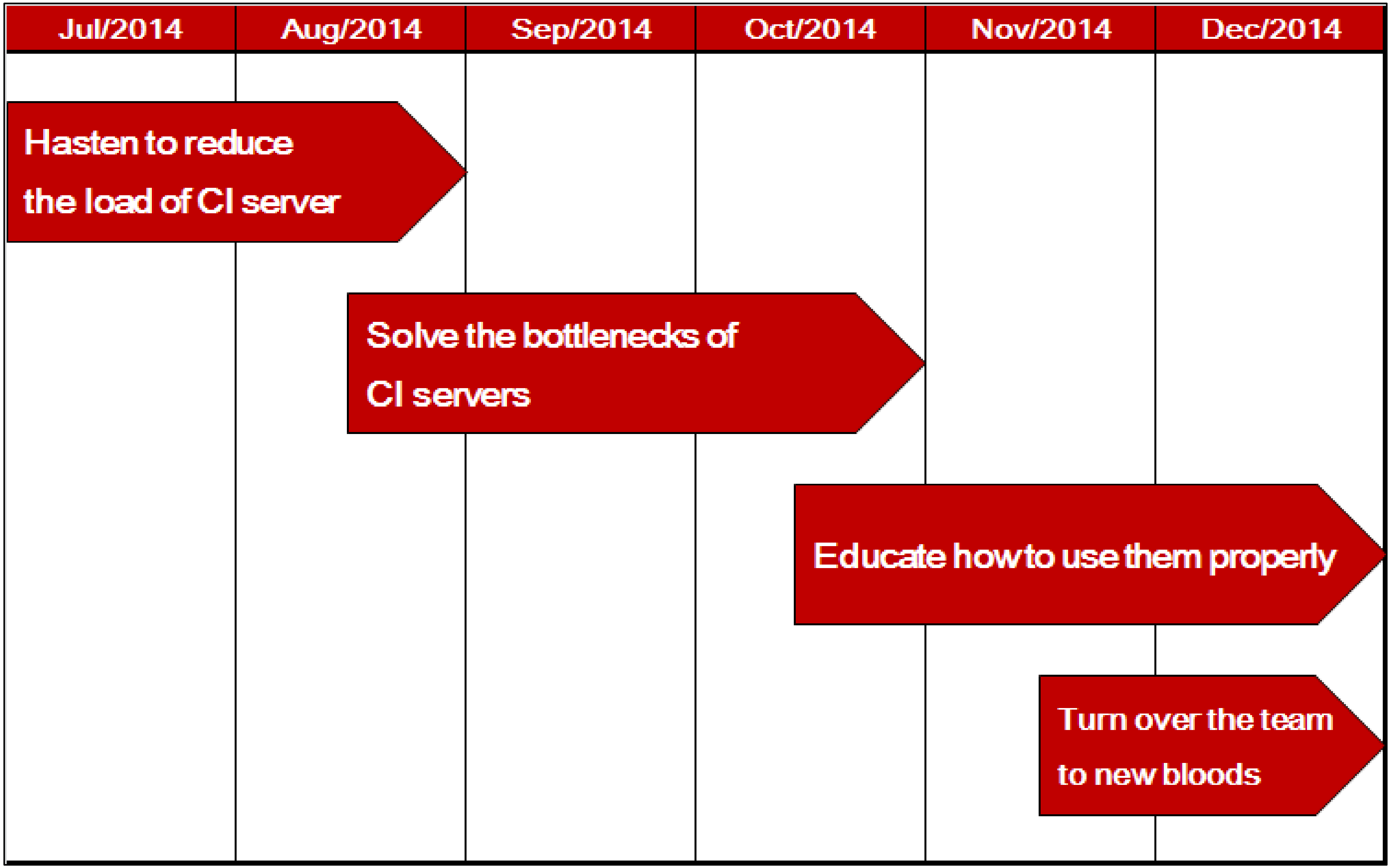
Delivered all products **on time**
and **prevented recurrence of delay**.

Scaled out CI servers (master/slave)

- **Over 550 jobs** are running

Turned over tasks to new bloods
by nurturing them.

Roadmap (planned)



Jul/2014 | Aug/2014 | Sep/2014 | Oct/2014 | Nov/2014 | Dec/2014

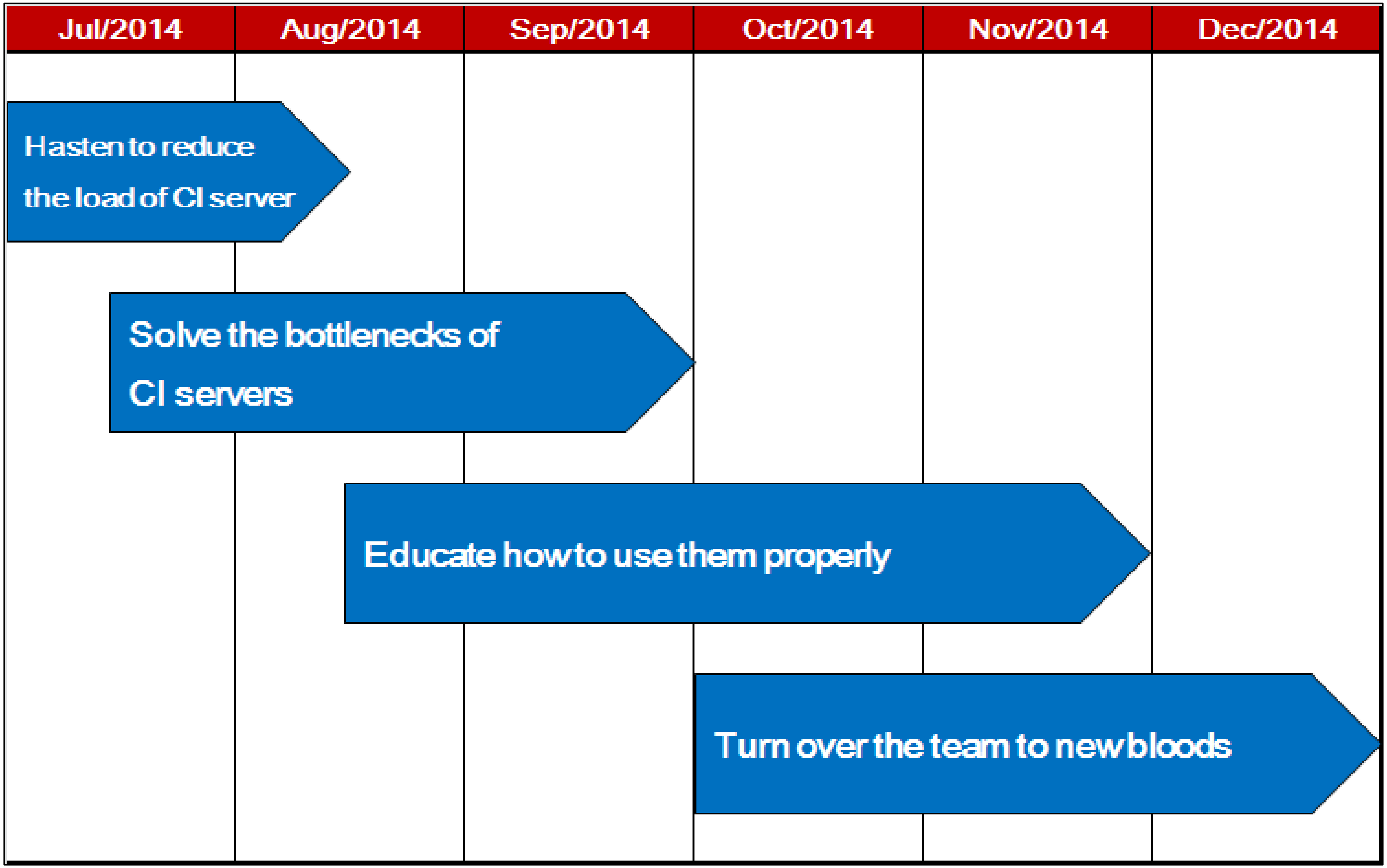
Hasten to reduce the load of CI server

Solve the bottlenecks of CI servers

Educate how to use them properly

Turn over the team to new bloods

Roadmap (actual)



Jul/2014

Aug/2014

Sep/2014

Oct/2014

Nov/2014

Dec/2014

Hasten to reduce the load of CI server

Solve the bottlenecks of CI servers

Educate how to use them properly

Turn over the team to new bloods

Remaining issues

IT/ST automation is not yet sufficient.

Data collision often occurs
on test environment.

- Solve by Immutable Infrastructure?

Release automation is not yet sufficient.

- Blue-green deployment?

New issues

- **Nearly gave some managers an easy mirage** that they can improve quality by only increasing UT coverage.
- **Often misunderstood that everyone can improve easily** due to the impact of our activities.

Agile for success

Run the PDCA cycle faster **by integrating strategy and (automation) techniques**.

Find/measure/review Project Metrics
to consolidate our PDCA cycle.

Nurture members/teams and
develop cooperative relationship **with above**.

The point of Project Metrics

Find useful information to think over

- Where is the current problem lurking?
- Was our action effective to solve problems?

Focus on the transition of numerical number

- Intelligence is on the transition of numerical number.
- If you find transition, you will win!

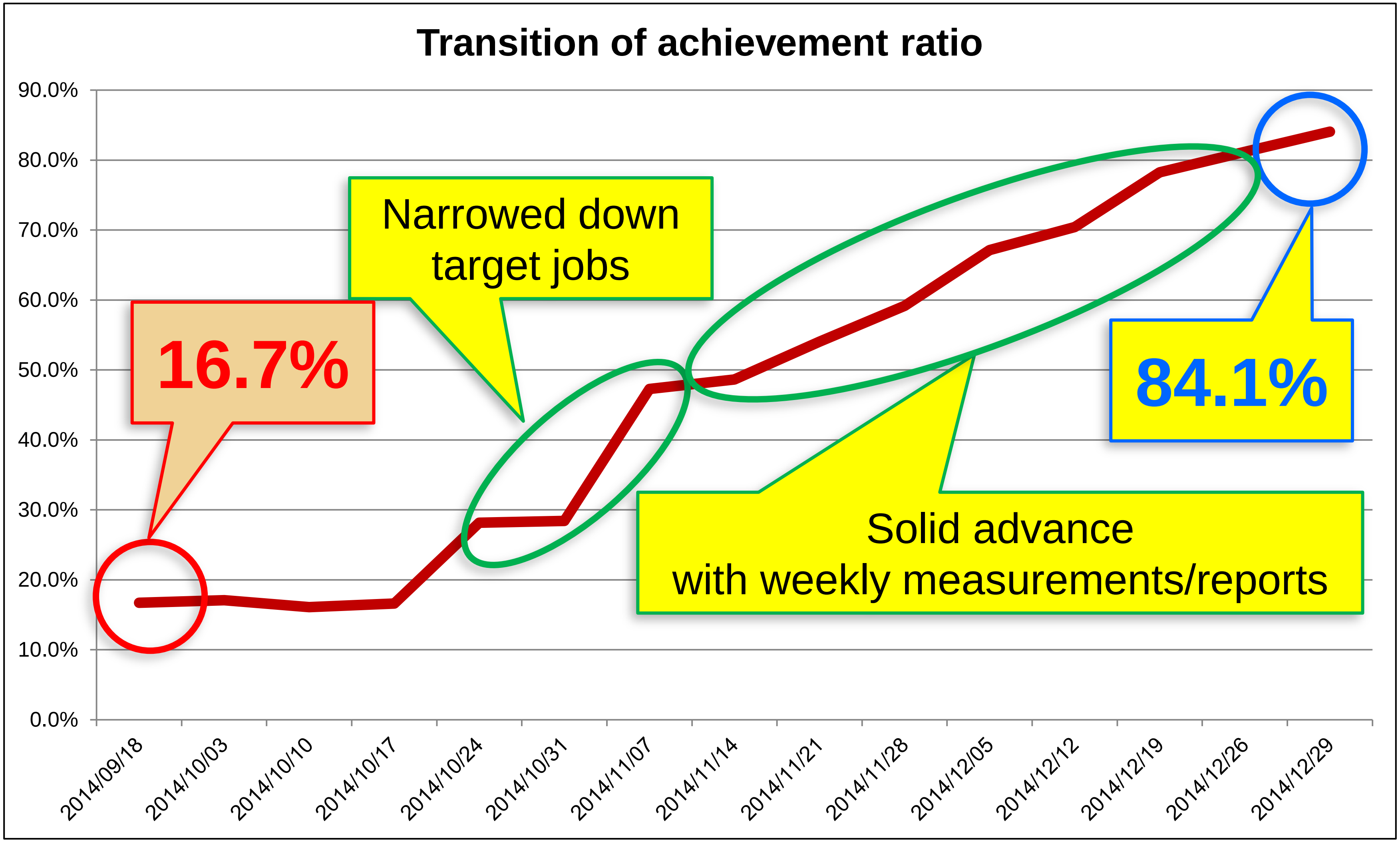
Review and improve them continuously

- Throw it away if it is useless.
- Create it if necessary.

Use them as a way of communication

- You can get additional ideas
  by talking members with Project Metrics.

It will start from measurement all.



Electrify your team!

Thank you very much!

@hageyahhoo (Twitter)

hageyahhoo (Facebook)

kazuhisa.naoi@rakuten.com

@naoinaoi (Twitter)

naoinaoi (Facebook)

Reference

“Useful Metrics in a Complex World” (Agile2014)
http://www.agilealliance.org/files/9814/0509/9343/ExperienceReport.2014.Power.pdf

“Moneyball for Software Projects” (Agile2014)
http://schd.ws/hosted_files/agile2014/f0/1272_Agile_2014_-_Software_Moneyball_%28Troy_Magennis%29.pdf

“Technology-Driven Development” (Agile2014)
http://www.agilealliance.org/files/5014/0509/9284/ExperienceReport.2014.Ito.pdf

“Visualize it with Project Metrics”
http://www.slideshare.net/ssuser968fab/xp-matsuri-2014ltthehiro