Debug Engineering

JaSST 09九州
09.11.12

テスト・デバッグのトレンド: 過去40年から近未来へ

～テストへの期待と変化～

DebugEng ㈲デバッグ工学研究所
http://www.debugeng.com/
松尾谷 徹　博士、法政大講師

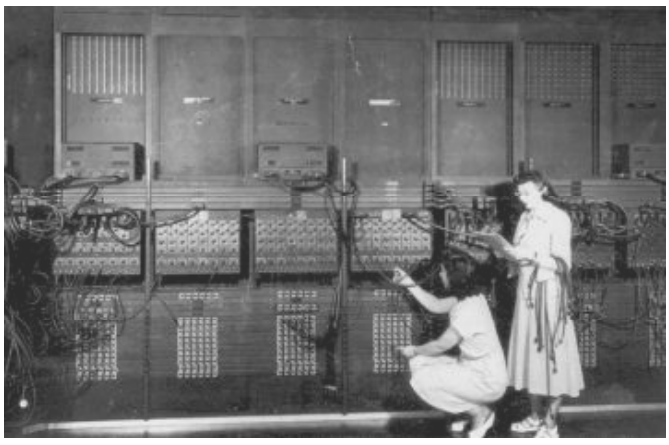
目 次

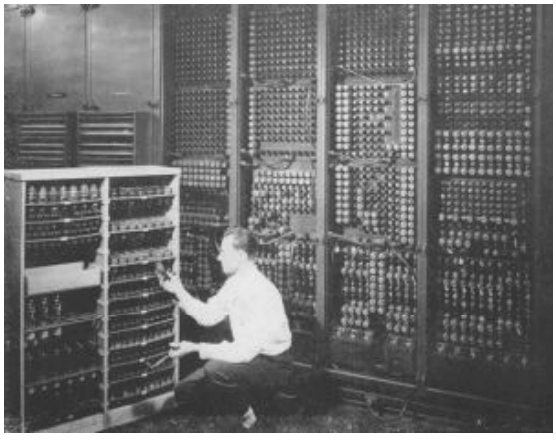
DEI

1. デバッグの始まり

コンピュータの出現

- **ENIAC** (Electronic Numerical Integrator and Computer)
- 18,800本もの真空管を使った大規模な電子計算機
- 床面積は 100m^2、重量 30トン、消費電力は 150KW
- 床面積60畳、車約20台分の重さの、1KW のストーブ 150台分





プログラミングの出現

DEI

- **1936 : Alan Turing**
  万能チューリング機械の論文を発表。"計算の理論"の基礎を築く
- **1945 : John von Neuman**
  プログラムとデータを統一的に内部メモリに表現することを提案
- **1946 : 真空管式計算機 ENIACを完成**

＜最初のプログラム＞

- **簡単なデモのためのプログラミング**
- **当時のコーディングは？**
  - ***コード（電線）を使った***

しかし、動作が変?!

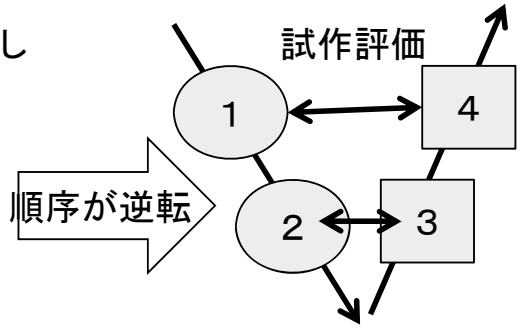
人類、SWのデバッグに遭遇

デバッグには2種類あった

DEI

1. プログラムの設計
2. コーディング : 電線の接続
3. 作業のデバッグ : 電線の接続確認
   （正しいものとの比較）
4. 設計のデバッグ : 設計の手直し
   （設計多重）

従来は、試作設計後に試作機を作って、評価し
製品設計を行うライフサイクル



表の仕事 Vs. 裏の仕事

DEI

- プログラミングは価値を生む表の仕事
- しかし、それを支えるデバッグが裏にある
- コンピュータの出現と共にデバッグが顕在化

- 同時に
- コンピュータの設計、製造は表の仕事
- 毎日故障する真空管の交換：保守の仕事が裏にある

メモ

DEI

テスト、デバッグのトレンド

2. 私のデバッグ遍歴

デバッグ遍歴 ＜前期＞

DEI

■ 1960年代後半 学生 趣味 アマチュア無線
- *意図した設計通りに動作しない無線機、テレビ送信機、アンテナ・・・!!*
- *危険なデバッグ:感電、コンデンサーの爆発、*

■ 1970年 修論で視覚系の応答シミュレーション
- *アナログコンピュータによるシミュレーション、後にFortranプログラム*
- *直流ドリフト、ペンの針がぶっとぶ、数値計算の精度に泣く・・・*

■ 1972年～:電気メーカ
- *周辺装置の制御開発、論理回と電気制御系が混在*
- *シンクロスコープ、ロジックレコーダ、連続運転でデバッグの連続*

・電子回路に対するデバッグ
・技術者個人のデバッグ力
・想定できなかった状態や要件に対する、設計の手直し

デバッグ遍歴　＜中期＞

DEI

■ 1980年代

- 汎用OSの開発、急激に接続端末の種類が増加した通信処理
- 仮想マシンを設計し、安定化
- 森口研究会：　森口繁一先生の指導を受ける
- 原因結果グラフの導入、CFDの開発
- 品質会計の開発、品質保証の立ち上げ

■ 1990年代

- 問題プロジェクト支援：性能問題、品質問題
- ISO/IEC　JTC1/SC7　WG9　インテグリティ水準

・ソフトウェア製品、システムのデバッグ
・複雑性に対するデバッグ技法

デバッグ遍歴　＜後期＞

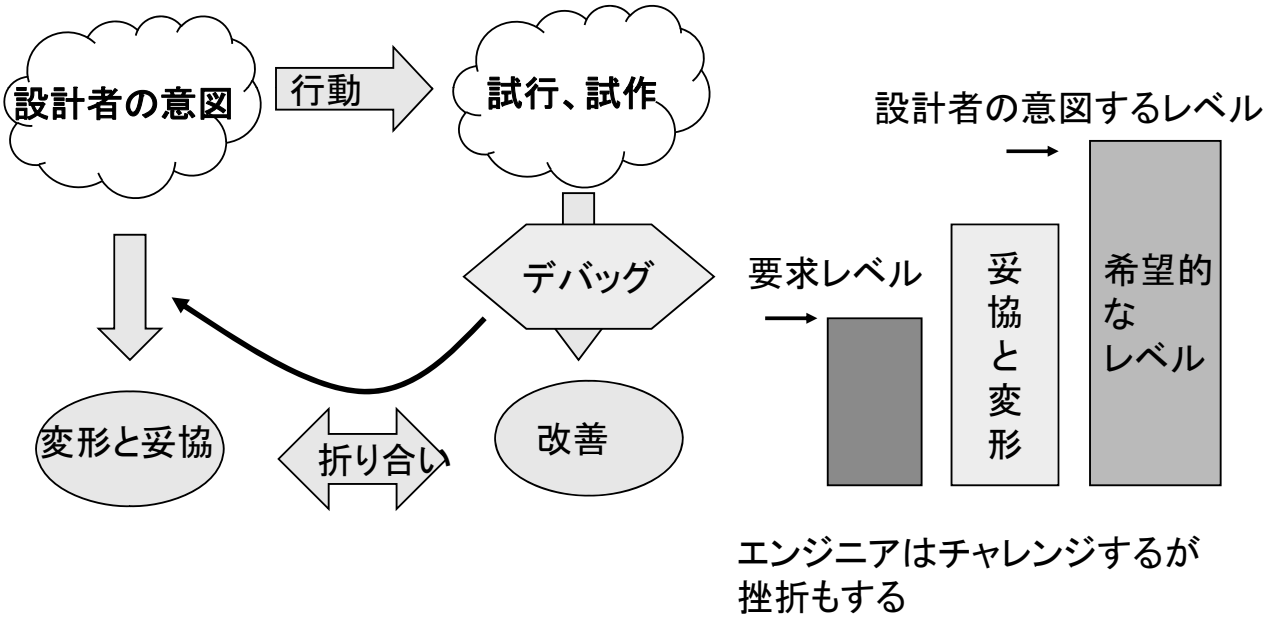
DEI

■ 2000年代

- ラインからスタッフ部門へ、モダンプロマネアカデミー設立（MPM）
- さらに早期退職、独立（2002年）
- ハーフリタイア：仕事のペースを落として、長く働く戦略
- 基本は、人材育成（技術者、マネジャーからチームの育成）
- より多くの経験、現場を見る
- デバッグ、テストから、ソフトウェア工学、さらにソフトウェア社会工学へ

・人が係わる活動のデバッグ
・技法やプロセスに対するデバッグ
・想定できなかった状態や要件に対する、プロセスの手直し

■ 適切な解を求める試行錯誤の過程



メモ

3. テストのトレンドと課題

テストに係る活動

DEI

- テストは対象分野や組織によって多様であるが
- 上位レベルのテスト
- 下位レベルのテスト

テスト レベル	観点 計画	テストケース設計	環境と実行	品質評価	特徴
上位レベル のテスト	リスク	リスクベース	実環境	リスク評価	統合者 チーム活動
下位レベル のテスト	全網羅	仕様ベース 実装ベース	開発環境	信頼性評価	開発 個別活動

上位レベルのテスト：統合テスト、総合テスト、実機テストなど呼び名は多様
下位レベルのテスト：単体テスト、結合テスト

歴史的に流れを追うと

＜下位レベルのテスト：ソフトウェアのテスト＞

■ 1970年代～1990年前半

G.J. マイヤーズ：　ソフトウェア・テストの技法

ソフトウェアの複合/構造化設計

ボーリス バイザー：ソフトウェアテスト技法

＜上位レベルのテスト：情報システムのテスト＞

■ 1990年後半～

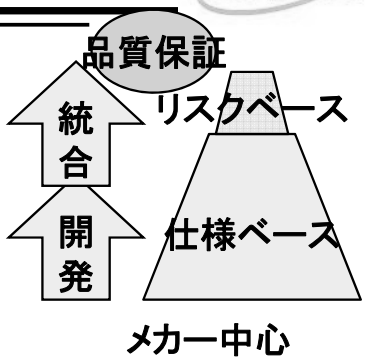
Cem Kaner 他　：基本から学ぶソフトウェアテスト

Rex Black：　基本から学ぶテストプロセス管理

産業界では

■ 1990年前半まで

- *主流：　テストは、**仕様ベース**で行うのが当たり前*
- *テストの主体は開発者*
- *ソフトウェア・テストの技法　G.J. マイヤーズ*
- *ソフトウェアテスト技法　ボーリス バイザー*



■ 1990年後半から現代

- *主流：　テストは、**リスクベース**で行うのが当たり前*
- *大きな変化：　テストの主体が統合者や利用者*
- *運用の範囲で、十分な品質*

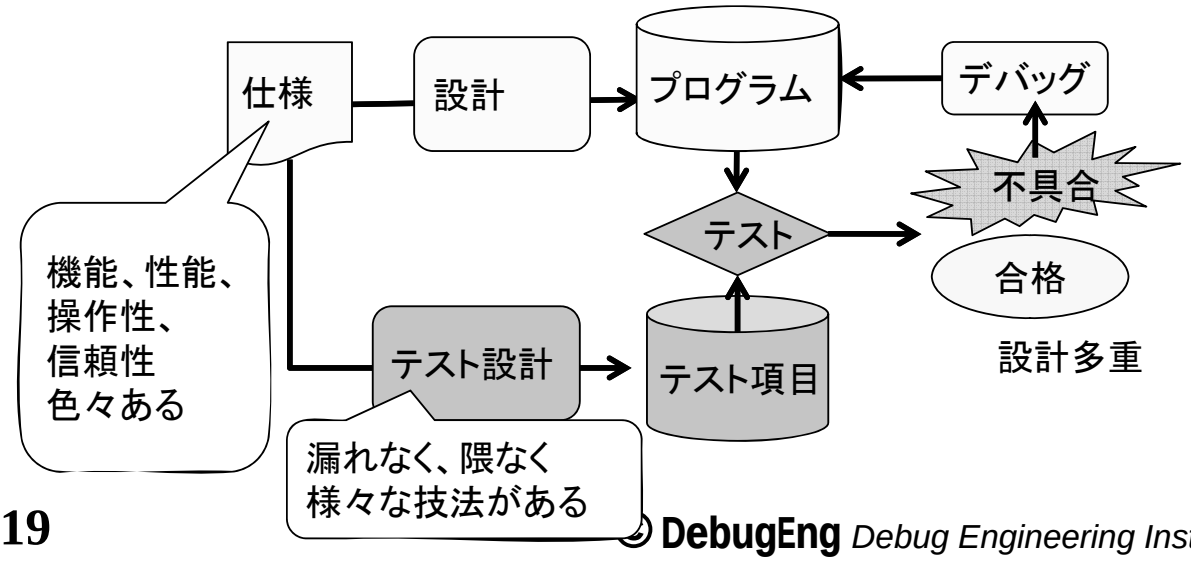
■ 両方共必要だが・・

分離した

下位ベースのテスト

DEI

- 開発者が自らの製品を磨くテスト
- 基本は、仕様ベースのテスト
- 下位の下位では、
  実装ベースのテスト（ホワイトボックステスト）



下位レベルのテストの課題

DEI

- 技法の進歩が停滞している
  - *テストに限らず、設計についても*
  - *基本は、機能（処理）、論理（条件）、順序性（状態やタイミング）を表現する*
  - *それらの組合せが生ずる、結合テストの技法が欠如*
- 技術者のスキル依存が大きい
  - *適切な技法が使われていない、使えない技術者*
  - *例えば、条件の組む合わせ・・・3つ以上の条件の組合せ表現がバラバラ*
  - *デシジョンテーブルを知らない（＝条件を組合せ表現ができない）*

上位レベルのテスト

DEI

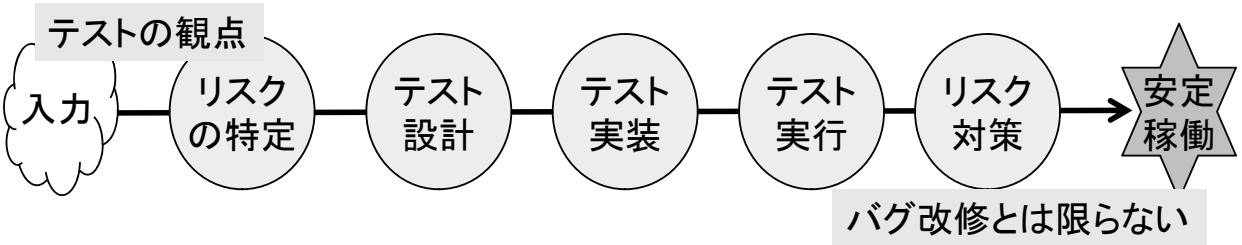
- 調達した製品でシステムを構成し、一部を開発/保守
- システム全体の詳細な仕様は解らない
- システムの利用者側に立ったテスト
- テストの目的：利用者のリスクを軽減する

- 背景
  - *90年後半から、メーカー主導が崩壊した*
  - *メーカーは、垂直分散で、ハード、OS，アプリのすべてを把握した*
  - *90年後半から、統合者にシステム評価の責任が生じた*
  - *仕様ベースのテストは不可能*

リスクベース・テストと課題

DEI

- リスクベース・テストの要素
  - *リスクを特定する*
  - *特定したリスクに対してテストを行う*
  - *発見した不具合に対して、リスク対策（回避やリスク低減）を行う*



- 課題は？
  - *何からリスクを特定するのか？　あぶない所が本当に予測可能か？*
  - *リスクからテストケースを設計できるのか？*

4. 近未来に向けて その1

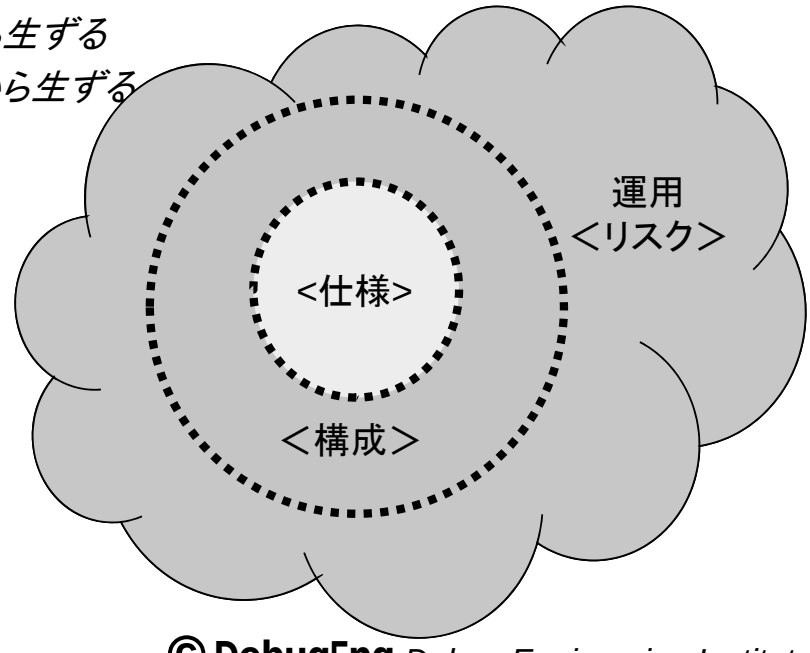
上位、下位の中間が必要

- 下位のテスト: 仕様ベース
- 上位のテスト: リスクベース
  - *リスクは運用から生じる*
  - *リスクはシステム構成から生ずる*
  - *リスクは下位テスト不足から生ずる*

リスクの分解が必要

- 階層に分離する

1. 仕様ベース
2. 構成ベース
3. リスクベース



テストの3層

DEI

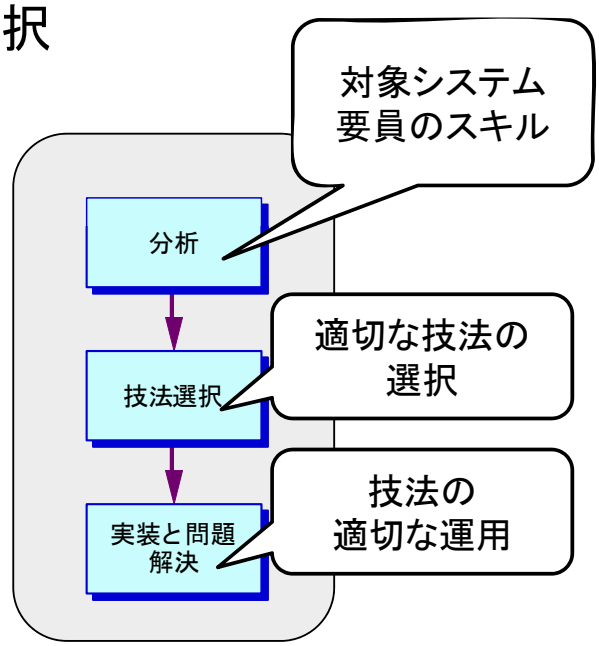
- 仕様ベースのテスト
  - ソフトウェアの設計、開発者としての責任
  - ソフトウェアの挙動を示す詳細な仕様と、
  - 製品であるソフトウェアの挙動が一致していることを確認する＜テスト＞
  - 不具合とは、仕様との不一致であり、改修する＜デバッグ＞
- 構成ベースのテスト
  - 統合者（SI）や装置製造者としての責任
  - 自社製、調達品を問わず構成したシステムや装置に対し、
  - 想定した運用の範囲*1 において問題のないことを確認する＜テスト＞
  - 不具合に対して、利用者の損失を低減する対策を講じる＜デバッグ＞
- リスクベースのテスト
  - 想定されるリスクを低減する目的で、テストを行う
  - 組込系では、シミュレーションは、実機や数学モデルを使う
  - エンプラ系では、実運用を基に行う

テストケースを作成する技法

DEI

- テストの観点が、3層化されると
- 次は、テストケースを作成する技法の出来栄え？
- その前に、適切な技法の選択
- 観点と状況から
- どんな技法が必要か？

1. 分析
2. 技法選択（技法は手段）
3. 適切な運用



5. 近未来に向けて その2

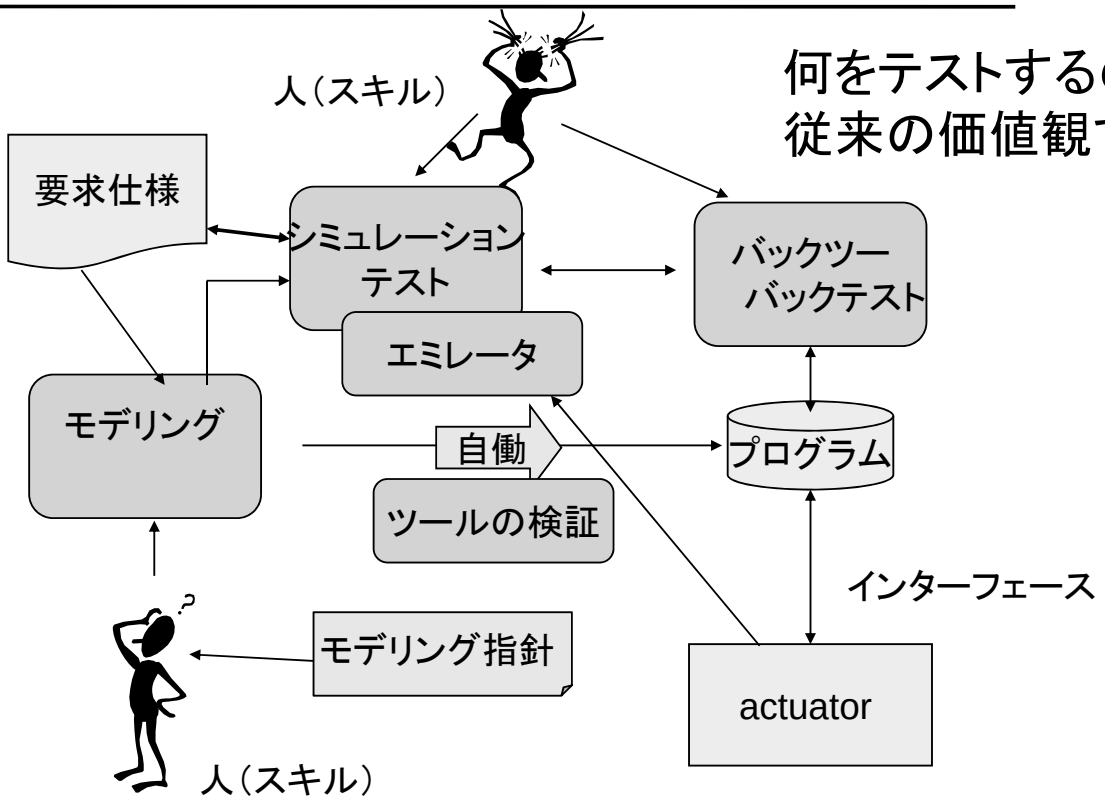
試作評価としてのデバッグ

- 永らくテストの目的は、当たり前品質の確保
  仕様通りに動作することを担保する・・・それが困難
- 製品（仕様）を高めるテスト（デバッグ）が欠如

■ 安全関連ソフトウェア（Safety-related Software ）を例に考えると2つの流れがある

1. プログラミング言語による開発
   1. *C言語、一部アセンブラー言語など*

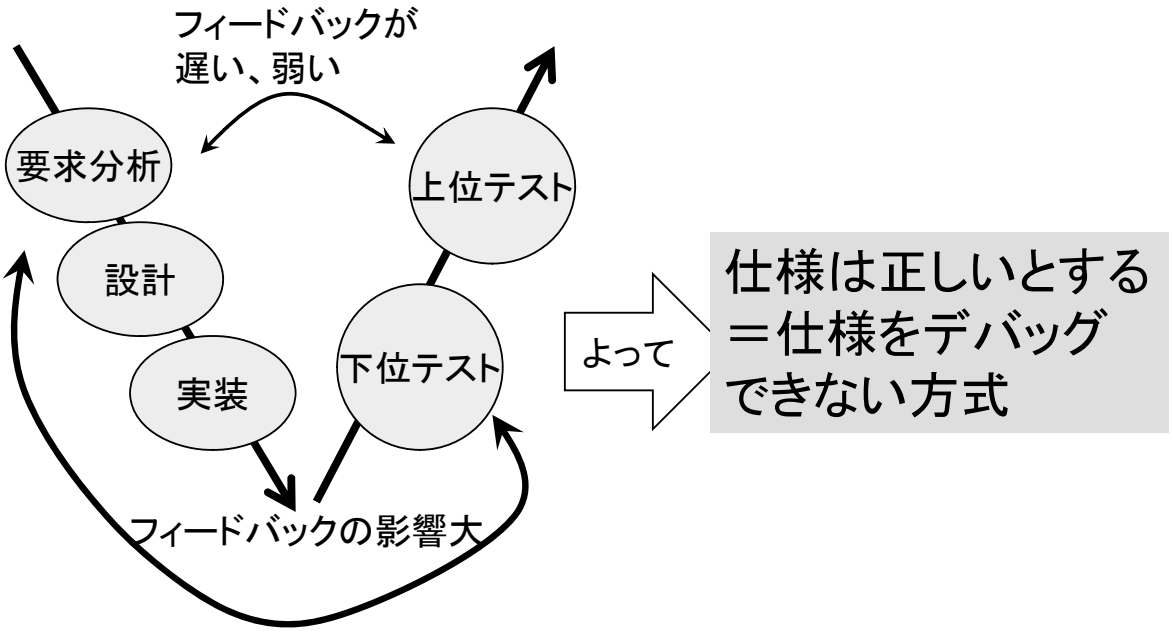
2. モデリング言語による開発 （モデルベース開発）
   - *さらに2系統*
   1. *VDM+など形式仕様と自働生成*
   2. *MATLAB/Simulink によるモデル、シミュレーション・テストと自働生成*
      MathWorks 社の製品



従来との違い

DEI

■ デバッグから試作評価の役割を奪った原因



モデリングとそのデバッグが必要

DEI

■ 上流工程におけるロジカルな定義

■ 下流工程の正確性だけでなく、自身のデバッグ手段

■ 複雑なものは、動作させなければ理解できない！！

■ これを支えるのは、技術者の新たなスキル

まとめ

DEI

- ４０年を振り返ってみると、デバッグの連続
- モノ作りの中の、小さな不具合を改善することから
- CFD技法や品質会計として抽象化
- 対象範囲を拡大
- まだまだデバッグが必要なのは、
- 複雑なプロセスに対するデバッグ
- 人のチームワークに対するデバッグ
- 人材育成に対するデバッグ
- など沢山あります

DEI

- ご静聴ありがとうございました