ソフトウェア構造解析のテスト設計への応用

RICOH
imagine. change.

SPI Japan 2018

リコーITソリューションズ株式会社
エンベデッドソリューション事業部 IT検証部
森 孝 司・増 子 聡・狩 野 薫

ビースラッシュ株式会社
山田大介

2018年10月11日

会社紹介 - 概要 -

RICOH
imagine. change.

会社名	リコーＩＴソリューションズ株式会社
設立年月日	1982年10月5日
本社事業所	神奈川県横浜市都筑区新栄町16-1
代表取締役 社長執行役員	石野 普之（いしの ひろゆき）
資本金	2億 5000万円
従業員数	958名 （2018年4月1日 現在）
主要事業	エンベデッド事業（組み込みソフトウェア開発,IT検証サービス） ソリューション事業, グループIT事業
事業所	北見/ 札幌/ 秋田/ 本社/ 新横浜/ 金沢/ 鳥取/ 鹿児島 ： 計 8拠点

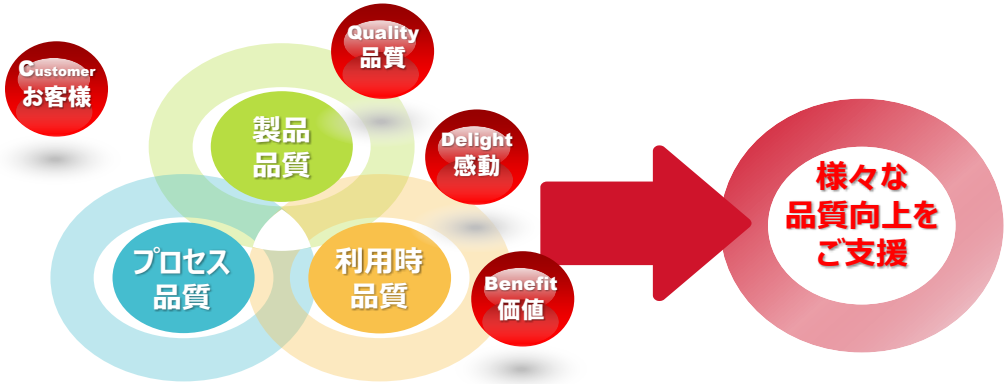
リコーITソリューションズ株式会社
代表取締役　社長執行役員
石野　普之

会社紹介 - IT検証サービス -

RICOH
imagine. change.

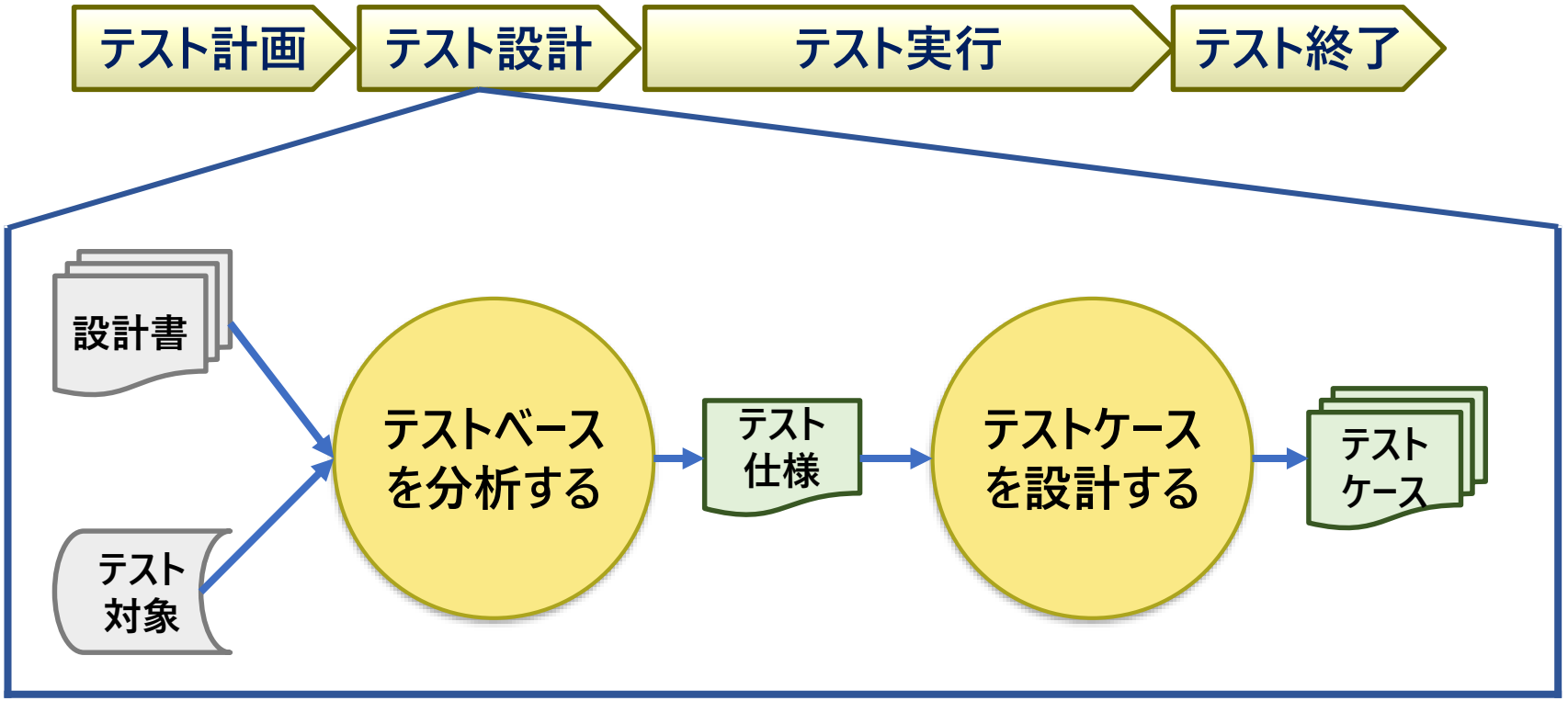
上流からのIT検証サービス

検証技術やプロセス改善等、リコーグループ内で培ったIT技術を駆使し、製品品質のみならずプロセス品質や利用時品質を含む品質全般の向上を支援します。あわせてIT検証のプロフェッショナル集団として、「品質の見える化」や上流工程からの品質マネジメントを支援します。



背景

RICOH
imagine. change.

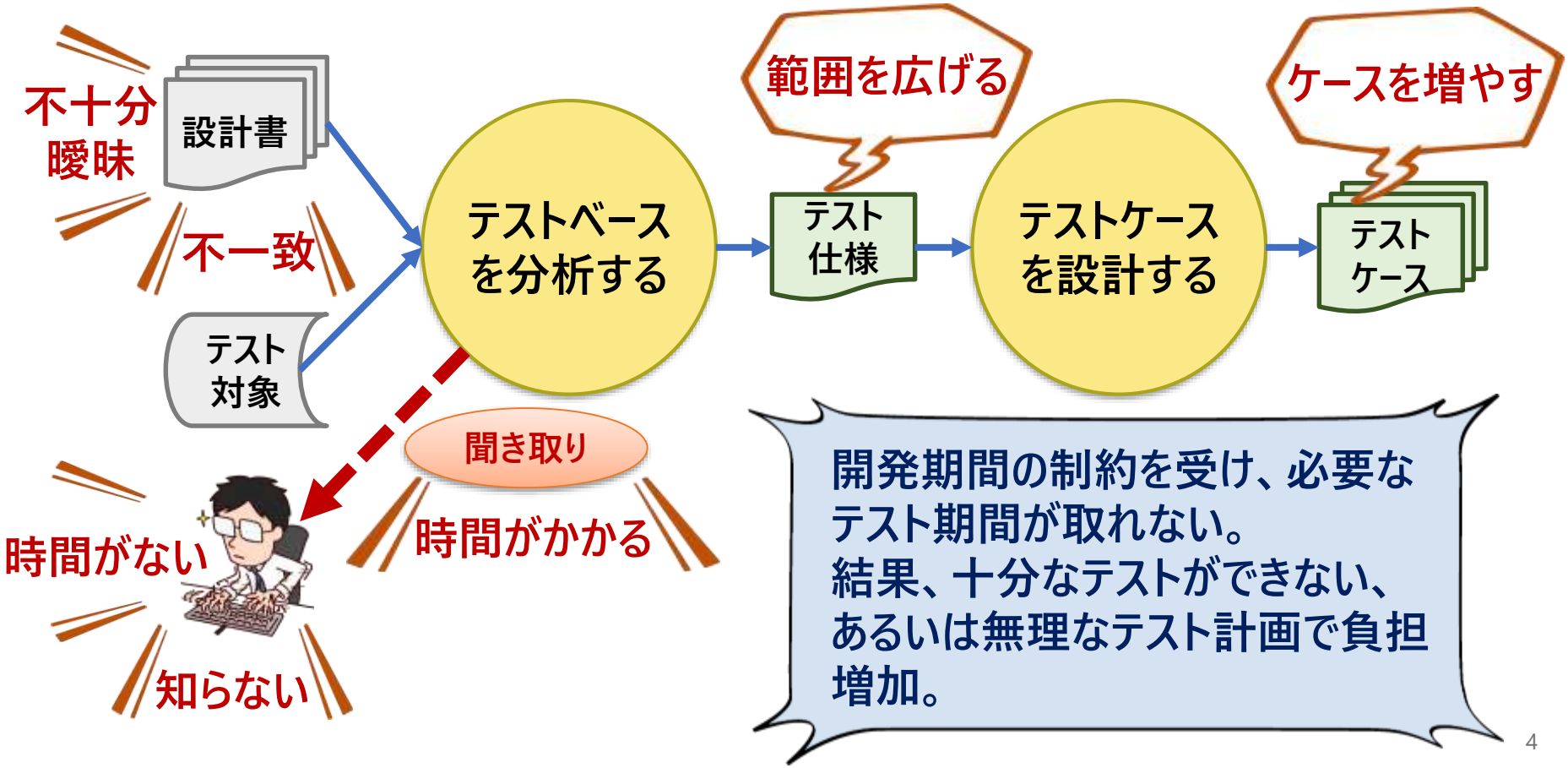


テスト設計プロセスとは

ソフトウェア要件、ソフトウェア構造図、テスト対象等のテストベースを基に、アプローチ方法、範囲等のテスト仕様とテストケースの設計を行う

背景

- 不十分あるいは曖昧な設計書
- 統合テストや探索テストのニーズ増加



課題

RICOH
imagine. change.

1. 設計文書がなく、ソフトウェアエンジニアの協力が得られない場合でも、、テスト範囲を客観的に判断する
   ① ソフトウェアの構造を把握する
   ② 欠陥発生箇所を予測する

2. １で得られたソフトウェア構造と欠陥発生予測よりテスト設計の十分さを客観的に判断する

3. テスト設計の質を上げ、かつテストエンジニアの負担を軽減する（テスト設計プロセスの改善）

4. テスト活動の成果をソフトウェア設計の改善に繋げる

方針

RICOH
imagine. change.

前提条件

- 設計書のソフトウェア構造図は当てにしない
- 当てになるのは実際に動作するもの（ソースコード）
- 結合テスト以降のテスト設計が対象

アプローチ

次の視点で静的解析技法を使い、テスト対象を分析する

- ソフトウェア構造…コンポーネントの依存関係
- 欠陥発生予測…プログラム上複雑な箇所

ソフトウェア構造の把握

RICOH
imagine. change.

ソフトウェア構造解析の狙い

- フォルダ、ソースコード、クラスなどの依存関係とインターフェースの所在を可視化すること

構造を可視化する方式

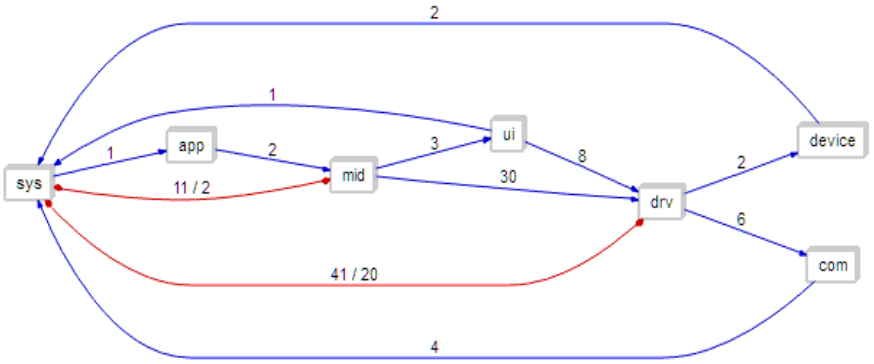
- DSM（Dependency Structure Matrix）
- Butteflyグラフ
- コンポジット構造図(Composite Structure Diagram)

ソフトウェア構造可視化の方式比較

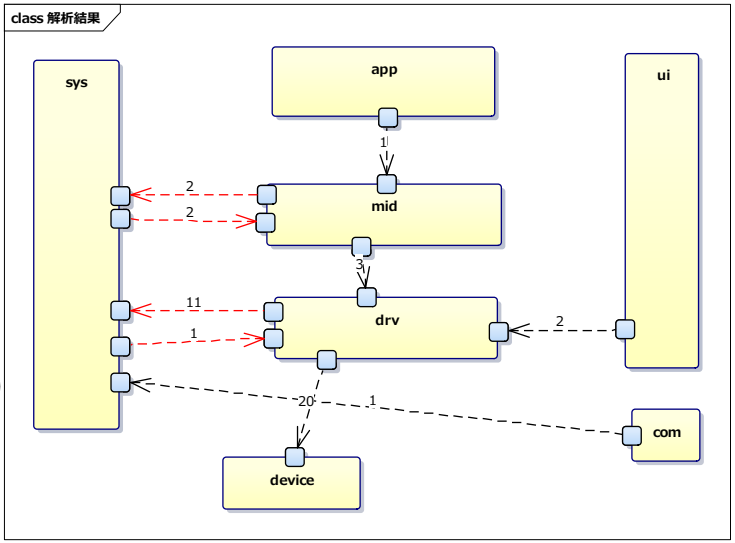
RICOH
imagine. change.

$root		RELATIVE::src.app 1	RELATIVE::src.mid 2	RELATIVE::src.ui 3	RELATIVE::src.drv 4	RELATIVE::src.com 5	RELATIVE::src.device 6	RELATIVE::src.sys 7
RELATIVE::src	RELATIVE::.src.app 1	12%						1
	RELATIVE::.src.mid 2	2	12%					4
	RELATIVE::.src.ui 3		1	4%				
	RELATIVE::.src.drv 4		11	2	38%			9
	RELATIVE::.src.com 5				1	8%		
	RELATIVE::.src.device 6				2		6%	
	RELATIVE::.src.sys 7		4	2	33	3	3	19%

DSM



Butteflyグラフ



コンポジット構造図

コンポーネント、ポート、インターフェース等のソフトウェア構造をソフトウェアモデルとして直感的に読取ることができる

欠陥発生箇所の予測

RICOH
imagine. change.

- メトリクスによる定量的な欠陥予測の研究事例
  - Daniel J. Sturtevant, System Design and the Cost of Architectural Complexity
  - Daniel J. Sturtevant, Ph.D., Technical Debt in Large Systems:Understanding the cost of software complexity
  - 綾井環・小向順・福永一寛, ソフトウェア構造を見れば品質がわかる！構造メトリクスとバクの関係

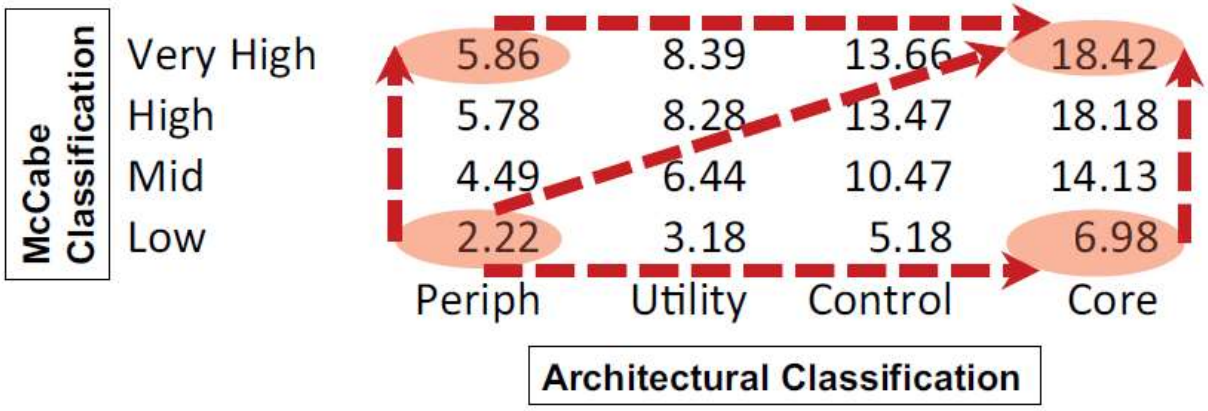
ソフトウェア構造と欠陥発生の関連

RICOH
imagine. change.

- 複雑度が増すと欠陥密度が増加する
- 複雑度が増すと潜在的コードエラーが増加する

引用：ソフトウェア構造を見れば品質がわかる！構造メトリクスとバクの関係(スライド22)

- Sturtevantの欠陥予測シミュレーション
  循環的複雑度（McCabe Classification）と構造的複雑度（Architectural Classification）より欠陥を予測する



引用：Technical Debt in Large Systems:Understanding the cost of software complexity (スライド39)

欠陥発生リスクの定義

RICOH
imagine. change.

リスクの定量化

- 各4段階の循環的複雑度と構造的複雑度よりリスクの大きさを16段階で定義する

循環複雑度	区分
1～10	無
11～20	低
21～50	中
51～	高

構造複雑度		区分
FanIn	FanOut	
Low	Low	無
High	Low	低
Low	High	中
High	High	高

循環複雑度					
	高	7	10	13	16
	中	5	9	12	15
	低	3	6	11	14
	無	1	2	4	8
		無	低	中	高
		構造複雑度			

欠陥発生リスクの影響範囲

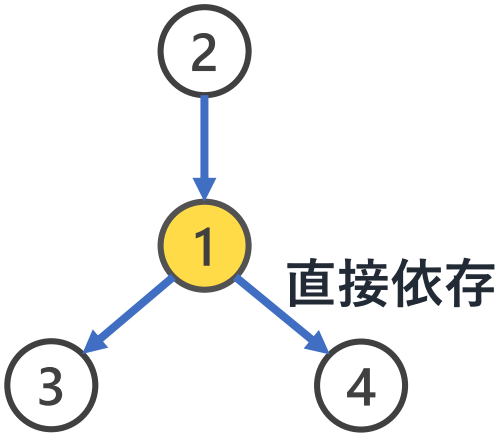
RICOH
imagine. change.

- ソースコード

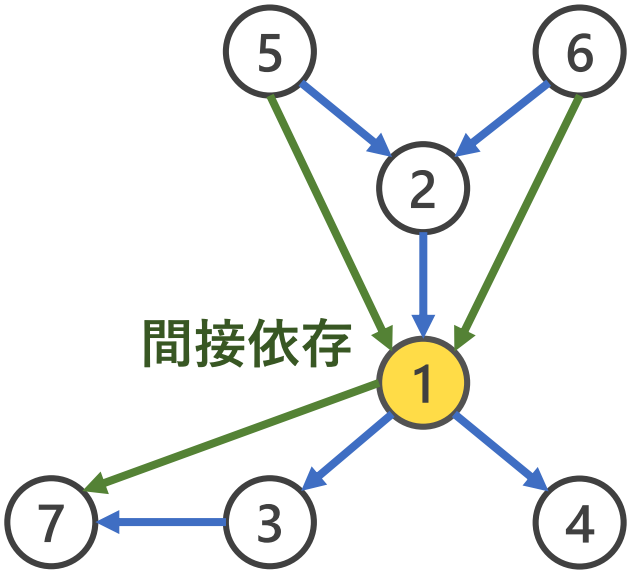
  ソースコードの直接的な依存関係が影響する範囲

- システム

  複数コンポーネントの間接的な依存関係が影響する範囲



ソースコード

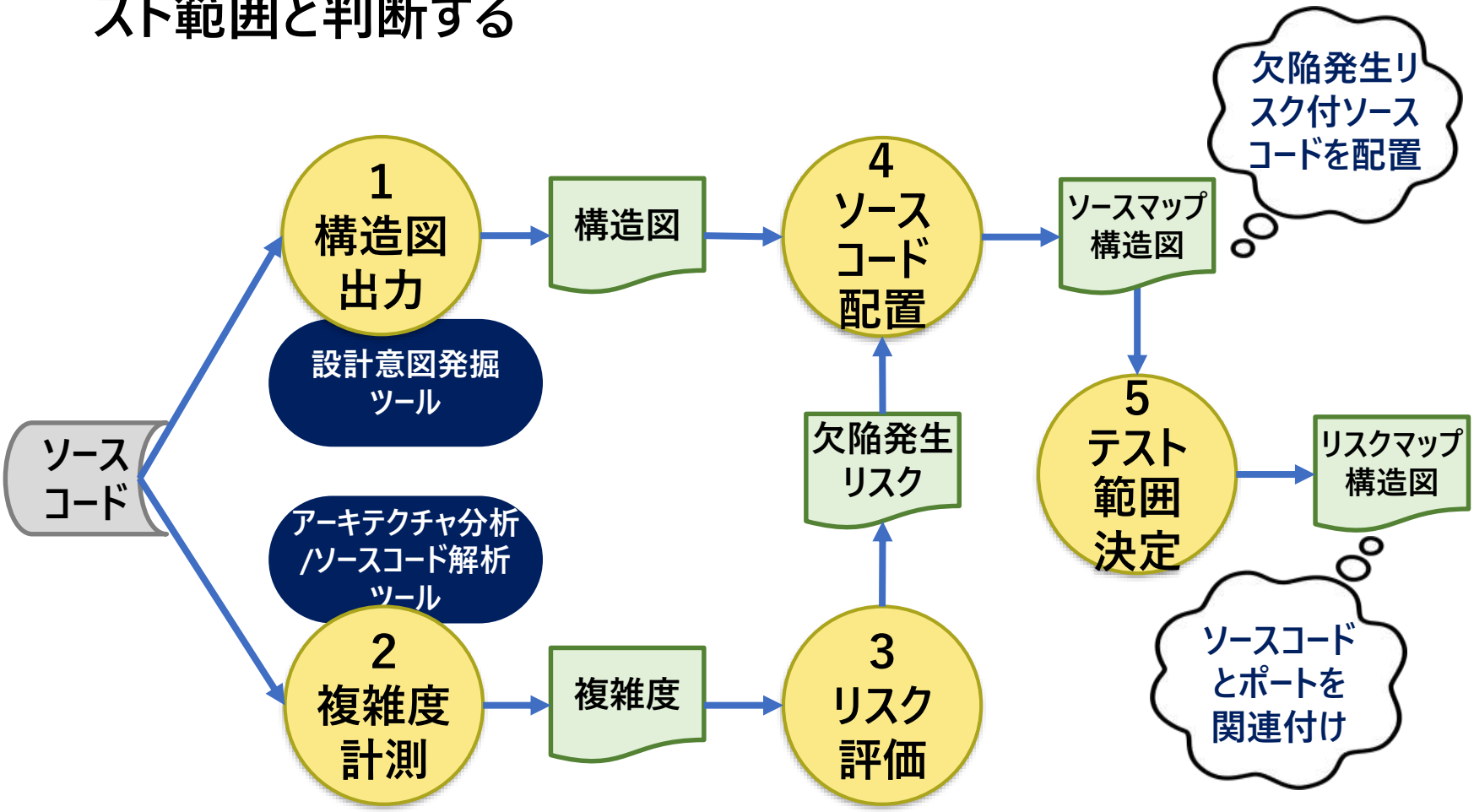


システム

テスト範囲の判断

RICOH
imagine. change.

- コンポーネント構造図上に、欠陥発生リスクが高いソースコードを配置、そのソースコードに含まれるポートに繋がるコンポーネントをテスト範囲と判断する



欠陥発生のリスク評価

RICOH
imagine. change.

ソフトウエアリスク	システムレベル		欠陥リスク														
			構造的複雑度														
			循環的複雑度														
	ソースコードレベル		欠陥リスク														
			構造的複雑度														
			循環的複雑度														
ソフトウェアメトリクス	コード行数総和		CountLineCode														
	複雑度総和		SumCyclomatic														
	最大複雑度		MaxCyclomatic														
	最大ネスト深さ		MaxNesting														
	直接影響度閾値		DirectFanIn														
	直接依存度閾値		DirectFanOut														
	間接影響度閾値		IndirectFanIn														
	間接依存度閾値		IndirectFanOut														

計測したメトリクスを基に
欠陥発生リスクを算出する

アーキテクチャ解析、ソース
コード解析のツールを使い
計測したメトリクス

ソースコード一覧

mid.DataStream.c
ur.Consumer.c
sumerC.c
DeviceM3.c
oducer.c
taLatch.c
sConfig.c
sConfig.h
pedef.h
Analyzer.c
Analyzer.h
ogApp.c
ogApp.h
eManager.c
app.ModeManager.h

欠陥発生のリスク評価結果

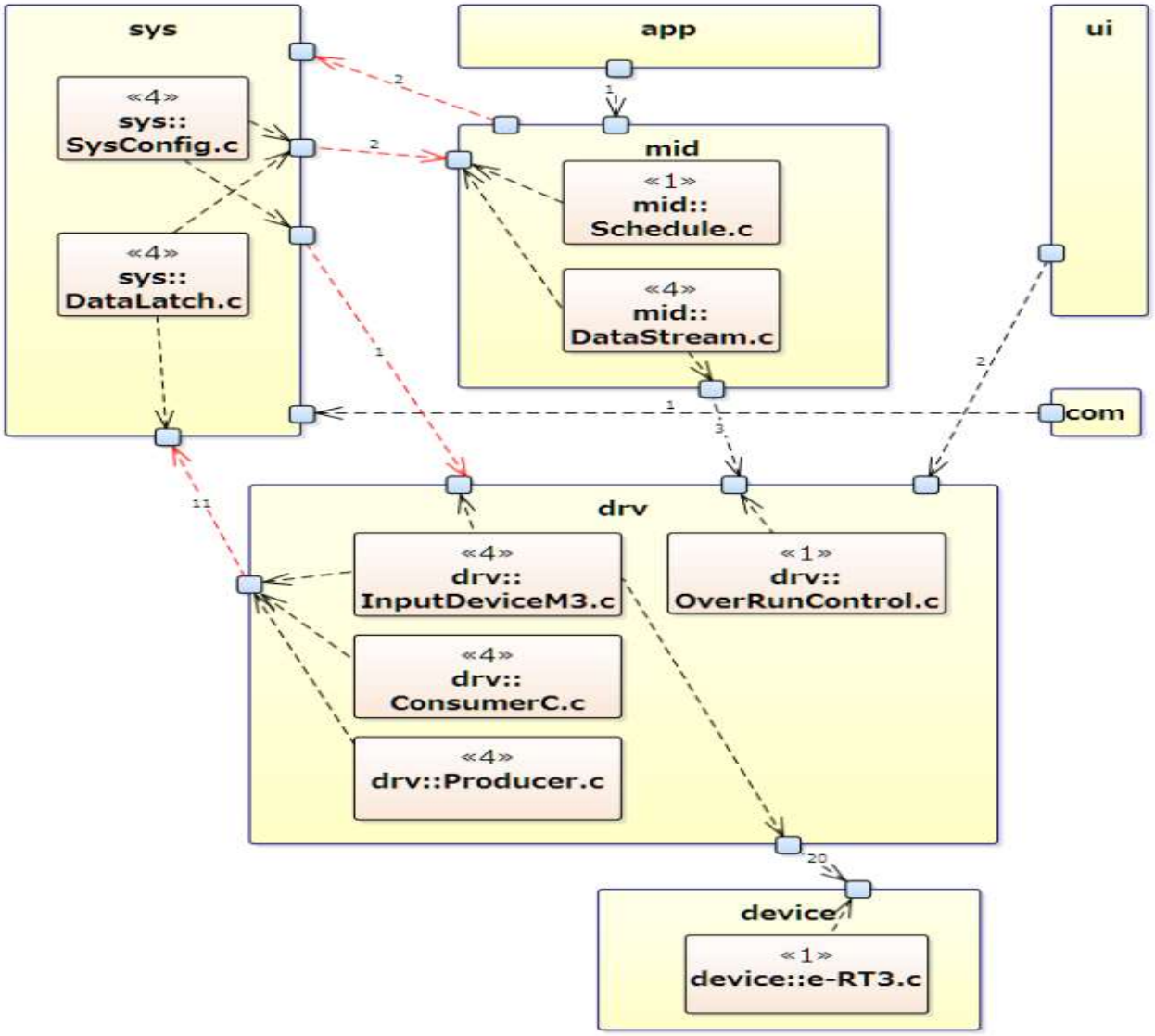
RICOH
imagine. change.

ソフトウエアリスク	システムレベル		欠陥リスク	4	1	4	4	4	1	4	2	2	1	1	1	1	1	1
			構造的複雑度	中	無	中	中	中	無	中	低	低	無	無	無	無	無	無
			循環的複雑度	無	無	無	無	無	無	無	無	無	無	無	無	無	無	無
	ソースコードレベル		欠陥リスク	**4**	**4**	**4**	**4**	**4**	**4**	**4**	2	2	1	1	1	1	1	1
			構造的複雑度	中	中	中	中	中	中	中	低	低	無	無	無	無	無	無
			循環的複雑度	無	無	無	無	無	無	無	無	無	無	無	無	無	無	無
ソフトウェアメトリクス	コード行数総和	1238	CountLineCode	36	39	68	104	18	38	13	8	9	3	1	4	1	4	1
	複雑度総和	119	SumCyclomatic	1	3	6	14	1	7	1	0	0	1	0	1	0	1	0
	最大複雑度	7	MaxCyclomatic	1	3	6	7	1	2	1	0	0	1	0	1	0	1	0
	最大ネスト深さ	3	MaxNesting	0	1	3	2	0	1	0	0	0	0	0	0	0	0	0
	直接影響度閾値	13.5	DirectFanIn	1	0	0	2	0	4	0	17	27	0	1	1	2	0	2
	直接依存度閾値	8	DirectFanOut	16	9	10	12	11	11	9	2	0	1	0	3	0	3	0
	間接影響度閾値	16.5	IndirectFanIn	2	1	1	3	1	6	1	18	33	1	2	2	3	1	3
	間接依存度閾値	14.5	IndirectFanOut	27	12	17	21	18	13	29	4	1	2	1	5	1	7	1
				mid.DataStream.c	drv.Consumer.c	drv.ConsumerC.c	drv.InputDeviceM3.c	drv.Producer.c	sys.DataLatch.c	sys.SysConfig.c	sys.SysConfig.h	sys.typedef.h	app.LogAnalyzer.c	app.LogAnalyzer.h	app.LogApp.c	app.LogApp.h	app.ModeManager.c	app.ModeManager.h

テスト範囲の判断

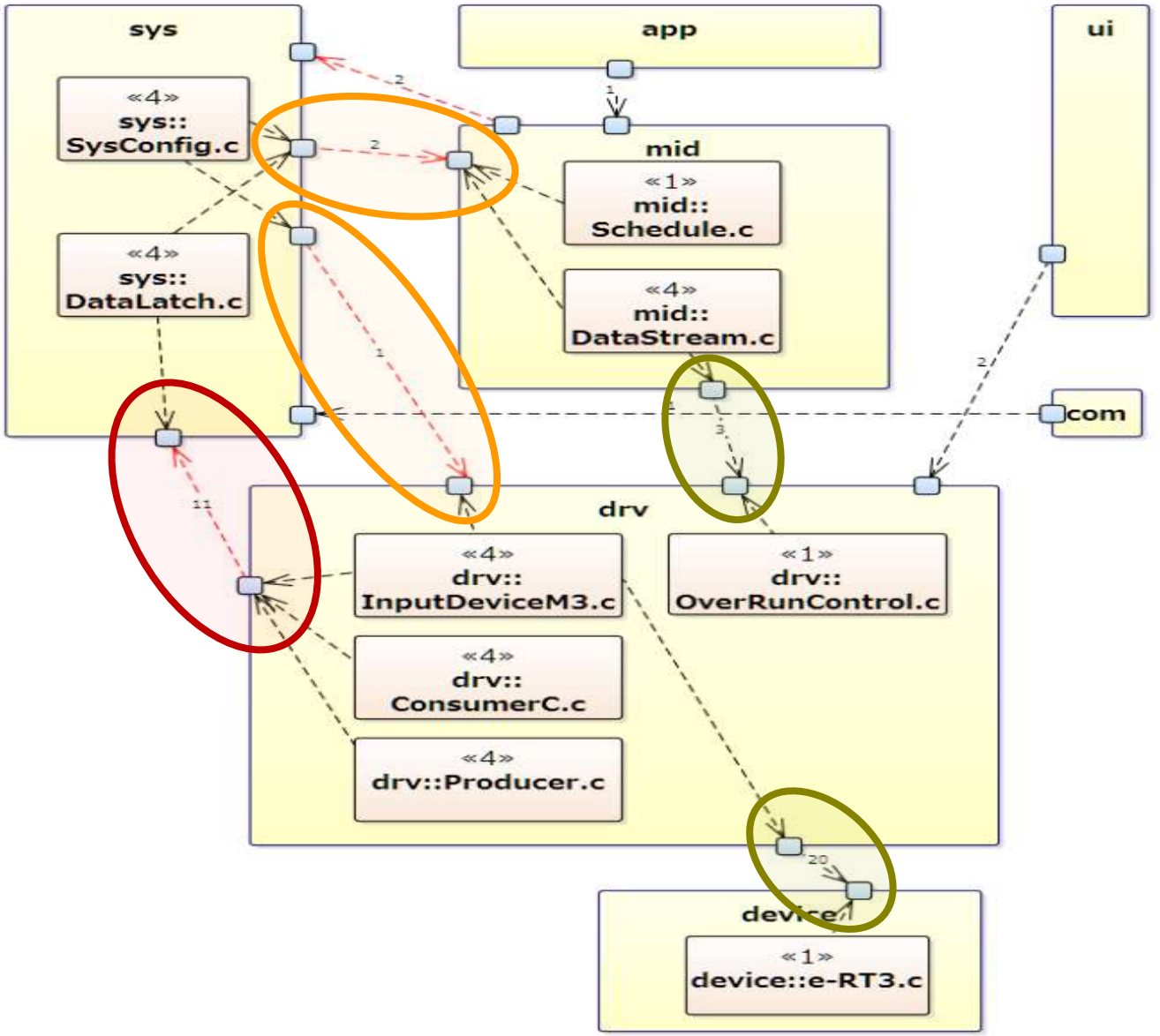
RICOH
imagine. change.

- 欠陥発生リスクの高いソースコードをコンポーネント構造図上に配置する



テスト範囲の判断結果

RICOH
imagine. change.



最重点

重点

推奨

テスト設計の十分さの評価

RICOH
imagine. change.

要件とソースコードの依存関係を示し、テストメトリクスの十分さを評価する

ソフトウェア要件	要件名称		依存関係													
	要件識別子	テスト網羅度														
		テスト密度														
		テスト項目総和														
		コード行数総和														
		複雑度総和														
		テストケース1件数														
		テストケース2件数														
		テストケース3件数														
ソフトウエアリスク		システムレベル	欠陥リスク	4	1	4	4	4	1	4	2	2	1	1	1	1	1
			構造複雑度	中	無	中	中	中	無	中	低	低	無	無	無	無	無
			循環複雑度	無	無	無	無	無	無	無	無	無	無	無	無	無	無
		ソースコードレベル	欠陥リスク	4	4	4	4	4	4	4	2	2	1	1	1	1	1
			構造複雑度	中	中	中	中	中	中	中	低	低	無	無	無	無	無
			循環複雑度	無	無	無	無	無	無	無	無	無	無	無	無	無	無
ソフトウェアメトリクス		コード行数総和	1238	36	39	68	104	18	38	13	8	9	3	1	4	1	4
		複雑度総和	119	1	3	6	14	1	7	1	0	0	1	0	1	0	1
		最大複雑度	7	1	3	6	7	1	2	1	0	0	1	0	1	0	1
		最大ネスト深さ	3	0	1	3	2	0	1	0	0	0	0	0	0	0	0
		直接影響度	13.5	1	0	0	2	0	4	0	17	27	0	1	1	2	0
		直接依存度	8	16	9	10	12	11	11	9	2	0	1	0	3	0	3
		間接影響度	16.5	2	1	1	3	1	6	1	18	33	1	2	2	3	1
		間接依存度	14.5	27	12	17	21	18	13	29	4	1	2	1	5	1	7
				mid.DataStream.c	drv.Consumer.c	drv.ConsumerC.c	drv.InputDeviceM3.c	drv.Producer.c	sys.DataLatch.c	sys.SysConfig.c	sys.SysConfig.h	sys.typedef.h	app.LogAnalyzer.c	app.LogAnalyzer.h	app.LogApp.c	app.LogApp.h	app.ModeManager.c

テスト網羅度：テスト項目総和/循環複雑度総和
テスト密度：テスト項目総和/コード行数総和
テスト項目総和：要件のテストケース項目数総和
コード行数総和：要件に依存するコード行数総和
循環複雑度総和：要件に依存する循環複雑度総和

プリンタ製品のサブシステムの事例

RICOH
imagine. change.

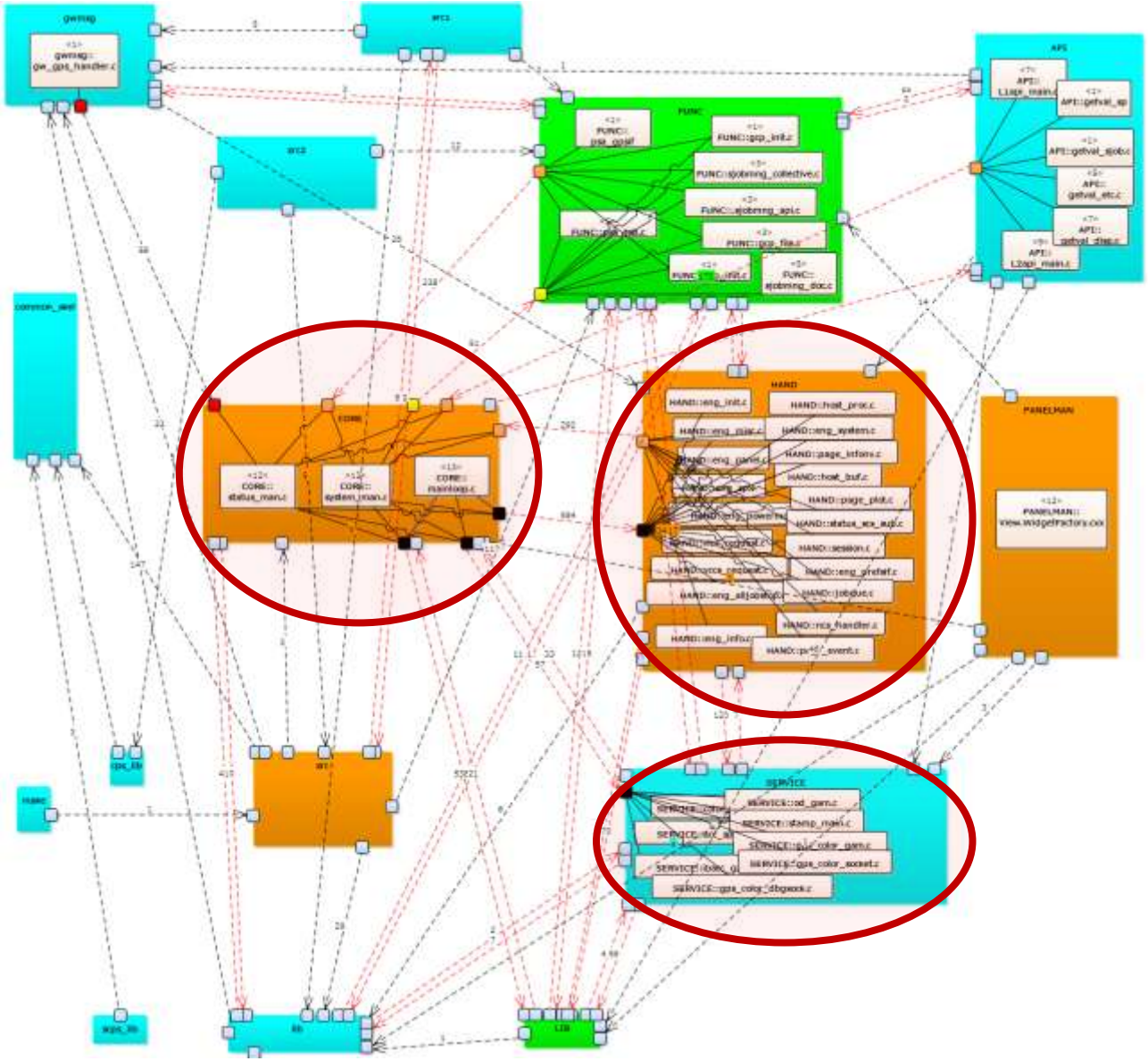
ソフトウェア要件	プリンタ言語仕様		依存関係				●	●											
	動作モード仕様		依存関係				◎	●											
	180	テスト網羅度	0.004878																
		テスト密度	1.004464																
		テスト項目総和	45																
		コード行数総和	44800																
		循環複雑度総和	9225																
		省エネ動作	24																
		ジョブリセット動作	6																
		オートリセット動作	7																
		一時停止動作	8																
	エラー動作仕様		依存関係				●	◎											
ソフトウェア品質		欠陥密度	0.00005																
		欠陥件数	15	1			3								1	1			
ソフトウエアリスク		システムレベル	欠陥リスク	16	16	12	15	15	13	13	13	13	13	13	13	13	13	13	13	13
			構造複雑度	高	高	中	高	高	中	中	中	中	中	中	中	中	中	中	中	中
			循環複雑度	高	高	中	中	中	高	高	高	高	高	高	高	高	高	高	高	高
		ソースコードレベル	欠陥リスク	13	13	12	12	12	7	7	7	7	7	7	7	7	7	7	7	7
			構造複雑度	中	中	中	中	中	無	無	無	無	無	無	無	無	無	無	無	無
			循環複雑度	高	高	中	中	中	高	高	高	高	高	高	高	高	高	高	高	高
ソフトウェアメトリクス		コード行数総和		285694	1161	4378	459	2231	2385	498	257	287	358	2081	2372	954	569	3773	478	275
		複雑度総和		52388	323	1066	112	462	478	90	99	75	65	482	565	259	126	793	164	69
		最大複雑度		1062	116	72	50	39	30	86	86	71	59	135	107	130	53	121	101	67
		最大ネスト深さ		8	3	5	5	4	5	6	4	5	5	5	7	5	3	6	3	5
		直接影響度閾値		159	2	8	7	43	31	2	2	3	2	3	3	31	8	15	5	3
		直接依存度閾値		48.5	67	49	97	70	50	16	18	26	33	38	27	42	15	23	12	8
		間接影響度閾値		361	432	432	47	432	432	47	47	47	47	47	47	75	77	79	77	5
		間接依存度閾値		486.5	690	690	919	690	690	919	919	919	919	919	919	815	753	751	735	818

printer.src.gps.src.CORE.system.mainloop.c
printer.src.gps.gpshell.cps.src.APP.WCPS.awcps_exchange.c
printer.src.gps.gps-printer.src.gps.src.CORE.system.system_main.c
printer.src.gps.src.CORE.status.status_main.c
printer.src.gps.gps-pm.src.PANELMAN.pm.typeII.View.NetSummaryTextWidget.cxx
printer.src.gps.gps-pm.src.PANELMAN.pm.typeII.View.GajiTextWidget.cxx
printer.src.gps.gps-pm.src.PANELMAN.pm.typeII.View.EventKeyWidget.cxx
printer.src.gps.gps-pm.src.PANELMAN.pm.typeII.View.SimpleTextWidget.cxx
printer.src.gps.gps-pm.src.PANELMAN.pm.typeII.Model.ActionExecutor.cxx
printer.src.gps.gps-pm.src.PANELMAN.pm.typeII.Model.MenuLayerLP.cxx
printer.src.gps.gps-pm.src.PANELMAN.pm.typeII.Model.PanelDataMan.cxx
printer.src.gps.gps-pm.src.PANELMAN.pm.typeII.Model.MenuMac.cxx
printer.src.gps.gps-pm.src.PANELMAN.pm.typeII.Model.MenuSystem.cxx
printer.src.gps.gps-pm.src.PANELMAN.pm.typeII.Model.MenuSupply.cxx
printer.src.gps.gps-pm.src.PANELMAN.pm.typeII.Controller.gwCommandSender.cxx

テスト網羅度、テスト密度が
共に十分でないことが直ぐに
判断できる

プリンタ製品のサブシステムの事例

RICOH
imagine. change.



テスト範囲

- 欠陥発生リスクの高いソースコードが含まれるコンポーネントとそれらに依存するコンポーネント（赤丸）
- 互いに依存（循環）する関係を持つコンポーネント（赤線）

結果

RICOH
imagine. change.

実現したこと

- **欠陥発生の可能性をコンポーネント図に示すこと**

  欠陥発生の可能性が高い箇所はテストを厚く、低い箇所はテストを薄くというようなテスト実装の判断材料にできる

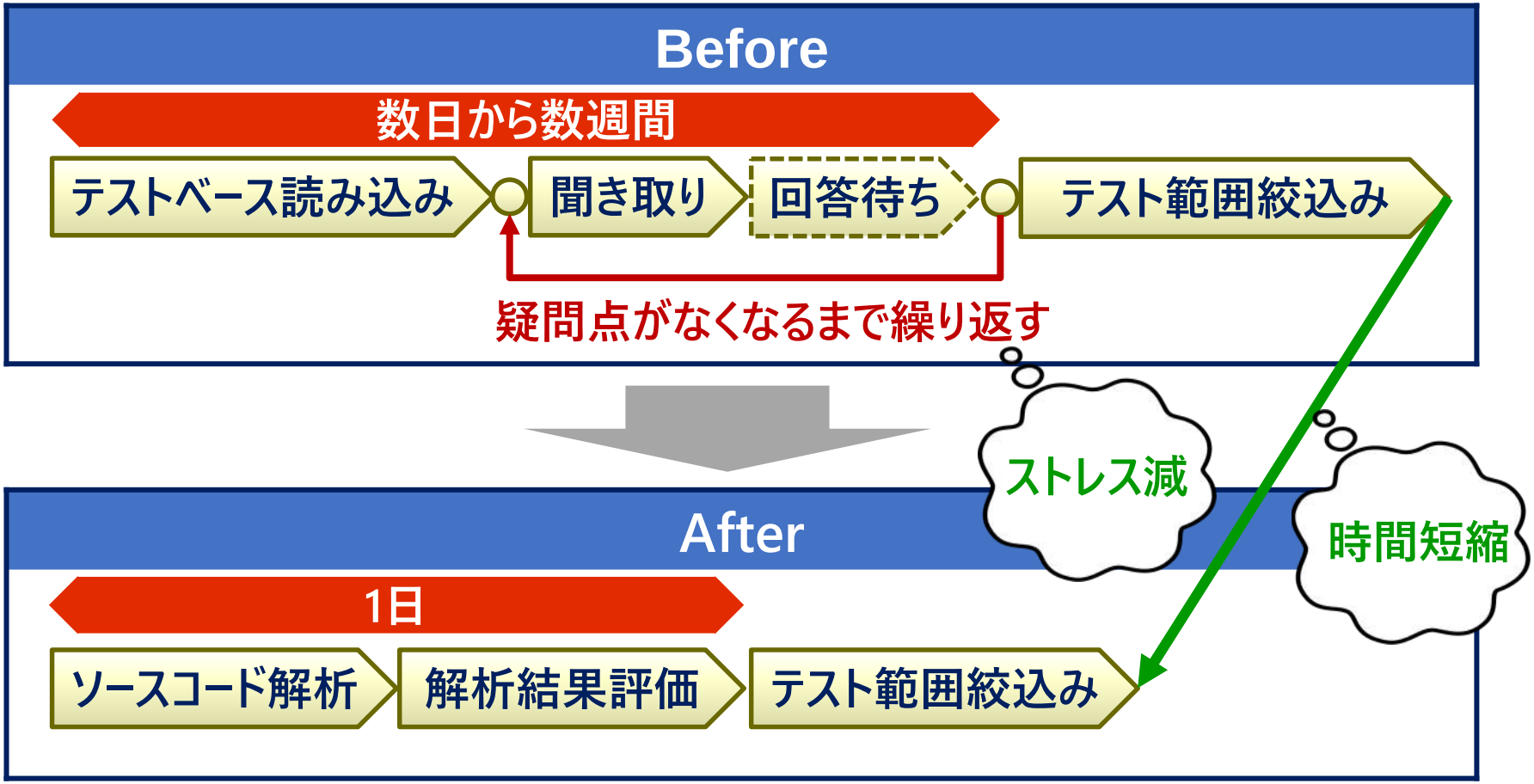
- **欠陥発生の可能性と要件のテストメトリクス（テストケース数、テスト網羅度、テスト密度）の関連を示すこと**

  欠陥発生の可能性とテストメトリクスの比較参照が、テストの十分さの判断材料できる

テストエンジニアの負担軽減効果

RICOH
imagine. change.

- ソフト設計者に確認する時間がなくなる（定量）
- 回答待ちや的確な回答がないストレスが減る（定性）



ソフトウェア設計の改善効果

RICOH
imagine. change.

- ソフトウェア品質の定量的指標の追加（品質UP）
- 再利用可能なコンポーネント構造図の提供（効率UP）

詳細設計
と
アーキテクチャ設計

Before	After
循環的複雑度 メソッドの長さ	循環的複雑度 メソッドの長さ 構造的複雑度 コンポーネント構造図

今後

RICOH
imagine. change.

- 繰り返し開発時のテスト範囲の絞込み効率化
  - 修正量・修正頻度に基づきテスト範囲を絞り込む
- テストケース設計の効率化
  - インターフェースを解析し、テストケースを生成する
- 効果測定の継続
  - 実施プロジェクトの拡大とその結果の蓄積

参照

RICOH
imagine. change.

- Daniel J. Sturtevant, “System Design and the Cost of Architectural Complexity”, MIT Engineering Systems Division, 2013
- Daniel J. Sturtevant, “Technical Debt in Large Systems: Understanding the cost of software complexity”, MIT SDM Webinar, May 6, 2013
- 綾井環ら,「ソフトウェア構造を見れば品質がわかる！　構造メトリクスとバグの関係」, SPI Japan 2014, Oct 15, 2014
- 設計意図発掘ツール,「AtScope」, ビースラッシュ株式会社 http://www.bslash.co.jp/AtScope/
- アーキテクチャ分析ツール,「Lattix」, テクマトリックス株式会社
- ソースコード解析ツール,「Understand」, テクマトリックス株式会社

ご清聴ありがとうございました

RICOH

imagine. change.