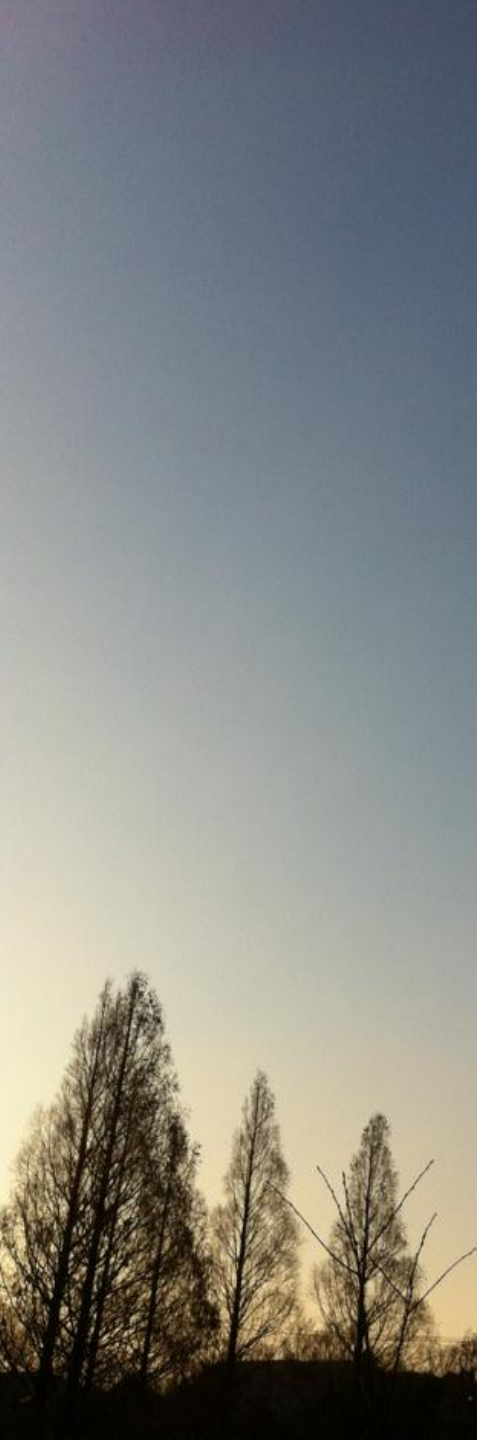


なぜバグ曲線は収束するのか

～Microsoft Excelを使って考えてみる～

JaSST ‘13 Tokyo
2013年1月31日

丹羽　岳雄
株式会社日本総合研究所

バグ曲線は、

ソフトウェア開発の品質管理ツール

の1つとして広く活用されている。

バグ曲線で“よく”議論されていること

より良いモデル
の構築？

曲線収束の判定方法？

最適なモデルの
選択方法？

横軸は、時間？工数？
テストケース数？

なにか　足りない・・・

~~バグ曲線で“よく”議論されていること~~

あまり議論されている**感じがしない**こと

バグ曲線は

なぜ

寝るの？

本セッションの構成

0. はじめに

1. バグ曲線収束理由の検討

2. 収束理由を踏まえたバグ曲線の活用

3. おわりに

バグ曲線収束理由の検討

バグ曲線が収束するとは、

結果論として、

バグがテスト期間の始めの方で
たくさん見つかっている

ということ

ならばそれを考えてみよう

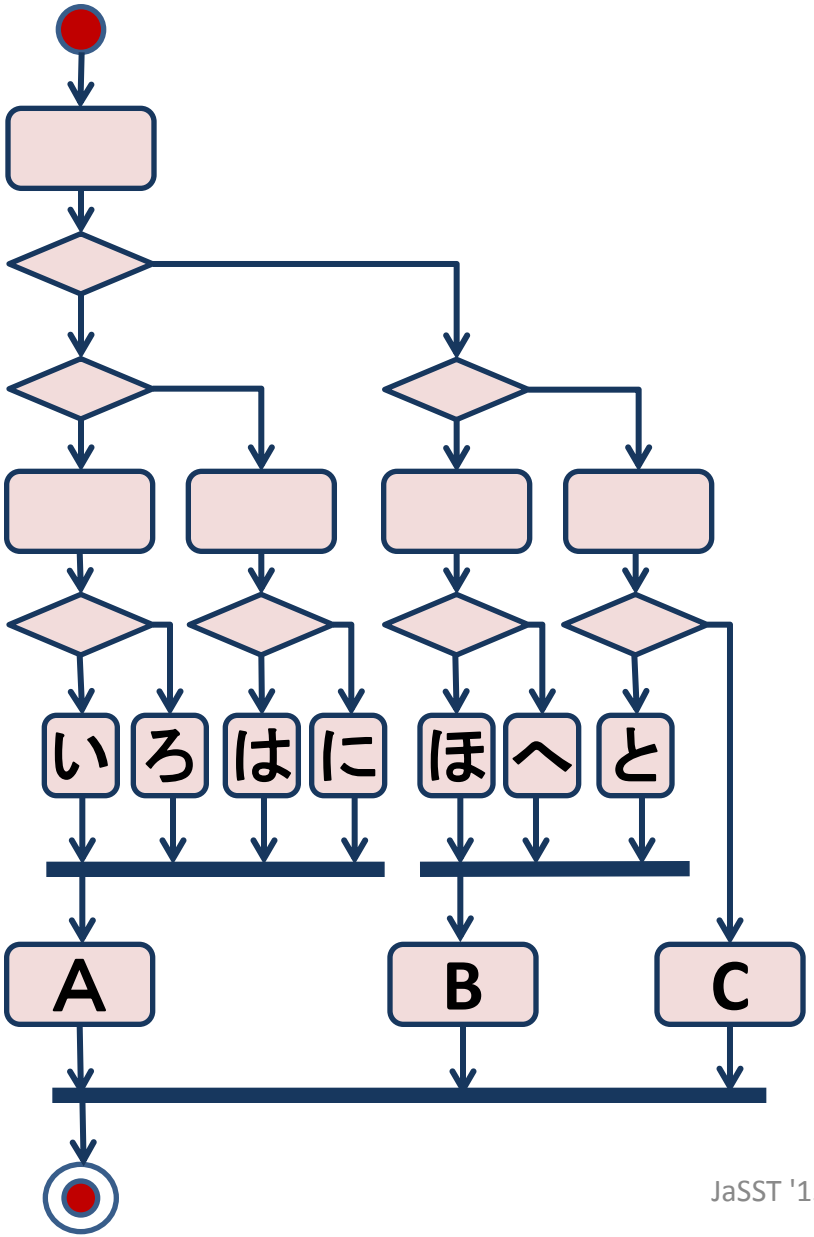
バグ曲線収束理由の検討

検討する収束理由

①テストケースの成り立ち上必然的に生ずるもの
「ある項目をテストしようとすると、
他の項目もテストしてしまう」

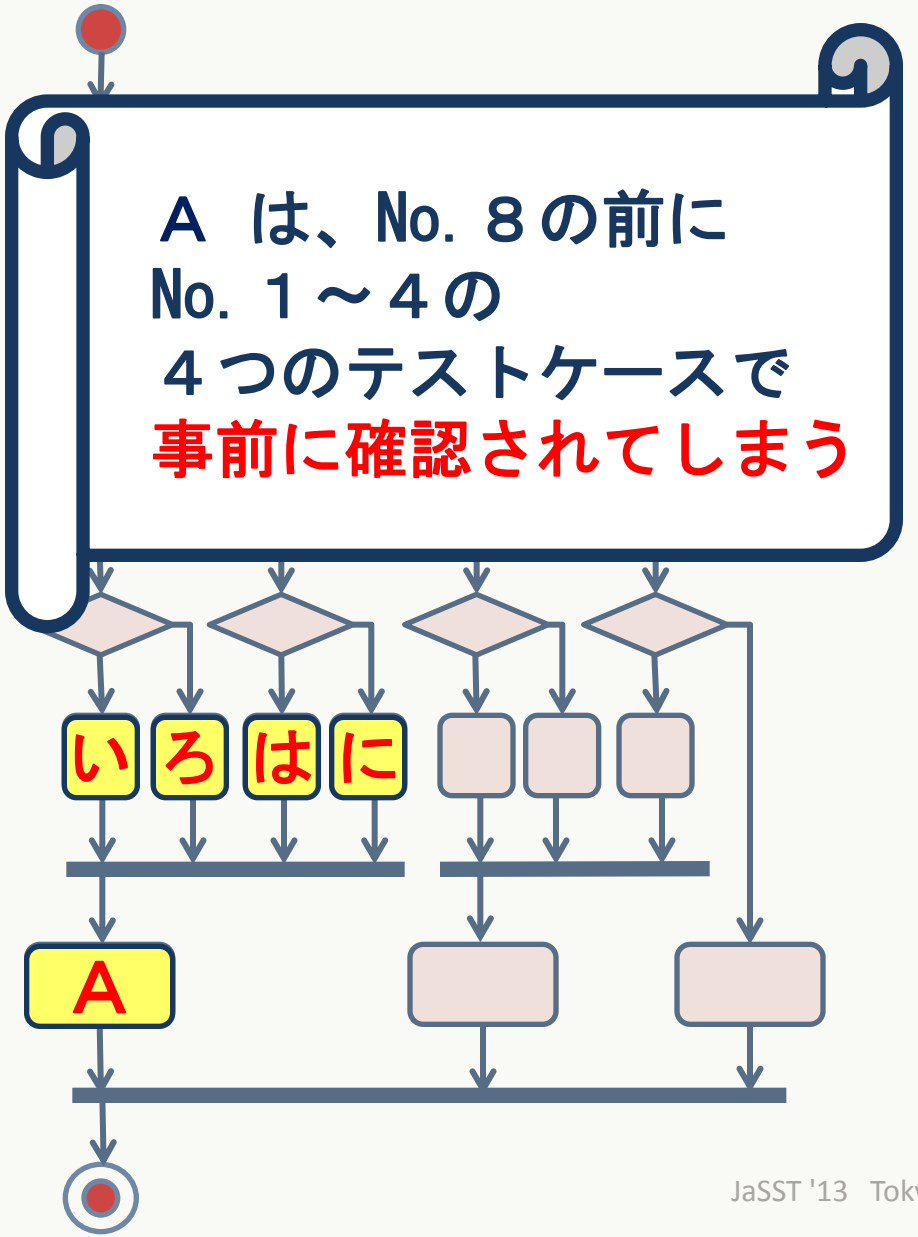
②テスト実施者の意図に由来するもの
【意図の例】
・バグ原因となる箇所を予測したテスト
・恣意的な実施順の決定

テストケースの成り立ち上必然的に生ずるもの



TestCaseNo.	確認したい項目
1	い
2	ろ
3	は
4	に
5	ほ
6	へ
7	と
8	A
9	B
10	C

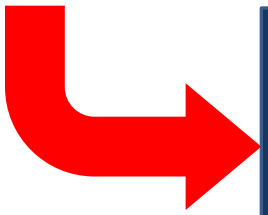
テストケースの成り立ち上必然的に生ずるもの



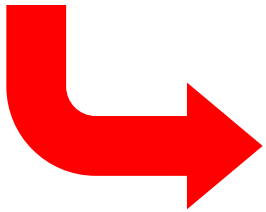
TestCaseNo.	確認したい項目
1	い
2	ろ
3	は
4	に
5	ほ
6	へ
7	と
8	A
9	B
10	C

テストケースの成り立ち上必然的に生ずるもの

Ａ　を確認するためのNo. ８のテストケース実行前に、
No. １～４の４つのテストケースで
Ａについて**事前に確認されてしまう**



個々のバグを発見できるのは、その部分の確認を意図するテストケースだけとは限らない。



個々のバグを発見できる
テストケース数が多ければ
バグ発見が早くなる（のでは？）

個々のバグを発見できるテストケース数が多ければバグ発見が早くなる(・・・のでは？)

"テストケーススイート1"
- テストケース　10000件
- バグ　200件

条件1　各バグは1つのテストケースのみで発見	【重複無】
条件2　各バグは3つのテストケースで発見	【3件重複】
条件3　各バグは5つのテストケースで発見	【5件重複】

個々のバグを発見できるテストケース数が多ければバグ発見が早くなる(・・・のでは？)

あるバグを発見可能なテストケースが、
何番目のテストケースで実行されるかは、乱数を生じさせるエクセル関数を使い、

RANDBETWEEN (1,10000)

と表すことができる。

3件重複であれば、バグを最初に発見できるテストケースの実施順は、

MIN (RANDBETWEEN (1,10000),
RANDBETWEEN (1,10000),
RANDBETWEEN (1,10000))

と表すことができる(厳密には3つの乱数がそれぞれ異なるような処理が必要)。

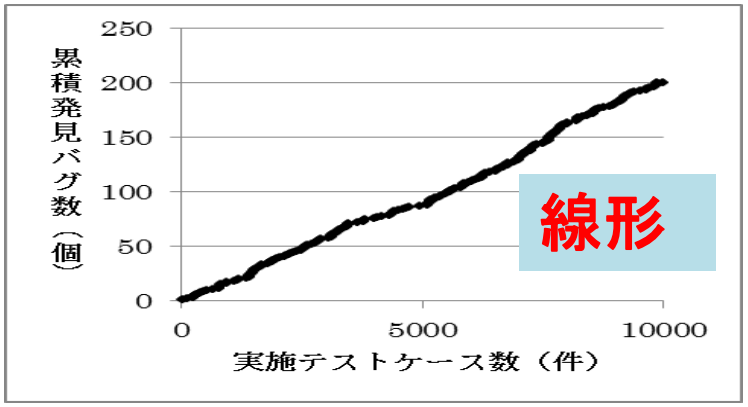
個々のバグを発見できるテストケース数が多ければバグ発見が早くなる（・・・のでは？）

- テストケース　10000件　　- バグ　　200件
は、それぞれの関数を200回実施して得られる値を
昇順に並べて累積的にグラフを描けばバグ曲線が得られる

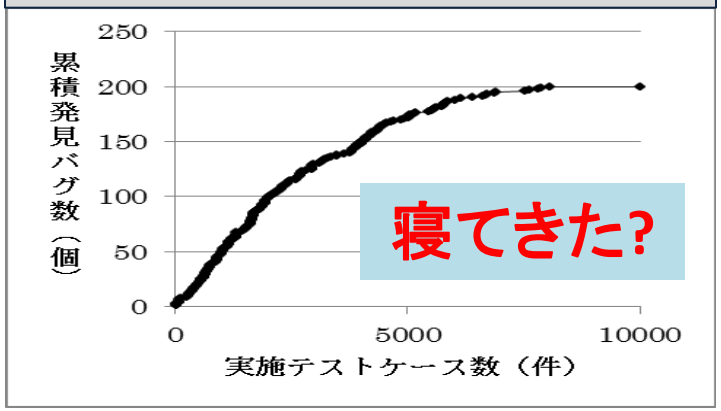
条件1	各バグは1つのテストケースのみで発見	【重複無】
	RANDBETWEEN (1,10000)	
条件2	各バグは3つのテストケースで発見	【3件重複】
	MIN (RANDBETWEEN(1,10000), RANDBETWEEN(1,10000), RANDBETWEEN(1,10000))	
条件3	各バグは5つのテストケースで発見	【5件重複】
	MIN (RANDBETWEEN(1,10000), RANDBETWEEN(1,10000), RANDBETWEEN(1,10000), RANDBETWEEN(1,10000), RANDBETWEEN(1,10000))	

個々のバグを発見できるテストケース数が多ければバグ発見が早くなる

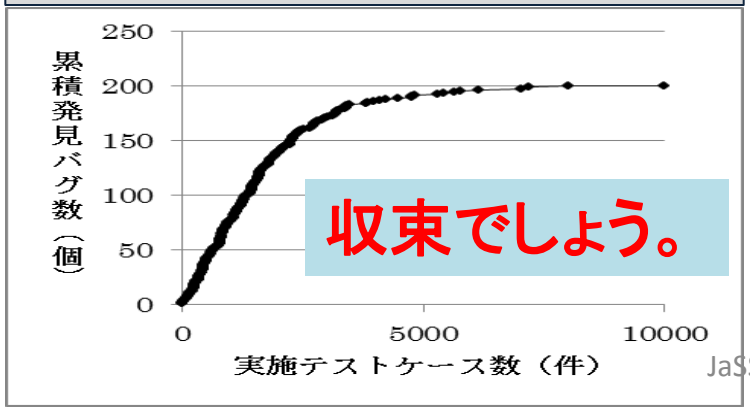
条件1： 各エラーは1つのテストケースのみで発見



条件2： 各エラーは3つのテストケースで発見



条件3： 各エラーは5つのテストケースで発見



テストケーススイート内に、同一バグを発見できるテストケースが多いほどバグ発見は早くなる

テストケースの成り立ち上必然的に生ずるもの

三段論法

あるテストケースが、
「他の項目もテストしてしまう」

バグ発見は早くなる

バグ曲線は収束しやすくなる

重複は
収束理由になる

バグ曲線収束理由の検討

検討する収束理由

①テストケースの成り立ち上必然的に生ずるもの
「ある項目をテストしようとすると、
他の項目もテストしてしまう」

②テスト実施者の意図に由来するもの
【意図の例】
・バグ原因となる箇所を予測したテスト
・恣意的な実施順の決定

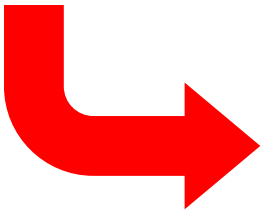
テスト実施者の意図に由来するもの

【バグを早く発見したいというテスト実施者の意図】

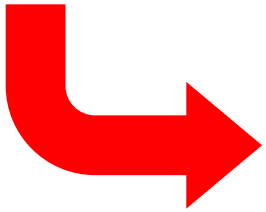
・バグ原因となる箇所を予測したテスト

・恣意的な実施順の決定　　etc

これらが上手くいけば



バグ発見は早くなる

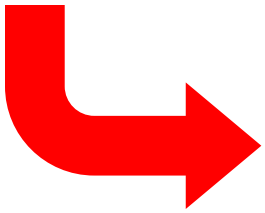


バグ曲線収束に貢献！

テスト実施者の意図に由来するものでバグ発見が早くなる（・・・とバグ曲線は収束するのでは？）

テストの早い段階ほど発見できるテストケースが、多いことは、例えば以下のように表すことができる。

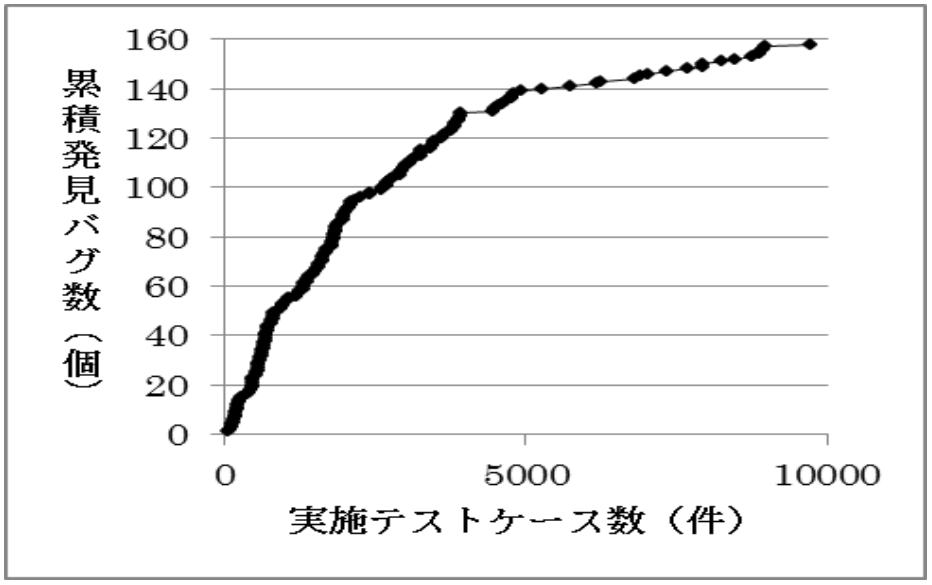
テストケース区間.	発見障害件数（重複無しとする）
1, 2000	90
2001, 4000	40
4001, 10000	20
6001, 10000	8



RANDBETWEEN (1,2000) が 90 個
RANDBETWEEN (2001,4000) が 40 個
RANDBETWEEN (4001,10000) が 20 個
RANDBETWEEN (6001,10000) が 8 個

テスト実施者の意図に由来するものでバグ発見が早くなりバグ曲線は収束に向かう(かな？)

RANDBETWEEN (1,2000)	が	90 個
RANDBETWEEN (2001,4000)	が	40 個
RANDBETWEEN (4001,10000)	が	20 個
RANDBETWEEN (6001,10000)	が	8 個



まあ、収束していく形に見えるように値を選んだのだから・・・
自演と言えば自演。

バグ曲線収束理由の検討

検討する収束理由

①テストケースの成り立ち上必然的に生ずるもの
「ある項目をテストしようとすると、
他の項目もテストしてしまう」

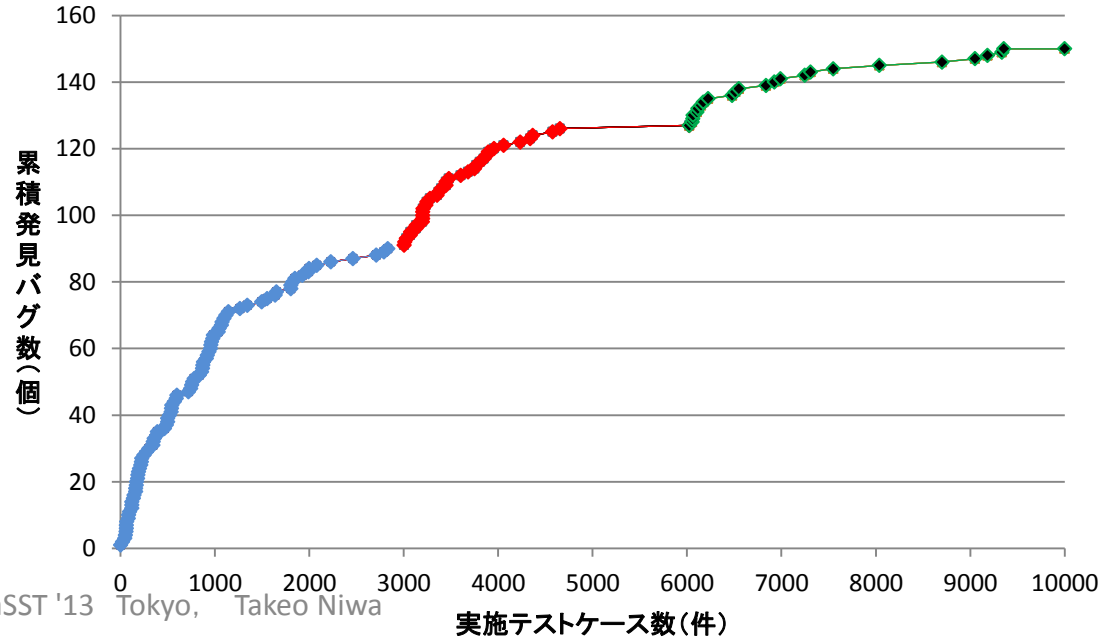
②テスト実施者の意図に由来するもの
【意図の例】
・バグ原因となる箇所を予測したテスト
・恣意的な実施順の決定

③①と②の複合形

テストケースの成り立ち上必然的に生ずるものとテスト実施者の意図に由来するものの複合形

テストケース区間.	発見障害件数				
	n=1	n=2	n=3	n=4	n=5
1, 3000	15	15	15	15	30
3001, 6000	0	6	6	12	12
6001, 10000	8	8	8	0	0

n ： 重複の数



本セッションの構成

0. はじめに

1. バグ曲線収束理由の検討

2. 収束理由を踏まえたバグ曲線の活用

3. おわりに

収束理由を踏まえたバグ曲線の活用

ここまでで確認したこと

次の2つはバグ曲線収束の原因となりうる。

①テストケースの成り立ち上必然的に生ずるもの【重複】

②テスト実施者の意図に由来するもの【意図】

→ 原因は他にもあるかもしれない

しかし

→ バグ曲線を活用するテスト管理において、
少なくとも2つの意識すべきポイントを得た

収束理由を踏まえたバグ曲線の活用

過去のバグ曲線の傾向や信頼度成長曲線を用いた
分析を行う際には以下の注意が必要

重複	テストケーススイートの網羅性が不十分でも、 重複していれば収束に向かう
意図	テスト実施者や実施環境によって、テスト実施順が 変わり、異なるバグ曲線となる可能性がある

本セッションの構成

0. はじめに

1. バグ曲線収束理由の検討

2. 収束理由を踏まえたバグ曲線の活用

3. おわりに

おわりに

本セッションでは、

テストケーススイートが存在する場合のバグ曲線

について検討した

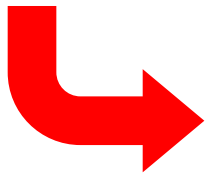
【先の繰り返し】

“過去のバグ曲線の傾向や信頼度成長曲線を
用いた分析を行う際には、
テストケース作成やテスト実施順に注意が必要“

おわりに

本セッションでは、
テストケーススイートが存在する場合のバグ曲線
について検討した

では、
ベータテストや運用テストのような
個々のテストケーススイートを前提としないテスト
の場合はどうだろうか？



テストケースとか恣意とか
あまり意識しない

おわりに

ベータテストや運用テスト

テストケースとか恣意とか意識せず、
どれだけの期間やるかがポイント

横軸はテストケースでなく時間でよい

普通の信頼度成長モデルが
使えるのでは？

おわりに（おまけ）

今回 Excel 関数で検討したのは・・・

バグ曲線での管理を指示する方々のPCには、
Excel が必ず入っている。
だから一緒にその特性を考えて欲しいのです。

・・・ということで、
最初に帰って。

バグ曲線は
どうして
寝るのでしょうか？

Thank You ！

E-mail: niwa.takeo@jri.co.jp

The opinions expressed in this presentation are solely those of the presenter and do not necessarily reflect the opinions of any organization.