無為而治

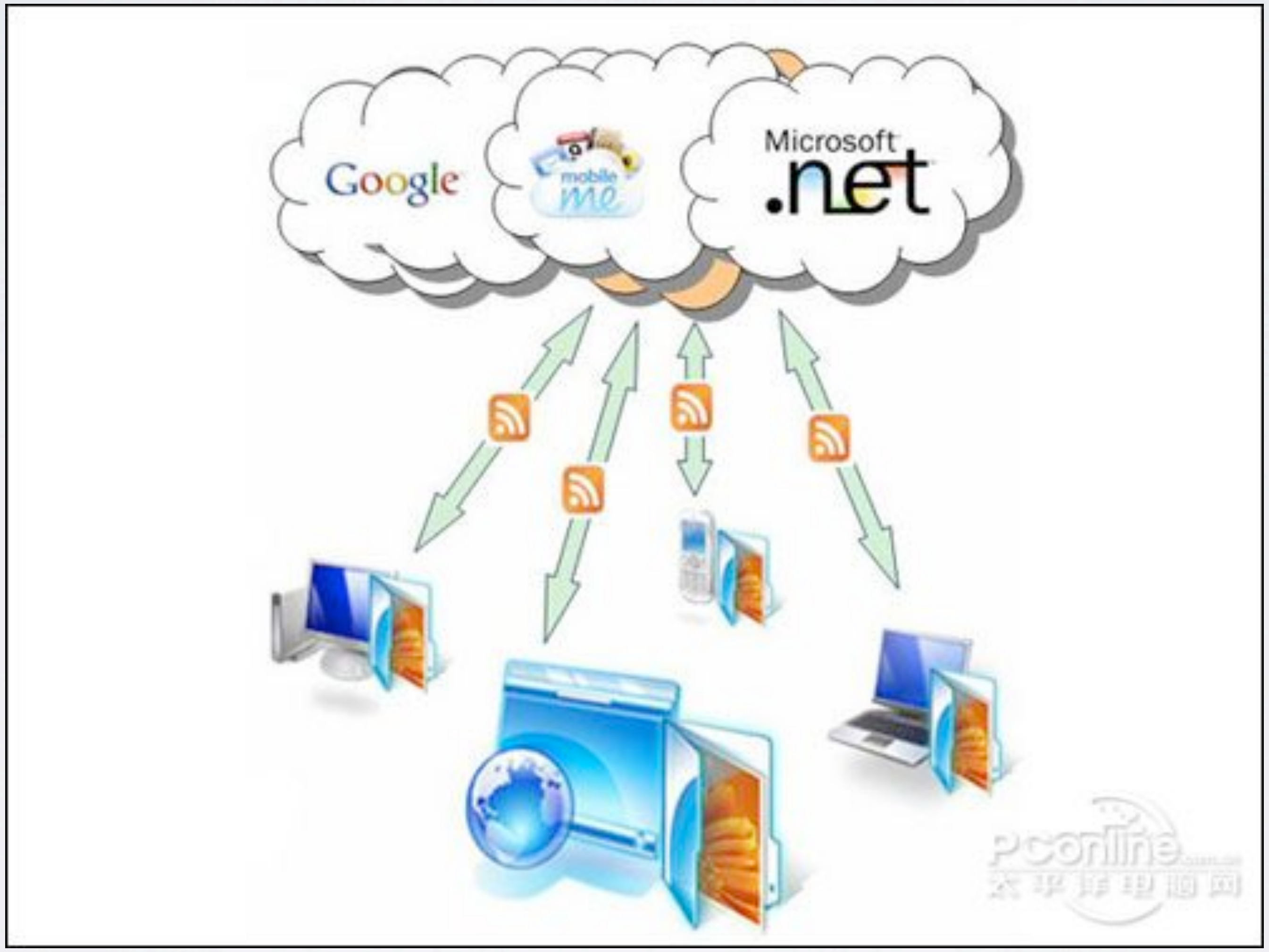
ファーストサーバ事件から話す

上海坦思計算機系統有限公司
王　洪雷

アジェンダ

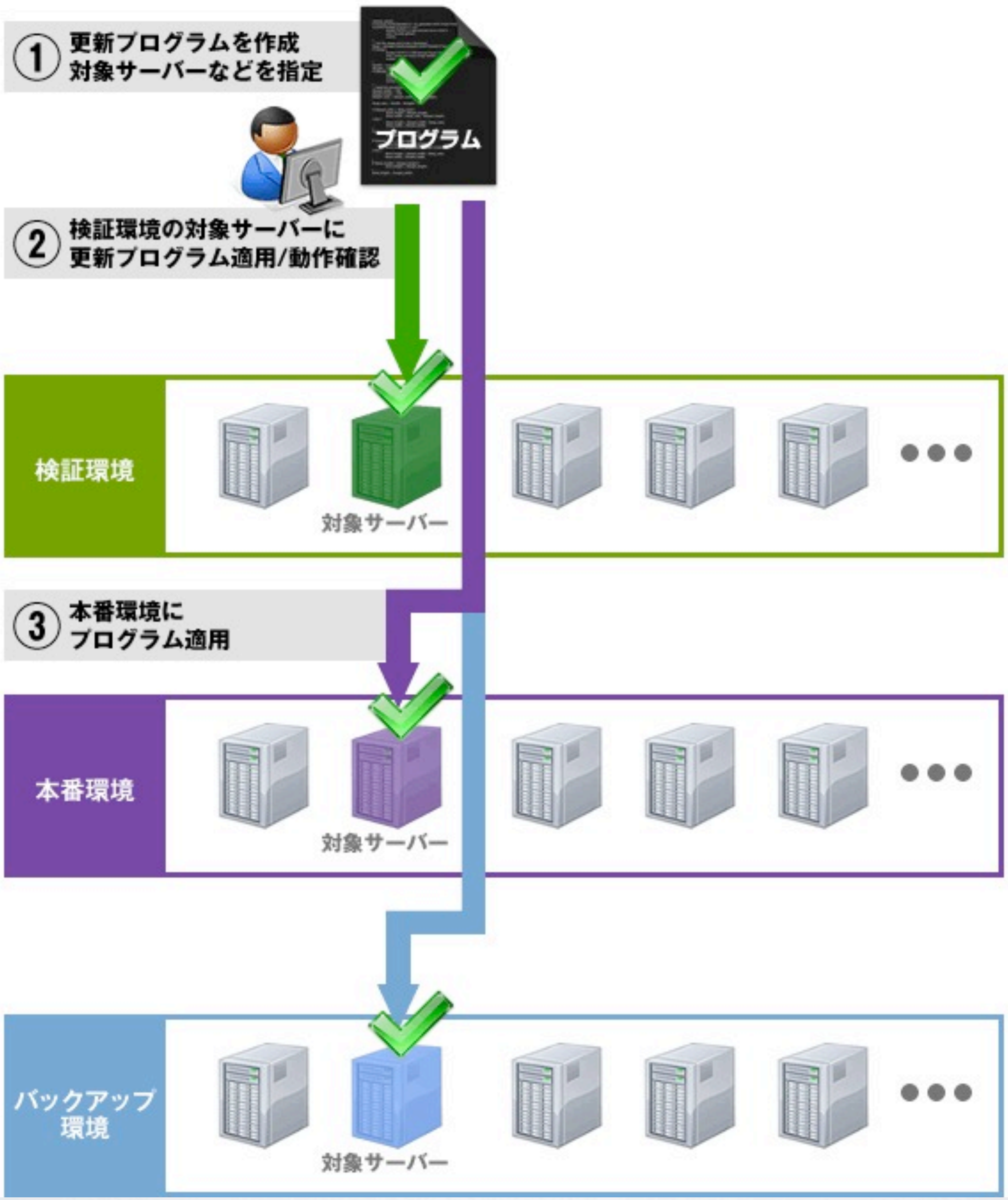
- ファーストサーバ事件の回顧
- 人間　vs　機器
- コーディング：Less is More
- レビュー：規約を減らす
- テスト：テストの作業量を減らす
- リリース：手順書不要

ファーストサーバ事件


Google
mobile me
Microsoft
.net
PConline
太平洋电脑网

ファーストサーバ事件

通常の状態



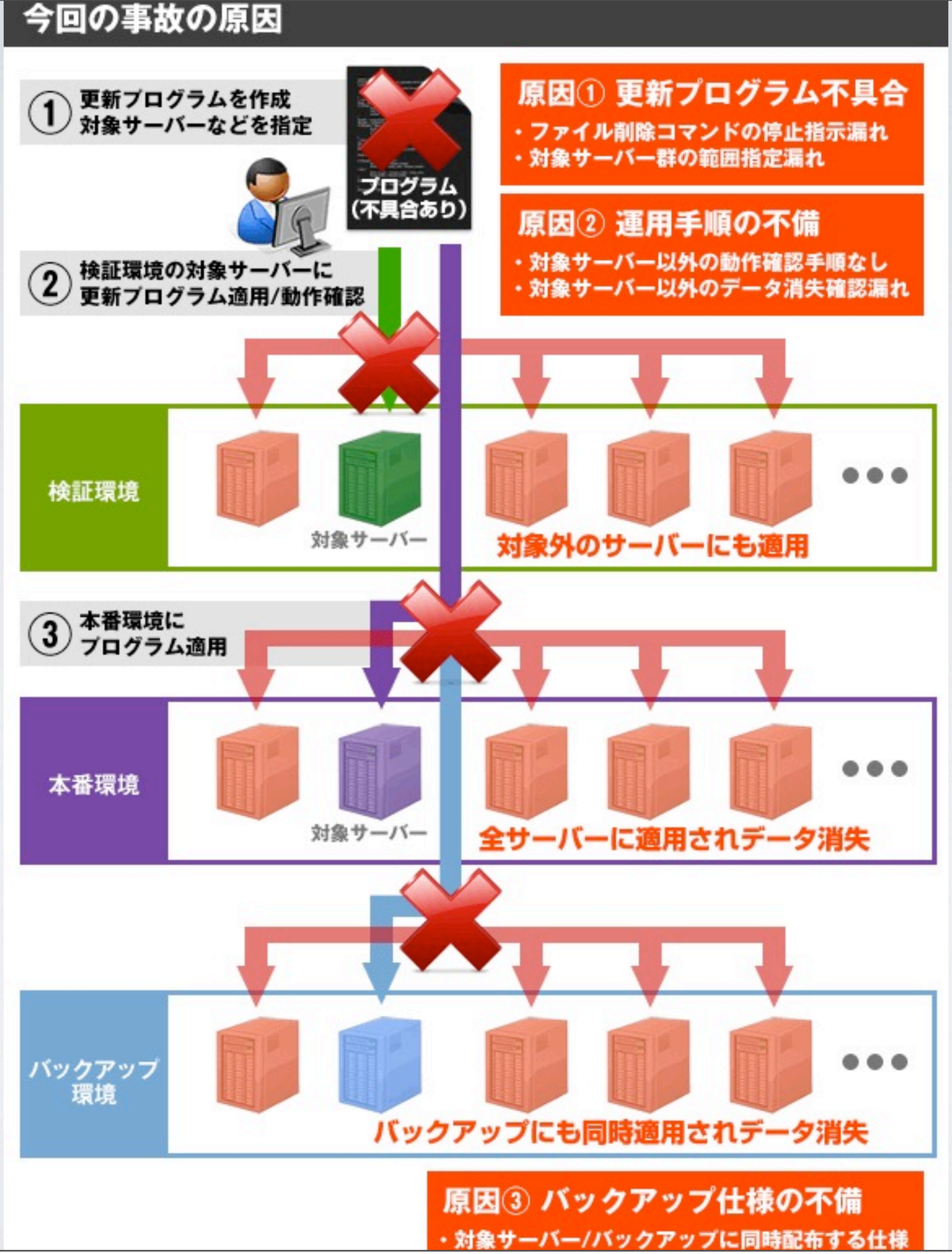
①更新プログラムを作成
対象サーバーなどを指定

②検証環境の対象サーバーに
更新プログラム適用/動作確認

③本番環境に
プログラム適用

プログラム
検証環境
対象サーバー
本番環境
対象サーバー
バックアップ
環境
対象サーバー

今回の事故の原因


① 更新プログラムを作成
対象サーバーなどを指定
プログラム
(不具合あり)
② 検証環境の対象サーバーに
更新プログラム適用/動作確認
検証環境
対象サーバー
対象外のサーバーにも適用
③ 本番環境に
プログラム適用
本番環境
対象サーバー
全サーバーに適用されデータ消失
バックアップ
環境
バックアップにも同時適用されデータ消失

原因① 更新プログラム不具合

- ファイル削除コマンドの停止指示漏れ
- 対象サーバー群の範囲指定漏れ

原因② 運用手順の不備

- 対象サーバー以外の動作確認手順なし
- 対象サーバー以外のデータ消失確認漏れ

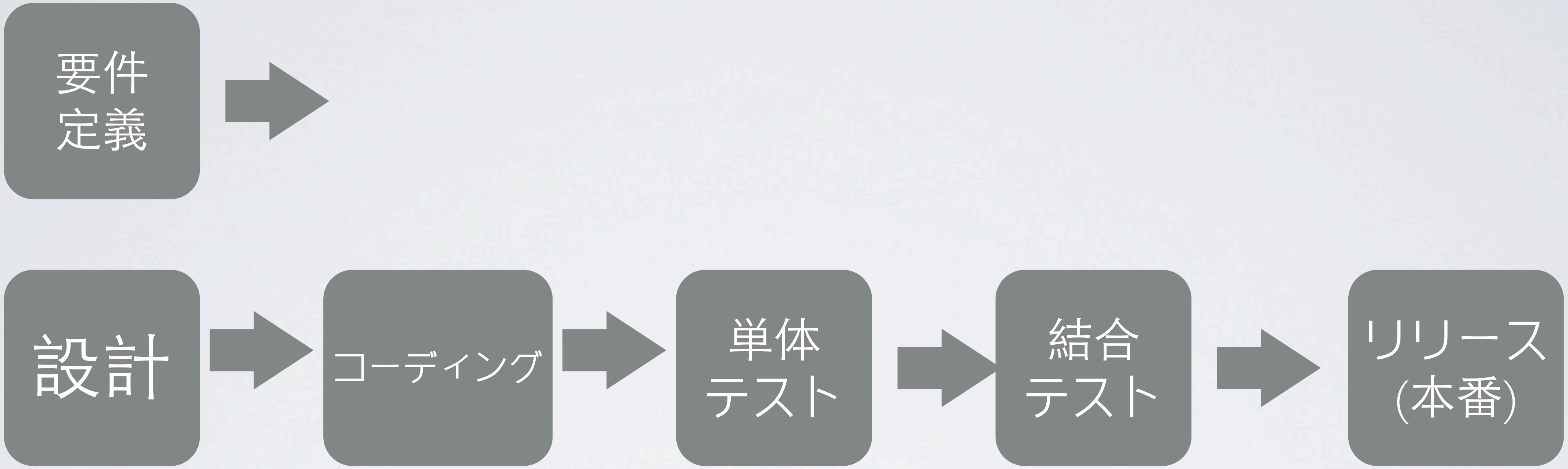
原因③ バックアップ仕様の不備

- 対象サーバー/バックアップに同時配布する仕様

人間　VS　機器

	人間	機器
計算 重複労働	遅い 安定しない	速い 安定
思考	創造性 想像力	できない

ソフト開発のプロセス



要件定義

設計

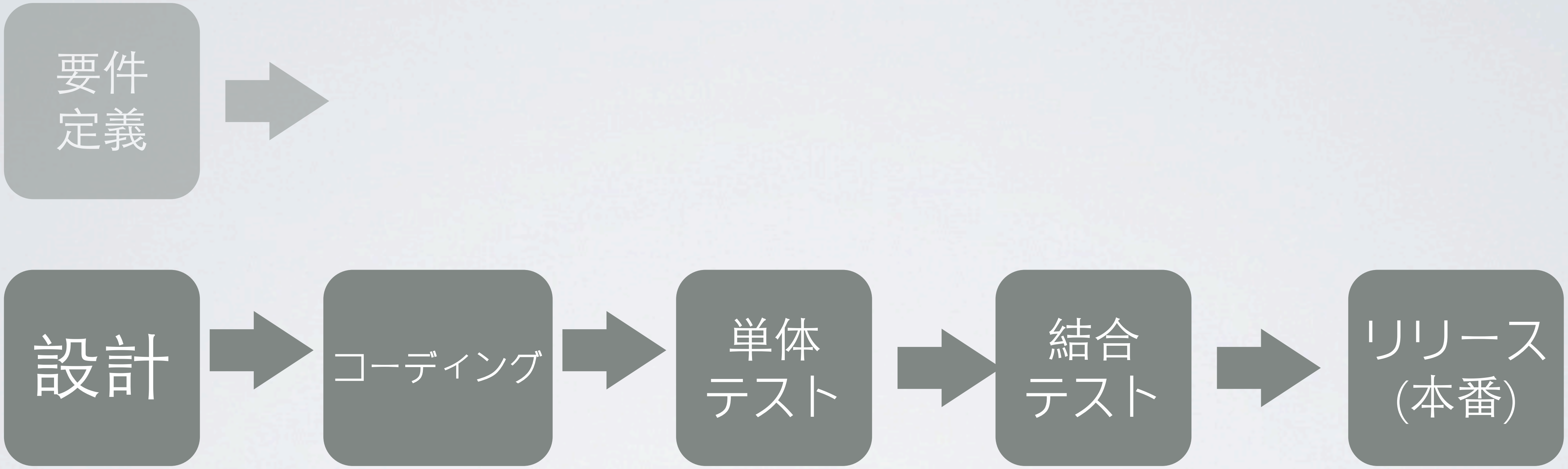
コーディング

単体テスト

結合テスト

リリース(本番)

ソフト開発のプロセス


要件
定義

設計

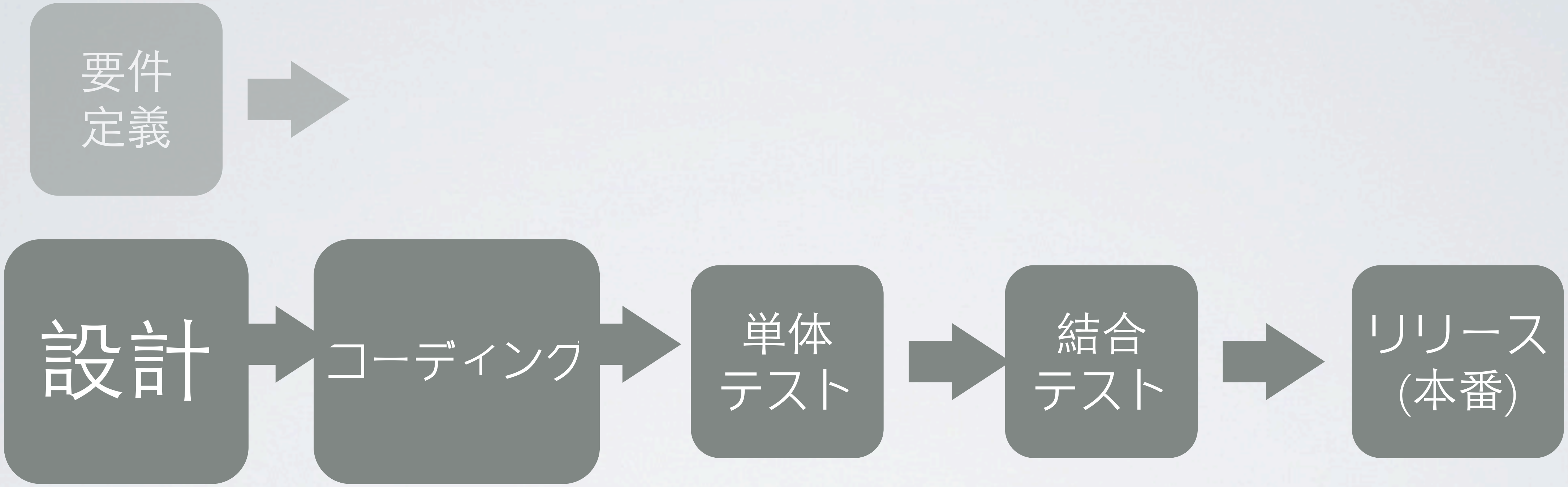
コーディング

単体
テスト

結合
テスト

リリース
(本番)

ソフト開発のプロセス



要件
定義

設計

コーディング

単体
テスト

結合
テスト

リリース
(本番)

言多必失

長くしゃべるとぼろが出る

- Strive to add function by deleting code.

- Strive to add function by deleting code.
- A designer knows he has achieved perfection not when there is nothing left to add, but when there is nothing left to take away. (Saint-Exupéry)

- Strive to add function by deleting code.
- A designer knows he has achieved perfection not when there is nothing left to add, but when there is nothing left to take away. (Saint-Exupéry)
- In software, the most beautiful code, the most beautiful functions, and the most beautiful programs are sometimes not there at all.

- Strive to add function by deleting code.
- A designer knows he has achieved perfection not when there is nothing left to add, but when there is nothing left to take away. (Saint-Exupéry)
- In software, the most beautiful code, the most beautiful functions, and the most beautiful programs are sometimes not there at all.
- Vigorous writing is concise. Omit needless words. (Strunk and White)

- Strive to add function by deleting code.
- A designer knows he has achieved perfection not when there is nothing left to add, but when there is nothing left to take away. (Saint-Exupéry)
- In software, the most beautiful code, the most beautiful functions, and the most beautiful programs are sometimes not there at all.
- Vigorous writing is concise. Omit needless words. (Strunk and White)
- The cheapest, fastest, and most reliable components of a computer system are those that aren’t there. (Bell)

- Endeavor to do more and more with less and less.

- Endeavor to do more and more with less and less.
- If I had more time, I would have written you a shorter letter. (Pascal)

- Endeavor to do more and more with less and less.
- If I had more time, I would have written you a shorter letter. (Pascal)
- The Inventor's Paradox: The more ambitious plan may have more chance of success. (Pólya)

- Endeavor to do more and more with less and less.
- If I had more time, I would have written you a shorter letter. (Pascal)
- The Inventor’s Paradox: The more ambitious plan may have more chance of success. (Pólya)
- Simplicity does not precede complexity, but follows it. (Perlis)

- Endeavor to do more and more with less and less.
- If I had more time, I would have written you a shorter letter. (Pascal)
- The Inventor's Paradox: The more ambitious plan may have more chance of success. (Pólya)
- Simplicity does not precede complexity, but follows it. (Perlis)
- Less is more. (Browning)

- Endeavor to do more and more with less and less.
- If I had more time, I would have written you a shorter letter. (Pascal)
- The Inventor’s Paradox: The more ambitious plan may have more chance of success. (Pólya)
- Simplicity does not precede complexity, but follows it. (Perlis)
- Less is more. (Browning)
- Make everything as simple as possible, but no simpler. (Einstein)

- Software should sometimes be seen as a soap bubble. (Perlis)

- Software should sometimes be seen as a soap bubble. (Perlis)
- Seek beauty through simplicity.

CODE IS A BY-PRODUCT

コードレビュー

- 静的コード解析ツールを活用
- コード規約の件数を減らして、理由を良く説明する

FXCOP

FXCOP

- COM (Interoperability) – rules that detect COM Interop issues.

FXCOP

- COM (Interoperability) – rules that detect COM Interop issues.
- Design – rules that detect potential design flaws. These coding errors typically do not affect the execution of your code.

FXCOP

- COM (Interoperability) – rules that detect COM Interop issues.
- Design – rules that detect potential design flaws. These coding errors typically do not affect the execution of your code.
- Globalization – rules that detect missing or incorrect usage of information related to globalization and localization.

FXCOP

- COM (Interoperability) – rules that detect COM Interop issues.
- Design – rules that detect potential design flaws. These coding errors typically do not affect the execution of your code.
- Globalization – rules that detect missing or incorrect usage of information related to globalization and localization.
- Naming – rules that detect incorrect casing, cross language keyword collisions, and other issues related to the names of types, members, parameters, namespaces, and assemblies.

FXCOP

FXCOP

- Performance – rules that detect elements in your assemblies that will degrade performance.

FXCOP

- Performance – rules that detect elements in your assemblies that will degrade performance.
- Security – rules that detect programming elements that leave your assemblies vulnerable to malicious users or code.

FXCOP

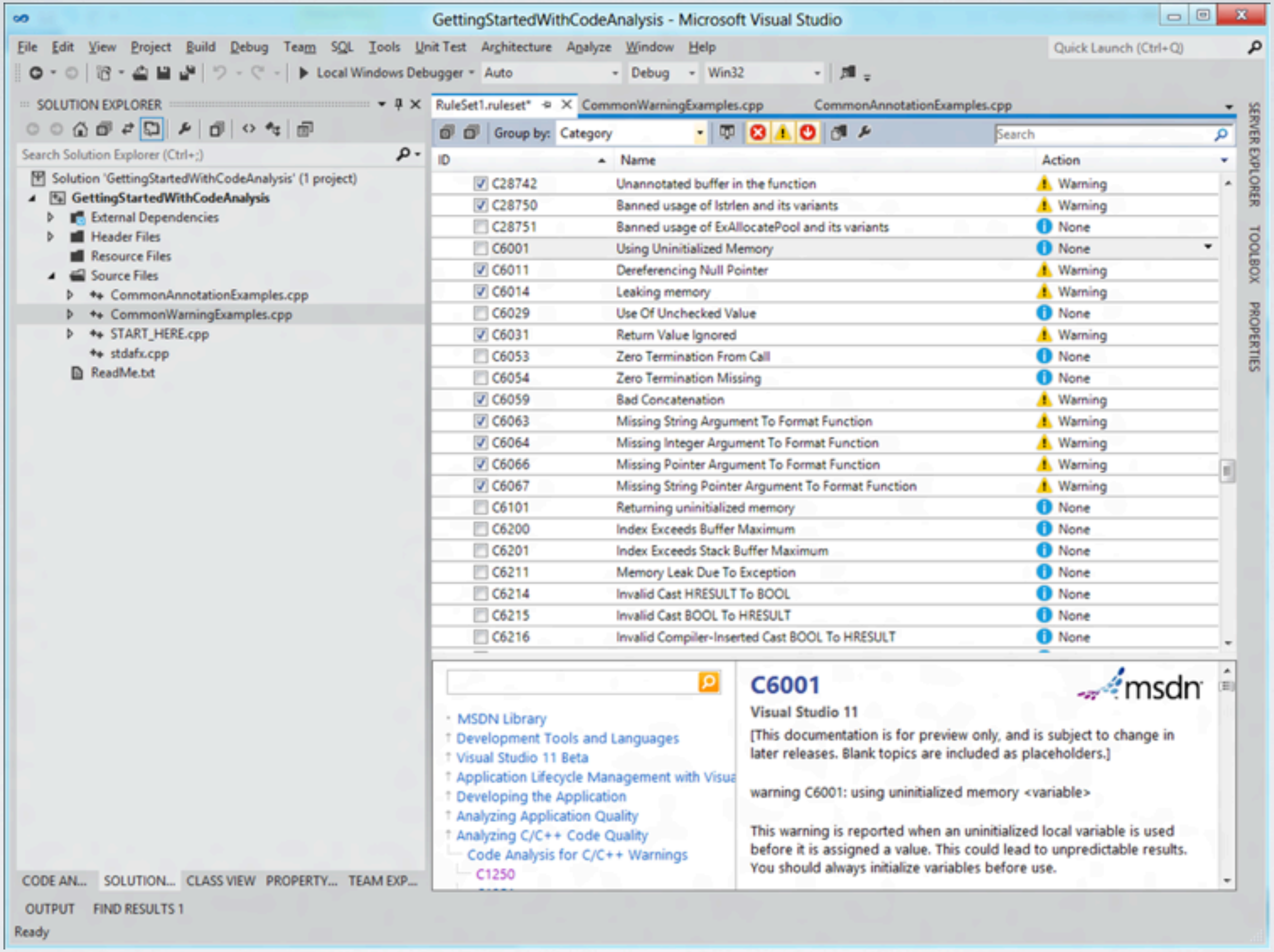
- Performance – rules that detect elements in your assemblies that will degrade performance.
- Security – rules that detect programming elements that leave your assemblies vulnerable to malicious users or code.
- Usage - rules that detect potential flaws in your assemblies that can affect code execution.

FXCOP

- Performance – rules that detect elements in your assemblies that will degrade performance.
- Security – rules that detect programming elements that leave your assemblies vulnerable to malicious users or code.
- Usage - rules that detect potential flaws in your assemblies that can affect code execution.
- Maintainability - rules that detect maintenance issues.

FXCOP

- Portability - rules that detect portability issues.
- Reliability- rules that detect correct memory and thread usage.


GettingStartedWithCodeAnalysis - Microsoft Visual Studio
File Edit View Project Build Debug Team SQL Tools Unit Test Architecture Analyze Window Help
Quick Launch (Ctrl+Q)
Local Windows Debugger
Auto
Debug
Win32
SOLUTION EXPLORER
Search Solution Explorer (Ctrl+;)
Solution 'GettingStartedWithCodeAnalysis' (1 project)
GettingStartedWithCodeAnalysis
External Dependencies
Header Files
Resource Files
Source Files
CommonAnnotationExamples.cpp
CommonWarningExamples.cpp
START_HERE.cpp
stdafx.cpp
ReadMe.txt
CODE AN...
SOLUTION...
CLASS VIEW
PROPERTY...
TEAM EXP...
OUTPUT
FIND RESULTS 1
Ready
RuleSet1.ruleset*
CommonWarningExamples.cpp
CommonAnnotationExamples.cpp
Group by:
Category
Search
SERVER EXPLORER
TOOLBOX
PROPERTIES
MSDN Library
Development Tools and Languages
Visual Studio 11 Beta
Application Lifecycle Management with Visua
Developing the Application
Analyzing Application Quality
Analyzing C/C++ Code Quality
Code Analysis for C/C++ Warnings
C1250
msdn

ID	Name	Action
☑ C28742	Unannotated buffer in the function	Warning
☑ C28750	Banned usage of lstrlen and its variants	Warning
☐ C28751	Banned usage of ExAllocatePool and its variants	None
☐ C6001	Using Uninitialized Memory	None
☑ C6011	Dereferencing Null Pointer	Warning
☑ C6014	Leaking memory	Warning
☐ C6029	Use Of Unchecked Value	None
☑ C6031	Return Value Ignored	Warning
☐ C6053	Zero Termination From Call	None
☐ C6054	Zero Termination Missing	None
☑ C6059	Bad Concatenation	Warning
☑ C6063	Missing String Argument To Format Function	Warning
☑ C6064	Missing Integer Argument To Format Function	Warning
☑ C6066	Missing Pointer Argument To Format Function	Warning
☑ C6067	Missing String Pointer Argument To Format Function	Warning
☐ C6101	Returning uninitialized memory	None
☐ C6200	Index Exceeds Buffer Maximum	None
☐ C6201	Index Exceeds Stack Buffer Maximum	None
☐ C6211	Memory Leak Due To Exception	None
☐ C6214	Invalid Cast HRESULT To BOOL	None
☐ C6215	Invalid Cast BOOL To HRESULT	None
☐ C6216	Invalid Compiler-Inserted Cast BOOL To HRESULT	None

C6001

Visual Studio 11

[This documentation is for preview only, and is subject to change in later releases. Blank topics are included as placeholders.]

warning C6001: using uninitialized memory <variable>

This warning is reported when an uninitialized local variable is used before it is assigned a value. This could lead to unpredictable results. You should always initialize variables before use.

JAVA 靜態分析

JAVA 靜態分析

- PMD （http://pmd.sourceforge.net/）

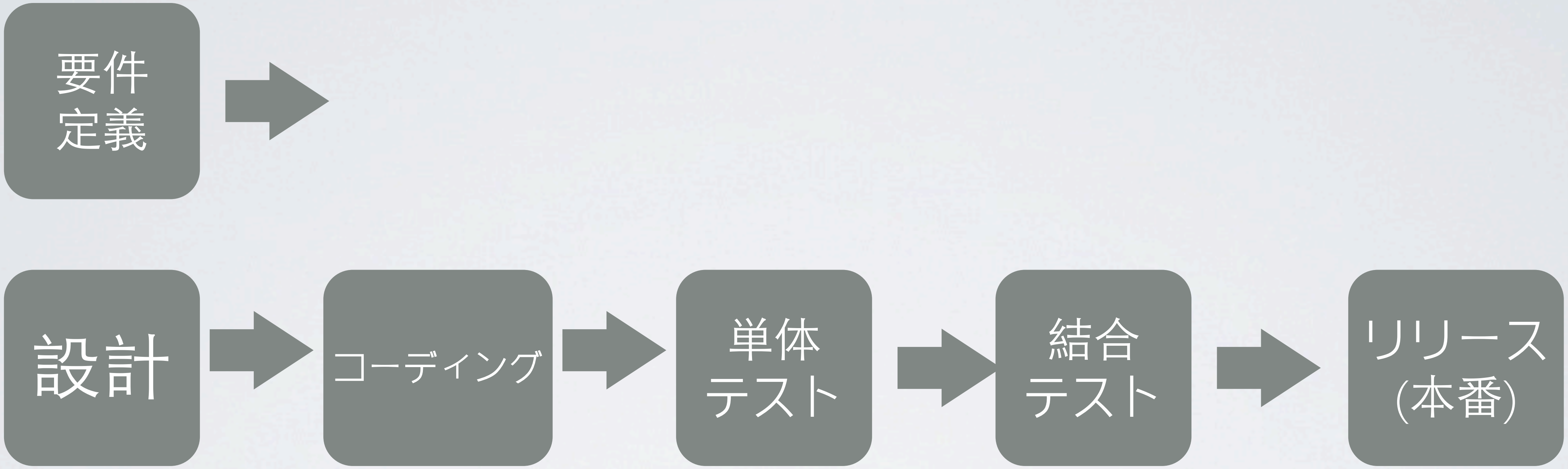
JAVA 靜態分析

- PMD　（http://pmd.sourceforge.net/）
- Checkstyle　（http://checkstyle.sourceforge.net/）

JAVA 靜態分析

- PMD （http://pmd.sourceforge.net/）
- Checkstyle （http://checkstyle.sourceforge.net/）
- FindBugs（http://findbugs.sourceforge.net/

ソフト開発のプロセス



要件定義

設計

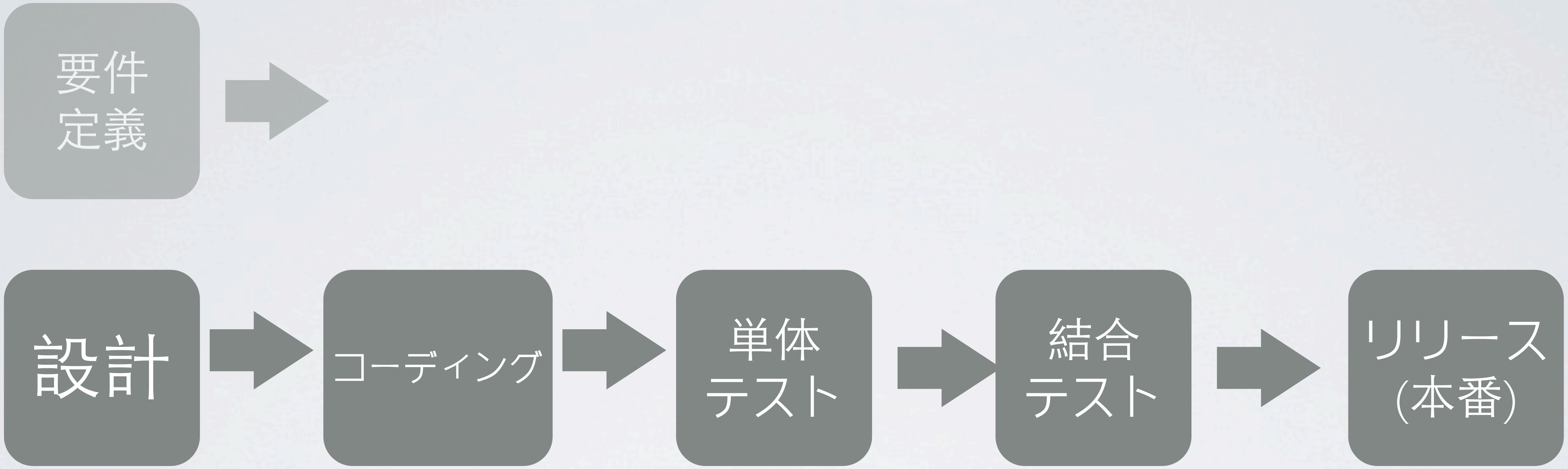
コーディング

単体テスト

結合テスト

リリース(本番)

ソフト開発のプロセス



要件
定義

設計

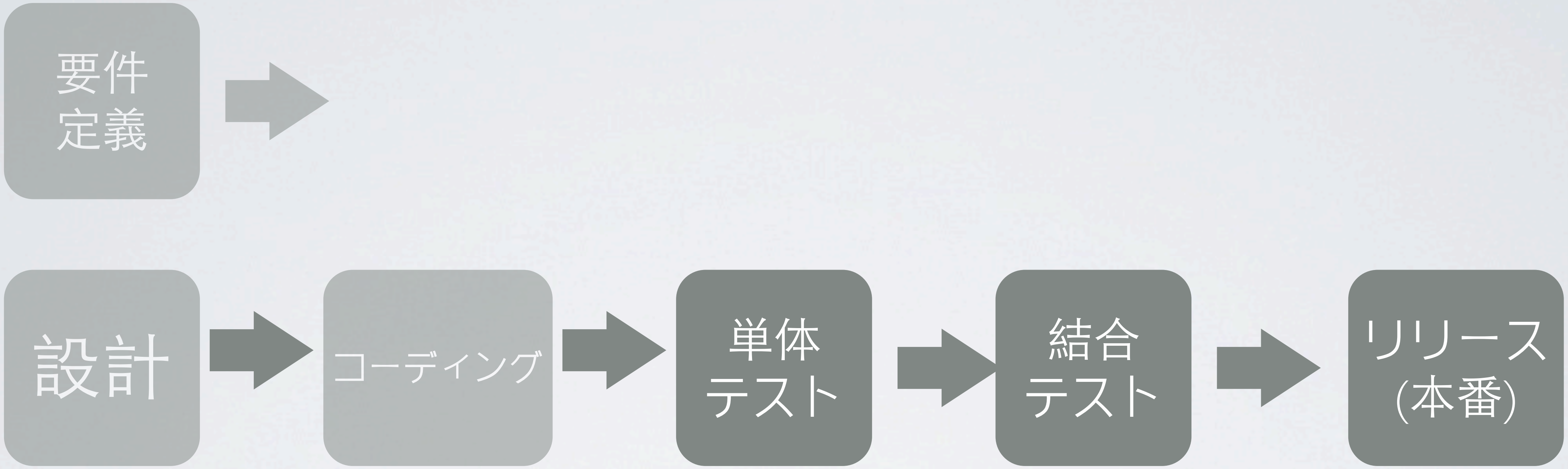
コーディング

単体
テスト

結合
テスト

リリース
(本番)

ソフト開発のプロセス



要件
定義

設計

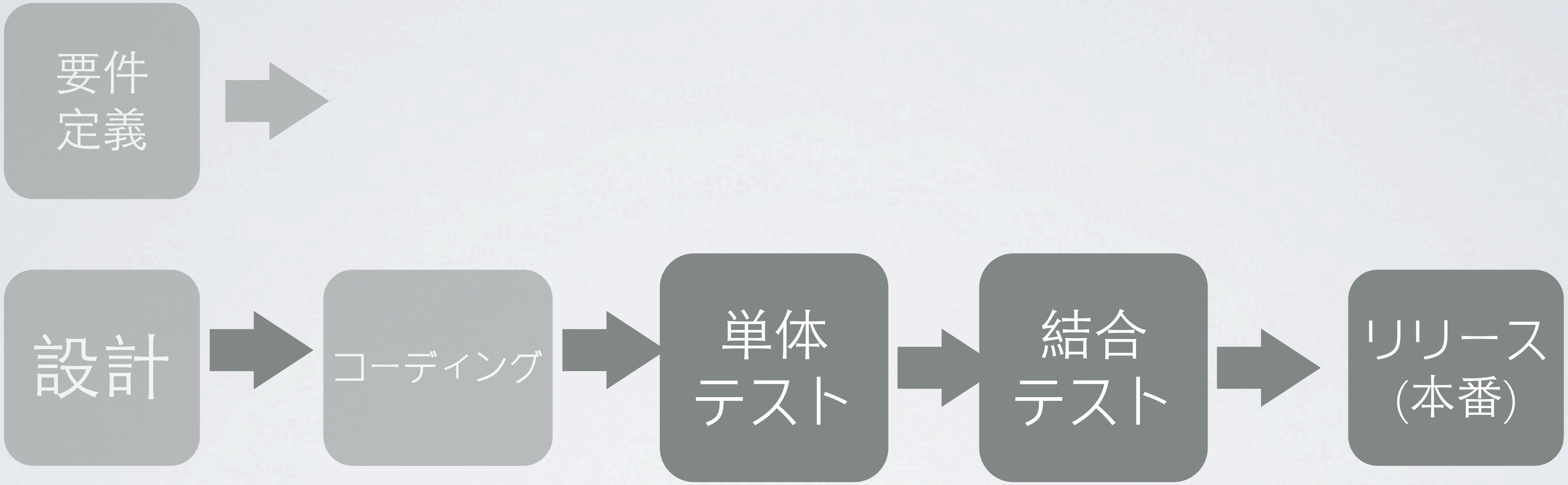
コーディング

単体
テスト

結合
テスト

リリース
(本番)

ソフト開発のプロセス



要件
定義

設計

コーディング

単体
テスト

結合
テスト

リリース
(本番)

テスト

テスト

- テスト = エビデンスの張付け?

テスト

- テスト = エビデンスの張付け?
- テストの跳ばす

テスト

- テスト = エビデンスの張付け?
- テストの跳ばす
- テスト結果の確認漏れ

テスト

- テスト = エビデンスの張付け?
- テストの跳ばす
- テスト結果の確認漏れ
- テストパタンの不足

自動化テストツール

- JUnit/NUnit
- Phantom.js
- should.js
- WebGhost

IOS 自動化テストツール

- UI Automation (Apple)
- Fone Monkey
- Instrument Driver

結合テスト

結合テスト

- ディグレッド問題

結合テスト

- ディグレッド問題
- 継続的インテグレーション

結合テスト

- ディグレッド問題
- 継続的インテグレーション
- Jenkis + hockey

結合テスト

- ディグレッド問題
- 継続的インテグレーション
- Jenkis + hockey
- Team Foundation Server

リリース自動化

リリース自動化

- リリースのスクリプト化

リリース自動化

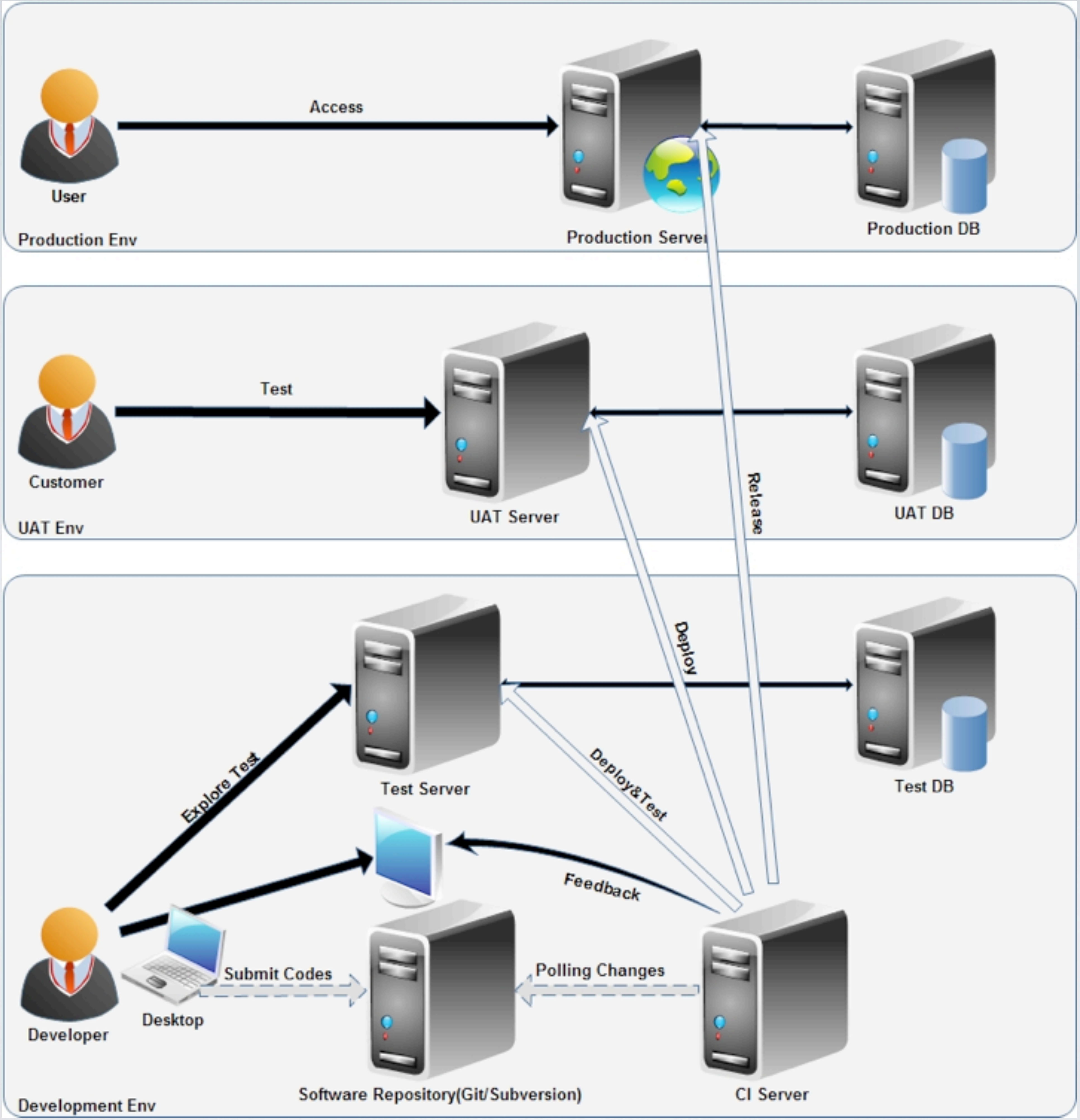
- リリースのスクリプト化
- テスト環境と本番環境の一致性を確保

リリース自動化

- リリースのスクリプト化
- テスト環境と本番環境の一致性を確保
- ロールバック自動化

リリース自動化

- リリースのスクリプト化
- テスト環境と本番環境の一致性を確保
- ロールバック自動化
- バックアップの復元できることを確認


User
Access
Production Server
Production DB
Production Env
Customer
Test
UAT Server
UAT DB
UAT Env
Release
Deploy
Test Server
Test DB
Explore Test
Deploy&Test
Feedback
Developer
Desktop
Submit Codes
Software Repository(Git/Subversion)
Polling Changes
CI Server
Development Env

まとめ

まとめ

- ファーストサーバ事件の回顧

まとめ

- ファーストサーバ事件の回顧
- 人間　vs　機器

まとめ

- ファーストサーバ事件の回顧
- 人間　vs　機器
- コーディング：Less is More

まとめ

- ファーストサーバ事件の回顧
- 人間　vs　機器
- コーディング：Less is More
- レビュー：静態分析ツールを活用し、ルールを減らす

まとめ

- ファーストサーバ事件の回顧
- 人間　vs　機器
- コーディング：Less is More
- レビュー：静態分析ツールを活用し、ルールを減らす
- テスト：自動化テストツールを活用、作業量を減らす

まとめ

- ファーストサーバ事件の回顧
- 人間　vs　機器
- コーディング：Less is More
- レビュー：静態分析ツールを活用し、ルールを減らす
- テスト：自動化テストツールを活用、作業量を減らす
- リリース：手順書不要

まとめ

為

まとめ

コンピュータ

為

速い
ルルーが守れる

まとめ

コンピュータ

為

速い
ルルーが守れる

人間

無為

創造性を発揮
ルルーを作る

無為而治

むいをしておさめる

ありがとうございます

上海坦思計算機系統有限公司
王　洪雷
WHL@tes-sys.com

2012/10/27