授業計画

① イントロダクション
② 開発プロセス（要件定義～初期設計）
③ 開発プロセス（論理設計）
④ 開発プロセス（実装）
⑤ 開発プロセス（結合テスト）
⑥ 開発プロセス（システムテスト～運用テスト）
⑦ 品質確保の取組１（レビュー）
⑧ 品質確保の取組２（コーディング）
⑨ 品質確保の取組３（テスト）
⑩ 構成管理（構成管理とは）
⑪ Webプラットフォーム１
⑫ Webプラットフォーム２
⑬ モバイルプラットフォーム１
⑭ モバイルプラットフォーム２
⑮ 開発プロセス（アジャイル１）
⑯ 開発プロセス（アジャイル２）
⑰ 構成管理（構成管理ツール）
⑱ プロジェクト演習（課題提示～方針決定）
⑲ プロジェクト演習（１サイクル目の要件設定～ソフトウェアサブシステム分割）

⑳ プロジェクト演習（設計1）
㉑ プロジェクト演習（実装～テスト1）
㉒ プロジェクト演習（評価1）
㉓ プロジェクト演習（2サイクル目の方針決定）
㉔ プロジェクト演習（2サイクル目の要件設定～ソフトウェアサブシステム分割）
㉕ プロジェクト演習（設計2）
㉖ プロジェクト演習（実装～テスト2）
㉗ プロジェクト演習（評価2）
㉘ プロジェクト演習（プロジェクト全体評価）
㉙ プロジェクト演習成果発表
㉚ プロジェクト演習成果発表

ソフトウェアシステム開発

第9回： 品質確保の取組３（テスト）

1. テストの概要

1.1 目的

- 成果物の一つであるプログラムを対象とし，仕様書などで求められている動作が実現できているかどうかを確認する．（動的検証）
- 達成された品質レベルを確認し，開発の終了や出荷の判断といった意思決定に必要なエビデンスを提供する．（品質レベル確認）

1.2 特性

- 実運用で起きりうる様々な事象（時間軸上連続する事象）を，スナップショットのような形で切り出して確認する（離散的な点で確認）ため，プログラム内に存在する不具合やバグを検出することはできるが，すべてのテスト項目を実施したとしても，プログラム内に不具合やバグが存在しないということは証明できない．

2. テストの種類と方法

2.1 テスト時期による区分

- 単体テスト
  プログラムユニット（関数，クラスなど）の機能動作の正しさを検証する．
- 結合テスト
  プログラムユニットを結合させ，1つの動作機能の塊としての正しさを検証する．
- システムテスト
  ソフトエアシステム全体としての動作機能の正しさを検証する．

また，

- 受入れテスト
  システムが稼働できるかどうかの妥当性確認を中心に行う．受入れテストでは，例えば，αテストやβテストに分けて行うことがある．

2.2 テスト方法による区分

- ホワイトボックステスト
  プログラムユニットなどの内部構造やロジックの正しさを確認する.
- ブラックボックステスト
  プログラムユニットなどの内部構造は確認せずに, その入出力や動作結果を中心に正しさを確認する.
- 状態遷移テスト
  状態遷移図上の遷移パスを実行させるイベントを発生させて, 実際に状態が適切に遷移していくかを確認する. 通常, 状態遷移の無限ループや状態のデッドロックなどを確認する.

2.3 テスト観点による区分

- 機能テスト
  設計書に記述された機能が実装されたかどうかを確認する. 正常系のテストと異常系のテストを含む.
- 性能テスト
  時間上の応答スピードの確認や, 過負荷テスト, メモリやストレージなどに関する容量テストなどを行う.
- 回復テスト
  システムに異常などが発生して, ソフトウェアが必要な機能サービスを提供するように回復できるかなどを確認する.

3. テスト項目の作成

3.1 ホワイトボックステストの場合

- 命令網羅テスト
  すべての命令を最低1回は実行するようにテストパスを作る.
- 分岐網羅テスト
  すべての分岐方向を必ず一度は通過させるようにテスト項目を設計する.
- 条件網羅テスト
  個々の条件を満たすすべての条件の組合せを用意しテストパスを設計する.

3.2 ブラックボックステストの場合

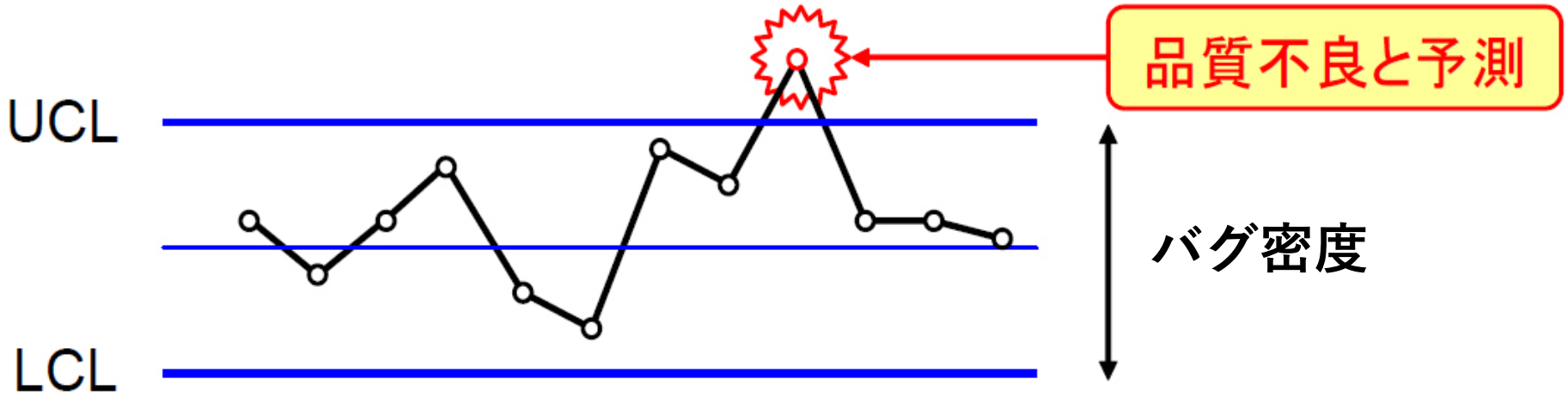
- 同値分割法
  範囲内の代表値（有効同値）を1つ, 範囲外の代表値（無効同値）を1つそれぞれテスト項目とする.

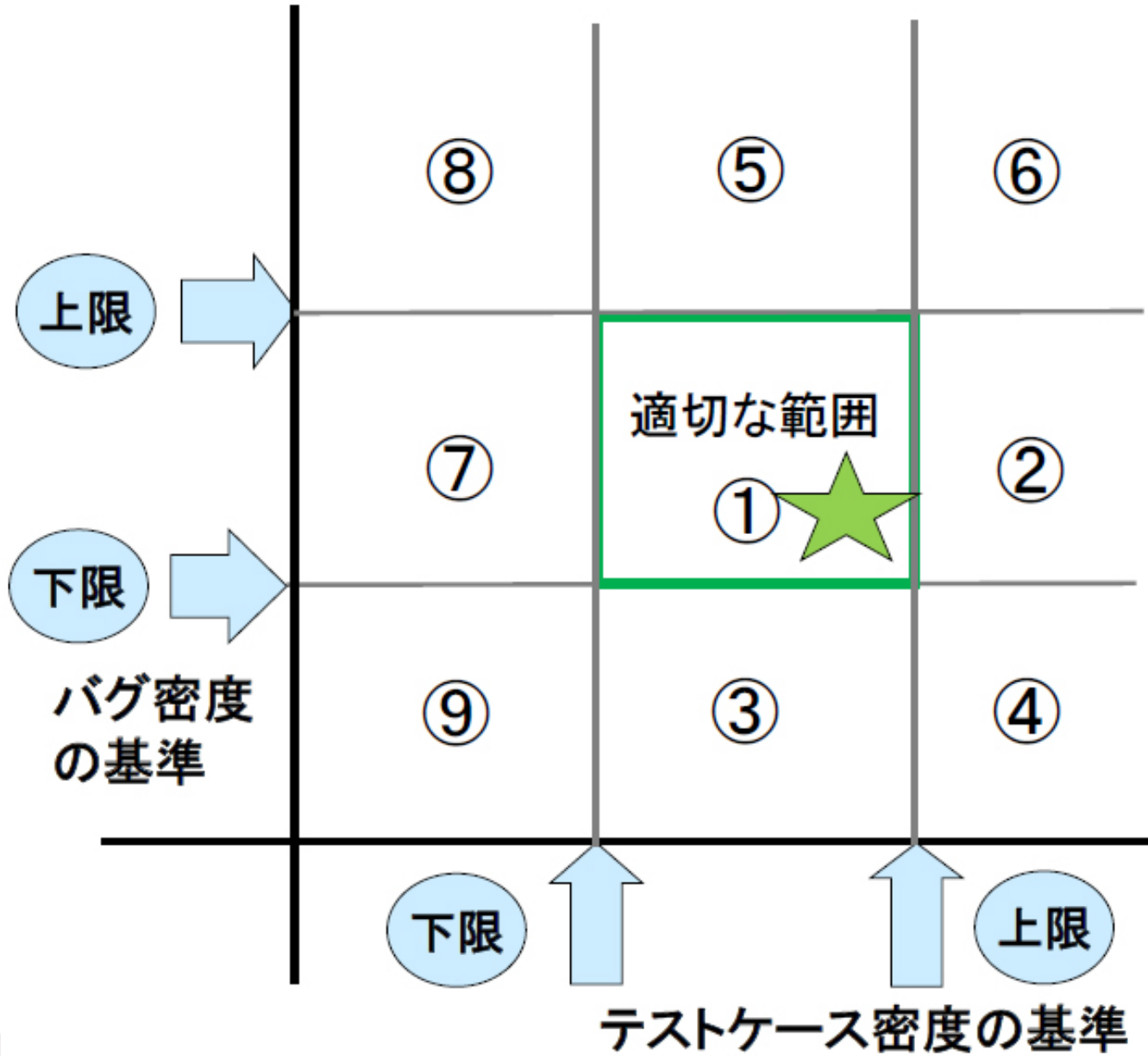
- 限界値・境界値分析
  上記の有効同値，無効同値に加えて，各同値域の端点をテスト項目に加える．

4. テスト管理

4. 1　管理図分析

- 閾値モデル
  - 概要　ある尺度の閾値によって分類するモデル
  - UCL(上部管理限界線　Upper Control Limit)
  - LCL(下部管理限界線　Lower Control Limit)
- 管理図分析
  - データの分布がUCLとLCLに対して，どの位置にあるかで，データが正常値であるか外れ値であるかを判定する





バグ密度が低い
⇒品質が高い？テストが不足？
テストケース密度が高い
⇒無駄なテスト？

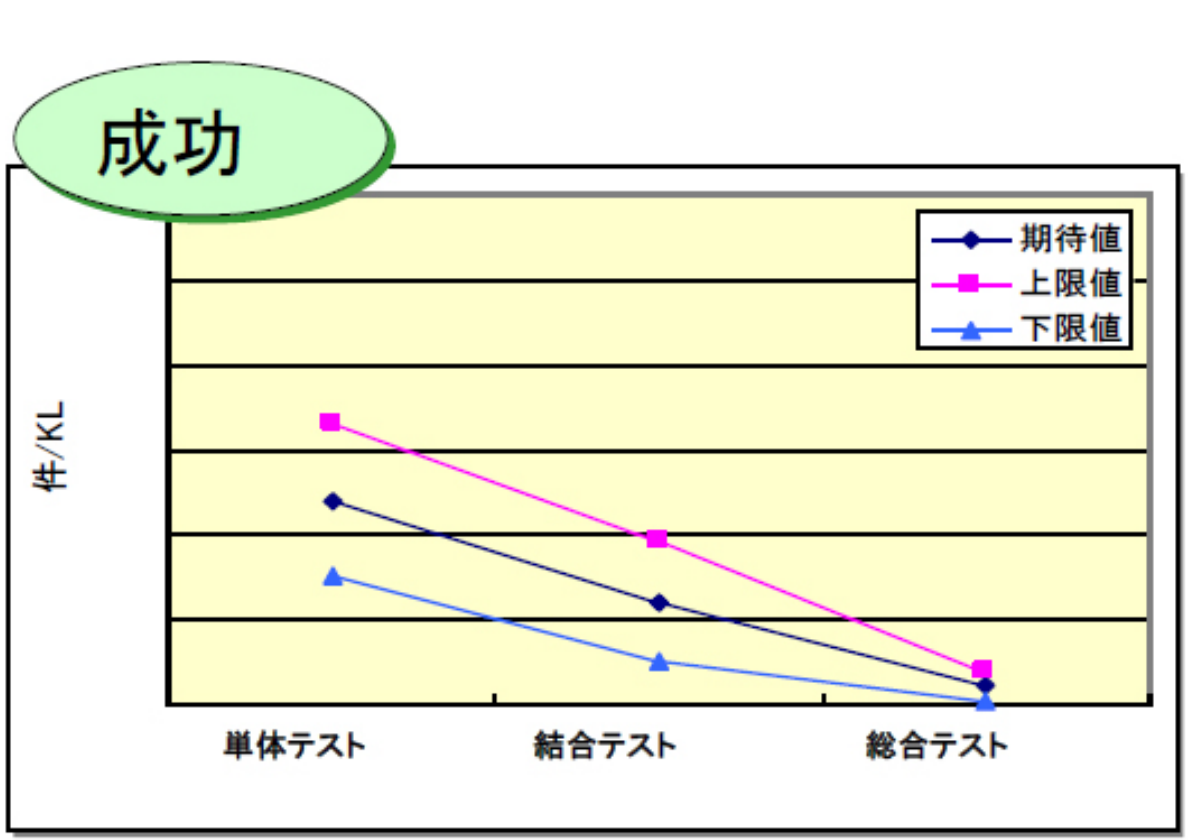
テストケース密度、バグ密度、
バグの内容(原因)の分析による
判断が必要

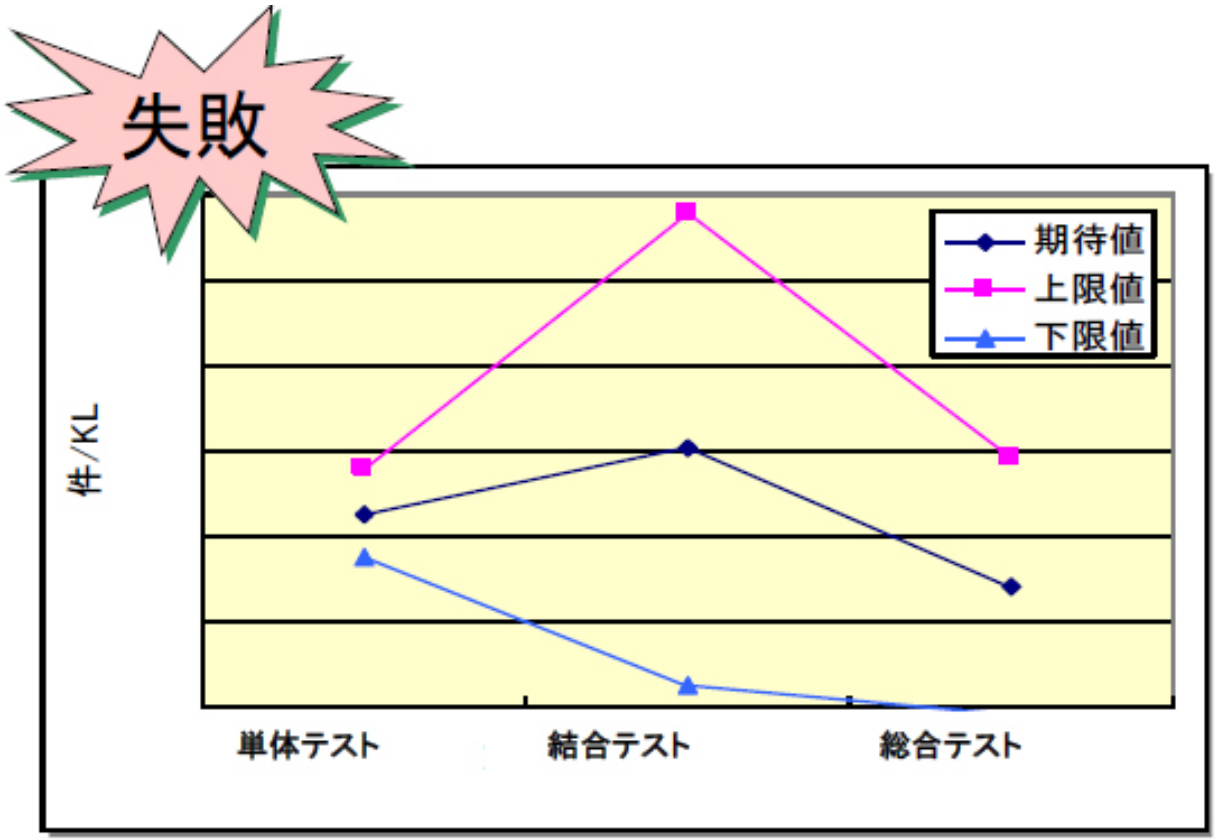
ゾーン	評価	品質
第1ゾーン	一応品質は良好、テスト効率も計画通り。	良
第2ゾーン	テスト効率がやや悪、テスト内容点検	低↑点検順位↓高
第3ゾーン	テスト内容が適切か点検	
第4ゾーン	テスト効率がやや悪、テスト内容点検	
第5ゾーン	前工程の品質確保不足、内容点検	
第6ゾーン	前工程の品質確保不足、内容点検	
第7ゾーン	テスト不足、前工程の品質確保不足、内容点検	
第8ゾーン	テスト不足、前工程の品質確保不足、内容点検	
第9ゾーン	テスト不足、内容点検	

大

4.2 テストフェーズにおけるバグ密度の推移図

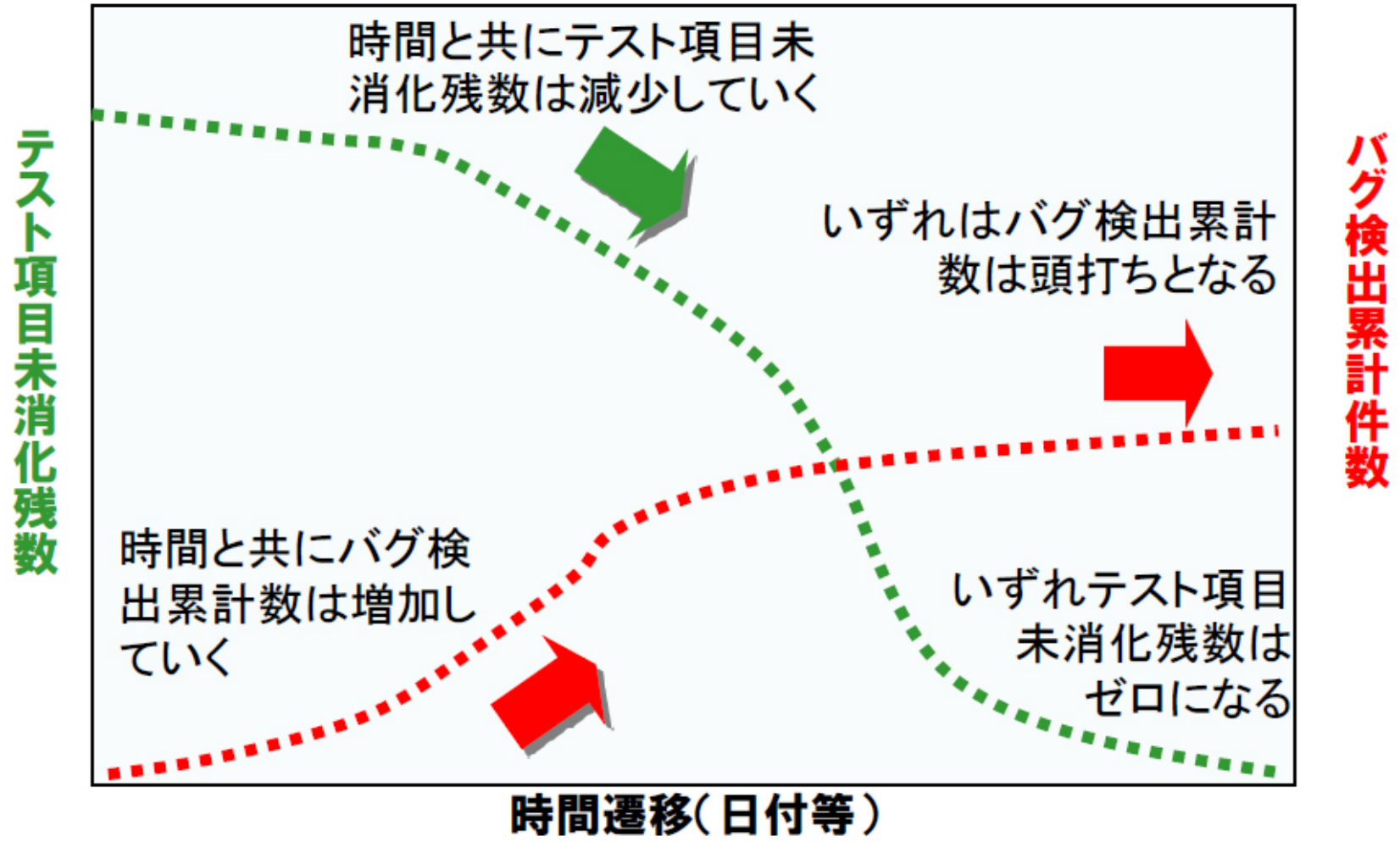
- 成功例と失敗例



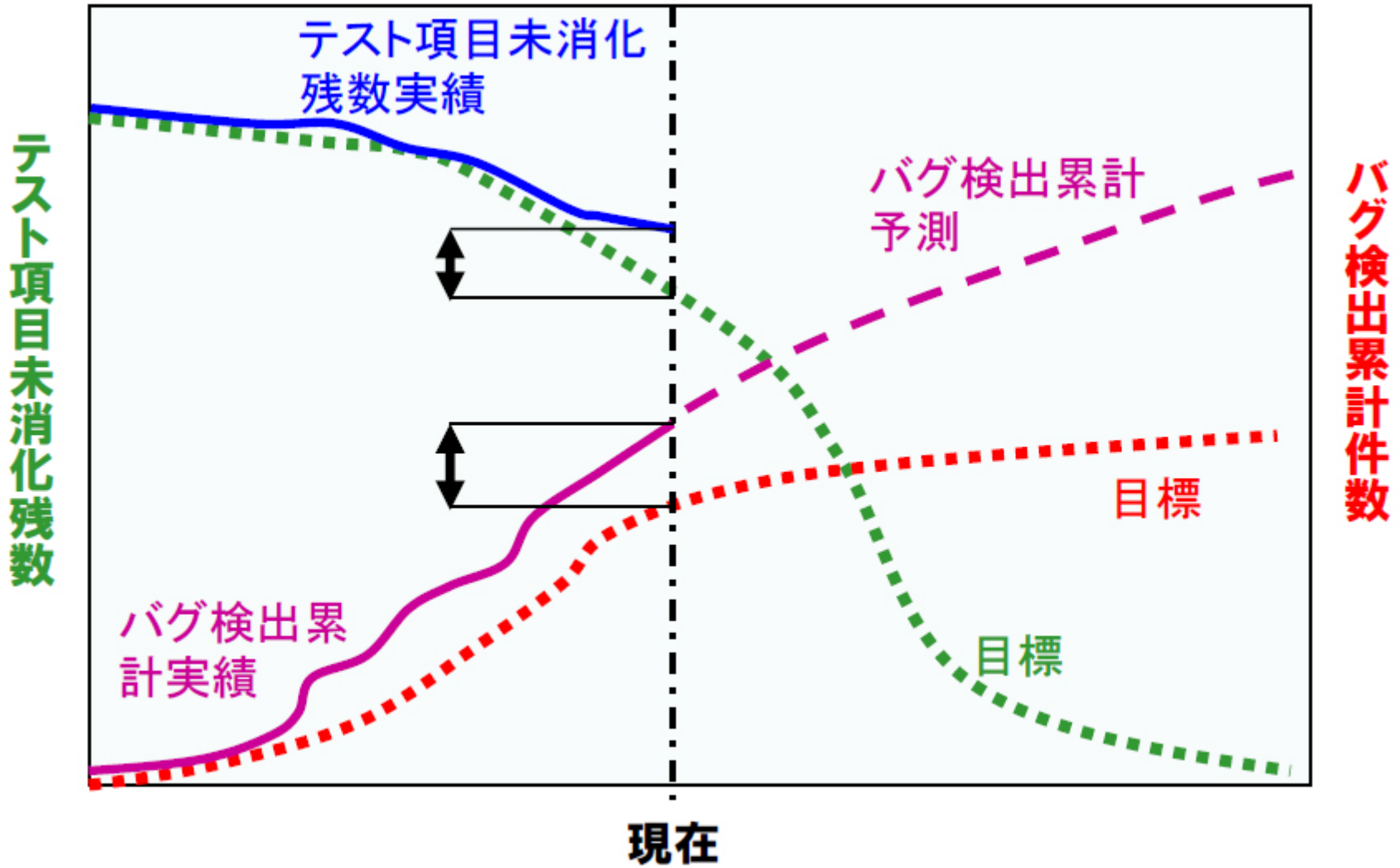


4.3 テストPB曲線（Passive Balance Curve）

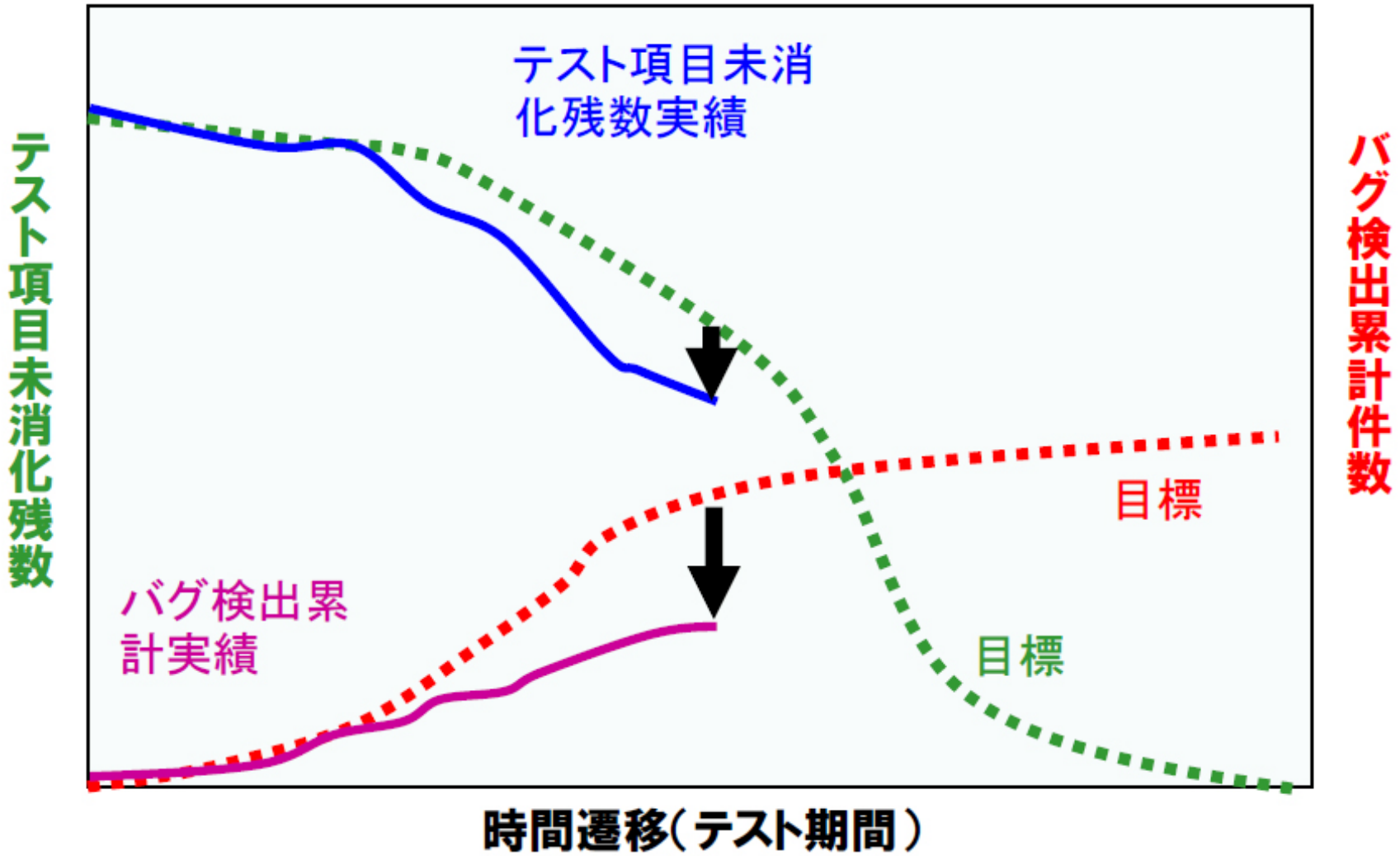
- PB曲線は，検出したバグの累積とテストの消化状況を表したもので，テストの完了見通しを判断するために使われる．



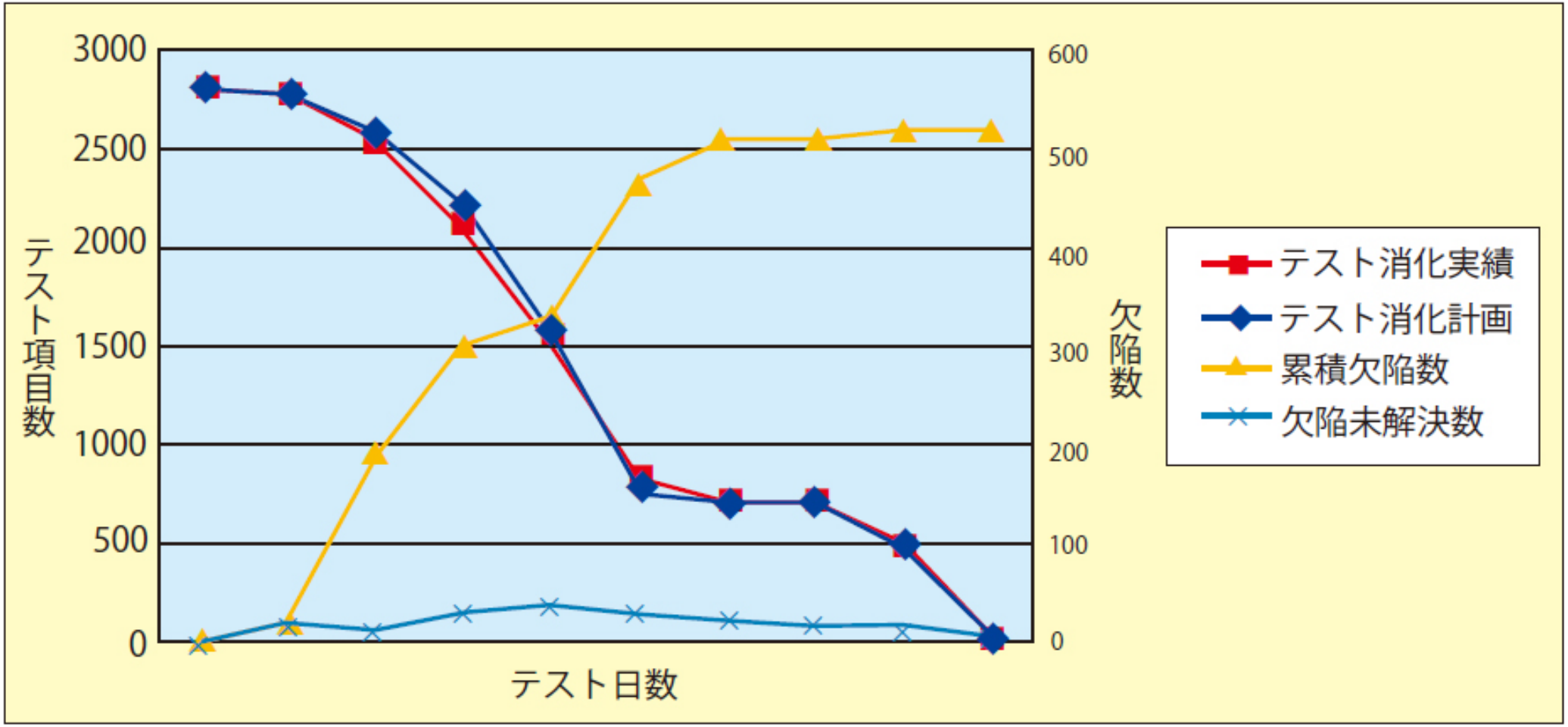
- この例では，テストの消化が目標より遅れているにもかかわらず，バグが目標よりも多く検出されているという状況で，品質状況に問題があると言わざるを得ない．



- この例では，テストが目標より進行しているが，バグの検出が目標より少ない．（→特定領域のテストが安易に進められている可能性，または難易度の低いテストばかりが偏って実施されている可能性がある）



- 実際には，テストの進行と同時に，検出されたバグが修正されるので，「欠陥未解決数」の線も描かれる．



ソフトウェアシステム開発

第10回： 構成管理（構成管理とは）

1. 構成管理とは

1.1 目標

- 設計書やソースコードなどの成果物を規定通りに整理し，管理する．

I.2 入力，出力およびタスク構成

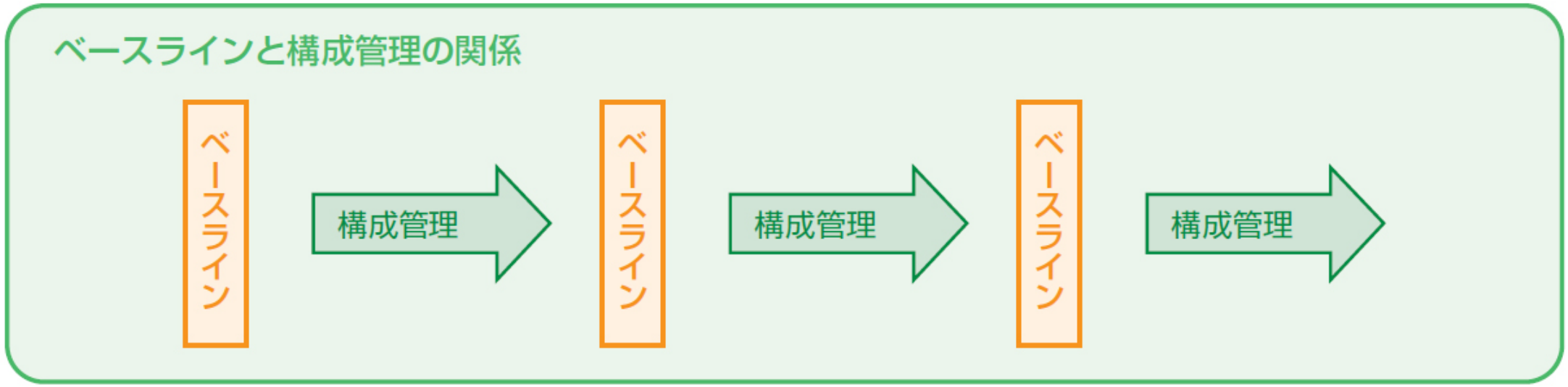
入力	タスク構成	出力
構成管理対象（成果物，開発環境，結合されたシステム等）	A) 構成管理対象物の把握 ①構成管理の方針と仕組みを明確にする • 構成管理の手順とルール 追跡管理できるように表を用いるなどして各成果物間のトレーサビリティを対応付けられるようにする． • ベースラインの予定と監査の方針など	構成管理票

入力	タスク構成	出力
	②構成管理の対象物を把握し決定する．管理対象になるのは各アクティビティの出力として，以下のものがある． ・ 仕様書，設計書，報告書などのドキュメント類 ・ ソースコード，結合されたシステムなど ・ 開発環境	

入力	タスク構成	出力
構成管理対象 （成果物，開発環境，結合されたシステム等） 構成管理票	B) 構成管理／変更管理履歴の管理 ①構成管理の対象物に変更が発生した場合にベースラインおよび変更履歴を更新しバージョン管理する ②ベースラインへのアクセスを管理する ③ベースラインからの成果物の生成を認可する ④構成管理情報を関係者に配布する	構成管理対象 （成果物，開発環境，結合されたシステム等） 構成管理票

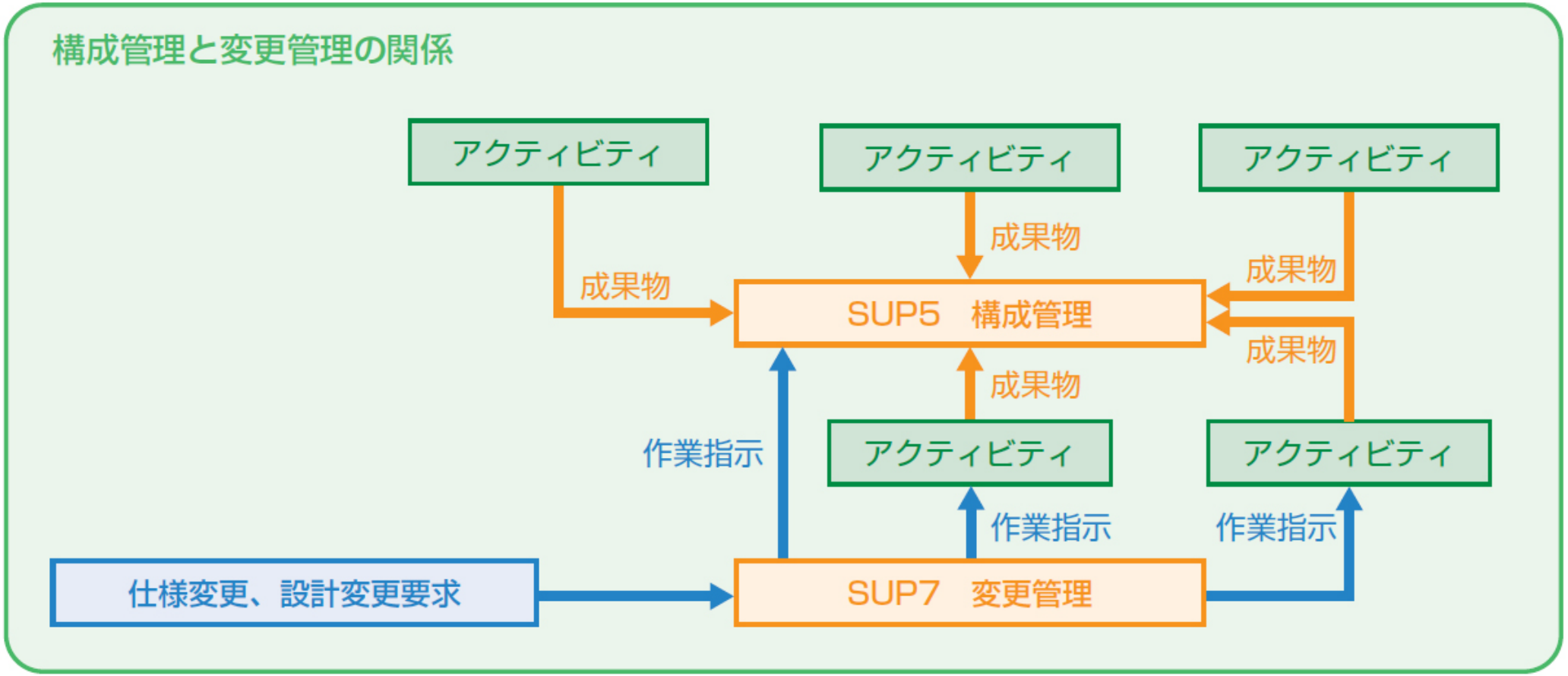
2. ベースラインと構成管理の関係

ベースラインとは，ある時点での構成要素をベースライン（出発点）として，以後の変更を管理する手法のこと．ブランチはベースラインから派生したものを指す．



3. 構成管理と変更管理の関係

一般的には成果物を管理するデータベースなどを用意して成果物の一元管理を行い，開発の際には成果物をチェックイン，チェックアウトして開発作業を進める．また，複数のバージョンが同時に進行する開発では，派生バージョンの管理を適切に行い，個々のバージョン間の関連を図示するなどして，明確にしておく必要がある．

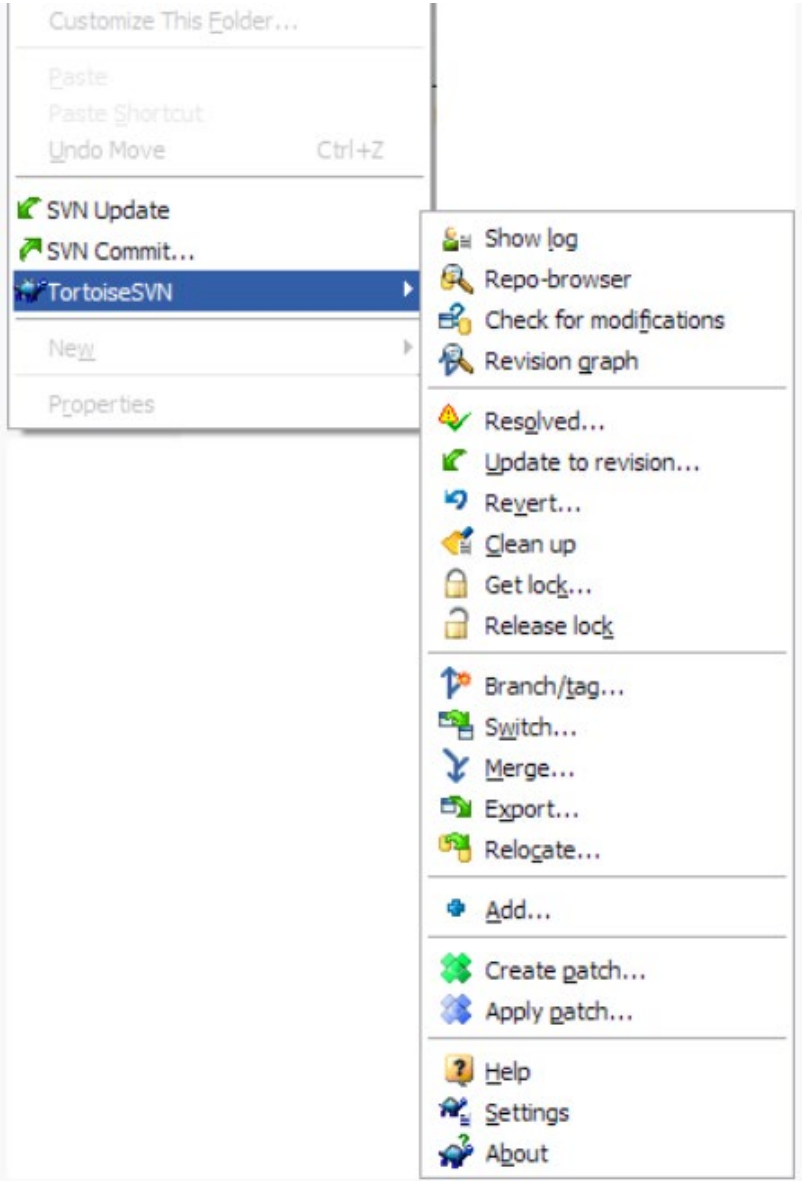


構成管理票は，表ソフト（例：Excel）などを使って製品リリースに含まれる各成果物とその保存場所が記載される一覧表である．

製品リリース番号	製品仕様書Verと保存場所	ソフトウェア要求仕様書Verと保存場所	ソフトウェア方式設計書Verと保存場所	ソフトウェア詳細設計書Verと保存場所	ソースコードブランチと保存場所	単体テスト仕様書兼成績書Verと保存場所	結合テスト仕様書兼成績書Verと保存場所	総合テスト仕様書兼成績書Verと保存場所
Rel_A001	Ver1.0 ¥…¥ProdSpec1.0.doc	Ver1.0 ¥…¥SoftReqSpec1.0.doc	Ver1.0 ¥…¥SoftArchDesg1.0.doc	Ver1.0 ¥…¥SoftFuncDesg1.0.doc	¥…¥SourceCodeBranchA001	Ver1.0 ¥…¥SoftUnitTest1.0.doc	Ver1.0 ¥…¥SoftIntTest1.0.doc	Ver1.0 ¥…¥SoftSysTest1.0.doc

4. 構成管理ツール

- Subversion管理ツール － TortoiseSVN
  - GNU General Public Licenseのもとで配布されているフリーソフトウェアである
  - Windowsのシェル拡張として実装されている
  - クライアント／サーバモデル
  - 以下の類似ツールが存在している．
    TortoiseGit=Git向けの同様のツール，
    TortoiseCVS=CVS向けの同様のツール，RabbitVCS=Linuxで使用できる同様のツール，など



- 分散バージョン管理システム　ー　Git
  - GNU General Public Licenseのもとで配布されているフリーソフトウェアである
  - Linuxコミュニティで開発されたもの
  - 中央サーバを持たない完全peer to peerモデル
  - リポジトリをまるごと各ユーザのローカルマシンにクローンして，リポジトリの完全なコピーを自分のワーキングディレクトリにそのまま持つ
  - 以下のようなステップの繰り返しで開発作業が行われる

1. git clone: 中心リポジトリをローカルに複製
2. git commit: ロカールリポジトリに変更履歴を記録
3. git push: ローカルの変更内容を中心リポジトリに反映. 変更内容が衝突する場合, git mergeで手動で解決
4. git pull: 更新された中心リポジトリ（他者の作業内容も統合されている）をローカルの複製に反映