ソフトウェア構造と ソフトウェアメトリクス に着目した 派生開発における テスト設計事例の紹介

SPI Japan 2019

2019/10/10(木)

リコーITソリューションズ株式会社
増子 聡・森 孝司・狩野 薫

ビースラッシュ株式会社
山田 大介

IT for Innovation

はじめに

RICOH imagine. change. BACKSLASH DESIGN

■本日お話ししたいこと

- 派生開発におけるテストに関する課題を解決するための、効果的なテスト設計手法をご紹介します

■お伝えしたい方々

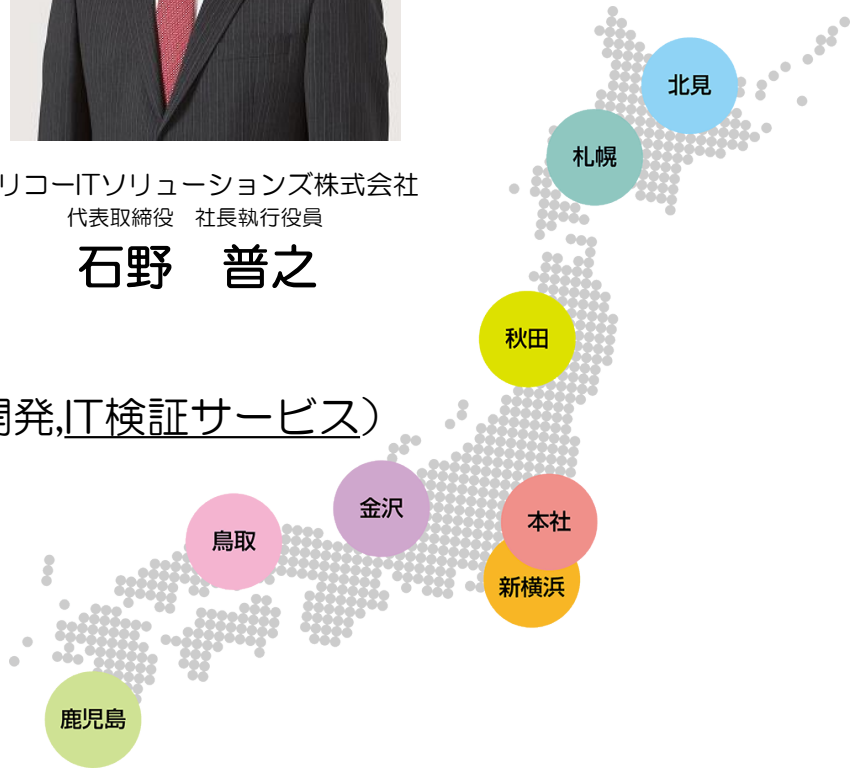
- テストスコープをうまく、簡単に絞り込みたい
- 短納期でも十分なテストを実施したい

Copyright © Ricoh IT Solutions Co., Ltd. All Rights Reserved.

会社紹介 － 会社概要 －

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	945名 （2019年３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

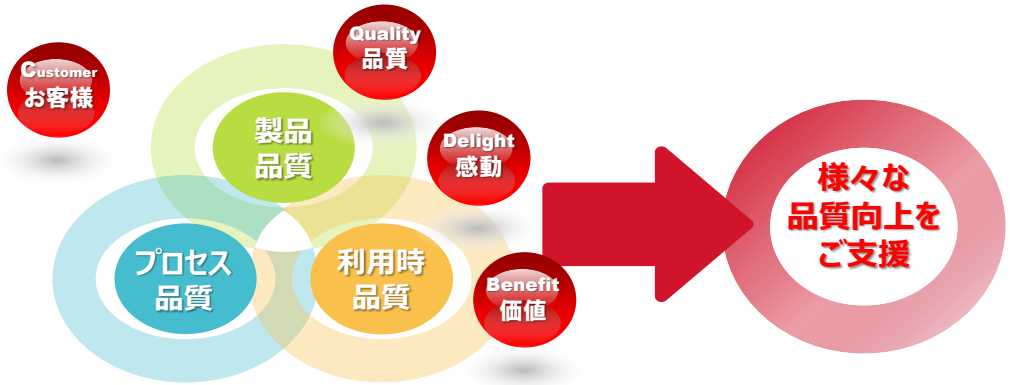


Copyright © Ricoh IT Solutions Co., Ltd. All Rights Reserved.

会社紹介 －IT検証サービス－

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。



Copyright © Ricoh IT Solutions Co., Ltd. All Rights Reserved.

自己紹介

- 名前：増子 聡

- 所属：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 事業推進センター
  IT検証部 検証推進グループ

- 経歴：入社24年目
  - 入社10年間は、MFP周辺のアプリケーションソフトウエアの開発に従事
  - その後、MFPモジュール開発の組織マネージメントに従事
  - 現在は、テストマネージャーとして、主にテスト業務の改善推進活動に従事

- 座右の銘：「好きこそものの上手なれ」

Copyright © Ricoh IT Solutions Co., Ltd. All Rights Reserved.

目次

1. 背景・目的

2. ソフトウェアメトリクスを用いた欠陥予測手法

3. 派生開発への適用方法

4. 派生開発での適用結果

5. 効果

6. 今後の取り組み

Copyright © Ricoh IT Solutions Co., Ltd. All Rights Reserved.

1. 背景・目的

- 派生開発におけるテスト現場の悩み
- 提案するアプローチ

Copyright © Ricoh IT Solutions Co., Ltd. All Rights Reserved.

派生開発におけるテスト現場の悩み

テストの現場での悩み	解決する手段
テストスコープの見極めが難しい	ソースコードの修正箇所の発見
	システム上の影響するテストパスの発見
テスト期間が短い	テストスコープの優先付け

Copyright © Ricoh IT Solutions Co., Ltd. All Rights Reserved.

提案するアプローチ

- 『ソフトウェアメトリクスを活用した欠陥予測手法』を派生開発に応用しテスト現場の悩みを解消するアプローチを提案します

Copyright © Ricoh IT Solutions Co., Ltd. All Rights Reserved.

2. ソフトウェアメトリクスを用いた欠陥予測手法*1

- 複雑度の組み合わせによるバグ発生予測
- テストメトリクスレポート例

*1 SPI Japan 2018 「ソフトウェア構造解析のテスト設計への応用」
http://www.jaspic.org/event/2018/SPIJapan/session3C/3C1_ID006.pdf

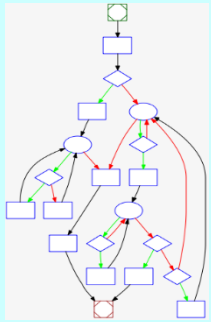
Copyright © Ricoh IT Solutions Co., Ltd. All Rights Reserved.

複雑度の組み合わせによるバグ発生予測

RICOH imagine. change.

BACKSLASH DESIGN

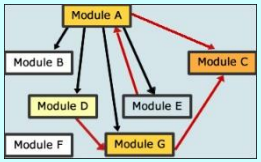
循環複雑度



循環複雑度	区分
51～	高
21～50	中
11～20	低
1～10	無

＋

構造複雑度



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

欠陥発生リスク

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			

２つの複雑度を組み合わせ、
ソースコードの欠陥発生リスクを
16段階で表現

出典:
http://understand-jp.blogspot.com/2012/06/blog-post.html
https://www.techmatrix.co.jp/product/lattix/function/impactanalysis.html

Copyright © Ricoh IT Solutions Co., Ltd. All Rights Reserved.

テストメトリクスレポート例

RICOH imagine. change.
BACKSLASH DESIGN

テストメトリクス

ソフトウェア要件と テストケース一覧

ソフトウェア品質

欠陥発生リスク

循環複雑度

ソフトウェアメトリクス

構造複雑度

DSM

ソフトウェア要件	ソフトウェア要件1			依存関係	●	◎	●		●		●	●			
	テスト網羅度		0.254	テスト網羅度		15									
	テスト密度		19.43	テスト密度		14.40366972									
	テスト項目総和		15	テスト項目総和		15									
	コード行数総和/要件		772	コード行数		436	69		40		130	97			
	複雑度総和/要件		59	複雑度総和		1	11		8		24	15			
	テストケース1		5	テストケース1		5									
	テストケース2		10	テストケース2		10									
	ソフトウェア要件2			依存関係				◎	●		●		●	●	
	ソフトウェア要件3			依存関係						◎		●	●	●	
ソフトウェア品質	欠陥密度														
	欠陥件数														
ソフトウエアリスク	システムレベル			欠陥リスク	4	2	2	2	2	2	2	2	4	2	2
	ソースコードレベル			欠陥リスク	4	2	4	1	4	1	4	4	4	1	1
ソフトウェアメトリクス	コード行数総和		1023	CountLineCode	100	436	69	20	40	61	130	97	68	1	1
	複雑度総和		111	SumCyclomatic	20	1	11	5	8	11	24	15	14	1	1
	最大複雑度		6	MaxCyclomatic	6	1	4	1	1	5	6	3	5	1	1
	最大ネスト深さ		3	MaxNesting	1	0	2	0	0	1	3	1	1	1	1
	直接影響度閾値		6.5	DirectFanIn	4	13	3	2	2	2	3	4	1	6	4
	直接依存度閾値		6.5	DirectFanOut	9	1	7	1	11	3	13	13	7	2	2
	間接影響度閾値		27.5	IndirectFanIn	27	40	28	30	30	30	30	29	1	55	32
	間接依存度閾値		152	IndirectFanOut	303	1	22	1	14	3	17	21	304	2	2

DSM	A.CPP	B.CPP	C.CPP	D.CPP	E.CPP	F.CPP	G.CPP	H.CPP	I.CPP	J.CPP	K.CPP
A.CPP		1									
B.CPP								1			
C.CPP	1	1							1		
D.CPP	1							1		1	
E.CPP		1		1				1			
F.CPP											
G.CPP		1		1					1	1	
H.CPP		1				1			1		

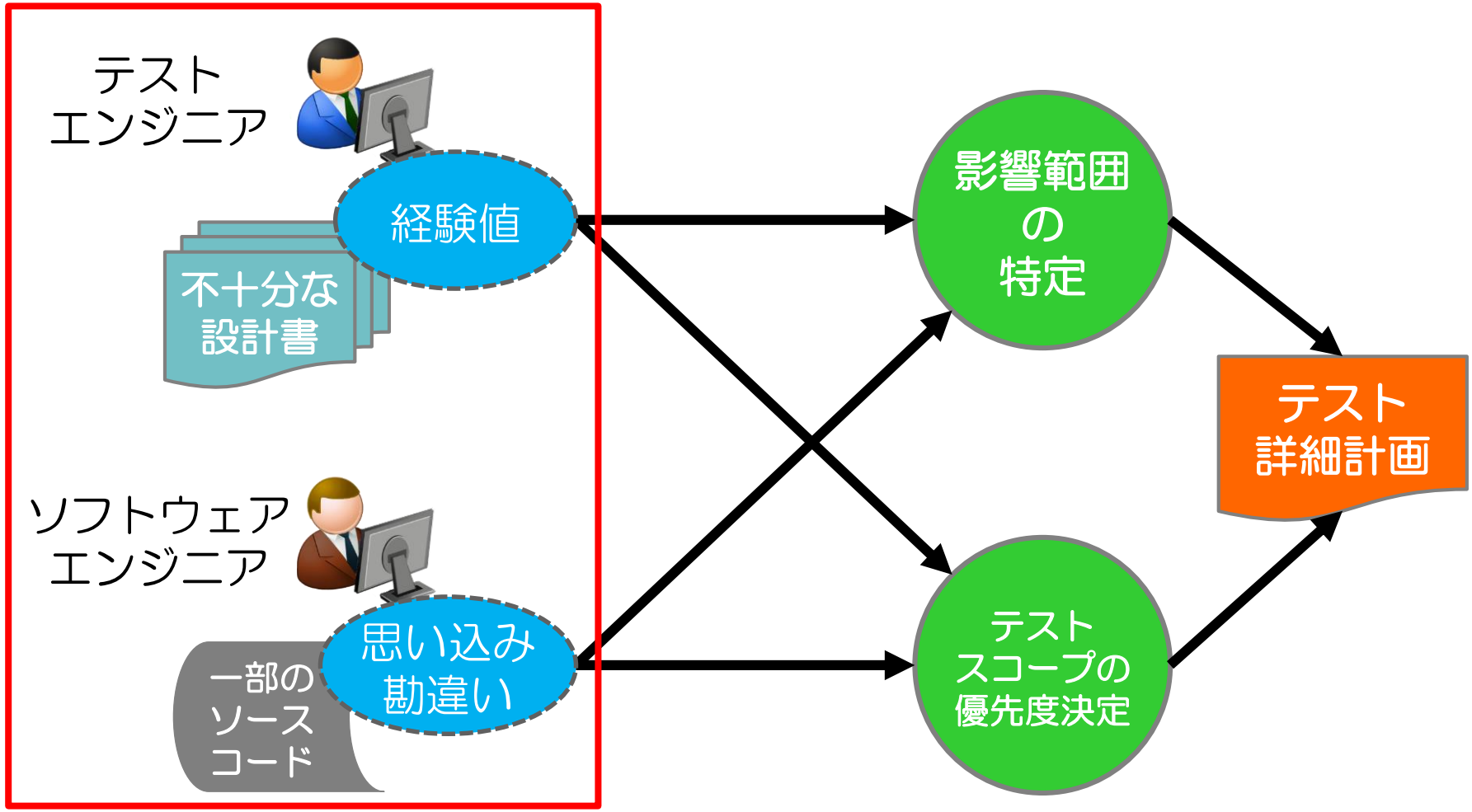
Copyright © Ricoh IT Solutions Co., Ltd. All Rights Reserved.

3. 派生開発への適用方法

- 派生開発への適用前のテストプロセス
- 派生開発への適用後のテストプロセス
- ステップ
  - Step1:修正箇所を見つける
  - Step2:修正箇所の影響範囲を見つける
  - Step3:テストスコープの優先度を決定する

Copyright © Ricoh IT Solutions Co., Ltd. All Rights Reserved.

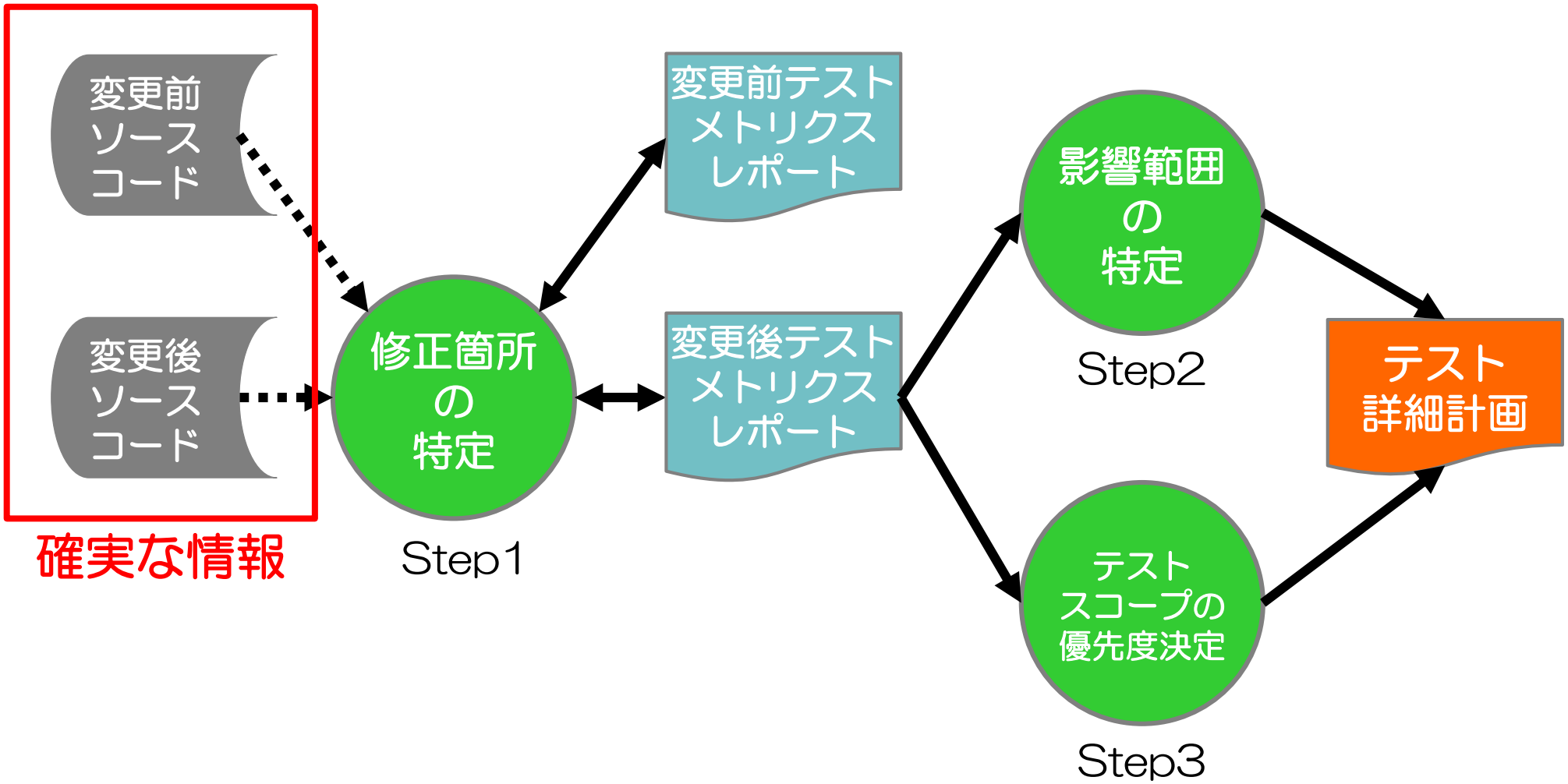
派生開発への適用前のテストプロセス



不確実な情報

Copyright © Ricoh IT Solutions Co., Ltd. All Rights Reserved.

派生開発への適用後のテストプロセス



Copyright © Ricoh IT Solutions Co., Ltd. All Rights Reserved.

Step1:修正箇所を見つける

■ソースコードの変更前と変更後のメトリクスを比較し、差異があった箇所を修正箇所（直接的テストスコープ）と判断する

ソフトウェアメトリクス			CountLineCode	100	436	69	20	40	61	130	97	68	1	1
	コード行数総和	1023	SumCyclomatic	20	1	11	5	8	11	24	15	14	1	1
	複雑度総和	111	MaxCyclomatic	6	1	4	1	1	5	6	3	5	1	1
	最大複雑度	6	MaxNesting	1	0	2	0	0	1	3	1	1	1	1
	最大ネスト深さ	[illegible]	DirectFanIn	4	13	3	2	2	2	3	4	1	6	4
	[illegible]	6.5	DirectFanOut	9	1	7	1	11	3	13	13	7	2	2
	[illegible]	6.5	IndirectFanIn	27	40	28	30	30	30	30	29	1	55	32
	[illegible]	27.5	IndirectFanOut	303	1	22	1	14	3	17	21	304	2	2
	間接依存度閾値	152												

メトリクス変化点

DSM	A.CPP	B.CPP	C.CPP	D.CPP	E.CPP	F.CPP	G.CPP	H.CPP	I.CPP	J.CPP	K.CPP
A.CPP		1									
B.CPP								1			
C.CPP	1	1							1		
D.CPP	1							1		1	
E.CPP		1		1				1			
F.CPP											
G.CPP		1		1					1	1	

修正箇所

Copyright © Ricoh IT Solutions Co., Ltd. All Rights Reserved.

Step2:修正箇所の影響範囲を見つける

RICOH imagine. change. BACKSLASH DESIGN

■修正箇所に依存するソースコードを追跡し、修正箇所の影響範囲（間接的テストスコープ）を見つける

ソフトウェアメトリクス													
	コード行数総和	1023	CountLineCode	100	436	69	20	40	61	130	97	68	1	1
	複雑度総和	111	SumCyclomatic	20	1	11	5	8	11	24	15	14	1	1
	最大複雑度	6	MaxCyclomatic	6	1	4	1	1	5	6	3	5	1	1
	最大ネスト深さ	3	MaxNesting	1	0	2	0	0	1	3	1	1	1	1
		6.5	DirectFanIn	4	13	3	2	2	2	3	4	1	6	4
		6.5	DirectFanOut	9	1	7	1	11	3	13	13	7	2	2
		27.5	IndirectFanIn	27	40	28	30	30	30	30	29	1	55	32
			IndirectFanOut	303	1	22	1	14	3	17	21	304	2	2

修正箇所

DSM	A.CPP	B.CPP	C.CPP	D.CPP	E.CPP	F.CPP	G.CPP	H.CPP	I.CPP	J.CPP	K.CPP
A.CPP		1									
B.CPP								1			
C.CPP	1	1							1		
D.CPP	1							1		1	
E.CPP		1		1				1			
F.CPP											
G.CPP		1		1					1	1	
H.CPP		1				1			1		
I.CPP				1		1		1	1		
J.CPP				1		1		1		1	
K.CPP								1	1		

影響範囲

Copyright © Ricoh IT Solutions Co., Ltd. All Rights Reserved.

Step3:テストスコープの優先度を決定する

RICOH imagine. change. BACKSLASH DESIGN

■欠陥リスクの高いソースコードを「重要度が高い」と判断しテストスコープの優先度を決定する

ソフトウエアリスク	システムレベル		欠陥リスク	4	2	2	2	2	2	2	2	4	2	2
	ソースコードレベル		欠陥リスク	4	2	4	1	4	1	4	4	4	1	1
ソフトウェアメトリクス	コード行数総和	1023	CountLineCode	100	136	69	20	40	61	130	97	68	1	1
	複雑度総和	111	SumCyclomatic	20	1	11	5	8	11	24	15	14	1	1
	最大複雑度	6	MaxCyclomatic	6	1	4	1	1	5	6	3	5	1	1
	最大ネスト深さ	3	MaxNesting	1	0	2	0	0	1	3	1	1	1	1
	直接影響度閾値	6.5	DirectFanIn	4	13	3	2	2	2	3	4	1	6	4
	直接依存度閾値	6.5	DirectFanOut	9	1	7	1	11	3	13	13	7	2	2
	間接影響度閾値	27.5	IndirectFanIn	27	40	28	30	30	30	30	29	1	55	32
	間接依存度閾値	152	IndirectFanOut	303	1	22	1	14	3	17	21	304	2	2
			DSM	A.CPP	B.CPP	C.CPP	D.CPP	E.CPP	F.CPP	G.CPP	H.CPP	I.CPP	J.CPP	K.CPP

テストスコープの優先順位は欠陥リスクの数値によって決まる
欠陥リスク４…A.CPP
欠陥リスク２… B.CPP
欠陥リスク１… J.CPP

Copyright © Ricoh IT Solutions Co., Ltd. All Rights Reserved.

4. 派生開発での適用結果

- 対象システム
- 結果
  - Step1:修正箇所を見つける
  - Step2:修正箇所の影響範囲を見つける
  - Step3:テストスコープの優先度を決定する

Copyright © Ricoh IT Solutions Co., Ltd. All Rights Reserved.

対象システム

- 製品名：ビースラッシュ(株) AtScope
- 対象とするバージョン：　V4.0　（26,056ステップ）
  V5.0　（27,780ステップ）
- 開発言語：C#.NET
- 製品の特徴：ソースコードから様々な構造図の出力を行うことで
  ソフトウェアの設計意図の「発掘」を支援する
  静的解析ツール

Copyright © Ricoh IT Solutions Co., Ltd. All Rights Reserved.

Step1:修正箇所を見つける

RICOH imagine. change.

BACKSLASH DESIGN

■ 修正箇所として認識したもの

項目	クラス数
全クラス数	346
修正クラス数（変更要求が明確なクラス）	50
修正クラス数（変更要求が不明なクラス）	28

コード行数総和	27780	CountLineCode
複雑度総和	4853	SumCyclomatic
最大複雑度	316	MaxCyclomatic
最大ネスト深さ	17	MaxNesting
直接影響度閾値	38	DirectFanIn
直接依存度閾値	20	DirectFanOut
間接影響度閾値	70.5	IndirectFanIn
間接依存度閾値	156	IndirectFanOut

	CountLineCode	SumCyclomatic	MaxCyclomatic	MaxNesting	DirectFanIn	DirectFanOut	IndirectFanIn	IndirectFanOut
$root	73	73	12	54	1	3	309	309
.12 ParserDeclarationDetector	105	16	12	5	4	7	30	73
.12 ParserDefinitionDetector	267	37	11	6	4	12	30	73
.22 DisplayElementDEStore	374	80	11	3	6	5	56	12
.31 IteratorHerFolderBox	60	13	4	3	6	6	98	54
AtScopeProjectProperties.Resources	441	1	1	0	13	1	39	1
.02 MenuForm TopMenu	67	7	3	2	2	3	27	3
.02 MenuForm TopMenu	969	83	9	4	2	28	26	309
.03 ToolFactory.CClassDiagram	19	5	1	0	3	8	26	309
.03 ToolFactory.COperationDiagram	20	5	1	0	3	8	26	309
.03 ToolFactory.CDataAccessNoParenDiagram	20	5	1	0	3	8	26	309
.03 ToolFactory.CDataAccessNoParenDiagram	20	5	1	0	3	8	26	309
.03 ToolFactory.CRefactoringScore	28	7	1	0	3	12	26	309
.03 ToolFactory.CTaskStructure	21	5	1	0	3	9	26	309
.03 ToolFactory.CTaskStructure	92	30	13	1	2	16	26	309
.03 ToolFactory.CreateFactory	44	11	1	0	9	14	26	309
.03 ToolFactory.ToolFactory	179	29	13	1	5	8	26	309
.03 ToolFactory.ToolFactory	67	14	1	1	3	5	30	309
.04 LogParsingLog	51	9	8	1	6	2	53	8
.04 LogParsingLog	44	9	6	1	14	4	40	5
.06 Configuration.FileConfig	148	32	2	1	25	10	45	2
.08 RuleSession.RuleSetting	73	15	2	1	2	5	27	12
.09 DisSession.DisSetting	85	17	5	2	7	5	27	59
.11 Scoping.OOScoping	1663	159	2	5	5	22	33	66
.11 Scoping.OOScoping	207	32	4	2	6	2	34	2
.12 ParserElementComposer	71	12	2	2	5	3	31	2
.12 ParserEnvSettings	50	8	3	3	6	2	28	56
.12 ParserFileInputList	71	13	3	1	5	2	33	1
.12 ParserHierarchyDetector	108	20	1	0	2	4	32	3
.12 ParserMethodDefinitionList	54	5	3	2	4	4	27	91
.12 ParserMethodDefinition	184	26	2	1	7	7	29	82
.12 Parser OOParser	13	2	2	4	0	3	1	1
.12 Parser OnPassAnalyzer	113	17	1	0	3	4	34	63
.12 Parser StmtExtension	92	1	5	4	3	3	29	64
.13 Analyzer FanIDCounter	100	20	4	4	4	4	29	56
.13 Analyzer GatheringDetector	110	17	5	4	3	3	34	59
.13 Analyzer PartsDetector	166	22	4	3	1	3	31	64
.13 AnalyzerPointerDetector	128	21	4	4	2	4	28	70
.14 CheckerProfiler	130	30	4	2	2	2	27	102
.16 FocusRefactoringScoreFocus	76	28	5	2	2	2	27	113
.16 FocusClassDiagramFocus	53	26	2	3	3	2	27	114
.16 FocusComponentDiagramFocus	116	15	1	1	2	1	27	111
.16 FocusDataAccessNoParenDiagramFocus	138	9	1	2	1	2	27	108
.16 FocusDataAccessNoParenDiagramFocus	133	18	2	3	5	2	28	104
.16 FocusDataAccessVariableSimilarFocus	432	23	2	3	4	2	28	101
.16 FocusDataAccessVariableSimilarFocus	574	22	3	3	5	2	34	101
.16 Focus Focus	133	67	2	2	4	16	27	120
.16 FocusRefactoringScoreFocus	99	83	2	1	5	2	33	83
.16 FocusTraceExpansion	66	17	1	2	5	4	28	85
.16 FocusTraceFunction	56	18	2	2	8	2	28	85
.16 Focus TraceVariable	56	12	2	2	4	2	28	85
.16 FocusRefactoringScoreRankCalculator	56	8	2	1	3	2	28	1
.16 FocusRefactoringScoreScoreCalculator	56	16	1	2	2	2	30	5
.16 FocusRefactoringScoreScoreCalculator	150	2	2	2	2	2	29	19
.17 Builder CoordinateMatrix	81	10	7	5	3	3	31	46
.17 Builder DisMatrix	175	16	5	2	2	2	33	9
.17 Builder DisplayConfig	19	5	1	0	5	1	30	17
.17 Builder Layouter	107	4	6	2	0	3	29	28
.17 Builder TreeNoParentLayouter	362	21	9	4	2	9	28	37
.18 DrawerGraphicDevice	82	79	5	4	4	14	30	48
.18 DrawerGraphicEngine	932	17	25	1	3	8	26	26
.18 DrawerEADrawerEAGraphicDevice	134	176	6	4	2	40	26	309
.18 DrawerEADrawerEAScoreFocus	199	23	2	3	6	22	26	309
.21 StructureElementLBox	38	43	3	0	1	4	17	17
.21 StructureElementLBoxAggregate	10	10	0	0	1	3	112	19
.21 StructureElementBoxTree	32	6	3	3	1	5	1	28
.21 StructureElementFaBoxList	270	53	1	10	42	17	137	17
.21 StructureElementFileBoxList	47	11	3	6	8	21	106	19
.21 StructureElementFileBoxIdentifier	74	15	1	4	4	19	103	21
.21 StructureElementFolderBoxList	101	22	2	6	8	21	103	17
.21 StructureElementFunctionBoxList	213	45	2	6	37	17	137	21
.21 StructureElementFunctionBoxList	92	19	3	6	3	21	104	54
.21 StructureElementSEComposer	295	53	3	6	76	27	98	17
.21 StructureElementVariableBox	90	19	2	4	32	4	137	15
.21 StructureElementLDEComposer	136	22	1	7	32	7	55	8
.22 DisplayElementDisplayBox	187	39	3	8	43	8	60	11
.22 DisplayElementDisplayLine	142	28	3	4	29	7	57	24
.31 IteratorIterDisplayElement	104	16	1	4	12	5	46	30
.31 IteratorIterEdgeElement	30	8	2	2	13	7	30	7
.31 IteratorIterList	113	27	2	3	3	5	38	23
.32 MoveingResources	726	1	0	1	5	16	43	1
.33 FieldVarIterFieldVarIdb	114	18	5	1	10	10	43	3

■ 実現したこと・わかったこと

- 修正箇所を漏れなくリスト化する
- 修正箇所には、変更要求が不明確なものも含まれている

Copyright © Ricoh IT Solutions Co., Ltd. All Rights Reserved.

Step2:修正箇所の影響範囲を見つける

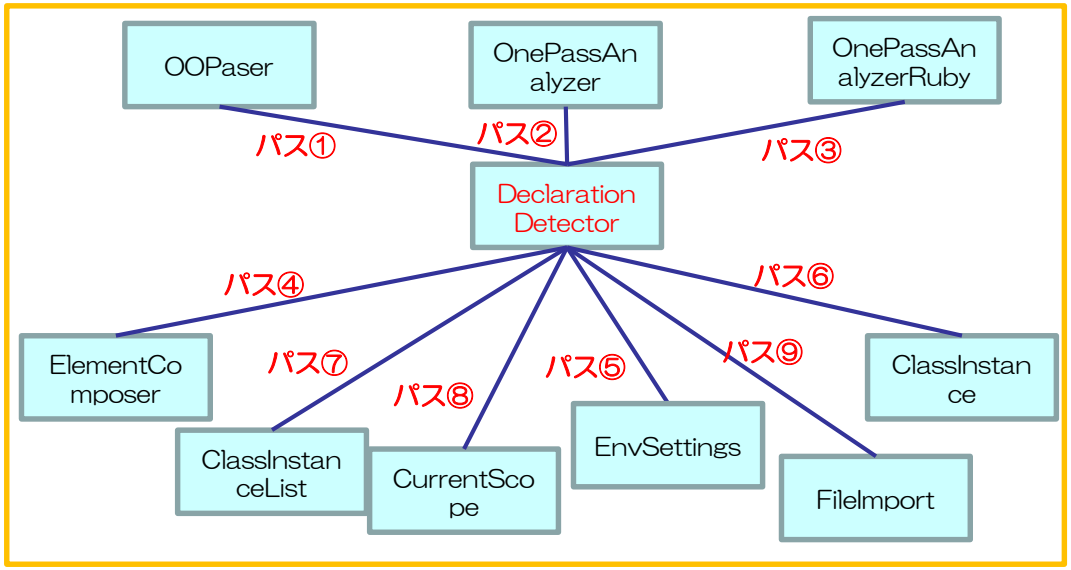
RICOH imagine. change. BACKSLASH DESIGN

■ 修正箇所の影響範囲として認識したもの

例）DeclarationDetectorの影響範囲

$root		_12_Parser.DeclarationDetector	_12_Parser.OOParser	_12_Parser.OnePassAnalyzer	_12_Parser.OnePassAnalyzerRuby
		129	147	148	149
_12_Parser.DeclarationDetector	129	.3%	1	1	1
_12_Parser.ElementComposer	131	1	1		
_12_Parser.EnvSettings	133	1	1		
_12_Parser.ClassInstance	115	1			
_12_Parser.ClassInstanceList	116	1	1	1	1
_12_Parser.CurrentScope	126	1		1	
_12_Parser.FileImport	135	1		1	1

項目	クラス数
修正箇所を呼び出しているクラス	3
修正箇所が呼び出しているクラス	6



■ 実現したこと・わかったこと

- 影響範囲を構造化（依存と影響で区分）する
- クラス単位の依存度と影響度よりテストパス数を算出する

Copyright © Ricoh IT Solutions Co., Ltd. All Rights Reserved.

Step3:テストスコープの優先度を決定する

RICOH imagine. change. BACKSLASH DESIGN

■ テストスコープの優先順の認識

早い → 実行順 → 遅い

ソフトウェア要件	インストール	1.1		依存関係																			
	ライセンス更新	1.2		依存関係																			
	バージョンアップ	1.3		依存関係																			
	アンインストール	1.4		依存関係																			
	アーキテクチャ解析	2.1		依存関係		●							●							●	●	●	●
	ファイル解析	2.2		依存関係		●					◎		●							●	●	●	●
	モジュール解析	2.3		依存関係		●						◎	●							●	●	●	●
	リファクタリングスコア	2.4		依存関係		●				◎			●	◎						●	●	●	●
ソフトウエアリスク	システムレベル			システム欠陥リスク	12	2	4	4	1	1	1	1	1	1	11	3	3	3	3	3	2	2	1
	ソースコードレベル			ソースコード欠陥リスク	12	8	4	4	4	4	4	4	4	4	3	3	3	3	3	3	2	2	2
ソフトウェアメトリクス	コード行数総和		27780	CountLineCode	932	295	960	134	1053	128	130	138	432	574	92	105	267	184	81	374	199	270	136
	複雑度総和		4853	SumCyclomatic	176	3	83	23	159	28	26	23	67	83	30	16	37	26	21	80	43	53	22
	最大複雑度		316	MaxCyclomatic	25	4	9	6	7	4	5	5	5	5	13	12	11	17	16	11	6	7	2
	最大ネスト深さ		17	MaxNesting	4	3	4	2	4	3	1	2	3	2	1	5	6	4	3	3	3	3	1
	直接影響度閾値		38	DirectFanIn	2	6	2	2	5	2	2	2	16	2	2	4	4	3	3	6	69	42	43
	直接依存度閾値		20	DirectFanOut	40	7	28	22	22	22	20	22	33	35	16	7	12	9	13	5	4	10	7
	間接影響度閾値		70.5	IndirectFanIn	26	98	26	26	33	28	27	28	34	27	26	30	30	29	29	56	137	137	55
	間接依存度閾値		156	IndirectFanOut	309	54	309	309	66	70	102	108	101	120	309	73	73	82	46	12	17	17	15
				$root	_18_Drawer.EADrawer.EAGraph	_21_StructureElement.SEComp	02_MenuForm.TopMenu	_18_Drawer.EADrawer.EAScore	_12_Parser.ElementComposer	_14_Checker.RefactoringScorel	_16_Focus.ClassDiagramFocus	_16_Focus.DataAccessVariable	_16_Focus.Focus	_16_Focus.RefactoringScoreFo	_03_ToolFactory.CreateFactory	_12_Parser.DeclarationDetector	_12_Parser.DefinitionDetector	_12_Parser.OnePassAnalyzer	_17_Builder.DispMat	_22_DisplayElement.DEStore	_21_StructureElementBox	_21_StructureElement.FileBox	_22_DisplayElement.DECompos

テストケース

テストスコープ

高い → 優先順 → 低い

■ 実現したこと・わかったこと

- 優先付けされたテストスコープから、テストケースの実行順を決定する
- 修正があるにもかかわらず、テストケースがないことがわかる

Copyright © Ricoh IT Solutions Co., Ltd. All Rights Reserved.

5. 効果

- 派生開発のテストでの効果

Copyright © Ricoh IT Solutions Co., Ltd. All Rights Reserved.

派生開発のテストでの効果

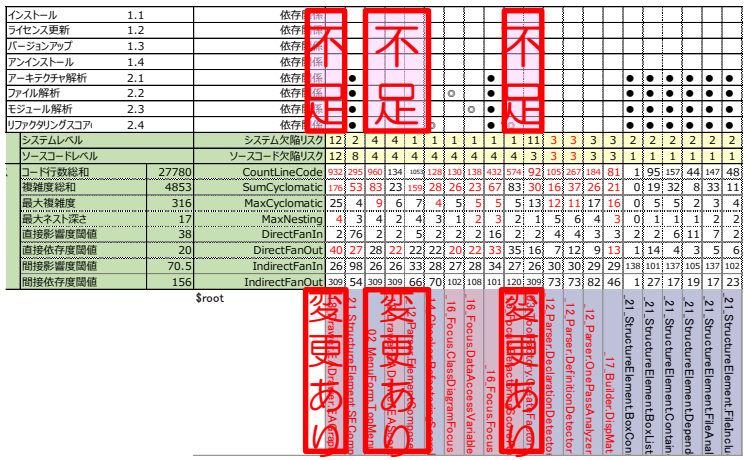
■効果１：修正箇所を迅速に発見できる

作業時間は、１時間前後！

■効果２：修正箇所、影響範囲の検出精度が高い

検出率は、ほぼ100%！

■効果３：テストケースの不足を発見できる



| | | |

派生開発のテストでの効果

■効果４：網羅とピンポイントを使い分けたテスト戦略をとることができる

- ●無駄なテストが減る（効率性）
- ●リスクの高い箇所を狙う（有効性）

例えば、一般的に回帰テストでは、すべてのテストを実行するが、本手法を使うことで無駄なテストを排除し有効なテストに絞ることができる！

| | | |

6. 今後の取り組み

Copyright © Ricoh IT Solutions Co., Ltd. All Rights Reserved.

今後の取り組み

- テストケース設計の効率化
  - ソースコードよりテスト条件を抽出し、テストケースを自動生成する

Copyright © Ricoh IT Solutions Co., Ltd. All Rights Reserved.

参照

- Daniel J. Sturtevant, “System Design and the Cost of Architectural Complexity”, MIT Engineering Systems Division, 2013
- 森 孝司, 増子 聡, 狩野 薫, 山田 大介, 「ソフトウェア構造解析のテスト設計への応用」, SPI Japan 2018, JASPIC, 2018
  http://www.jaspic.org/event/2018/SPIJapan/session3C/3C1_ID006.pdf
- 狩野 薫, 森 孝司, 増子 聡, 山田 大介, 「ソフトウェア構造解析を用いた結合テスト技法の紹介」, JaSST’19 Hokkaido, JaSST, 2019
  http://www.jasst.jp/symposium/jasst19hokkaido.html
- 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/

Copyright © Ricoh IT Solutions Co., Ltd. All Rights Reserved.

ご清聴ありがとうございました

RICOH
imagine. change.

BACKSLASH
DESIGN