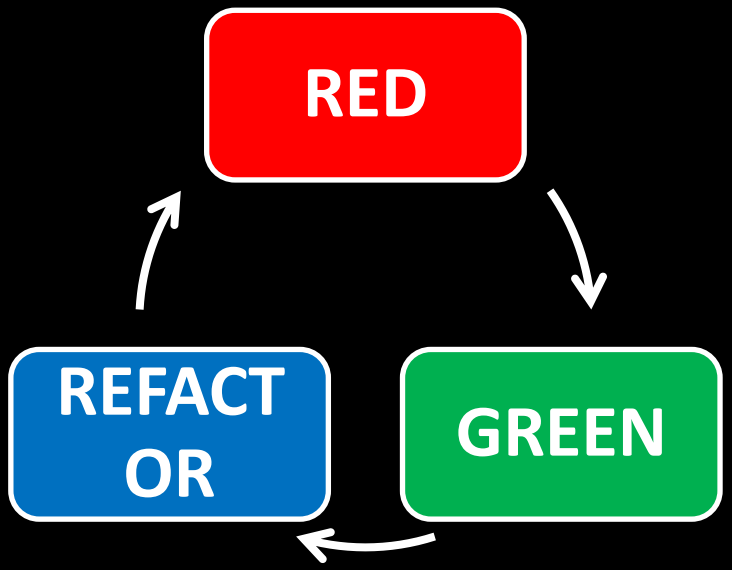
拡張型TDD

TDD研究会　井芹 洋輝

従来のTDDについて

テスト駆動開発（TDD）

- 次のサイクルを繰り返す開発手法
  1. 失敗するテストを書く（RED）
  2. テストをパスするコードを書く（GREEN）
  3. リファクタリングする（REFACTOR）

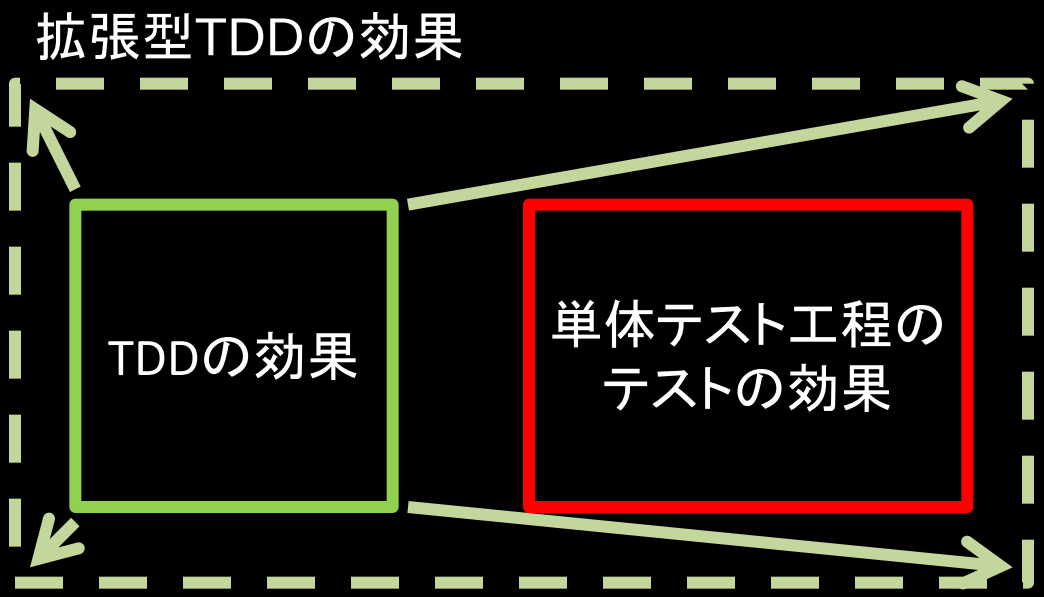


TDDライブ

課題分析

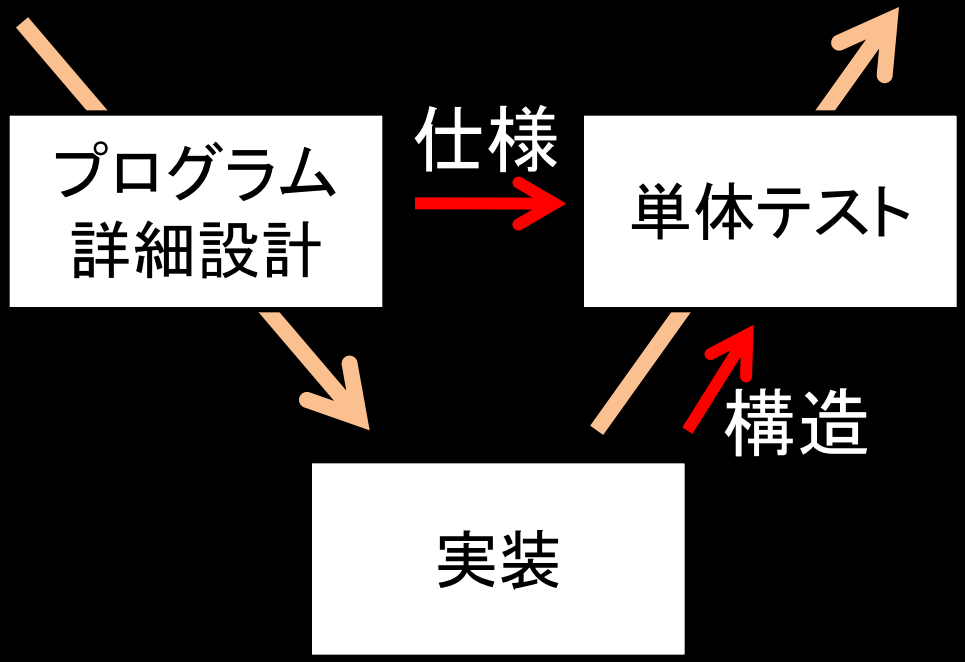
検討内容

- 今回の論点
  - 品質改善効果に優れたTDD運用ノウハウを明らかにする
  - TDDを拡張し、従来の単体テスト工程でのユニットテスト設計をTDDの活動の中に取り込む



単体テスト工程でのユニットテスト

- 実装工程後に行うコードベースのユニットテスト
- テスト対象を分析し、網羅的にプログラムの欠陥検出、仕様への適合性検証を行う



TDDのテスト
単体テスト工程のテスト

	TDDのテスト	単体テスト工程のテスト
目的	プログラミングのサポート	欠陥検出 仕様との適合性検証
フェーズ	プログラミング中継続的に	実装作業後
条件や網羅度	荒い条件や網羅度 「プログラミングの効率を落とさない」を優先	細かい条件や網羅度 「十分に欠陥や仕様未達を検出する」を優先

TDD拡張の目標

- 相乗効果
  - TDD→単体テスト工程:
    - テスタビリティの確保、テストコードの流用
  - 単体テスト工程→TDD
    - より品質の高いプログラムの実現

- 相乗効果を活かし、TDDの中で単体テスト工程でのテストを確保する

TDD拡張にあたっての課題

- 課題１：網羅的なテストを構築する
  - 網羅的なテストを確保
  - テストの品質を保証する仕組みを確保

- 課題２：テストの品質を維持する
  - 堅牢製が高くテスト対象の変更を許容するテストを確保

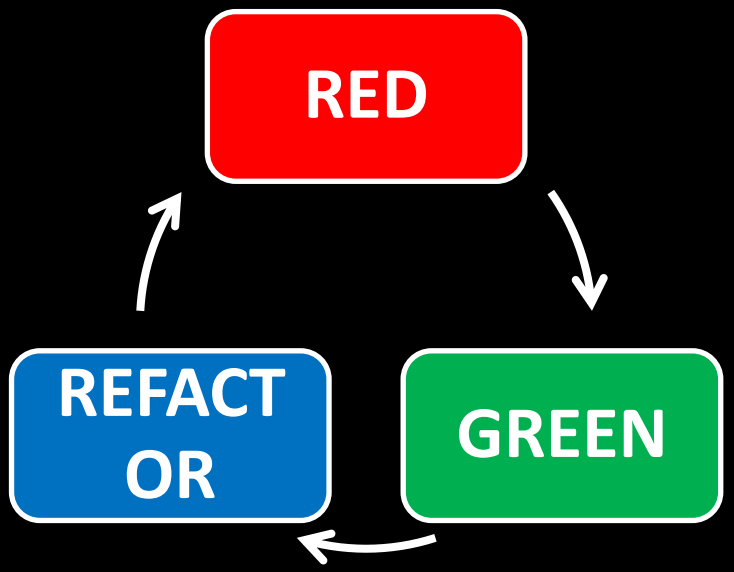
拡張型TDD

拡張型TDDとは

- TDDの運用ノウハウ
  - TDD＋網羅的なテストの構築＋テストの保守
  - 従来のTDDを否定するものでない

1. 拡張型TDDのコア
2. 製品コード/テストコードの構造
3. ユニットテストの評価・作りこみ

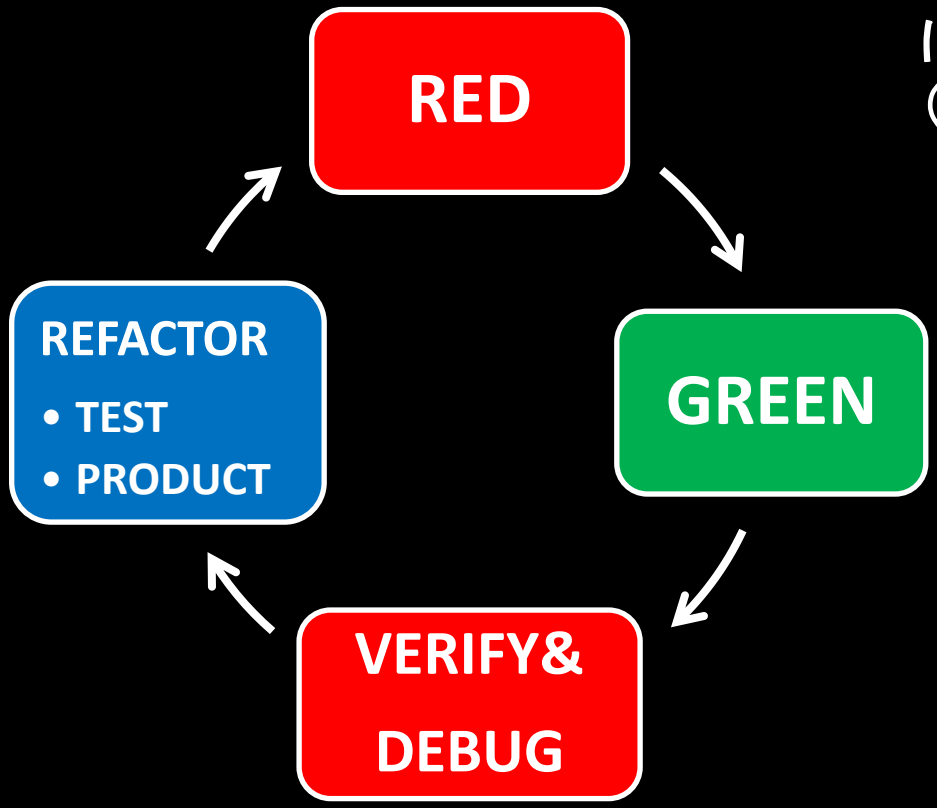
TDDのコア



Assertファースト
による追加・変更
（RED→GREEN）

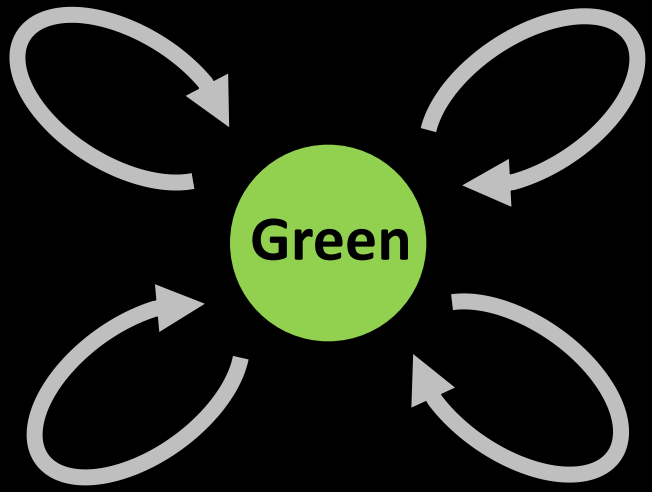
リファクタリング
（Refactor）

拡張型TDDのコア



Assertファースト
による追加・変更
（RED→GREEN）

テストコードの
設計改善
（REFACTOR[TEST]）



リファクタリング
(Refactor[PRODUCT])

テスト設計の洗練
(VERIFY&DEBUG)

Verify&Debug

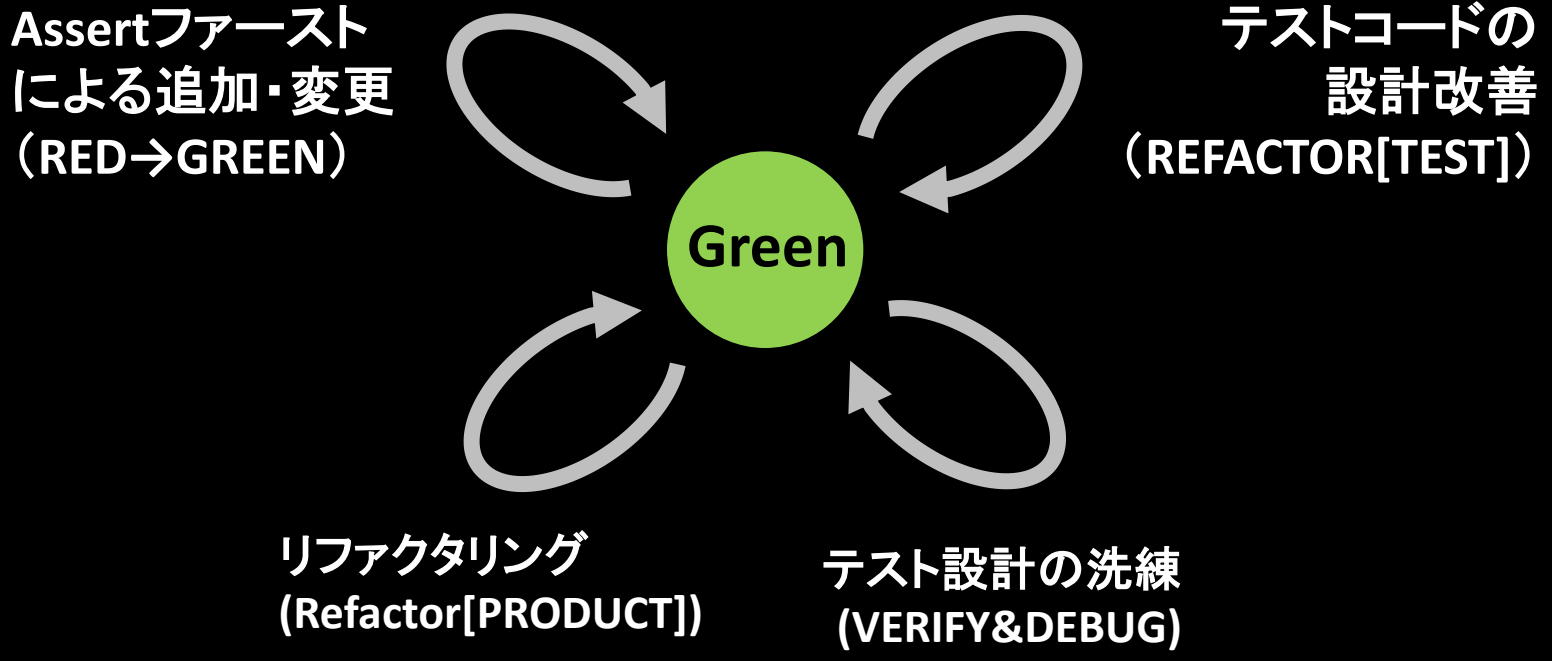
- テスト設計を作りこむ
  - テストケースを追加・洗練させる（Verify）
  - VerifyでREDになれば修正する（Debug)
- Verifyでの付属的活動
  - 冗長なテストの削除
  - 仕様の確保

Refactor[Product/Test]

- Refactor[Test]
  - テストの入力値・期待値を変更せず、テストコードの記述改善を行う
- コードの記述改善でテストの堅牢製・保守性を向上させる

まとめ

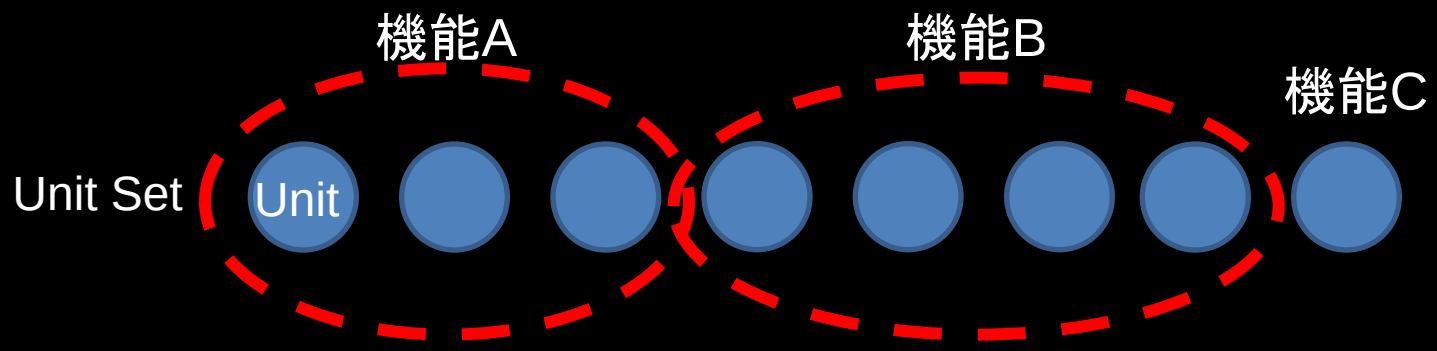
- 課題達成のための活動を継続的に行う
  - Verify&Debugで網羅的なテスト設計を確保
  - Refactor[Test]でテストの堅牢製を確保



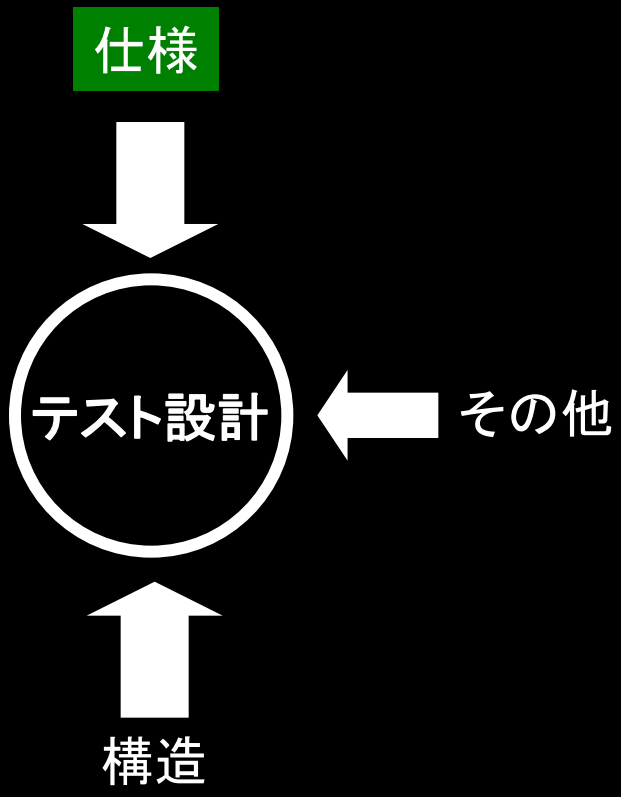
1. 拡張型TDDのコア
2. 製品コード/テストコードの構造
3. ユニットテストの評価・作りこみ

製品コードの構造

- **テスト設計のやりやすさ**を観点にユニット（Classや関数）を整理する
- 観点は2つ
- 観点にそぐわないユニットはグルーピングして粒度を調整する

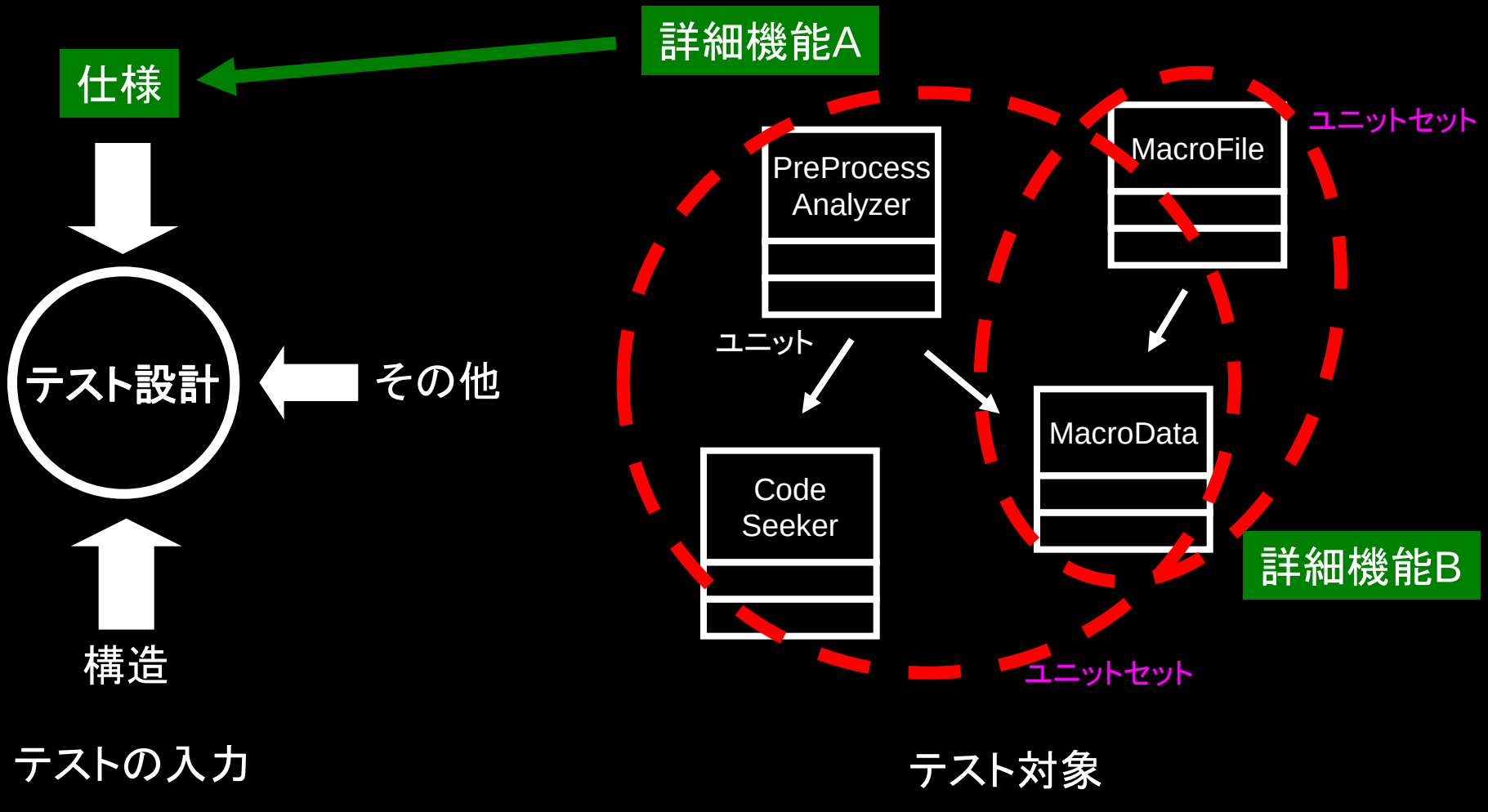


観点１：担当機能が具体的で、仕様ベースのテスト設計を一まとめにできる

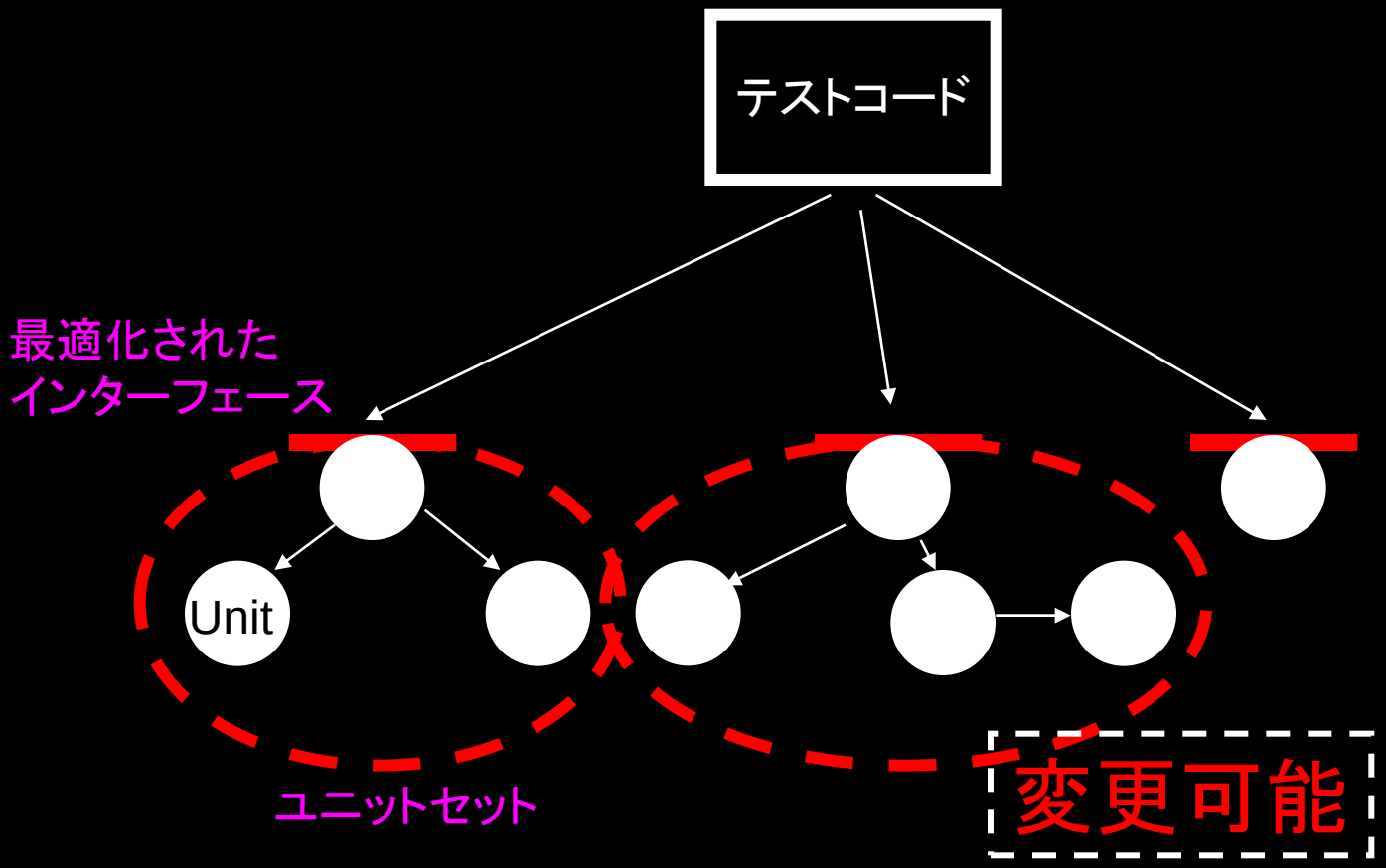


テストの入力

観点1：担当機能が具体的で、仕様ベースのテスト設計を一まとめにできる



観点２：テストが限られたインターフェースを通して全体にアクセスできる



テストコードの構造

- 目的でテストを2つに分類する

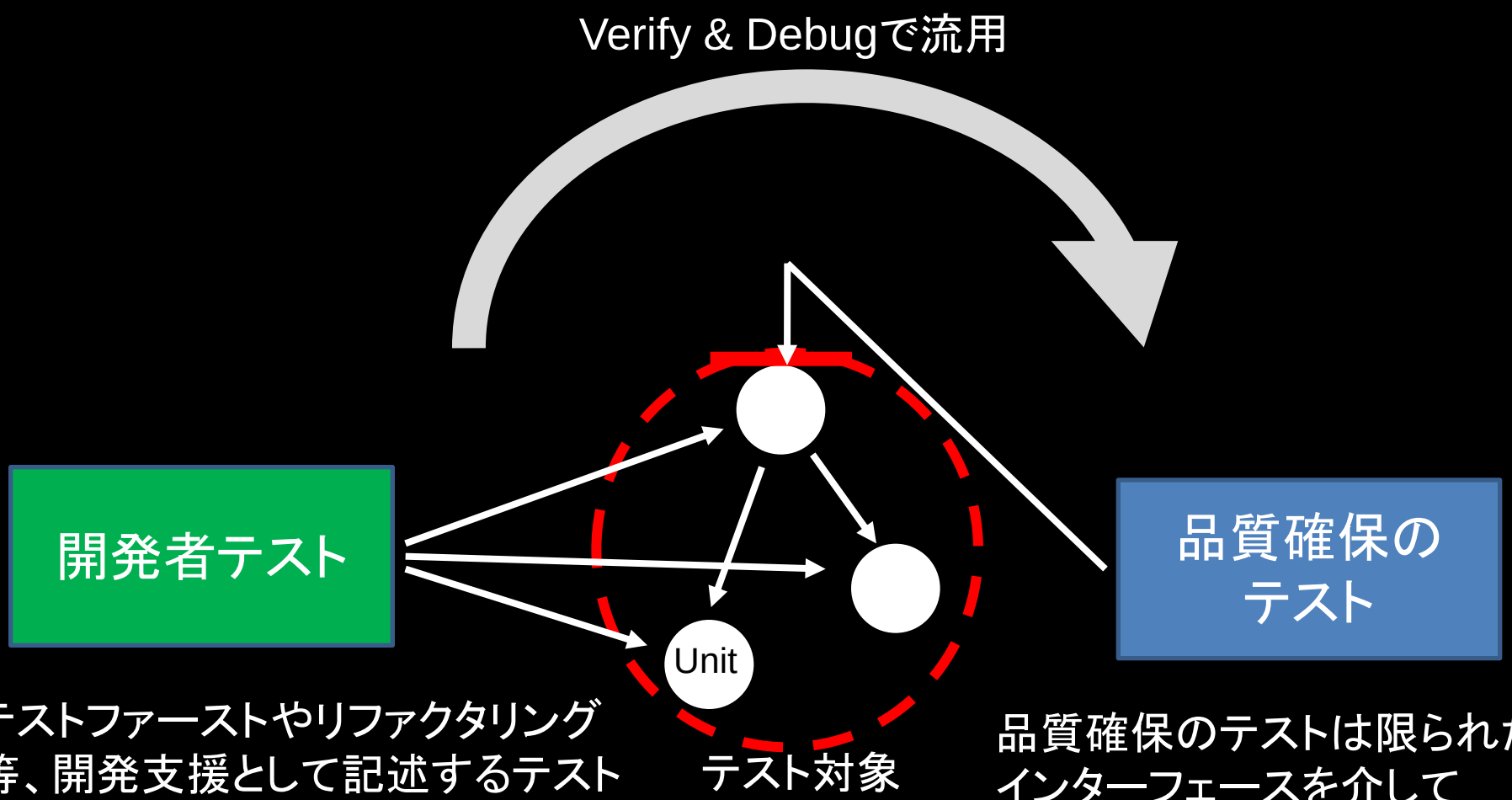
開発者テスト

- 開発作業(テストファースト/リファクタリング/欠陥局所化等)のために記述されたテスト

品質確保のテスト

- 網羅的にコードを検証するテスト
- 単体テスト工程のテストを指向

テストコードの構造

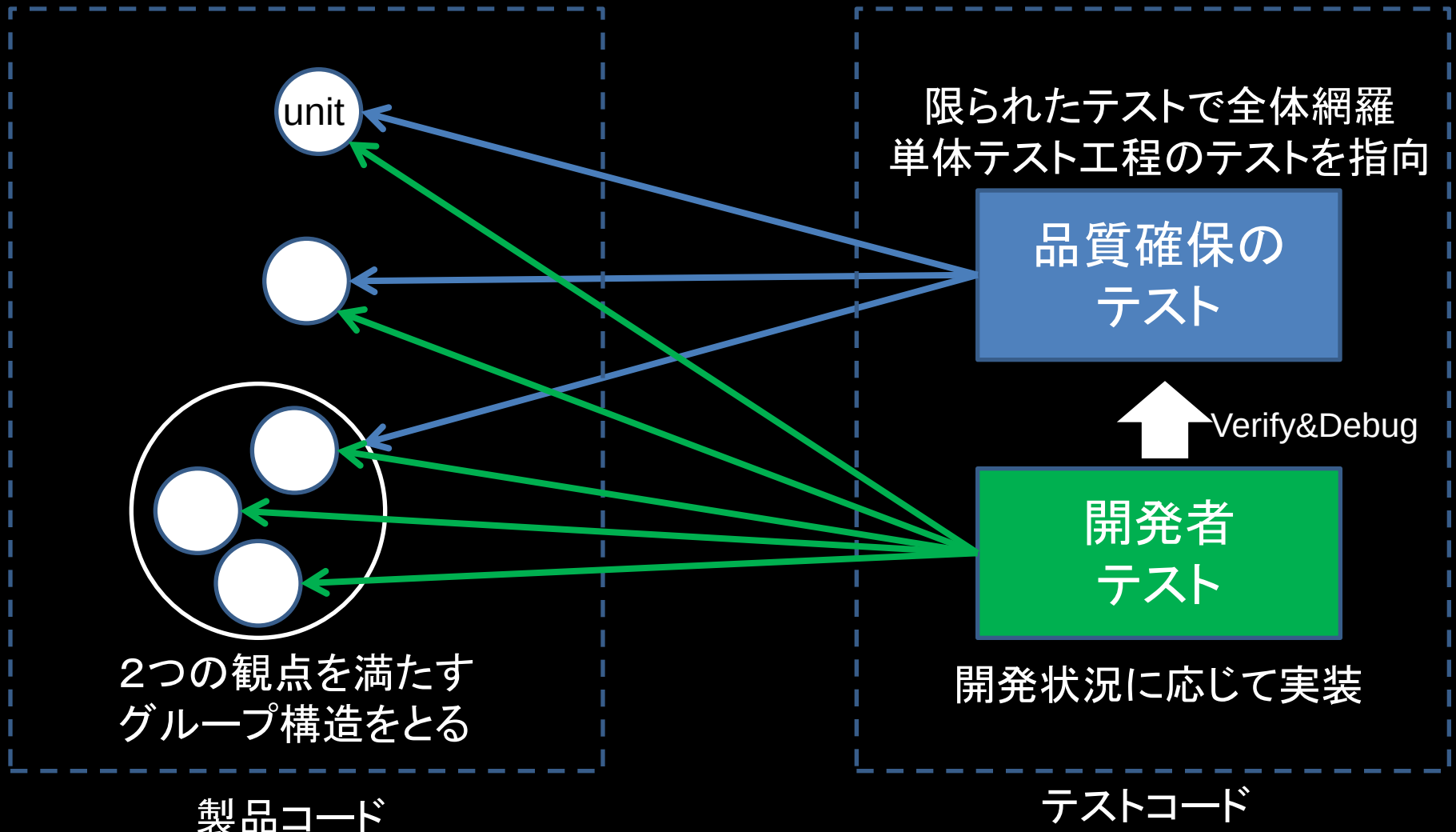


テストファーストやリファクタリング
等、開発支援として記述するテスト

テスト対象

品質確保のテストは限られた
インターフェースを介して
全体を網羅

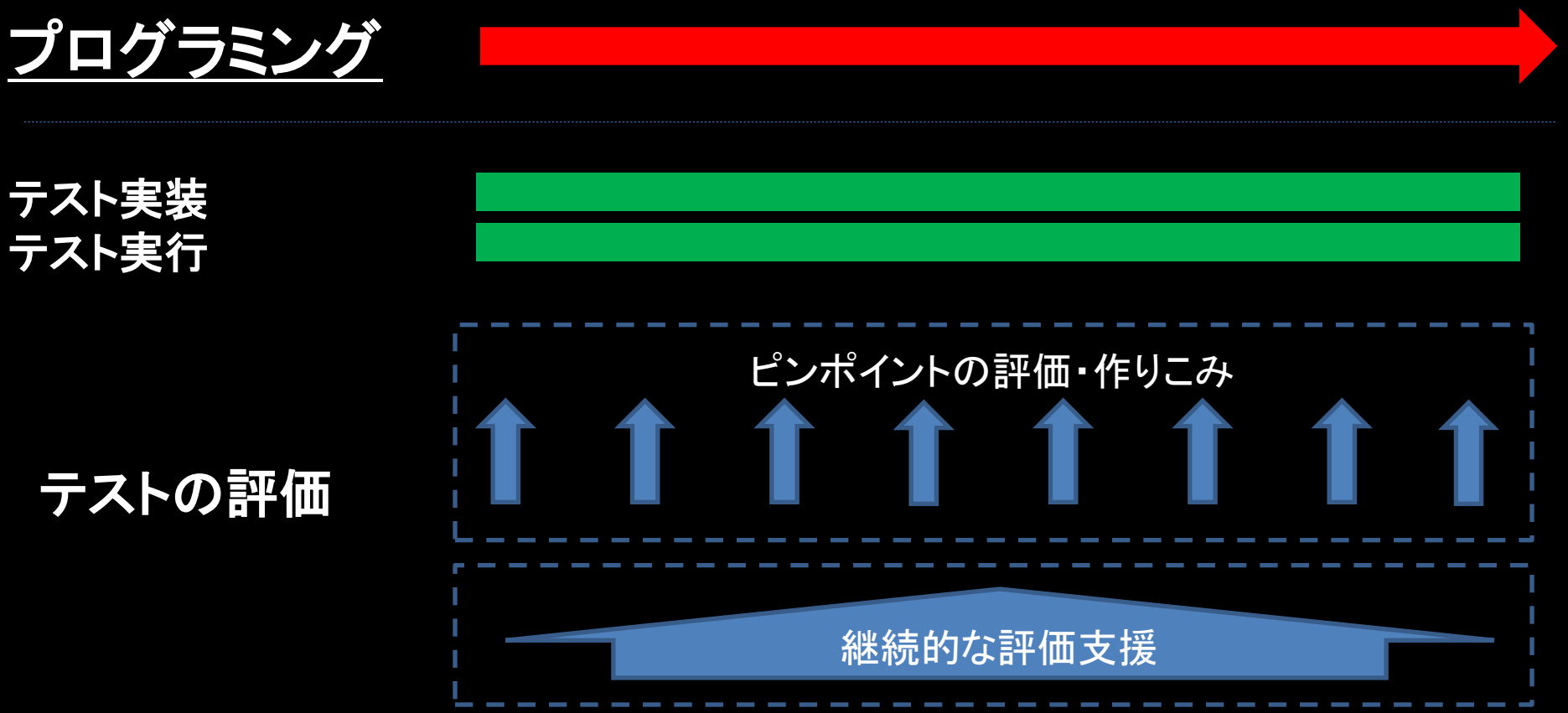
全体像


unit
限られたテストで全体網羅
単体テスト工程のテストを指向
品質確保の
テスト
Verify&Debug
開発者
テスト
2つの観点を満たす
グループ構造をとる
開発状況に応じて実装
製品コード
テストコード

1. 拡張型TDDのコア
2. 製品コード/テストコードの構造
3. ユニットテストの評価

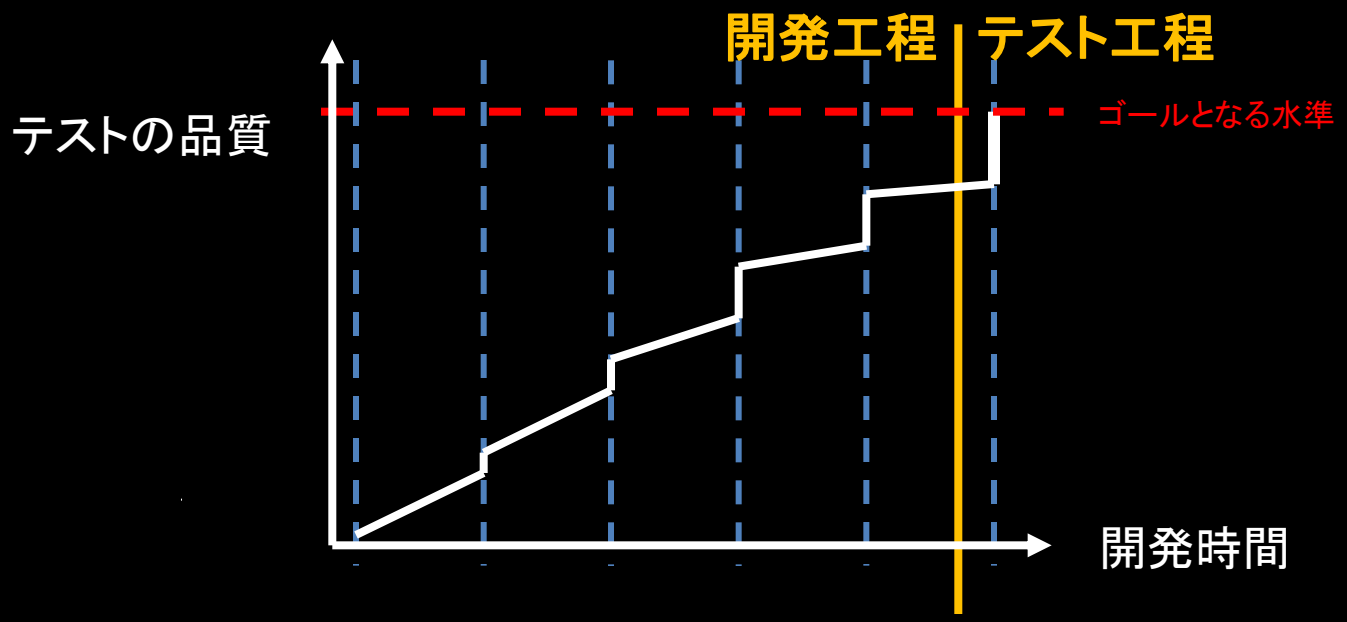
ユニットテストの評価

- 「ピンポイントの評価」「継続的な評価支援」で
  ユニットテストの品質を確保する



ピンポイントの評価

- 品質確保のテストはイテレーティブに評価・修正

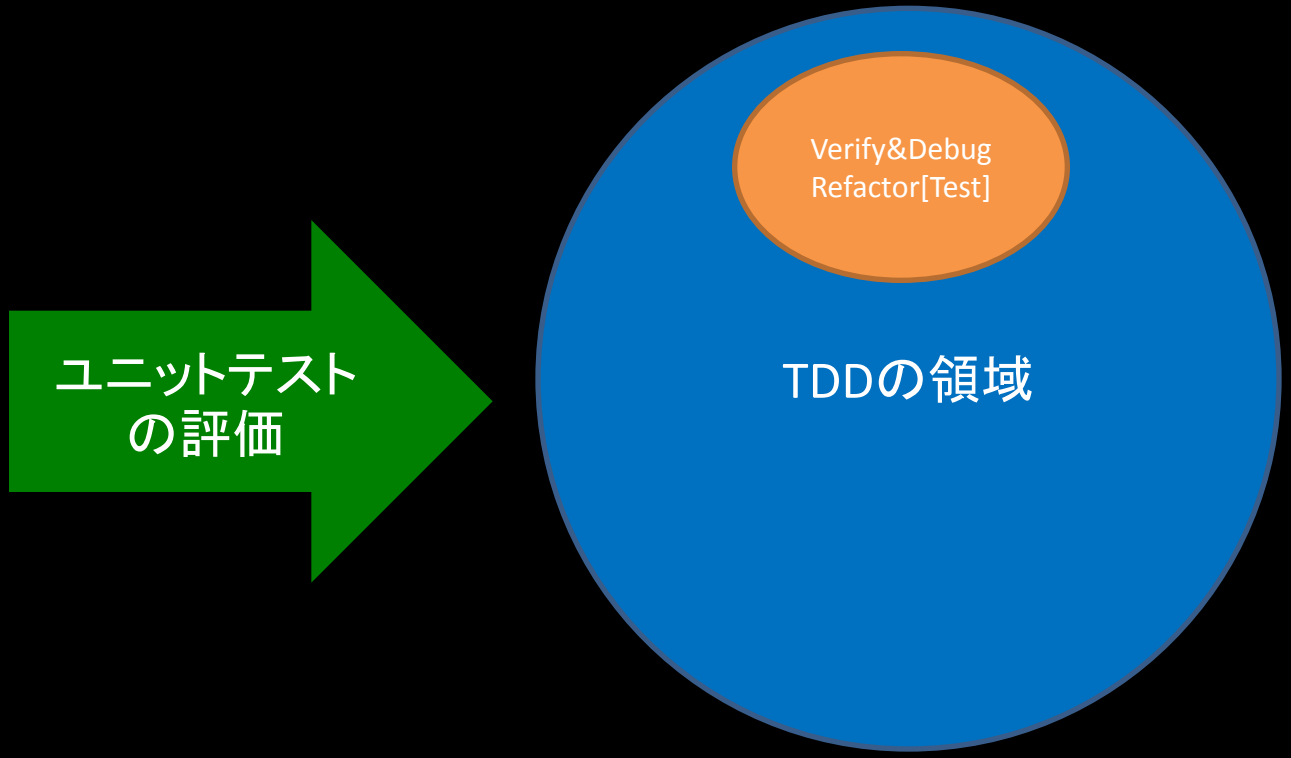


- 効果
  - 信頼できるテストを確保。後工程の負荷を最小化
  - テストは継続的に自動実行。後工程の手戻りを最小化

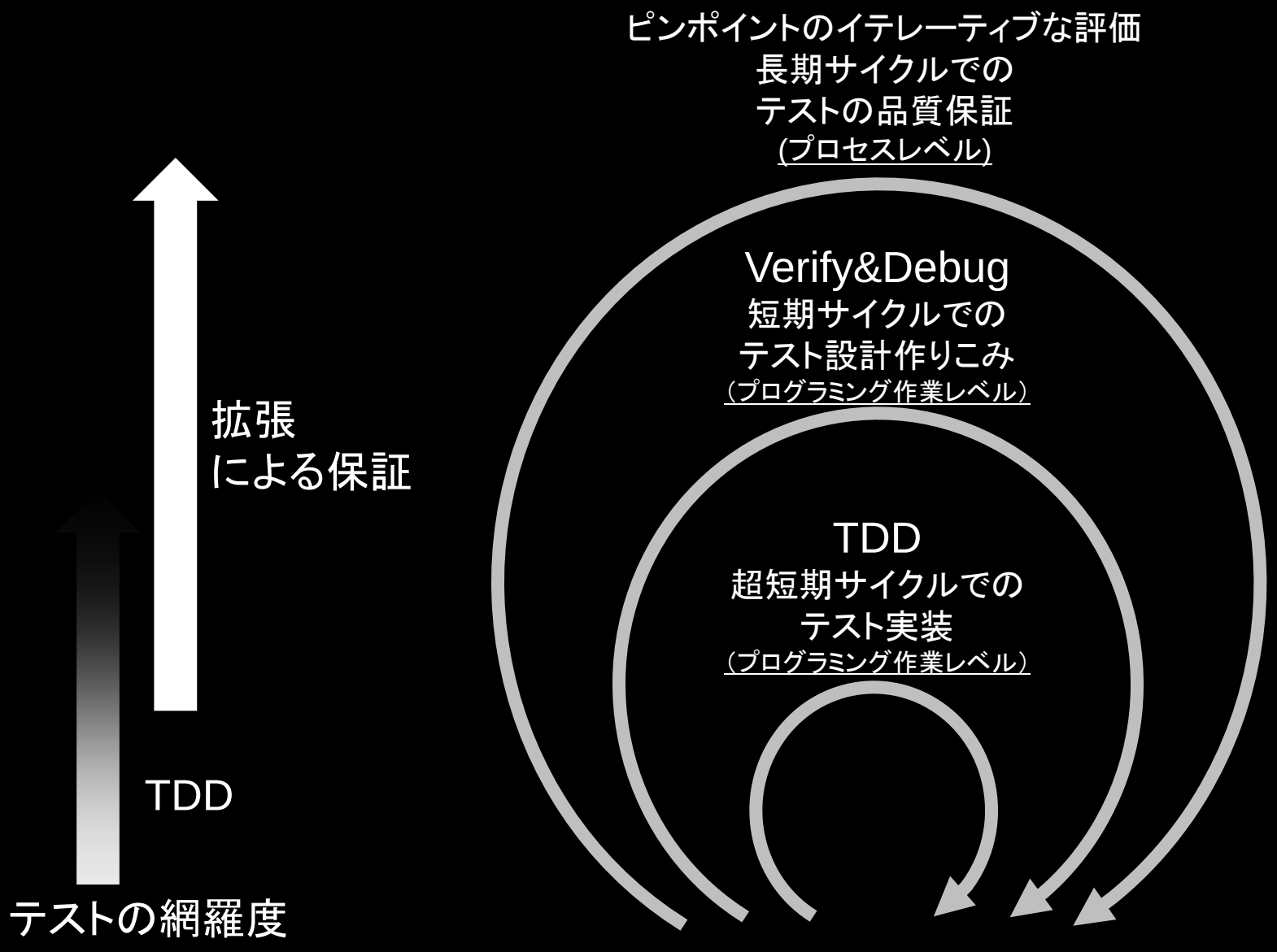
継続的な評価支援

- ユニットテストの継続的な評価を行い、ピンポイントの評価を容易にする体制を整備する
  - 推奨例：ペアプログラミング（★）
    - テストエンジニアとプログラマのペアプログラミングを推奨
    - チェックインの時点で適切な評価がなされている状態を確保

TDDと拡張部分



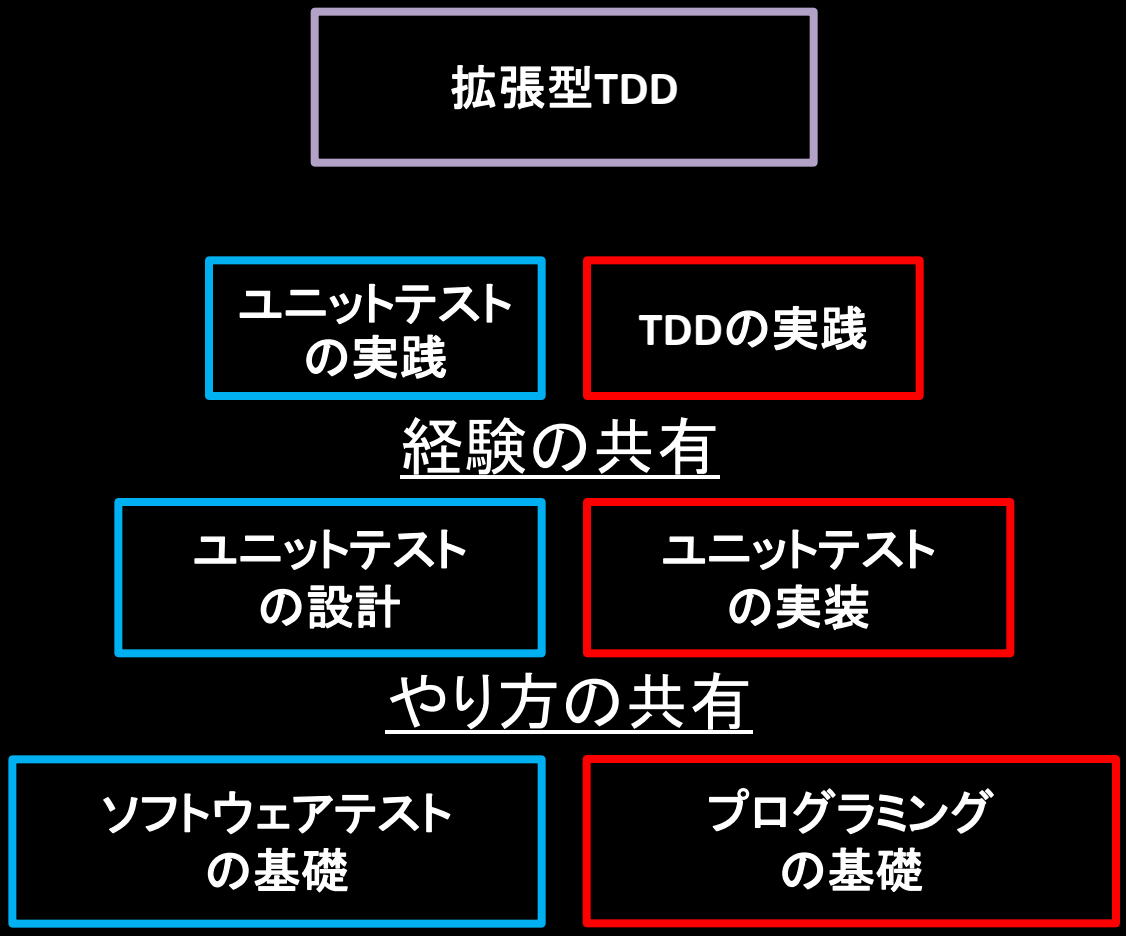
テスト設計の作りこみ拡張



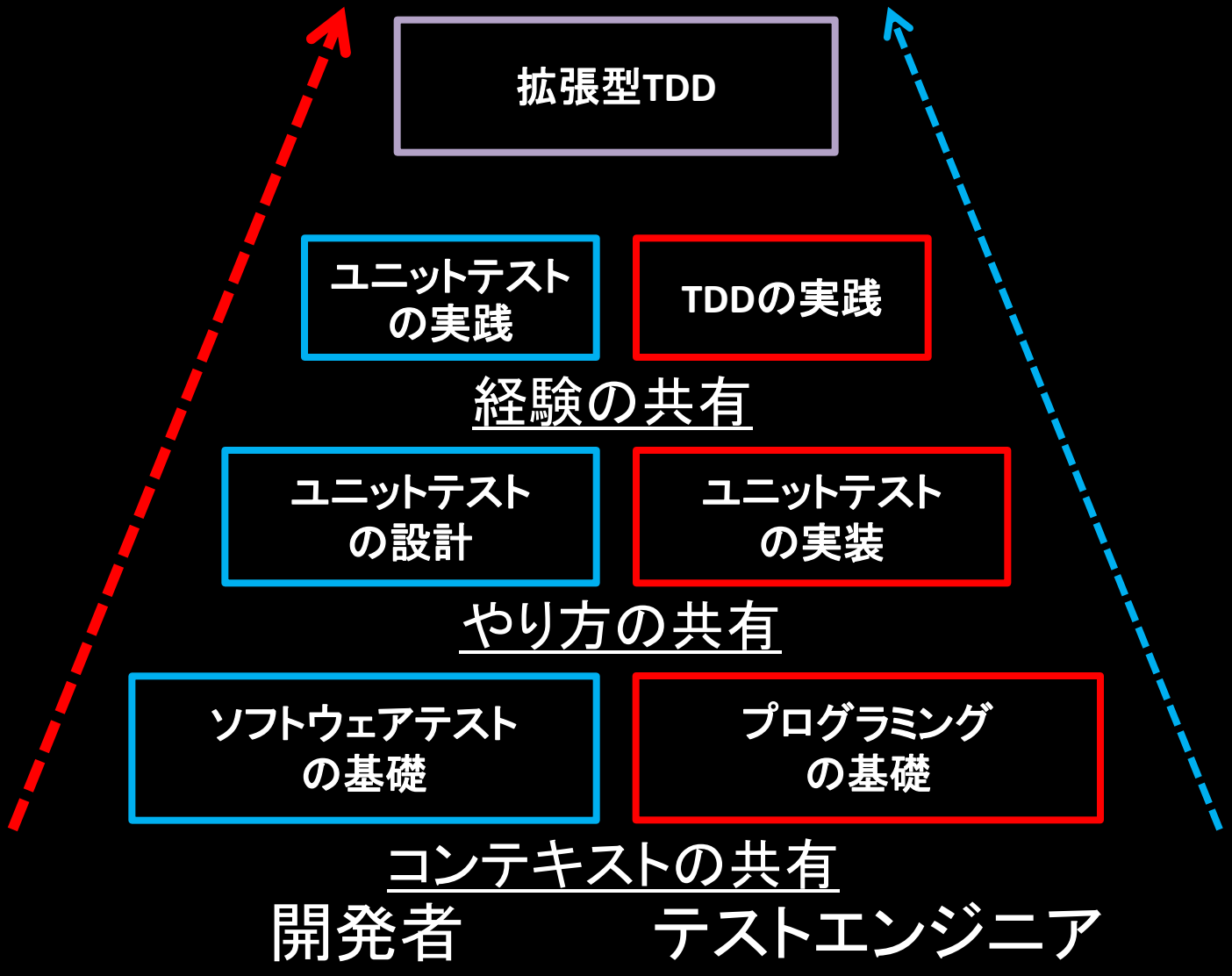
拡張型TDDライブ

拡張型TDDを
効果的に運用するには

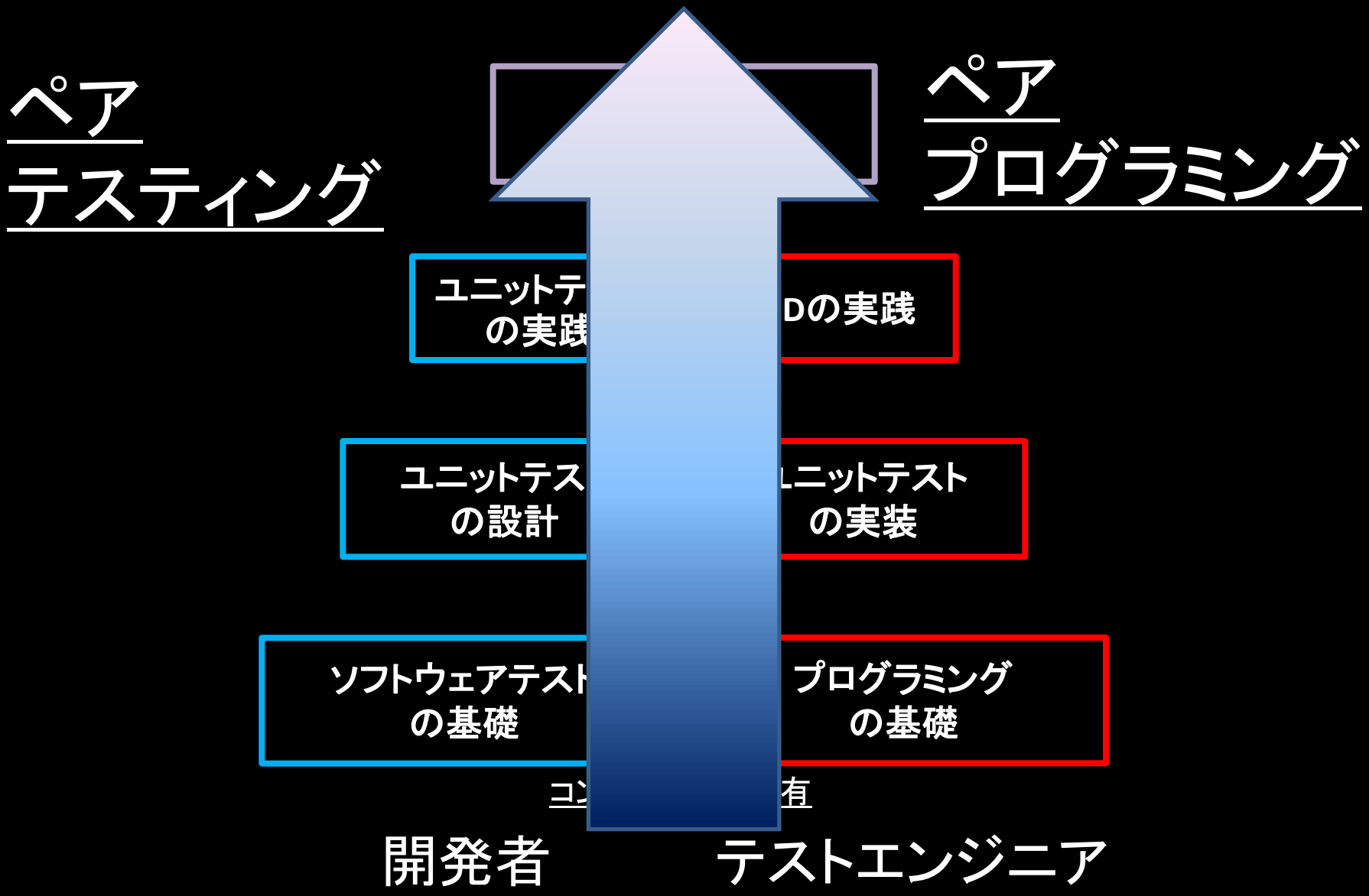
拡張型TDDを支える能力モデル



拡張型TDDを支える能力モデル



拡張型TDDを支える能力モデル



拡張型TDDを支えるもの

- 拡張型TDDの効率化はチーム文化の課題
- TDDにおいてテストエンジニアと開発者のコラボレーションの相乗効果は大
- テストエンジニアと開発者がコンテキスト、手法、経験を共有し、より効率的なTDDを！

ご清聴ありがとうございました

- 拡張型TDD
  - ４つの活動
  - 開発者テスト/品質確保のテストの両立
  - ピンポイントのテストの評価・作りこみ/継続的な底上げ