IPA Software Engineering Center
Information-technology Promotion Agency、Japan

# 導入へのガイダンス(B)

SEC形式手法人材育成作業部会(編)

# このモジュールについて



主に技術者としての立場から、形式手法を円滑に導入するために考慮すべき事項について理解し、形式手法導入の計画立案を開始できるようになるためのモジュール

- 事前知識・経験
  - 形式手法の有用性を理解した上で導入に前向きに検討している
  - 形式手法を具体的に適用する知識・スキルはない
- 学習目標
  - 形式手法の導入に必要な知識とスキル
  - 技術者として円滑な導入を実践するために必要な姿勢
- 主な学習項目
  - 目的の明確化
  - 適用レベルと前提
  - 代表的な「戒律」と事例
  - 開発プロセスと上流での仕様明確化の重要性

# 形式手法を適用する理由



[FME=http://www.fmeurope.org/?page-id=264]

- 開発プロセス全体の厳密性を高め品質を向上させる
- システムの完全無欠度や信頼性などの性質を向上させる
- 仕様の誤りを減少させる
- 要求定義の信頼性を改善させる
- 設計文書を改善し、設計の理解度を向上させる
- 保守や拡張のためのより堅固な土台を提供する
- 設計アーキテクチャの性質を精査する
- テストデータ選択のより合理的な基礎を提供する
- 設計と実現に誤りがないことを可能な限り保証する
- 特定の顧客や標準からの要求に適合させる

# 形式手法適用レベル



[http://en.wikipedia.org/wiki/Formal-methods]

- レベル0: 形式仕様記述
  - 数学的な記法を用いて厳密な仕様を記述し、証明や分析までは行わずに、この記述を基にしてプログラムを開発する
- レベル1: 形式的開発および検証
  - プログラムの性質を証明したり、詳細化により仕様からプログラムを作成する
- レベル2: 機械支援による証明
  - 定理証明器や証明支援器を用いてプログラムの性質を証明する

# 普及に向けて



- Practice & Experience
  - 共有 / 再利用
  - ガイドライン / Cook Book
- 先進的な研究よりは、むしろ基本的な知識と技術
  - ソフトウエアエ学
  - 形式手法
  - Technical Writing
  - 日常的なコミュニケーション
- 例えば、Ten Commandments of Formal Methods の実現

# Ten Commandments of Formal Methods



[J. P. Bowen & M. G. Hinchey: Ten Commandments of Formal Methods, IEEE Computer, Vol.28, No.4, pp.56–63, 1995]

#1: Thou shalt choose an appropriate notation.

#2: Thou shalt formalize but not over Formalize.

#3: Thou shalt estimate costs.

#4: Thou shalt have a formal methods guru on call.

#5: Thou shalt not abandon thy traditional development methods.

#6: Thou shalt document sufficiently.

#7: Thou shalt not compromise thy quality standards.

#8=Thou shalt not be dogmatic.

#9=Thou shalt test, test, and test again.

#10:Thou shalt reuse.

# 形式手法の十戒



#1: 汝、適切な表記法を選ぶべし
#2: 汝、形式化を行うべし、されど過ぐること勿れ
#3: 汝、コスト予測をすべし
#4: 汝、形式手法の師匠を身近に持つべし
#5: 汝、従来よりの開発法を棄つること勿れ
#6: 汝、十分に文書化すべし
#7: 汝、自らの品質標準を危うくすること勿れ
#8: 汝、独善となる勿れ
#9: 汝、テストすべし、またテストすべし、さらにテストすべし
#10: 汝、再利用すべし

# FeliCa の事例から学ぶ導入/適用方法



- 実践経験
  - 開発プロセス全体にうまく組入れる
- 継続と発展
  - 導入方法、適用方法
  - 要素技術開発
  - 管理方法
- FeliCa の成功事例 vs.十戒
  - いちいち当てはまる
  - 分析/評価
  - 導入ガイドラインとして取りまとめ中
    - 栗田太郎、中津川泰正、荒木啓二郎: 形式手法適用の実際と教訓—「形式手法の十戒」に照らし合わせて—, ソフトウェアエ学の基礎ワークショップ(FOSE2009)論文集, 近代科学社, 2009年11月.

# 一般的な企業の例

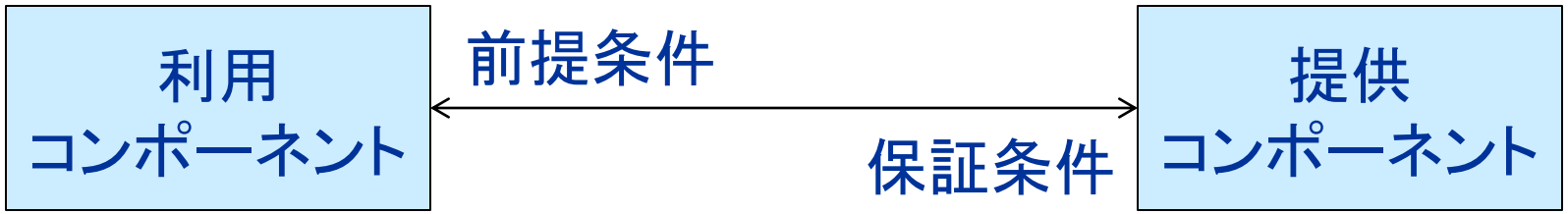


- 現場の技術者の高い能力
  - 形式仕様記述の修得は容易
- それぞれ自社の開発プロセスを所有
  - 典型的な(教科書的な)ソフトウエア開発プロセス
  - ソフトウェア工学の基本知識が不十分
- 形式手法と自分自身の開発業務との関わりに対する意識がない/誤認識
  - 高度に理論的
  - 完全、完璧、理想の世界

# 形式手法とアーキテクチャ

- アーキテクチャの特性と相互作用に対して形式手法を活用できる

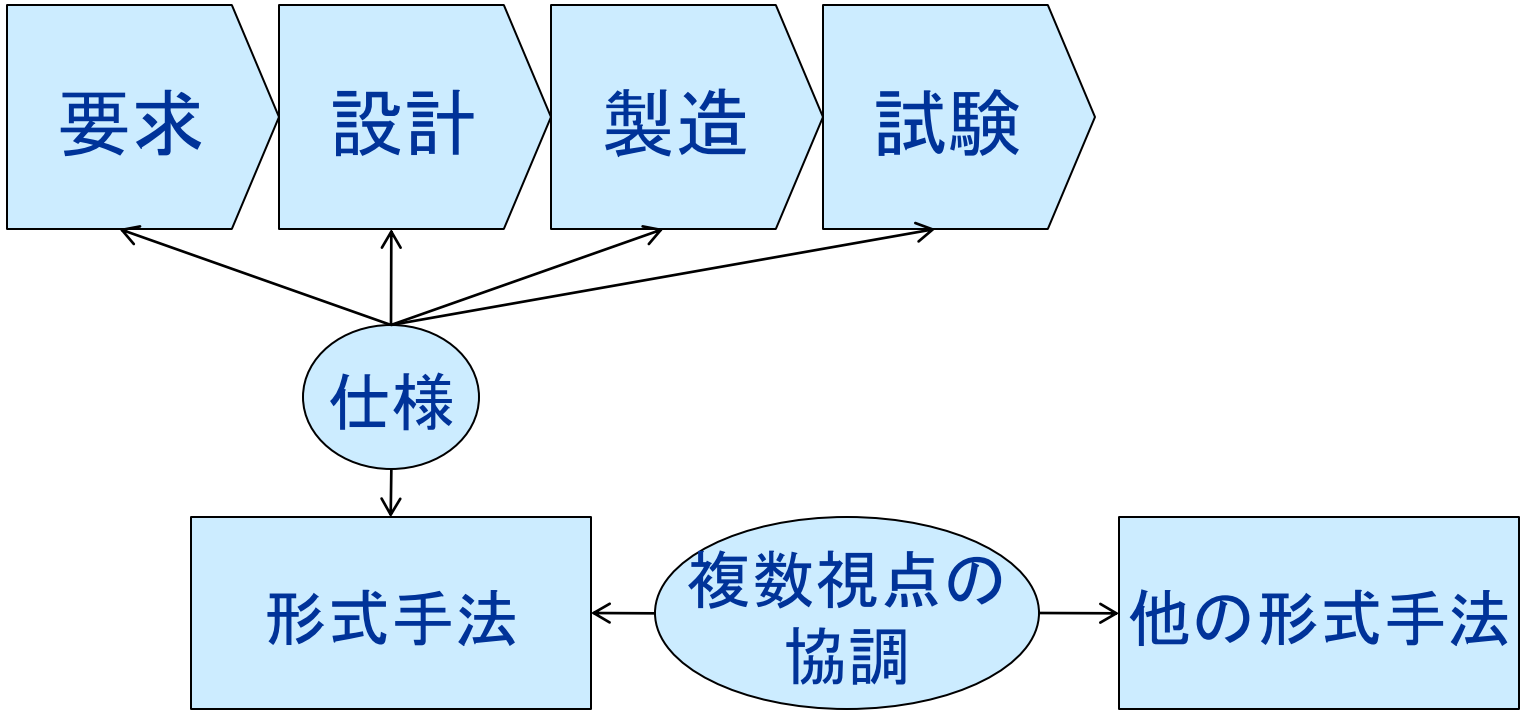
| 検証対象 | 形式手法の活用方法の例 |
|---|---|
| 特性 | 構造モデルで構成要素を定義<br>構成要素の操作と状態に基づいて、振舞いをモデル検査 |
| 相互作用 | 利用コンポーネントの前提条件に対して、提供コンポーネントの保証条件による充足性を検証 |



# 開発プロセスと形式手法



- どの開発工程に形式手法を適用するか？
- 適用工程と他の開発工程との関係をどうするか？
- 適用工程内で、形式手法と他の手法の関係をどうするか？
- 複数の形式手法を用いる場合、どのように組み合わせるか？



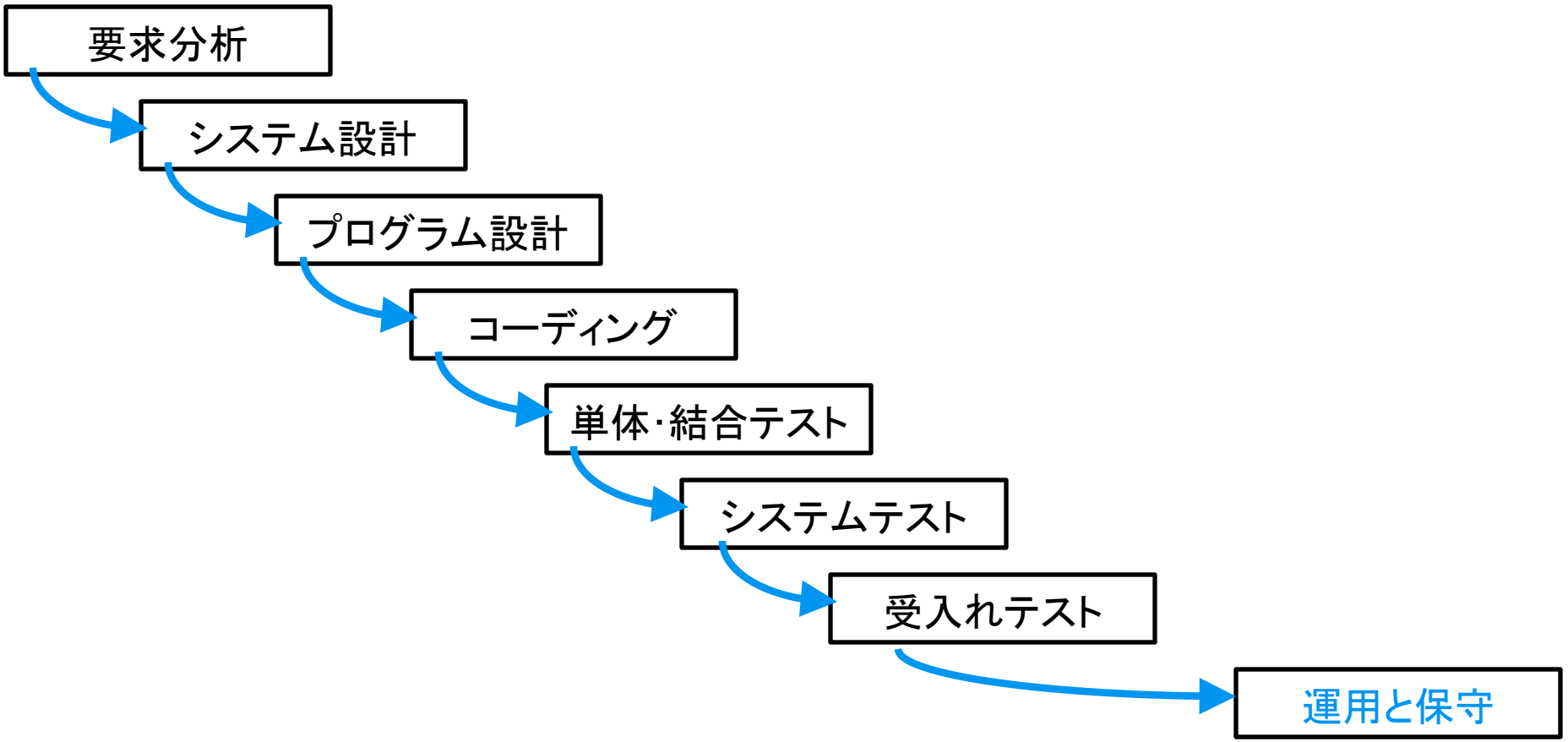
# ソフトウェア開発の過程(プロセス)



- 要求分析･定義: 利用者の意図や目的
- 仕様記述: システムが何をするか
- 設計: どのようにして実現するか
  - 概要設計
  - 詳細設計
- コーディング: プログラム作成
- テスト･デバッグ: 検査と修正
- 運用･保守: 利用と変更･拡張

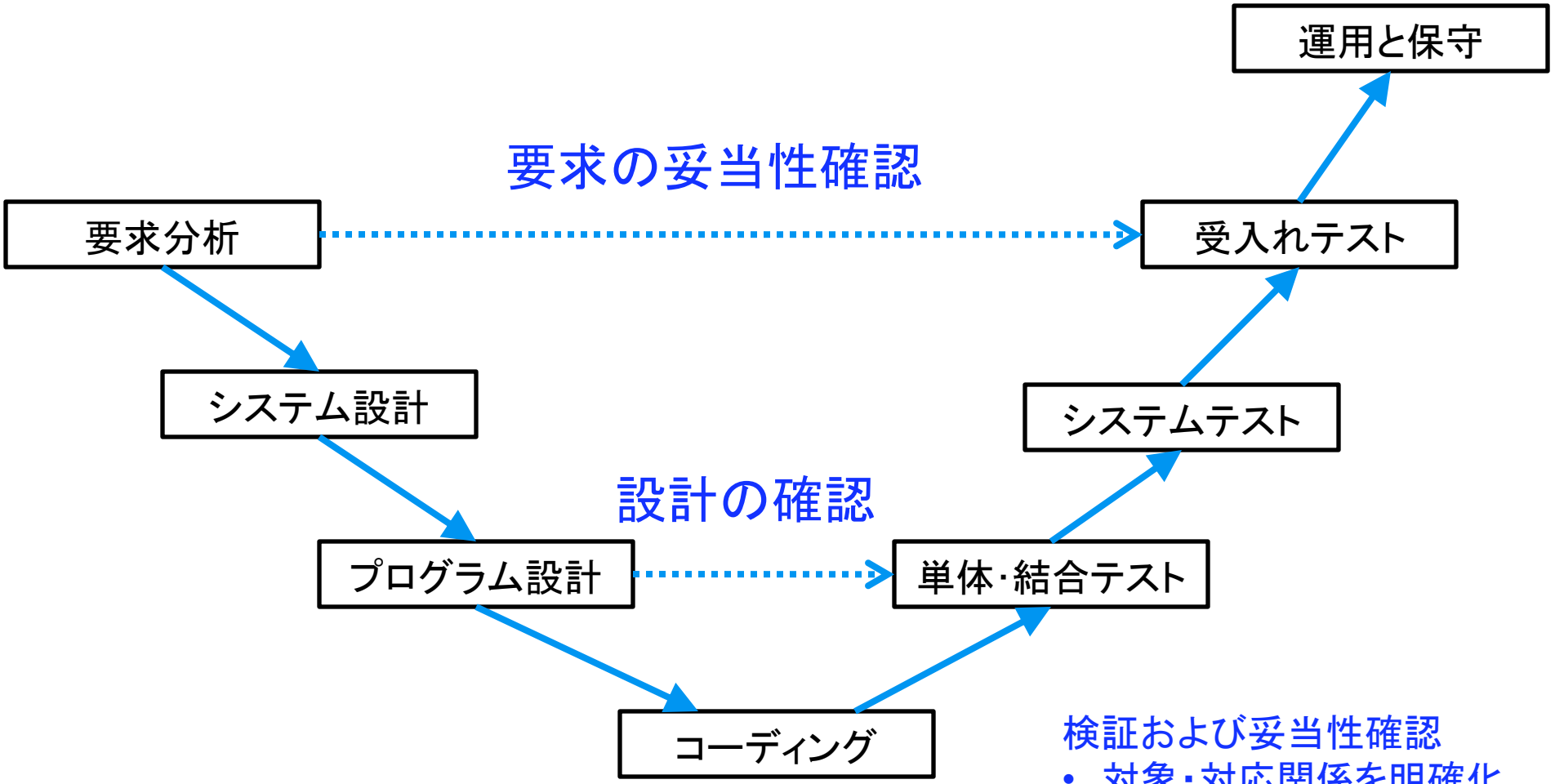
# 落水（ウォーターフオール）モデル





# V字モデル

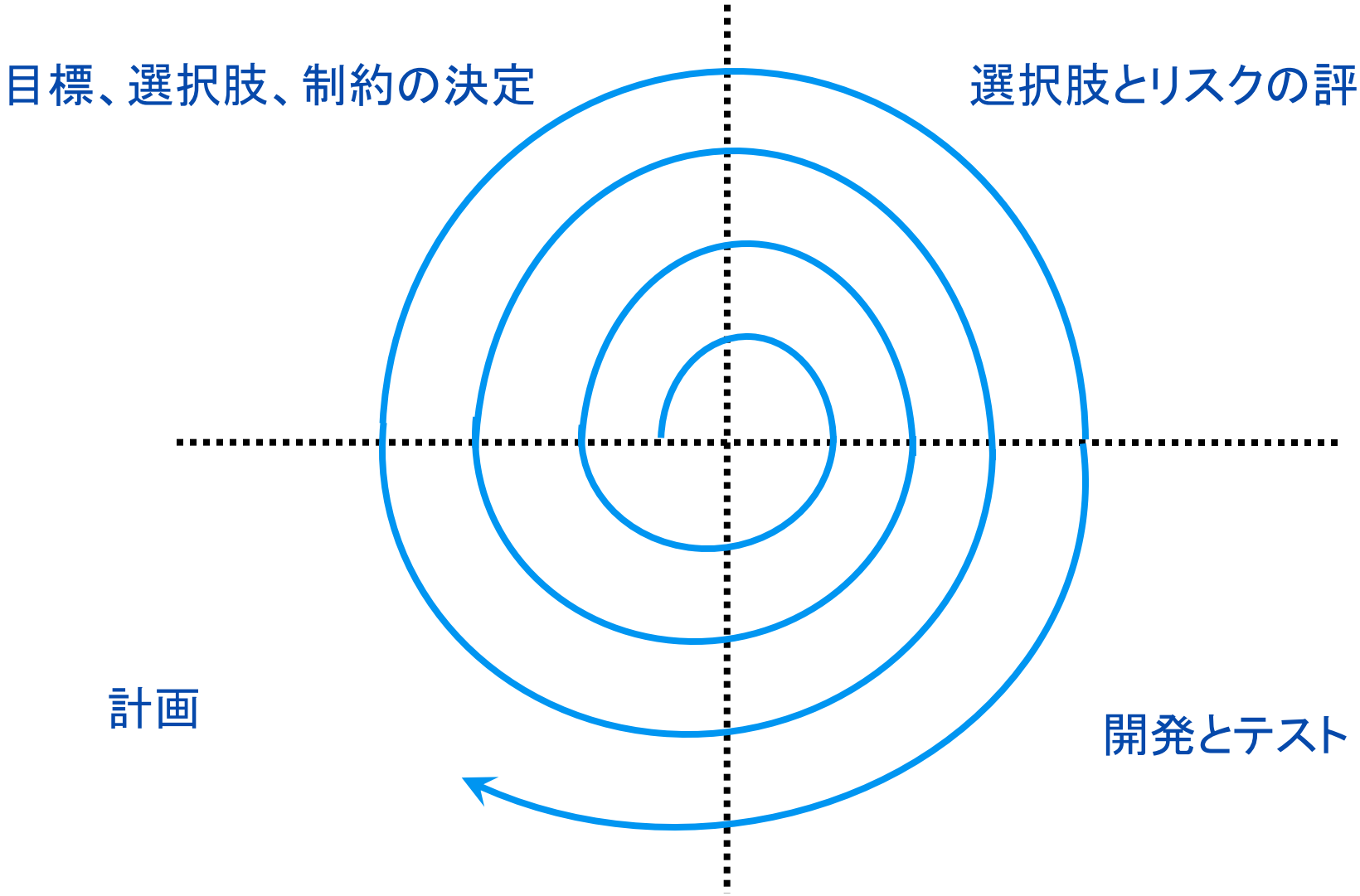





検証および妥当性確認
- 対象・対応関係を明確化
- 上流工程の段階から考慮

# スパイラルモデル





# 曖昧な仕様による問題



- ソフトウェアの開発現場では、曖昧な仕様に起因するトラブルが多い
- 下流工程の修正コストは、上流工程よりも大きくなる
- 正しくないプログラムの欠陥修正は非常に難しい、または不可能である

- 自然言語により曖昧かつ不完全な仕様を記述している
- 仕様記述と称して、手続的日本語プログラミングを行っている
- 業務を知らないコーダーが手続的にプログラミングを行っている
- 結果として、重大な欠陥を含み、保守性・再利用性のないソフトウェアが大量に生産されている

# 上流工程の重要性



- 失敗プロジェクトの原因
  - 不完全な要求(13.1%)
  - 利用者とのコミュニケーション不足(12.4%)
  - 資源の欠如(10.6%)
  - 非現実的な期待(9.9%)
  - 管理サポート不足(9.3%)
  - 要求と仕様の変更(8.7%)
  - 計画の欠如(8.1%)
  - もはや必要とされないシステム(7.5%)
- 修正コストの例
  - 要求：設計：コーディング：単体テスト：出荷後

    1 ： 5 ： 10 ： 20 ： 200
- 手戻り=全開発費の30～50%

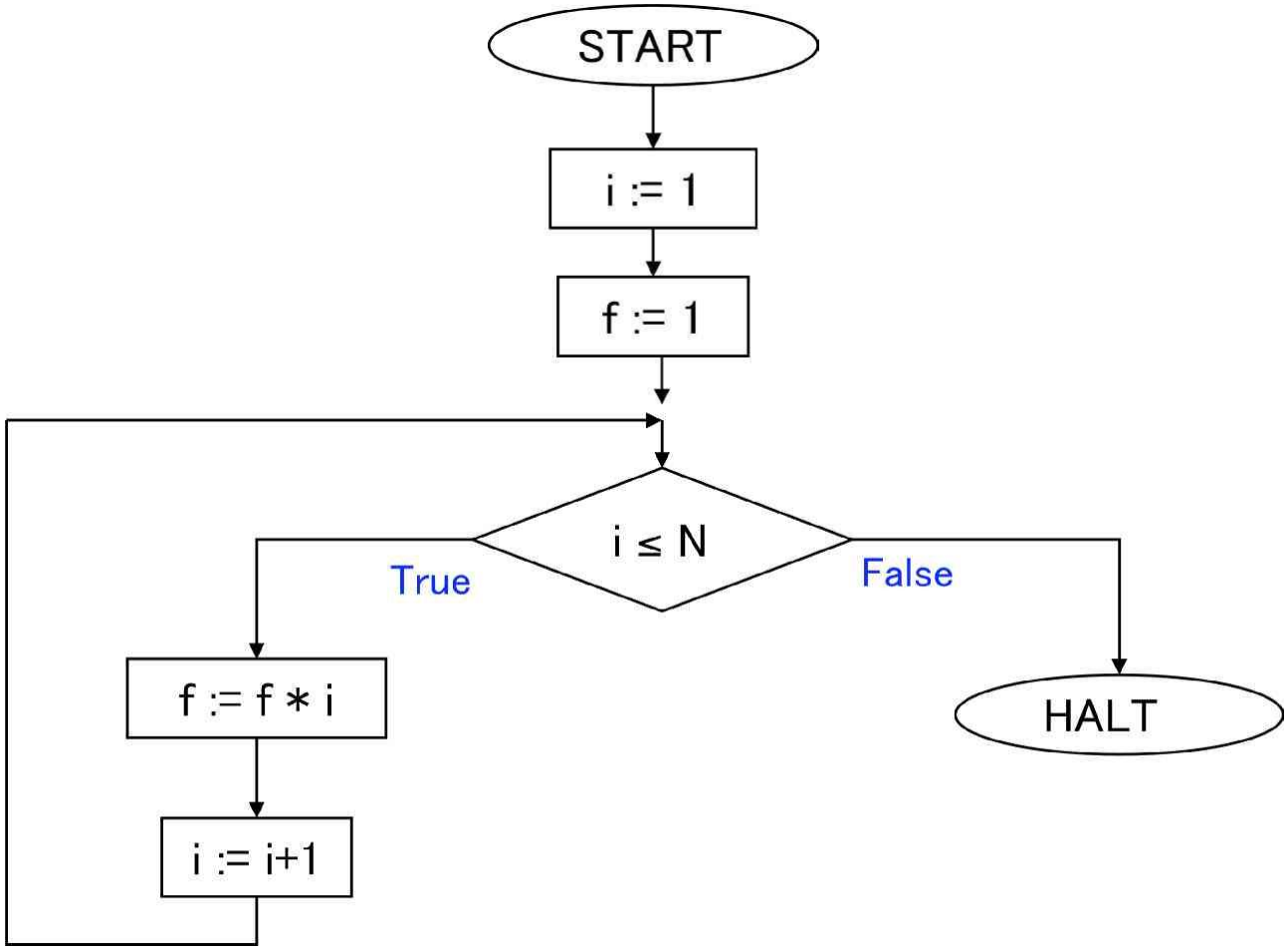
# 失敗の原因



- 前頁の失敗プロジェクトの原因のなかで、要求にかかわるものはどれですか?

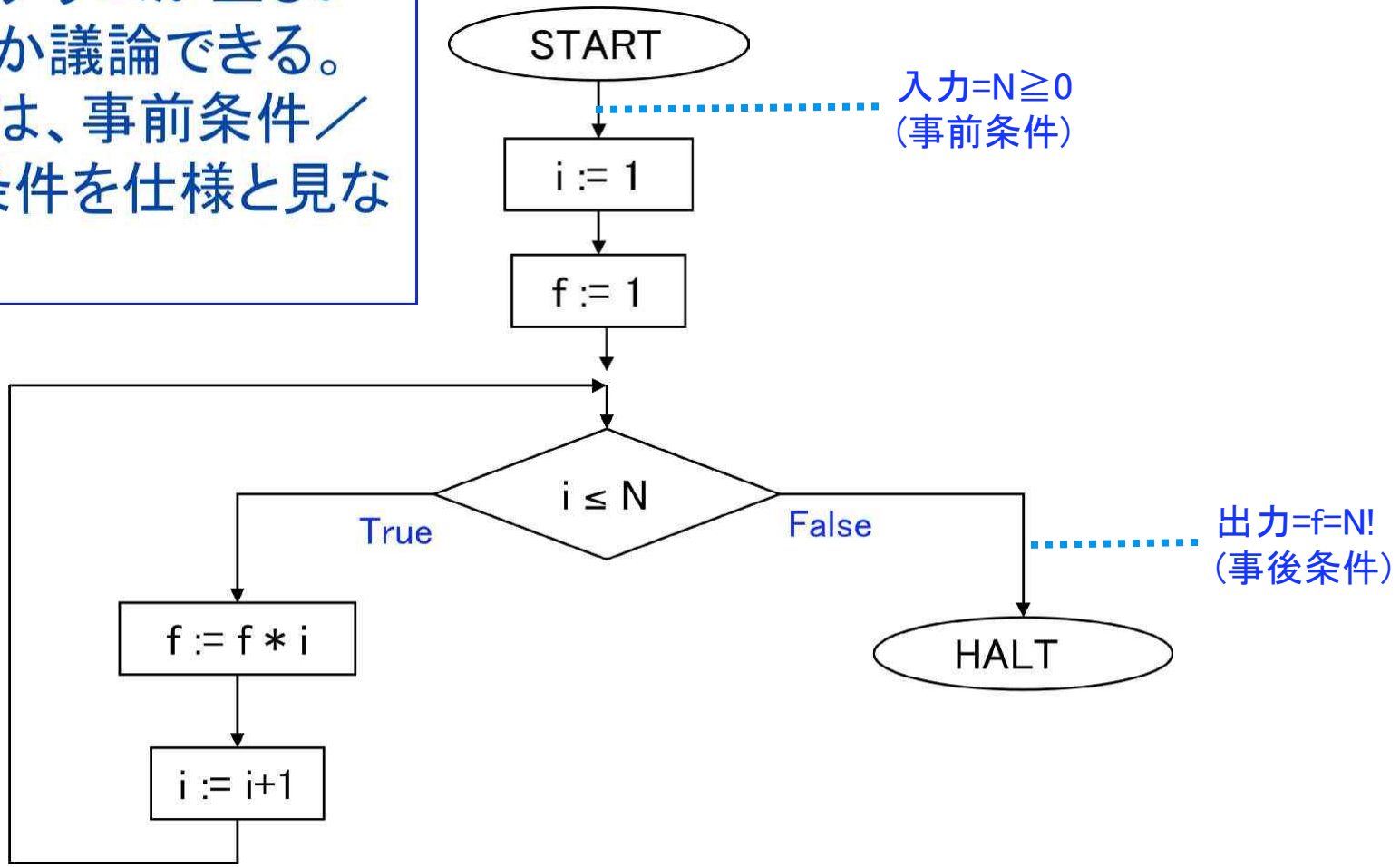
# 正しいプログラムとは？

以下のプログラムは、正しいプログラムでしょうか?



# 階乗の流れ図プログラム

仕様が与えられて初めてプログラムが正しいかどうか議論できる。ここでは、事前条件／事後条件を仕様と見なす。



# 階乗に関するホーアの論理

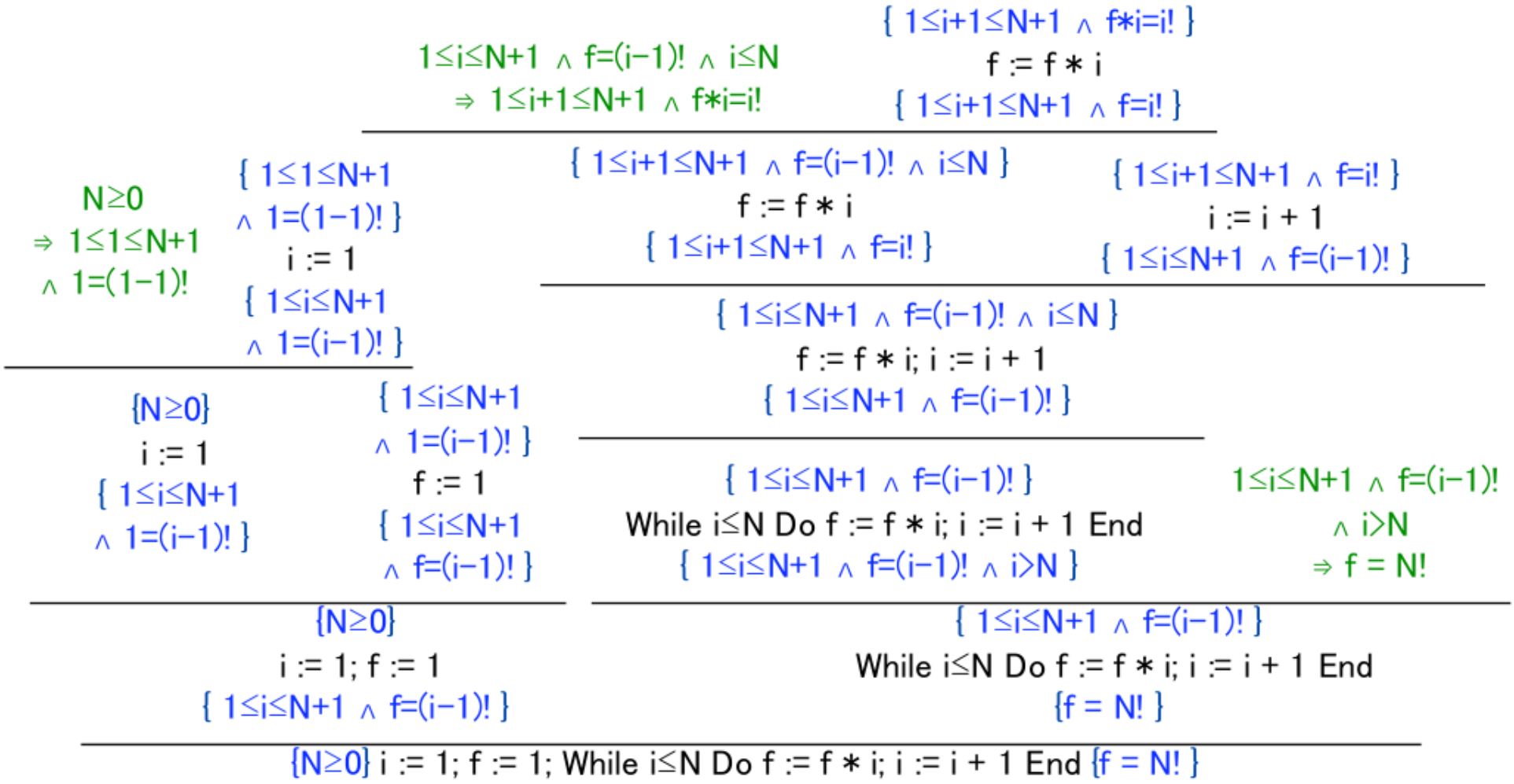


{N ≧ 0}

```
i := 1;
f := 1;
While i ≦ N
Do
   f := f * i;
   i := i + 1
End
```

{f = N!}

# 証明図(これが証明です!)



# サマリー



主に技術者としての立場から、形式手法を円滑に導入するために考慮すべき事項について理解し、形式手法導入の計画立案を開始できるようになることをめざし、主に以下を学習

- 目的の明確化
- 適用レベルと前提
- 代表的なガイドラインと事例
- 開発プロセスと上流での仕様明確化の重要性