日科技連 第32年度 ソフトウェア品質管理研究会
第5分科会「ソフトウェアテスト」 教育チーム

テスト初心者に向けた テスト実施スキル向上手法 （SOHT法）の提案

2017年2月24日

主査：秋山 浩一　　副主査：喜多 義弘

武田 匡広
雨宮 真実　株式会社メタテクノ
小此木 済　（株）インテリジェンスビジネスソリューションズ
清水 専一　株式会社コスモコンピュータセンター
藤野 真澄　富士通株式会社

目次

1．現場の課題

テストでバグが
見つからない

出荷後にバグが
見つかる

課題の
解決が難しい

対策案1
テストケースの拡充

課題1
全てテストできない
納期に間に合わない

対策案2
探索的テストの導入

課題2
ベテラン不在で導入できない

対策案1と2はどちらもうまくいかなかった

2. 解決の方向性

「教育チーム」のテーマ

テスト現場の人材育成

ベテランは少ない

ビギナーのスキル底上げ

具体的にどうすればいいのか

3. ベテランとビギナーの違い ①何が違う？

ベテランとビギナーの違いに着目しました

テストケースにない
パターンを推測

テストケースの
手順どおり

ここがバグなら
ここもたぶん・・・

こっちがこうなら
ここはどうだ？

こういう場合は
ここが怪しい

ここを押す

豊富な
経験と知識

ベテラン

その次は
ここを押す

次は
ここを押す

ここを押す

ビギナー

3. ベテランとビギナーの違い ②ベテランの思考（仮説） 6/18

ある局面における特定の操作

↓

設計上の弱点を想定

↓

類似バグの探索

ベテランは、どんな時にどんなバグが出やすいか知っている。
見つけたバグから設計上の弱点を考え、
さらに類似のバグを見つけることができている。

ベテランの思考を利用できないだろうか？

4. “SOHT”を用いたトレーニング ①SOHTとは

SOHT = Situation/Operation/Hypothesis Table
ソット [状況] [操作] [仮説] （表）

バグを検出したときの
「操作」を行った状況

「状況」「操作」から
考えられる
設計上の弱点

バグを検出したときの操作

「仮説」は開発者に確認（判断できない場合）

4. “SOHT”を用いたトレーニング ②SOHTの例

状況	操作	仮説		
		ID	想定される弱点	関連
モニタ解像度の変更が認められている.	SVGAで画面サイズを設定したのち，XGAなどでより小さい画面で起動する.	1	モニタ解像度の違いを考慮した設計が不十分.	2
	縦画面・横画面(ピボットモニター)を切り替えてアプリケーションを運用してみる.			
	ウィンドウサイズを記憶している場合，モニタ解像度変更時の動作が考慮されているかを確認する.			
DPIの変更が認められている.	72DPIに設定しアプリケーションを運用する.	2	モニタ解像度の違いを考慮した設計が不十分.	1
	120DPIに設定しアプリケーションを運用する.		ノートPCやタブレットでの運用の考慮不足.	-

ベテランの持つ「バグ検出の思考過程」を表にした

4. “SOHT”を用いたトレーニング ③SOHT作成手順

作成手順

(1) 要約

欠陥の検出手順（再現手順）を一言に要約する

「状況XXXのときに操作xxxをしたら・・・となった」の形

(2) グループ化

同じ状況で発生する操作をまとめる

(3) 弱点の関連付け

「操作」から「推測される弱点」洗い出し、
類似の弱点を関連づける

（状況XXXでxxxの操作をしてバグを見つけた場合、同様の弱点を有するYYYでyyyをしてもバグか見つかる可能性が高いと考える）

4. “SOHT”を用いたトレーニング ④SOHTの構成

状況	操作	仮説		
		ID	想定される弱点	関連
モニタ解像度の変更が認められている.	SVGAで画面サイズを設定したのち，XGAなどでより小さい画面で起動する.	1	モニタ解像度の違いを考慮した設計が不十分.	2
	縦画面・横画面（ピボットモニター）を切り替えてアプリケーションを運用してみる.			
	ウィンドウサイズを記憶している場合，モニタ解像度変更時の動作が考慮されているかを確認する.			
	設定しアプリケーションを運用	2	モニタ解像度の違いを考慮した設計が不十分.	1
	設		ノートPCやタブレットでの運用の考慮不足.	-

状況ごとにどの様な操作を試すべきかを整理

バグを見つけたら遡って他のバグを見つけに行く

類似の弱点を関連付けしている点が特徴

4. “SOHT”を用いたトレーニング ⑤トレーニング手順11/18

①SOHTの準備
テストケース外で検出された過去バグ情報を用いてビギナー本人がSOHTを作成する（**ベテランの指導あり**）

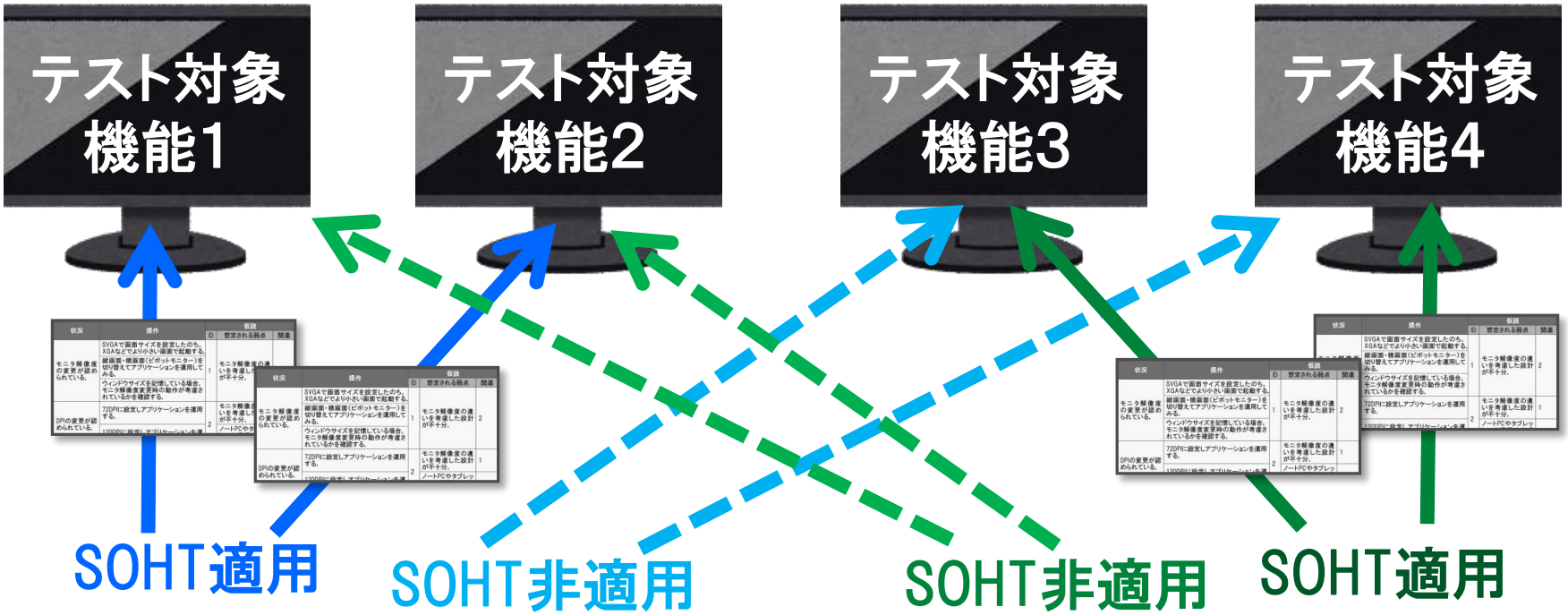
②テスト実施
SOHTに記載済みの「状況」の場合に「操作」を実施する
バグを見つけたら類似バグも探す

③SOHTの評価
設定した「仮説」に対して**実際にバグを検出することができたかどうか**評価する

④SOHTの更新
評価にもとづいて**SOHTが肥大化しないように**追加／削除する

実際のテスト環境下で繰り返し実施し、鍛える

5. 検証方法 ①構成



ビギナー Bチーム

**4機能のテストを
2チームに
分かれて実施**

ビギナー Aチーム

SOHTの適用でバグ検出力に差は出るか？

5. 検証方法　②テスト対象機能の比較

機能名	開発規模	テスト密度	GUI
機能1	約 10 KLOC	3.5 件/KLOC	なし
機能2	約 20 KLOC	10 件/KLOC	あり
機能3	約 25 KLOC	10 件/KLOC	あり
機能4	約 40 KLOC	15 件/KLOC	あり

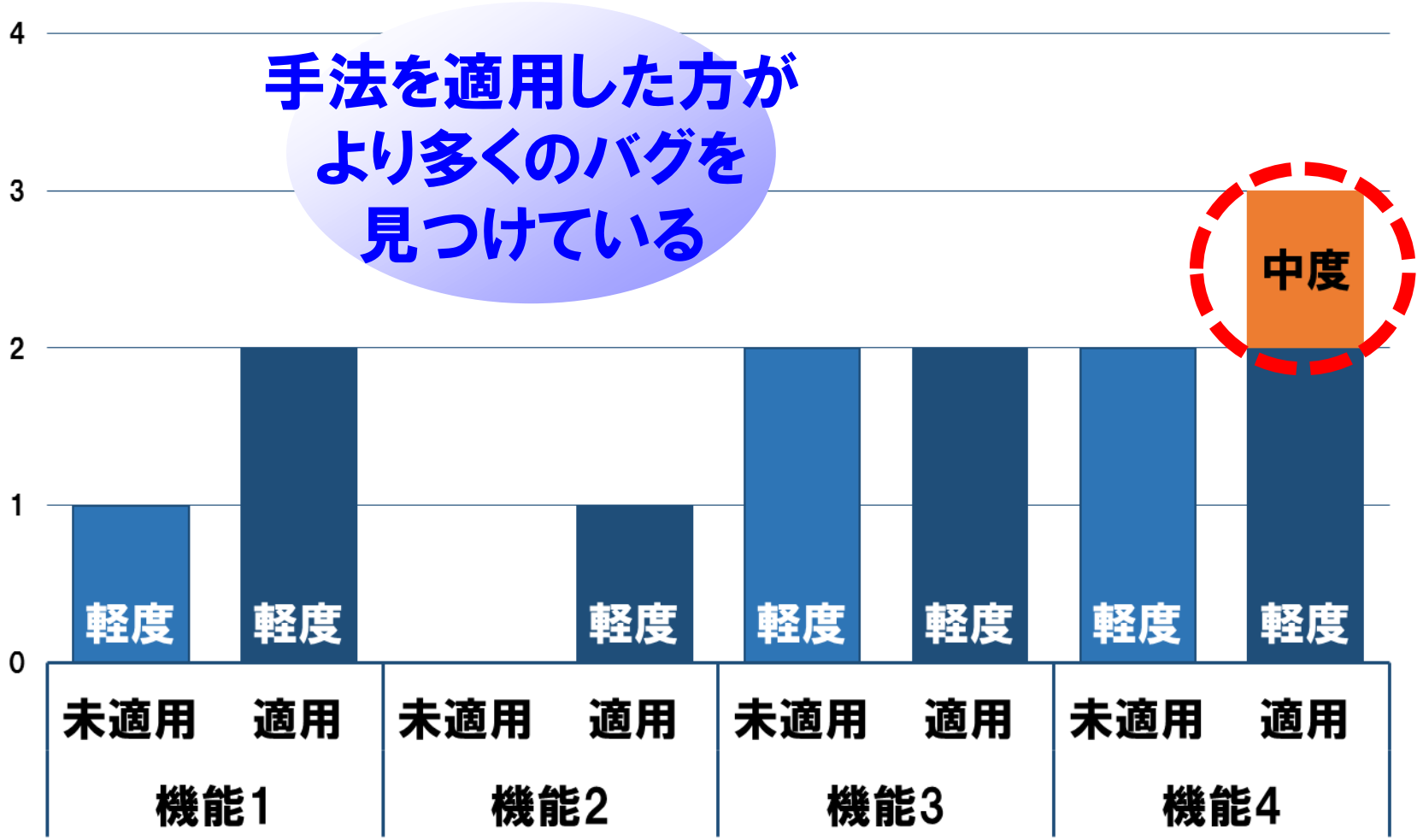
※テスト実施は各機能1回のみ

※各チームが満足するまでテストを実施

5. 検証方法　③バグ重篤度の基準

重篤度	定義	具体例
重	・ユーザーが著しい被害/損害を被る欠陥	・データが紛失する，誤ったデータを生成する ・意図しないアプリケーションの終了/ハングアップ
中	・重,軽のいずれにも該当しない欠陥	・画像の拡大機能が解除できないなど画面表示の問題
軽	・操作性に問題はあるが作業の継続が可能な欠陥 ・機能上は問題ないがユーザーの満足感/安心感を損なう欠陥	・レスポスが悪いなど操作性の問題 ・画面のちらつき，スペルミスなど，見た目の問題

6. 検証結果　①検出バグ数の比較



効果を確認！

6. 検証結果　②テスト実施時間の比較

チーム	機能1	機能2	機能3	機能4
A	0.3	0.5	0.8	1.1
B	0.4	0.6	0.5	0.5

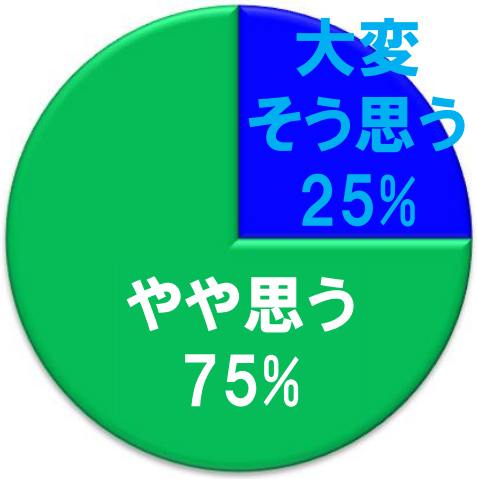
（人/h，下線＝手法を適用）

どの機能においても
提案の手法を適用した方が
テスト実施の時間は長かった

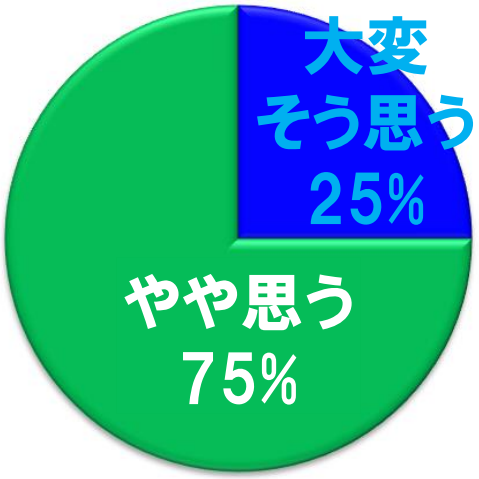
効果を確認！

6. 検証結果　③アンケート結果

トレーニングは
欠陥検出に有効
であると思うか？



トレーニングで
テストの幅は
広がったか？



トレーニングにより、
より重篤な欠陥が
検出できると思うか？

n=4

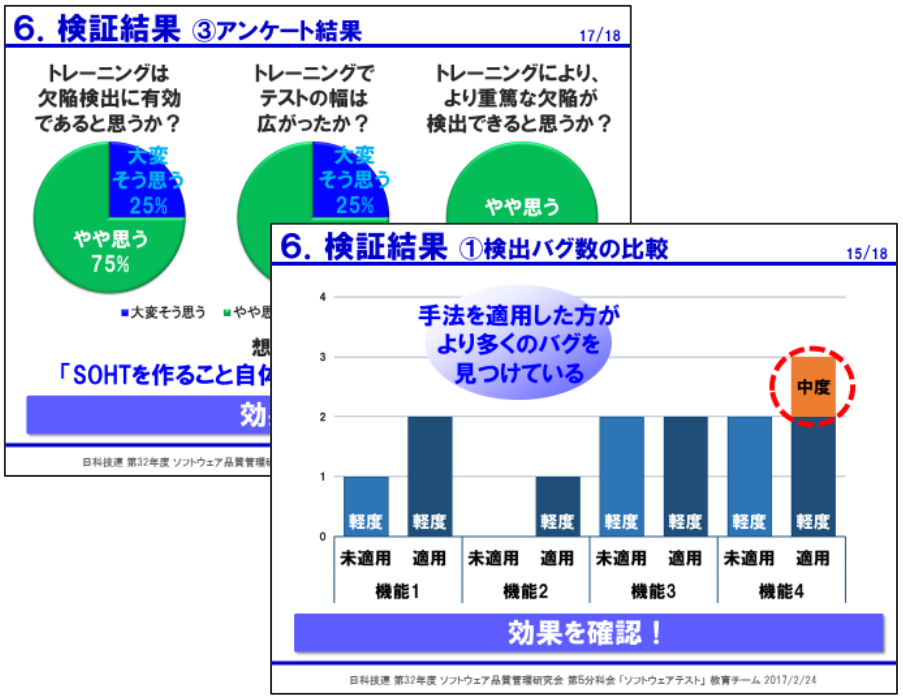
■大変そう思う　■やや思う　■あまり思わない　■まったく思わない

想定外の意見

「SOHTを作ること自体が良いトレーニングになった」

効果を確認！

7. 今後に向けて



「効果を確認」
とは言うものの・・・

課題① サンプルが少ない

課題② 同根欠陥を検出する可能性

まだまだ課題は多い、今後も研究が必要！