日産自動車における ソフトウェア品質向上活動
—SWEEP—

日産自動車(株)
ソフトウェア品質グループ
菊池光彦
2006年1月30日

目次

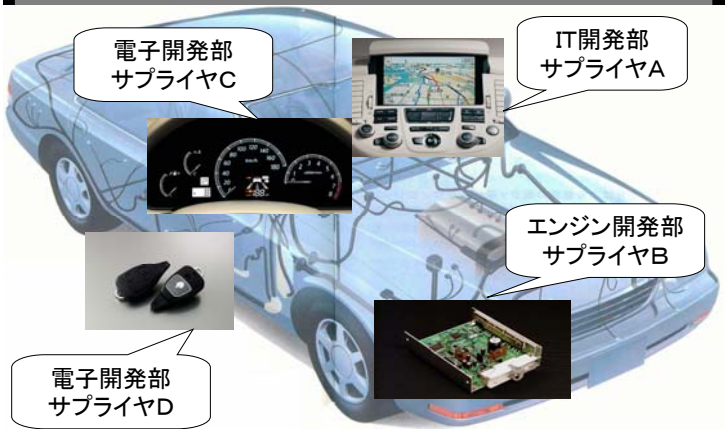
1. 歴史
クルマは常に進化し続ける

環境
省エネ・CO_2

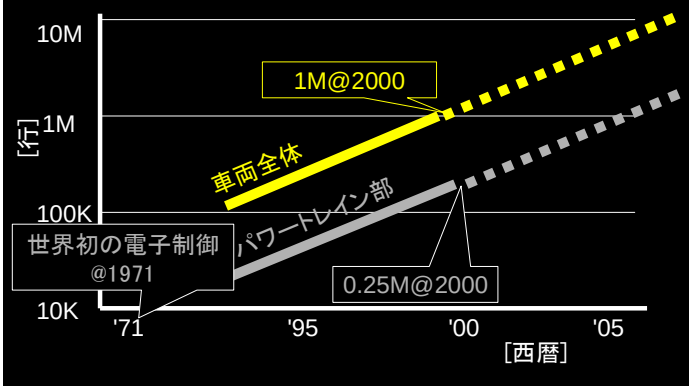
魅力品質

安全性

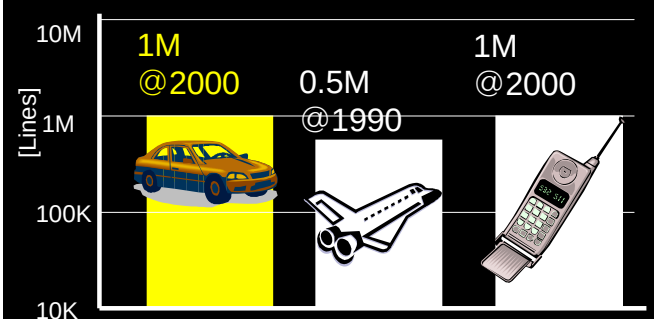
1. 歴史
1台の車には数十個のECUがある



1. 歴史
ソフトウェアサイズは膨大 @2000

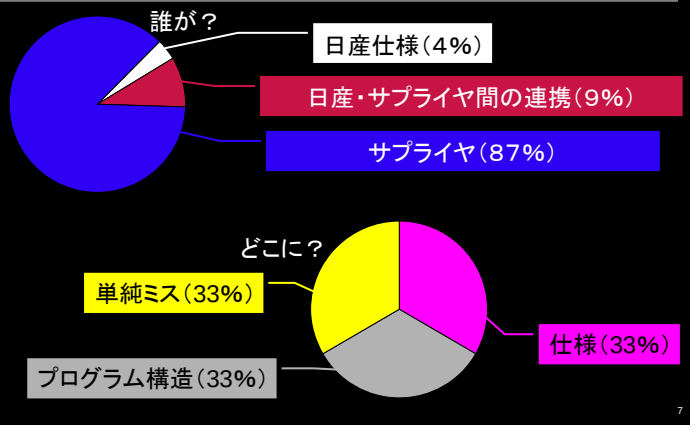


1. 歴史
ソフトウェアサイズは膨大 @2000



1. 歴史

開発中のソフトウェア不具合の分析 @2000



1. 歴史

2001年4月にSWEEP活動を開始

Soft**W**are
Engineering
Evolution
Program

目次

2. SWEEP

SWEEP活動の開始

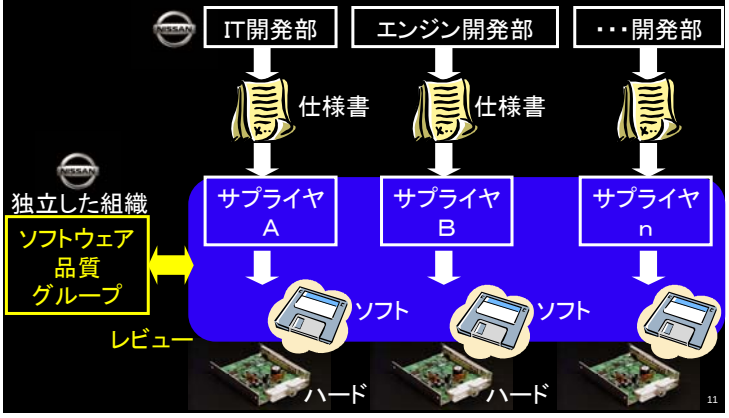
■ソフトウェア品質グループの設立

■ソフトウェア開発プロセスの構築

■ソフトウェア開発のルールの確立

2. SWEEP

ソフトウェア品質グループの役割



2. SWEEP

ソフトウェア品質グループの役割

範囲：
車載されるすべてのECUのソフトウェア

役割:
- **日産標準の開発、社内適用・サプライヤ適用のフォロー**
- **サプライヤのソフト開発プロセスのレビュー、プロセスどおりに進んでいることの第3者レビュー**
- **ハイリスク部品に対する特別活動**
- **日産役員への定期報告**
- **よりよいツールや手法の調査と開発**

2. SWEEP
標準プロセス・ルールの構築

個人の経験・ノウハウ

過去の成功体験　組み込みエンジニアの常識　過去の失敗体験



再利用可能な仕組みとしての〝標準〟

2. SWEEP
実践的アプローチを採用した

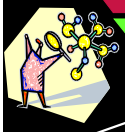
プロセス・ルール構築のポイント

- 確立した技術
- 技術とプロセスの整合
- 重大不具合の再発防止
- 成功体験の展開

2. SWEEP
ソフトウェア開発のプロセス


プロセスレビュー

仕様設計	実車テスト
構造設計	結合テスト
詳細設計	単体テスト
コーディング	コードレビュー

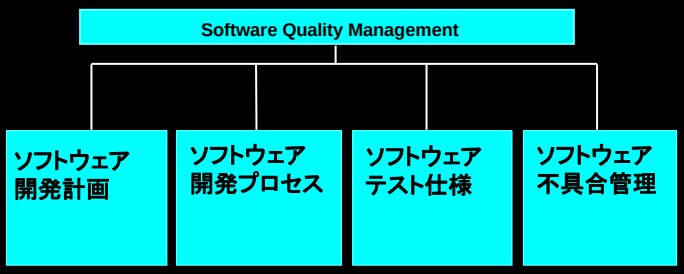



仕様レビュー
構造レビュー

テスト仕様レビュー
SQAツールレビュー

2. SWEEP
ソフトウェア開発のルール

4カテゴリの標準ルールを規定



目次

3. ソフトウェアレビュー
日産／サプライヤの開発分担

日産

・ヌケモレや矛盾の無い仕様を作成する。
・実車を用いて、システム間機能評価や車両性能評価を実施する。

仕様設計	実車テスト
構造設計	結合テスト
詳細設計	単体テスト
コーディング	コードレビュー

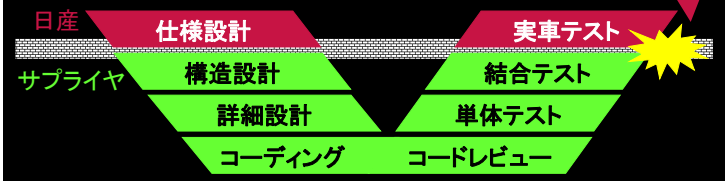
サプライヤ

・要求仕様を分析し、整理する
・ソフトウェアを設計／開発し、テストする。
・ソフト品質向上のため、節目毎にレビューを行う。

3. ソフトウェアレビュー

日産／サプライヤの開発分担

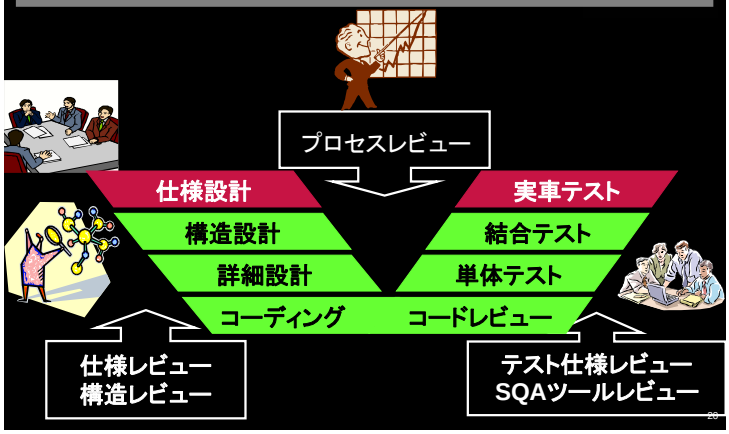
✓日産で不具合を発見したとしても、
日産ではソフトの詳細を直接解析できない！
✓実車不具合は再現困難！



ソフト開発はサプライヤの分担
ソフトはサプライヤの資産

3. ソフトウェアレビュー

日産／サプライヤによるソフト品質レビュー



3. ソフトウェアレビュー

レビューの内容

全体

プロセスレビュー
- サプライヤ評価：弱点を分析し、今後のレビュー方針を決める
- プロセスアセスメント：各プロセスが標準プロセスどおりに実施されていることを確認する

設計工程

仕様レビュー
- 設計部署とサプライヤ間で実施し、仕様のヌケモレ・矛盾を無くす

構造レビュー
- シンプルな構造になっていること、テスト容易な構造になっていることを確認する

テスト工程

テスト仕様レビュー
- ソフトウェアが仕様書通りにできている事を確認する
- ゼロ割りなどの純粋なソフトバグの観点でテストされている事を確認する

ソースコードレビュー (SQAツール解析)
- 設計基準やコーディングルールが守られていることを確認する

3. ソフトウェアレビュー

日産のすべきこと

■プロセスレビュー

■要求仕様レビュー

■テスト仕様レビュー

■構造レビュー

■SQAツールレビュー

3. ソフトウェアレビュー

必要理由

■プロセスレビュー

■要求仕様レビュー

■テスト仕様レビュー

■構造レビュー

■SQAツールレビュー

3. ソフトウェアレビュー

プロセスレビューは必要！

・作ろうとしているソフトにあった設計技術・手法の確立

・個人に依存せず、組織として作業・工程が標準化されていることを確認

・強み／弱みを把握し、補強

3. ソフトウェアレビュー

たとえ話

・設計技術・手法の確立

＝おいしいラーメンのレシピを持っていること

・個人に依存せず、組織として作業・工程が標準化されていることを確認

＝毎回おいしいラーメンを繰り返し作れること

・強み／弱みを把握し、補強

＝レシピはわかりやすく書くこと

25

3. ソフトウェアレビュー

必要理由

■プロセスレビュー

■要求仕様レビュー

■テスト仕様レビュー

■構造レビュー

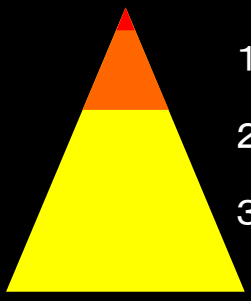
■SQAツールレビュー

26

3. ソフトウェアレビュー

ハインリッヒの法則（1:29:300）

労働災害の事例から導き出された統計値
H. W. Heinrich (米), **出典: フリー百科事典『ウィキペディア』**

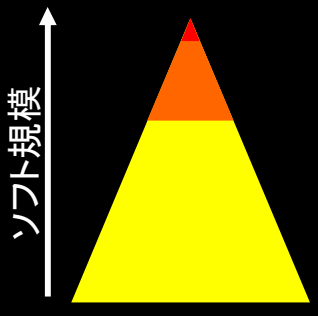


1件の重大災害
の影には
29件の軽微災害
があり
300件のヒヤリハット
が隠れている

27

3. ソフトウェアレビュー

「見える不具合」は氷山の一角

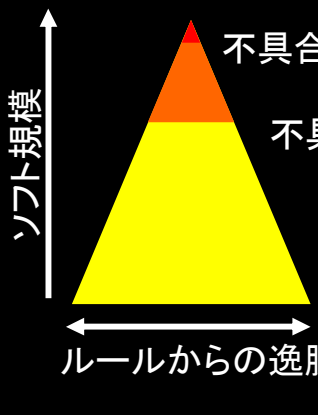


1件の重大不具合
の影には
29件の軽微不具合
があり
300件のルール違反
が隠れている

28

3. ソフトウェアレビュー

氷山が小さければ不具合は少ない



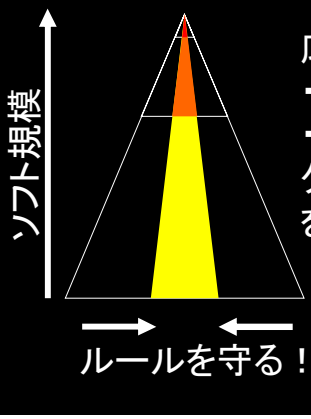
不具合の総数＝3角形の面積

不具合を減らすには
・底辺
・高さ
を小さくする

29

3. ソフトウェアレビュー

ルールを守れば氷山は小さくなる

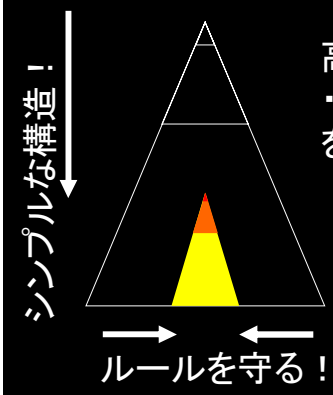


底辺を小さくするには、
・標準プロセス
・ガイドラインやコーディングルール
を守る！

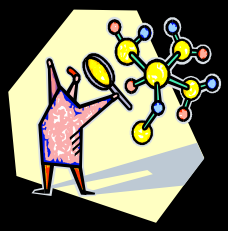
30

3. ソフトウェアレビュー

シンプルに作れば氷山は小さくなる

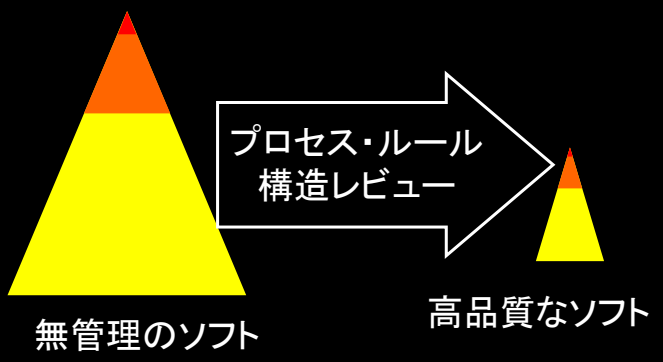


高さを小さくするには、
・シンプルな構造
を設計する！



3. ソフトウェアレビュー

氷山を小さくすることは可能



3. ソフトウェアレビュー

必要理由

■プロセスレビュー

■要求仕様レビュー

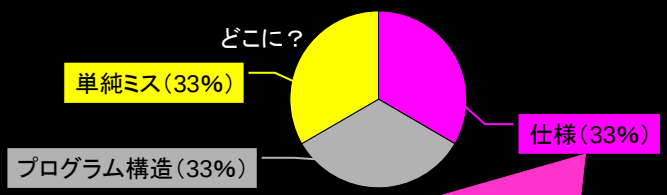
■テスト仕様レビュー

■構造レビュー

■SQAツールレビュー

3. ソフトウェアレビュー

仕様バグには仕様レビューが必要



日産／サプライヤ間の仕様の誤理解・誤伝達

↓

要求仕様分析と要求仕様レビュー

3. ソフトウェアレビュー

必要理由

■プロセスレビュー

■要求仕様レビュー

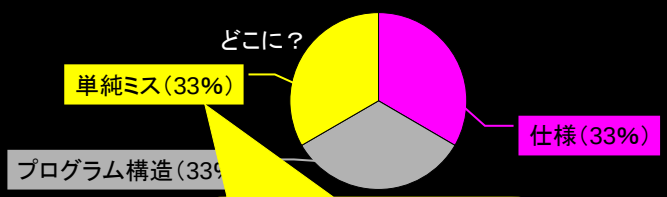
■テスト仕様レビュー

■構造レビュー

■SQAツールレビュー

3. ソフトウェアレビュー

単純ミスにはテスト仕様レビューが必要



テスト不足

↓

テスト項目の規定
テスト仕様レビュー

3. ソフトウェアレビュー

必要理由

■プロセスレビュー

■要求仕様レビュー

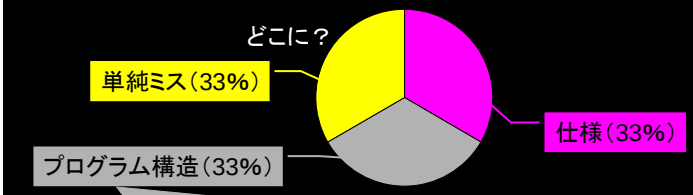
■テスト仕様レビュー

■構造レビュー

■SQAツールレビュー

3. ソフトウェアレビュー

構造バグにはソフト構造レビューが必要



複雑な構造による想定外タイミングの動き

シンプルな構造とソフト構造レビュー

3. ソフトウェアレビュー

必要理由

■プロセスレビュー

■要求仕様レビュー

■テスト仕様レビュー

■構造レビュー

■SQAツールレビュー

3. ソフトウェアレビュー

効率向上＆正確化にはSQAツールが必要

・多彩で詳細なルールや手順による非効率化

・完全にゼロにはできないヒューマンエラー

SQAツールによる効率向上と正確化

3. ソフトウェアレビュー

サプライヤのすべきこと・依頼事項

■プロセスレビュー
→標準の開発プロセスの構築

■要求仕様レビュー
→要求分析の実施

■テスト仕様レビュー
→トレーサビリティの確保

■構造レビュー
→アーキテクチャ設計の実施

■SQAツールレビュー
→ガイドラインやコーディングルール

3. ソフトウェアレビュー

観点を整理してからレビューに臨む

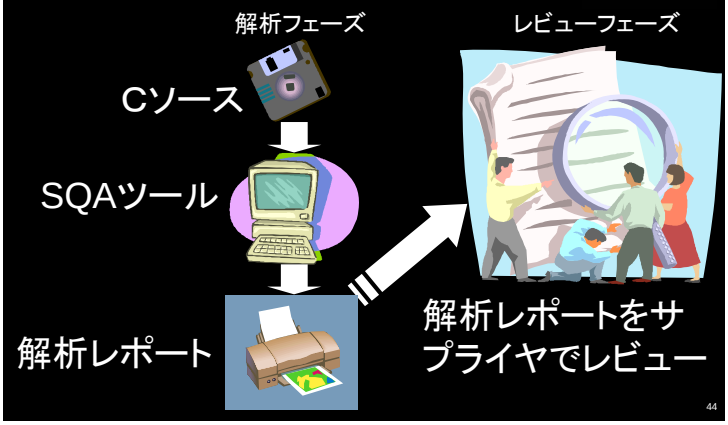
観点 / 工程	仕様（設計部署の観点）	ロバスト性（ソフト品質部署の観点）
システムテスト	仕様は正しく作られているか？	システムダウンしないか？（想定外の使われ方）
結合テスト	ソフトは仕様通りに作られているか？	タイミングBUGやタスク間I／Fに問題はないか？（想定外のタイミング）
単体テスト	ソフトは仕様通りに作られているか？ 例外処理は考えられているか？	ゼロ割やオーバーフローなどはないか？（想定外の入力）

目次

1. 歴史
2. SWEEP
3. ソフトウェアレビュー
4. SQAツールレビュー
5. まとめ

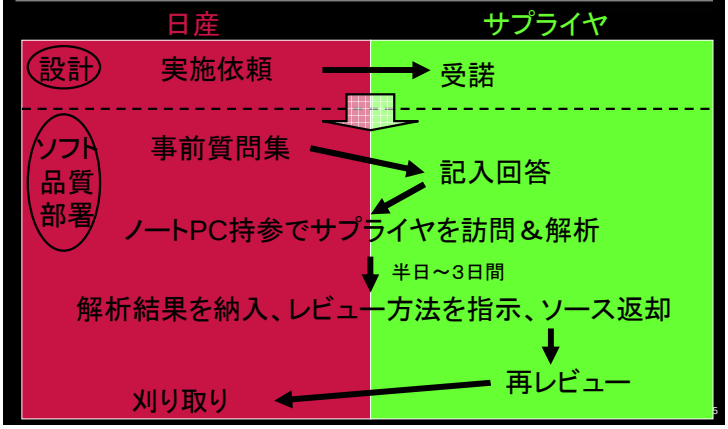
43

4. SQAツールレビュー
解析フェーズとレビューフェーズ



44

4. SQAツールレビュー
サプライヤと日産とで協調して実施

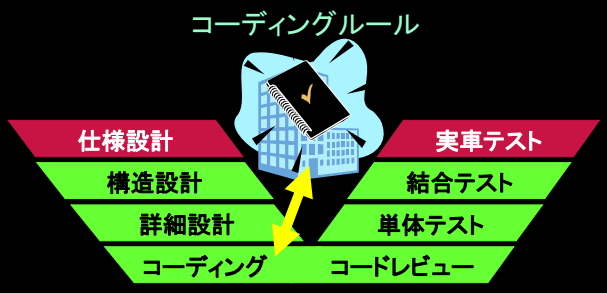


4. SQAツールレビュー
ソースコードは保護される

- SQAツール解析はサプライヤ・日産の合意が前提
- サプライヤサイトで実施することで、ソースコード流出を防止
- 日産所有のPCへのコピーと削除は、サプライヤ担当者の同席が前提
- SQAツール解析中はサプライヤ担当者の同席が前提

46

4. SQAツールレビュー
SQAツールでコードレビューを支援



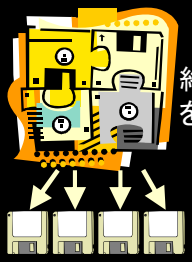
・テストしやすいコードであること←SQAツールで支援
・ルール通りのコードであること←SQAツールで支援
・仕様どおりのコードであること

47

4. SQAツールレビュー
テストしやすいコードであること

スパゲティコードは
デバッグ困難

コードのスパゲティ度合いを数値化
→SQAツールでメトリックス測定



絡まっている箇所を特定

単純な構造に作り変える

48

4. SQAツールレビュー

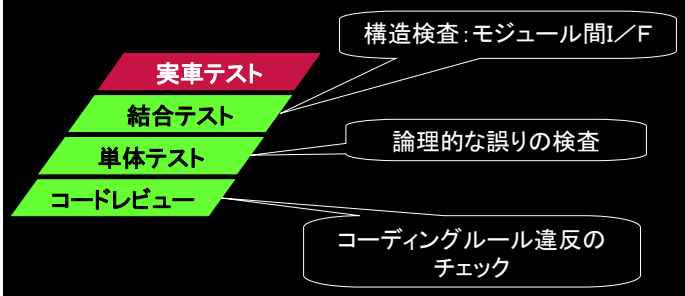
メトリックスで複雑箇所を特定

プログラム構造の複雑さ
- ・モジュール間I／F

関数の複雑さ
- ・静的パスカウント
- ・実行行数

4. SQAツールレビュー

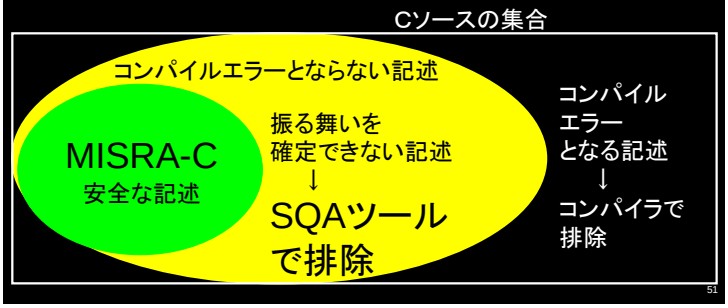
ルール通りのコードであること



4. SQAツールレビュー

コーディングルール違反

たとえばMISRA-C　　http://www.misra.org.uk
(社)自動車技術会より日本語版が発行



4. SQAツールレビュー

論理的な誤り

- ✓ オーバーフロー
- ✓ アンダーフロー
- ✓ ゼロ除算
- ✓ 配列の範囲外アクセス
- ✓ 不正なポインタアクセス
- ✓ 実行されないコード

過去の不具合経験から
代表的な6個を重視

4. SQAツールレビュー

構造的な誤り：モジュール間I／F

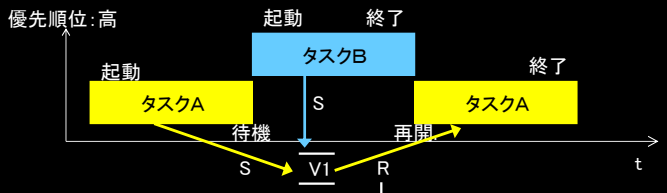
- ✓ 多重書き込み変数
- ✓ リエントラント関数
- ✓ 再帰関数
- ✓ 書き込まれない変数 ／読まれない変数
- ✓ Z変数
- ✓ 意味のない演算
- ✓ タスク間共有変数
- ✓ 割り込み許可／禁止のペア
- ✓ DFDのループ

過去の不具合経験から
代表的な9個を重視

4. SQAツールレビュー

要レビュー！多重書き込み変数

タスクや割り込みルーチンから変数の値を上書きする箇所

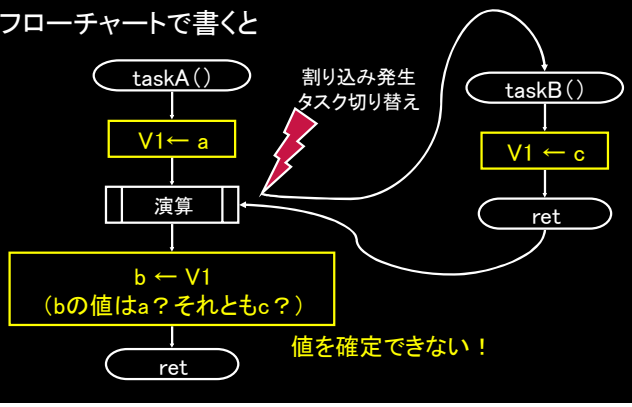


タスクAが変数V1の値を読み込む時、V1の値は、
- ・タスクAが自分で書き込んだ値？
- ・タスクBが途中で上書きした値？　振る舞いを確定できない！

4. SQAツールレビュー

要レビュー！多重書き込み変数

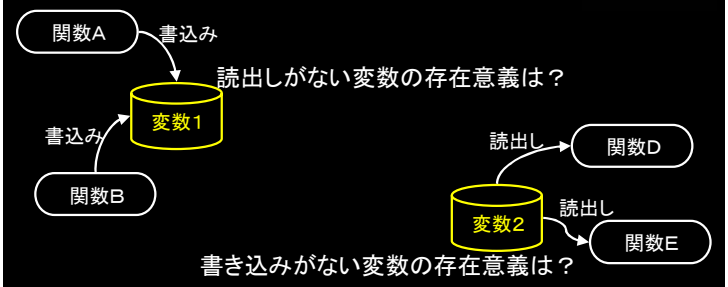
フローチャートで書くと



値を確定できない！

4. SQAツールレビュー

要レビュー！書き込まれない 変数



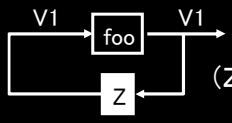
データフロー図を作成すると、すべての要素が線で結ばれているはず！

4. SQAツールレビュー

要レビュー！Z変数

Z変数とは？ ”1周期前の値を保持する変数„

”新鮮ではない値を保持する変数„



ブロック線図
（Zは1周期遅れの意味）

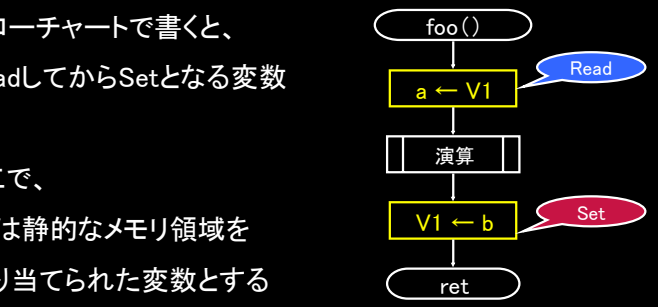
変数V1はZ変数
・関数fooは周期的に実行される
・関数fooは変数V1に値を書き込む
・関数fooは1周期前の変数V1の値を参照する

4. SQAツールレビュー

要レビュー！Z変数

フローチャートで書くと、
ReadしてからSetとなる変数

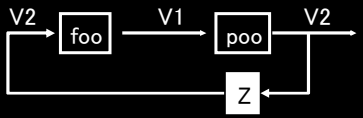
ここで、
V1は静的なメモリ領域を
割り当てられた変数とする



4. SQAツールレビュー

要レビュー！Z変数

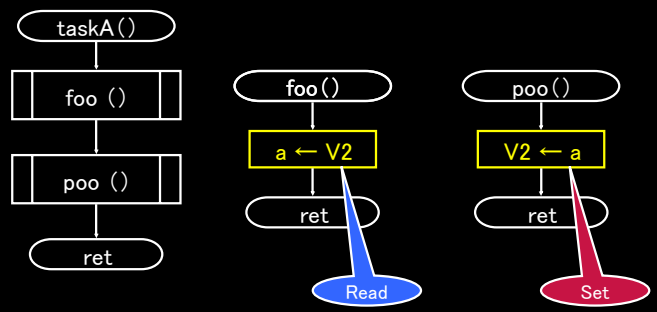
1つの関数の中でのZ変数だけではなく、
タスク内の関数についてZ変数を検出する。



4. SQAツールレビュー

要レビュー！Z変数

タスク全体を見ると、実はV2はZ変数



4. SQAツールレビュー

要レビュー！Z変数

組み込みソフトでの使われ方

・内部状態を保持する状態変数／フラグ

状態遷移で設計されたモジュールの状態変数

・入力信号の変化を見るための前回値

SWのON→OFF／OFF→ONのエッジ検出

アナログ入力の変化の度合いの測定

4. SQAツールレビュー

要レビュー！Z変数のリスク

想定するリスク

・プログラマの単純ミスではないか？

（本当はZ変数ではないのでは？）

・Z変数の場合、初期値は正しいか？

（電源ON時の初回実行時の動作は検討されているか？）

・新鮮でない値を使って制御することに問題はないのか？

4. SQAツールレビュー

要レビュー！Z変数

レビューポイントの抽出

「タスク内の一連の処理の中で、Setする前に一度もReadされない変数」

4. SQAツールレビュー

要レビュー！Z変数のレビューの観点

・Z変数なのは意図どおりか？

・正しい初期値で初期化しているか？

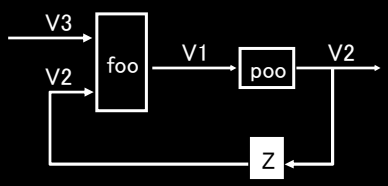
（ゼロなどの固定値で初期化するのではなく、電源ON直後の初期化ルーチンで入力値を読み込んだ上で、その値をZ変数の初期値としているか？）

・初期値や初回の動作は設計されレビューされているか？

4. SQAツールレビュー

要レビュー！Z変数

注意すべきパターン



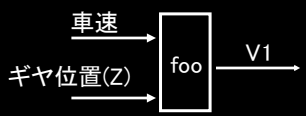
関数foo()は、前回値と今回値を参照している

つまり、値の新鮮さが異なる

4. SQAツールレビュー

要レビュー！Z変数

疑問に思う？

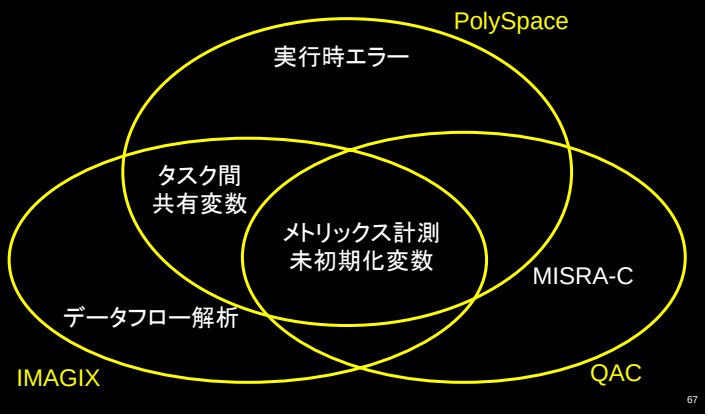


現在の車速が30km/hで、ギヤ位置がリバース

……本当？

4. SQAツールレビュー

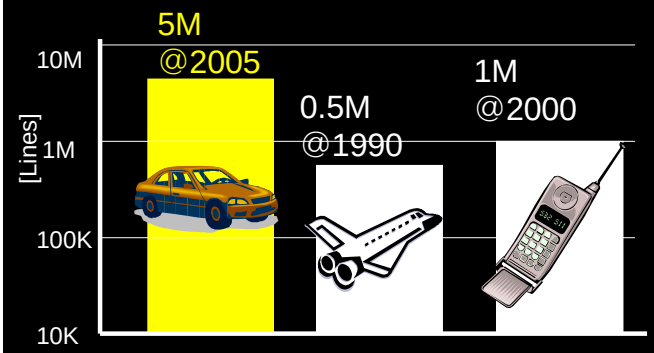
3つのSQAツールを活用



目次

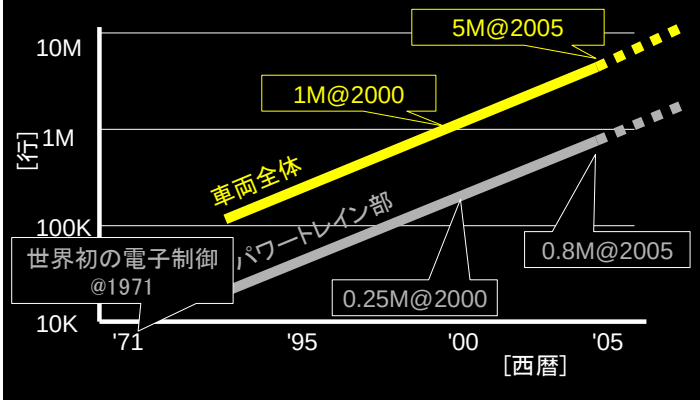
5. まとめ

ソフトウェアはさらに膨大に！ @2005



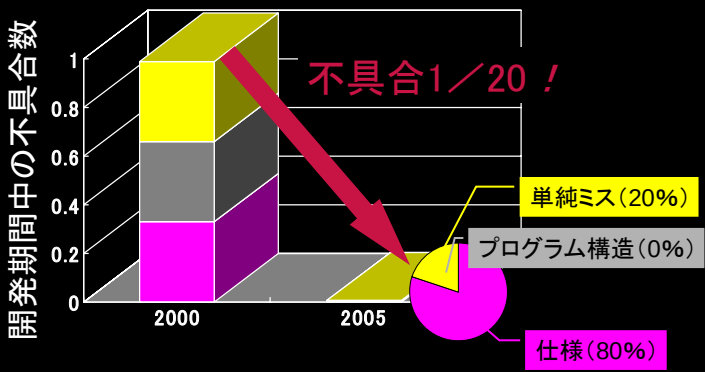
5. まとめ

ソフトウェアはさらに膨大に！ @2005



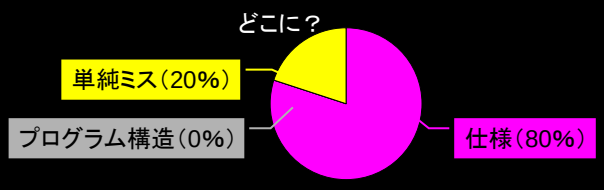
5. まとめ

ソフトウェア不具合は1／20に！ @2005



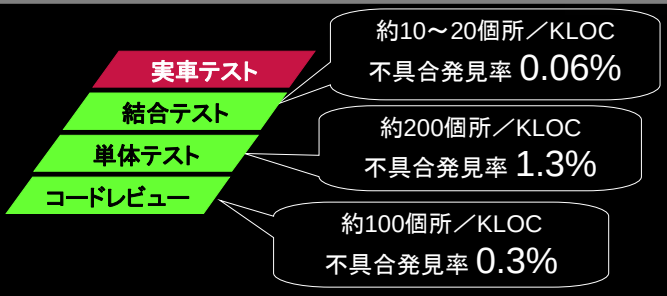
5. まとめ

構造バグはゼロ！ @2005



5. まとめ

SQAツールで万に1つも徹底撲滅！



$$\text{不具合発見率} = \frac{\text{真の不具合数}}{\text{ツール検出数}}$$

（値は日産ソフト品質グループの経験値）

5. まとめ

成功要因

- 個人ノウハウから共有ルールへ
  →1つの成功事例をどのプロジェクトでも再現できた

- 実践的アプローチ：新規技術ではなく既存技術中心の活動
  →特別な導入教育なく、すぐに、全員に、浸透できた

5. まとめ

成功要因

- ソフト品質部署の立場から「サプライヤの生資料を使ってレビューすることで、真の問題点を共有できた」

- サプライヤの立場から「自分の文書・コードを他人に示すことで、自分のやるべきことがよく見えるようになった」

5. まとめ

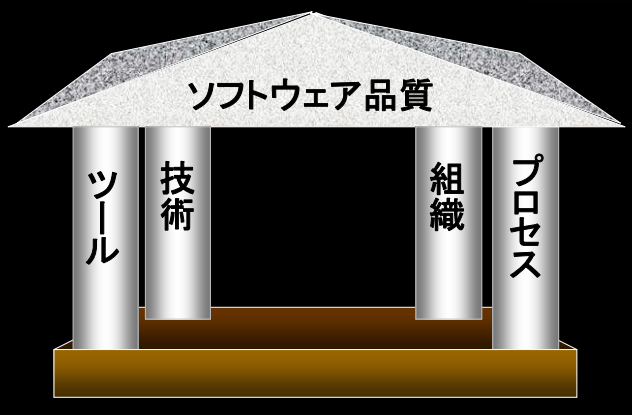
成功要因

- 「やってあたりまえ」の内容
  →全員参加できた
  - 日産設計部
  - 日産ソフト品質グループ
  - すべてのサプライヤ

- 継続的な活動
  →体質・基礎体力として定着できた
  例）社内監査体制

5. まとめ

ソフトウェア品質を支える4本柱



5. まとめ

4本柱：技術

・経験を積むだけでは再利用できない

・技術的視点での問題意識

↓

"デザインルール" や

"チェックリスト" へ昇華させる

5. まとめ

4本柱：ツール

・多彩で詳細なルールや手順による非効率化

・完全にゼロにはできないヒューマンエラー

SQAツールによる効率向上と正確化

5. まとめ

4本柱：プロセス

・「なすべきことをせよ」

・プロセス通りに実施していることのチェック＝メタプロセス

担当者／マネージャの業務処理基準

開発プロセスの定義

5. まとめ

4本柱：組織

・製品開発を担当するグループ

・プロセスやインフラ定義を担当するグループ

・プロセス通りであることを監査するグループ

主体的な行動者の定義、権限の明確化

チェック体制

「それは私がやるんだ」

NISSAN SHIFT_the future