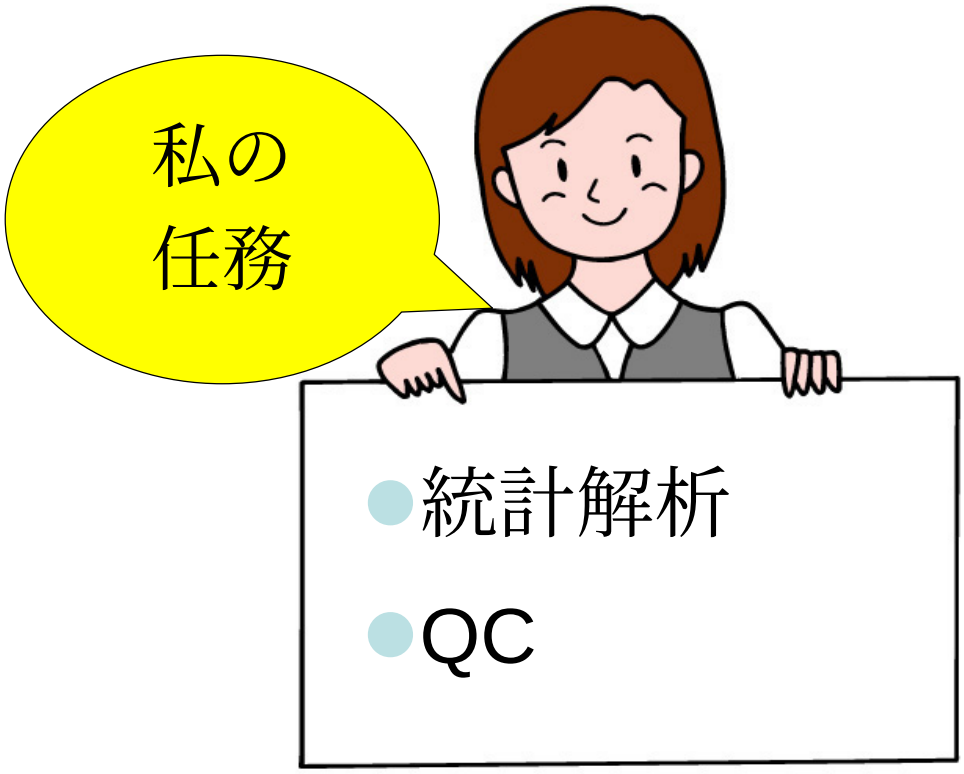
統計解析における
高度作業プロセスのバリデーション

㈱バイオスタティスティカル リサーチ　生物統計部　板垣　智代


私の
任務
統計解析
QC

私の業務

解析結果の
照合･検証

文書管理

バリデーション

私の経歴
システムエンジニア

ダブルプログラミングと文書管理だけでは不十分

本日の内容

1. 医薬品開発とシステム開発のバリデーションの比較
2. 現状におけるバリデーションの問題点
   (ダブルプログラミングの位置づけ)
3. ダブルプログラミングエラーの解決策
4. レビュープロセスにおけるバリデーションの推奨
   解析用データセット作成プロセスを例として
5. まとめ

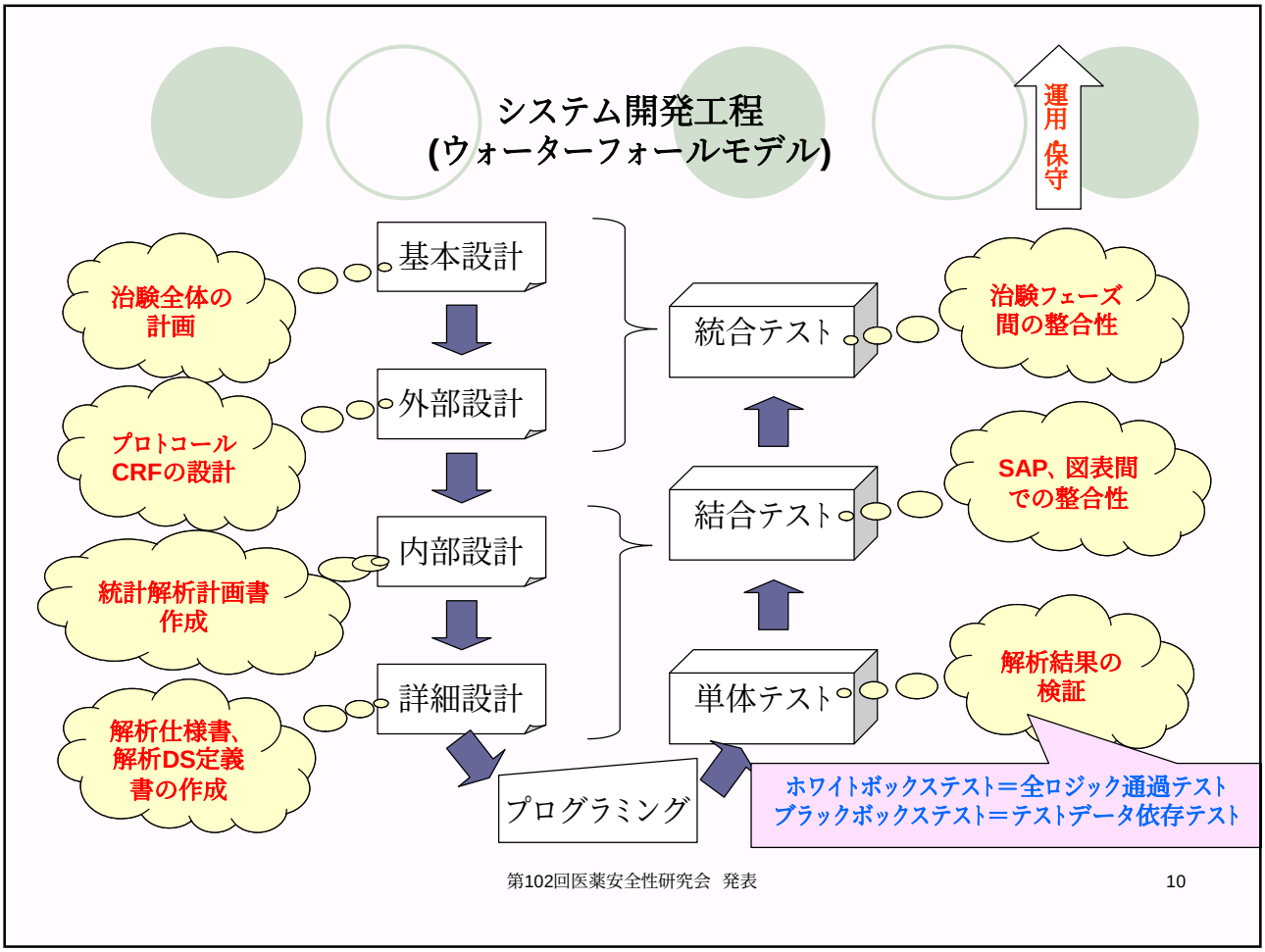
統計解析のバリデーション
本日の範囲

開発計画時の問題	○主要評価項目の選定 ○例数設計 ○解析手法の決定 ○解析対象集団
解析計画時の問題	○手法の妥当性 ○データ採否の妥当性
解析用データセットの問題	○データセットの設計 ○作成プロセス
解析プログラムの問題 ダブルプログラミング	○作成プロセス ○結果の妥当性

用語の定義

- DS：データセット
- PGM：プログラムまたはプログラミング
- W-PGM：ダブルプログラミング
- SAP：Statistical Analysis Plan(解析計画書)
- LOCF：Last Observation Carried Forward

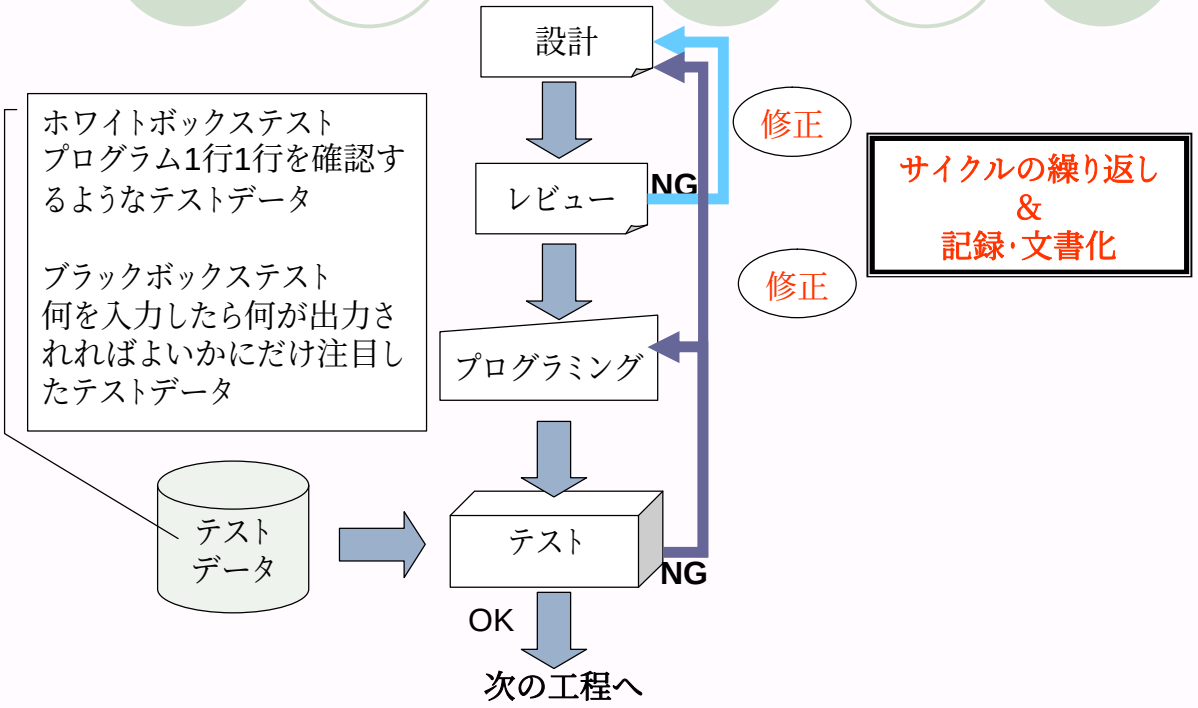
1.医薬品開発とシステム開発のバリデーションの比較


システム開発工程
(ウォーターフォールモデル)
運用・保守
基本設計
外部設計
内部設計
詳細設計
プログラミング
統合テスト
結合テスト
単体テスト
治験全体の計画
プロトコールCRFの設計
統計解析計画書作成
解析仕様書、解析DS定義書の作成
治験フェーズ間の整合性
SAP、図表間での整合性
解析結果の検証
ホワイトボックステスト＝全ロジック通過テスト
ブラックボックステスト＝テストデータ依存テスト
第102回医薬安全性研究会　発表
10

単体テスト

- プログラムごとに動作確認･検証
  - ホワイトボックステスト
    すべてのロジックを通過するテストデータを用いる
    結果の保証範囲はロジックに依存
  - ブラックボックステスト
    検証したいデータの組み合わせでテストする
    ロジックは無視
    結果の保証範囲はデータに依存

バリデーションとは・・・


設計
レビュー
NG
修正
プログラミング
修正
テスト
NG
OK
次の工程へ
テスト
データ
ホワイトボックステスト
プログラム1行1行を確認するようなテストデータ
ブラックボックステスト
何を入力したら何が出力されればよいかにだけ注目したテストデータ
サイクルの繰り返し
&
記録･文書化

バリデーション

- プロセスバリデーションの一般原則に関するFDAガイドライン(1987年5月)
- 「特定のプロセスが一貫して事前に定められた規格および品質特性に合致する製品を製造できることを高度に保証する根拠を示す文書を作成すること」

2.現状におけるバリデーションの問題点 (ダブルプログラミングの位置づけ)

ダブルプログラミングの問題

- 同じ目的の解析プログラムを独立に2人以上が作成し、同じテストデータ(臨床試験データ)を用いて得た両者の結果を照合することでミスを発見する方法
- 解析プログラム作成時におけるプログラムミスを発見するための方法のひとつ。

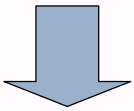
- 利点→エラー率減少効果
- プログラムに残るエラー率

シングルプログラミング(検証なし)	最大100%
ダブルプログラミング	0.15%

- 菅波秀規、益田隆史　ダブルプログラミングによる統計解析の品質管理　57,SUJI-J2000

ダブルプログラミングの問題

- 同じ目的の解析プログラムを独立に2人以上が作成し、同じテストデータ(臨床試験データ)を用いて得た両者の結果を照合することでミスを発見する方法



実はダブルプログラミングとは・・・
ブラックボックステストの検証の手段のひとつ

ダブルプログラミングの問題

- データに依存した結果の一致性の確認のみ
  →2人が出した答えが一致しているから、正答率は確率的に高いというだけ
- シングルプログラミング＋手計算での確認と同じか、下手をすると低いレベルの検証でしかない
  →2人とも同じミスをしたときは気が付かない
  →技術的に間違いやすい傾向にあるものは、2人でやっても間違いやすい
- 真の正確性、設計との乖離の確認はできていない
  →結果の内容の妥当性を保証するものではない

SAPの段階で保証をしておかなければならない

ダブルプログラミング以外にないのか？

- ブラックボックステストはシングルプログラミングで可能。全症例データから手計算で解析結果があっていることを目で確認すればよい
- ブラックボックステストよりも品質保証のレベルが高いホワイトボックステストをすればいい

- 実施可能性→非現実的

ダブルプログラミング以外にないのか？

- 臨床試験の統計解析ではコーディング～単体テストにかけている時間がない
  - システム開発　500ステップ／人月
  - 統計解析　100ステップ×50PGM
    ＝5000ステップ／人月
- いつかは固定されるデータ＝データバリエーションが増えることがない
  - ホワイトボックステストを行うことは無駄
  - そもそも、ありもしないデータの処理ロジックを組むことさえ省略化可能

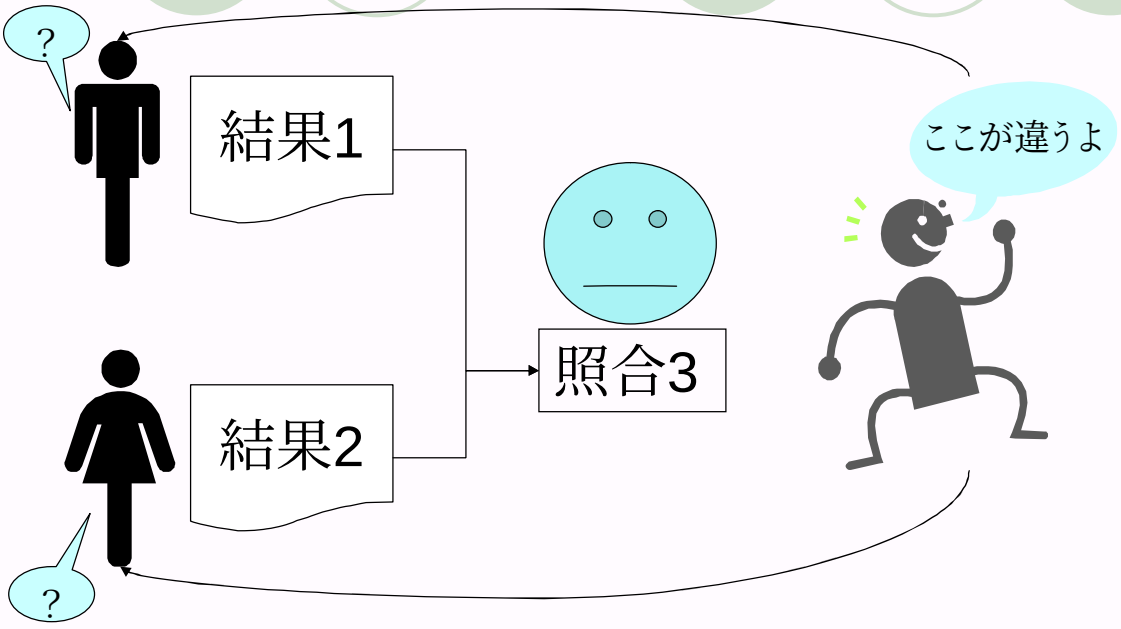
- W-PGMによるエラーを補う行為が必要

3.ダブルプログラミングエラーの解決策

ダブルプログラミングエラーの活用

- 独立照合エラー判定システム
- プログラミングエラーDictionaryの作成とプログラミング仕様書への反映･未然予防

従来の照合システム



従来の照合システム

従来の照合システムのデメリット

- 照合不一致の事実のみを担当者にフィードバック
  1) 問題解決に時間がかかる
  2) 正しくない結果に修正する可能性
  3) 最悪、お互いの結果に合わせようとする、相談（独立性の欠落）

独立照合エラー判定システム

- プログラム担当者とは独立な照合判定者が、不一致の原因を追求。各プログラム担当者にエラー箇所と理由を通達。

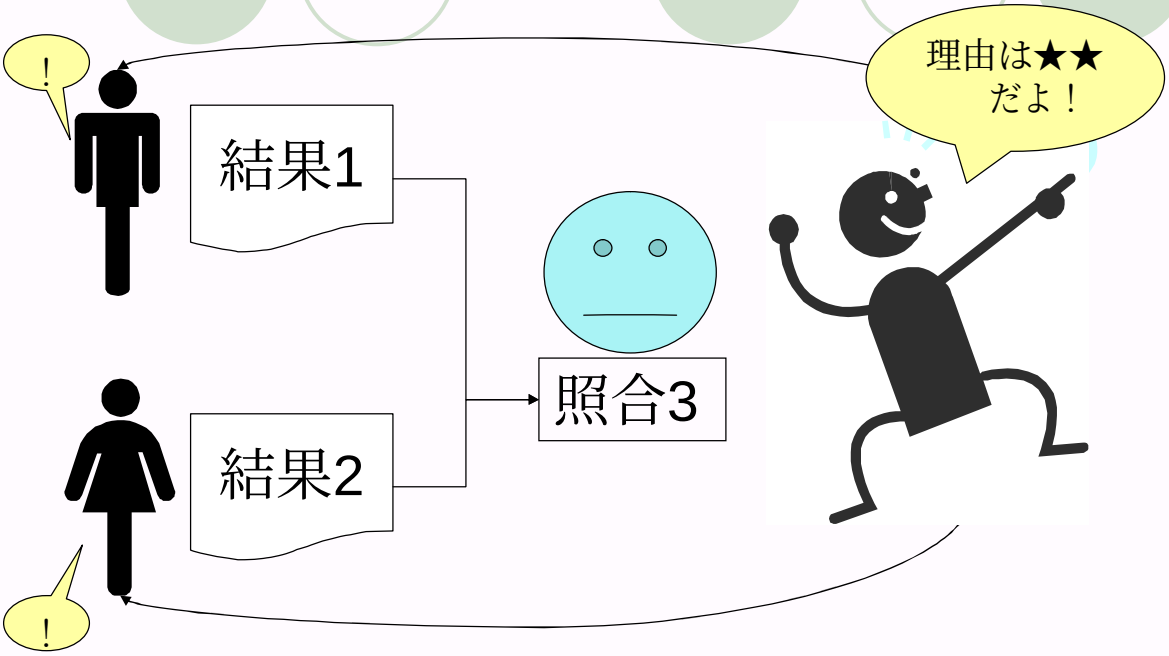
- 利点
  1) 系統的なエラーの発見
  2) 解析全体から判断するため、問題解決の論理的妥当性が保証される
  3) エラーの原因、結果の文書記録化が容易

- 欠点
  質の高いQC担当者の確保
  - 大幅なコストの上昇
  - ただでさえ人員不足、優秀な人材をQCに割けるか？

- 結論

これなしでは、ダブルプログラミングの精度を保証しきれない

独立照合エラー判定システム

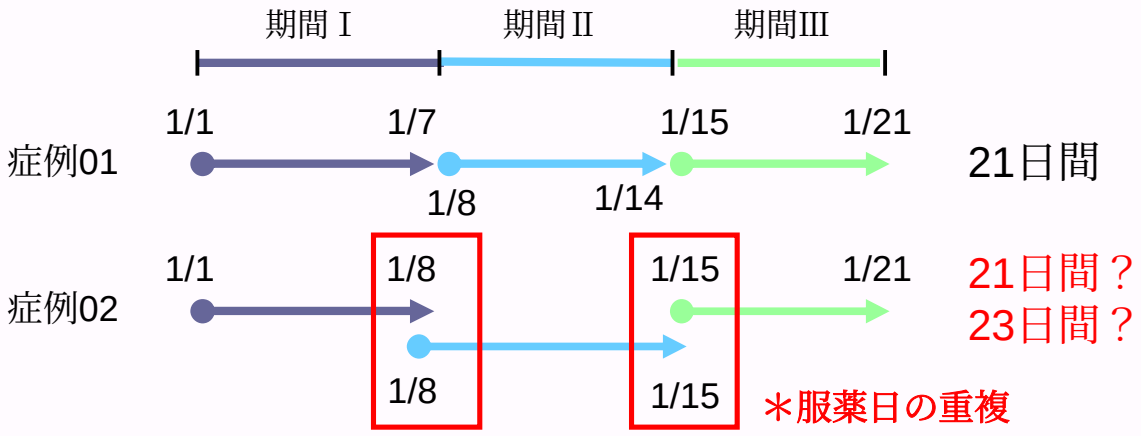


独立照合エラー判定システムでの修正例
服薬日数計算

・観察期間に3つの区分

・解析計画書の記述

「服薬日数は各期間の服薬日数の合計とする」

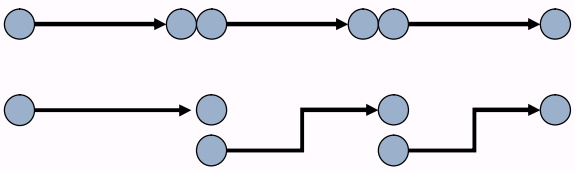


独立照合エラー判定システムでの修正例
服薬日数計算

症例02についてのダブルプログラミングエラー

プログラマーA　:服薬日数　21日間
重複した日数を除いて計算

プログラマーB　:服薬日数　23日間
期間ごとに計算した日数を
そのまま合計



不一致の事実のみ伝達されると・・・

どちらもコーディングにミスが無い。ロジックも間違っていない。

→上級者の結果に、無理に合わせてしまうことも・・・
第三者の判断が重要になる。

プログラミング　エラーディクショナリー

- 重大なプログラミングエラー、ダブルプログラミングエラーは辞書化し、プログラミング仕様書への反映（仕様書計画時）

リスクアセスメントの観点から
データベースを構築

リスクアセスメント表

リスクアセスメント		3	2	1
		確実に起こりうる	おそらく起こる	起こりそうに無い
4	許容不可・危機的	HIGH	HIGH	MEDIUM
3	好ましくない・重大	HIGH	MEDIUM	MEDIUM
2	見直しにより受け入れられる	MEDIUM	MEDIUM	LOW
1	見直さずに受け入れられる/無視できる	MEDIUM	LOW	LOW

HIGH:	設計段階で回避する、かつ、 W-PGM結果が一致しても確認する
MEDIUM:	設計段階で回避する、または、 W-PGM結果が一致しても確認する
LOW:	設計段階で回避する、または、 W-PGM結果が一致すればよい

エラーディクショナリー

NO	エラー発生フェーズ	内容	原因	重要度	発生頻度	総合的な危険度
1	W-PGM	服薬日数の計算	解析計画書記述のあいまいさ データ構造上の特徴(症例ごとに日付の入力ルールが異なる)	4	3	HIGH
2	W-PGM	観察日の補完	コーディングミス	3	2	MEDIUM
3	W-PGM	イベントの有無	解析計画書記述のあいまいさ	4	2	HIGH

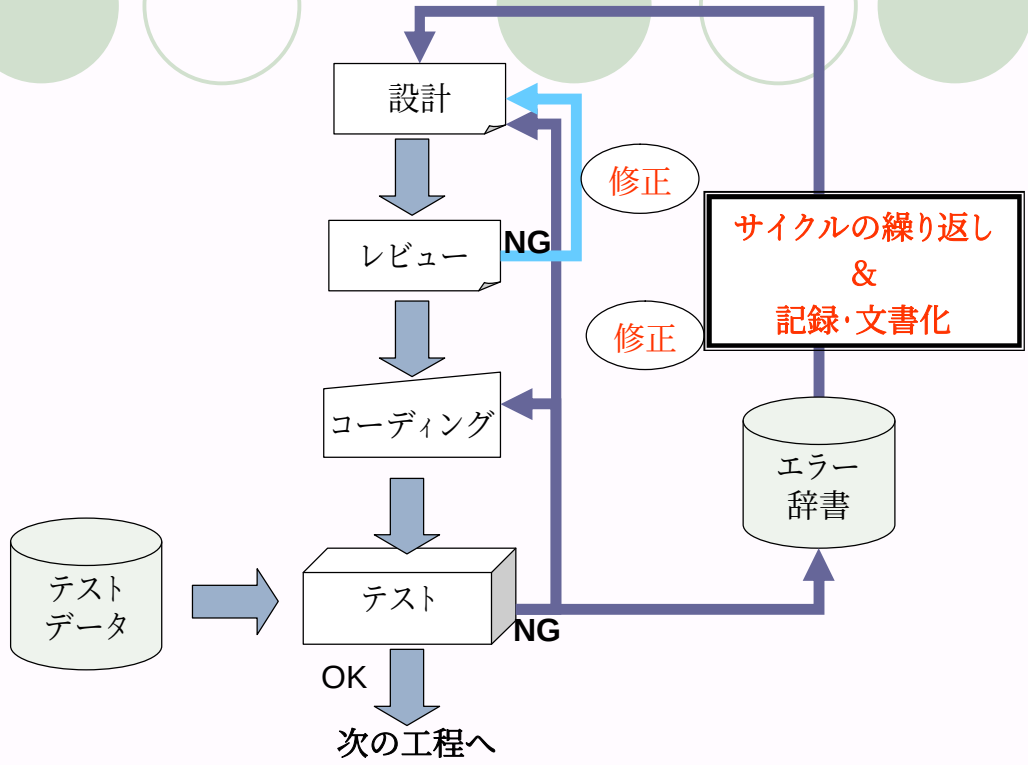
解析計画書（仕様書への反映）

- **従来の記述**
  ・期ごとの服薬日数(i)=投与終了日（i）－投与開始日（i）＋1
  ・服薬日数＝合計・期ごとの服薬日数(i)

- **データレビュー指示**
  期間（i）の投与終了日と期間(i+1)の投与開始日の重複の有無を確認すること

- **レビュー結果の仕様への反映**
  もし、重複が確認された場合は、下記を仕様書に追加
  ・IF　投与終了日（i）=投与開始日(i+1)　THEN
  投与開始日(i+1)=投与開始日(i+1)+1
  ・期ごとの服薬日数(i)=投与終了日（i）－投与開始日（i）＋1
  ・服薬日数＝合計・期ごとの服薬日数(i)

エラーディクショナリーの活用



4.レビュープロセスにおけるバリデーションの推奨

解析用データセット作成プロセスを例として

統計解析のバリデーション

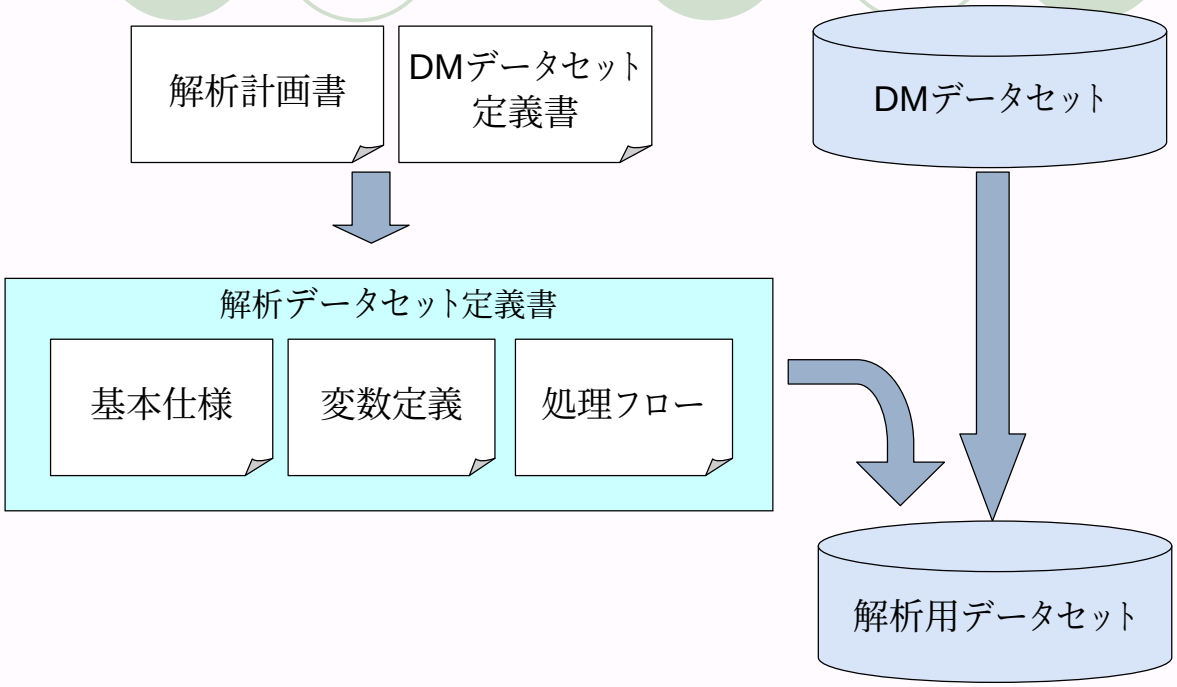
開発計画時の問題	○主要評価項目の選定 ○例数設計 ○解析手法の決定 ○解析対象集団
解析計画時の問題	○手法の妥当性 ○データ採否の妥当性
解析用データセットの問題	○データセットの設計 ○作成プロセス
解析プログラムの問題	○作成プロセス ○結果の妥当性

解析用データセット

- 治験データ収集用データベース（DMデータセット）から、統計解析・レポーティングに必要なデータを抽出したデータセット

- 利点
  1. 解析用データセット作成プロセスを独立化することで、解析プロセスを単純化できる
  2. プログラマーのスキルによって解析用データセット作成と統計解析の専門性を分けることで生産性があがる
  3. 解析計画へのフィードバックが早くなる
  4. エラー発生時の原因究明の効率化

解析用データセット作成のフロー



解析用データセット作成時のバリデーション

- 解析用データセット定義書に以下を定義
  1. 基本仕様
  2. 変数定義
  3. 処理フロー
- 解析用データセット定義書のレビュー実施
  1. レビュー参加者：解析計画書作成者・DM担当者
- 各プロセスにおけるエラーのフィードバックと記録
  1. 定義書作成時
  2. レビュー時
  3. コーディング時
  4. テスト時

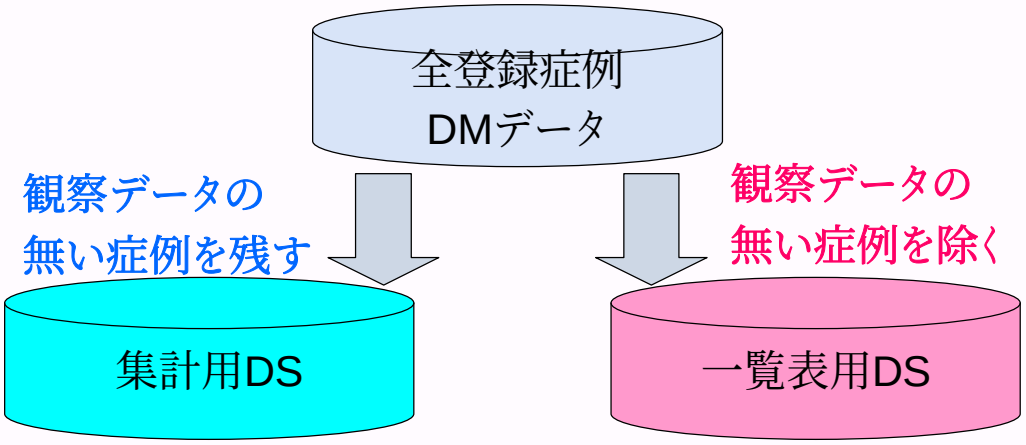
解析用データセット定義書・基本仕様

変数の定義以外にこれらを記述しておく

1. データセット定義
   - 属性
   - オブザーベションタイプ(1:N)
   - 観察データのない症例の取り扱い
2. 解析の目的　例：time-to-event、変化量、時系列データ
3. FLAG定義　FLAG作成or欠測
4. データの補完方法　例：LOCFルール
5. フォーマット定義

解析用データセット定義書・基本仕様

- なぜ必要か？
- 例：観察データのない症例の取り扱い
- データセット毎に解析目的によって変更



＊例：被験者の内訳
観察データの無い症例も
分母に含める必要がある

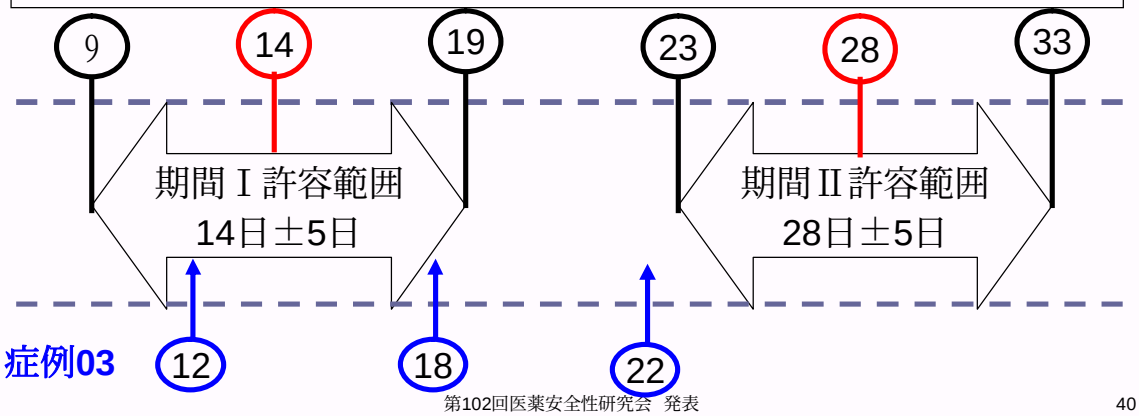
＊例：臨床検査値の一覧
データのない症例は対象外

解析用データセット定義書・基本仕様

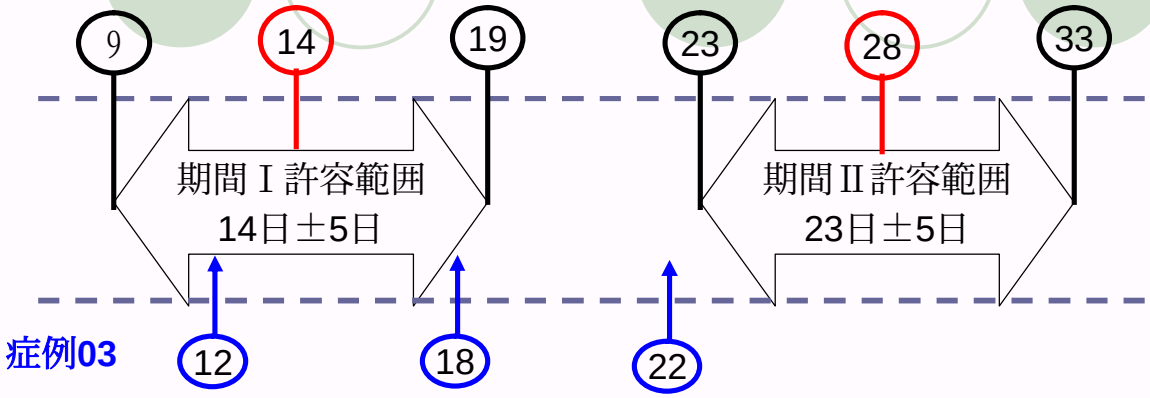
- なぜ必要か？
- 例：期間内のデータの選択ルールとLOCFの2つのルール
- ルールの相違によって選択されるデータが異なる

①期間内のデータの選択ルール：各期間の許容範囲内で、基準日に最も近い来院日

②ＬＯＣＦルール：欠測値は直前のデータで補う



解析用データセット定義書・基本仕様



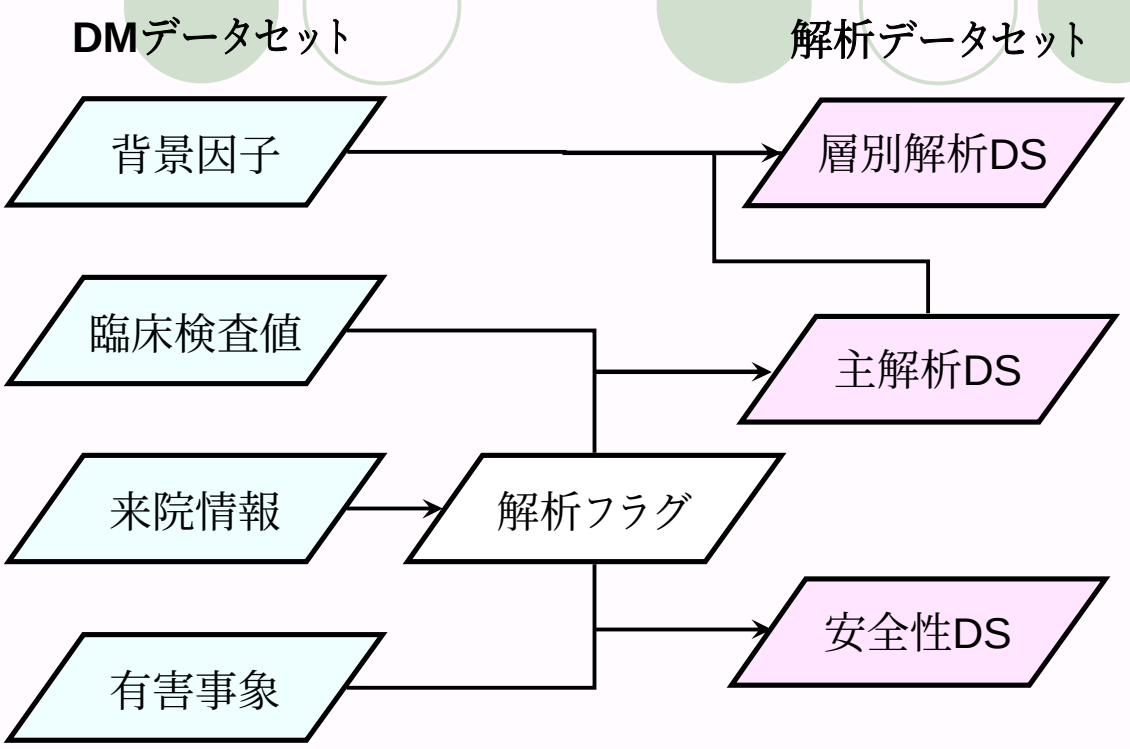
LOCFで選ばれるデータ

処理順1:①→②	→12日目のデータ
処理順2:②	→22日目のデータ
処理順3:①の許容範囲→②	→18日目のデータ

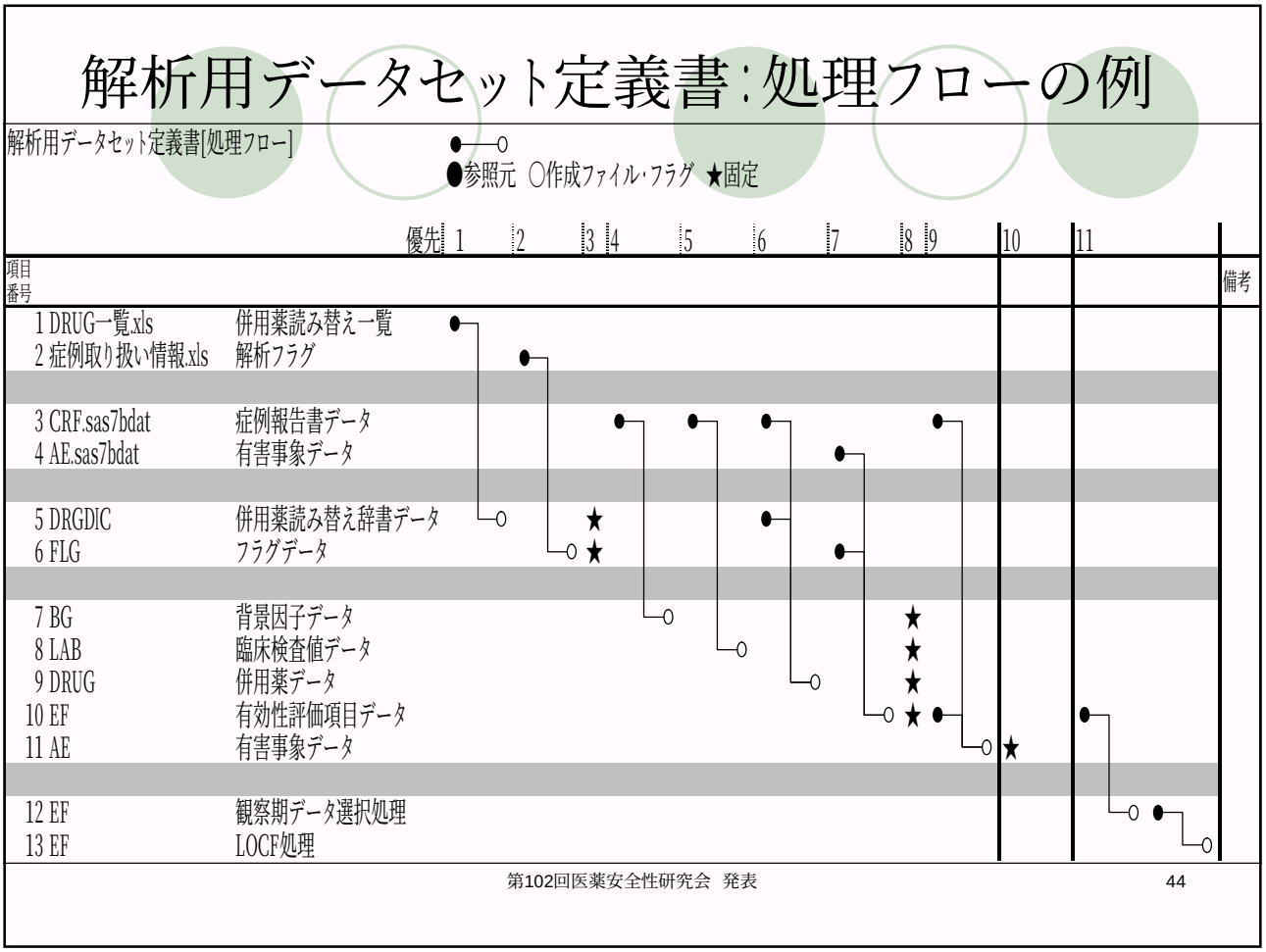
解析用データセット定義書：処理フロー

- 解析用DS同士で相互に参照して計算するようなとき、処理順がことなることで計算結果がかわってくることがある
- 変数定義だけではわからない処理順序を視覚的に指示するようなフロー図

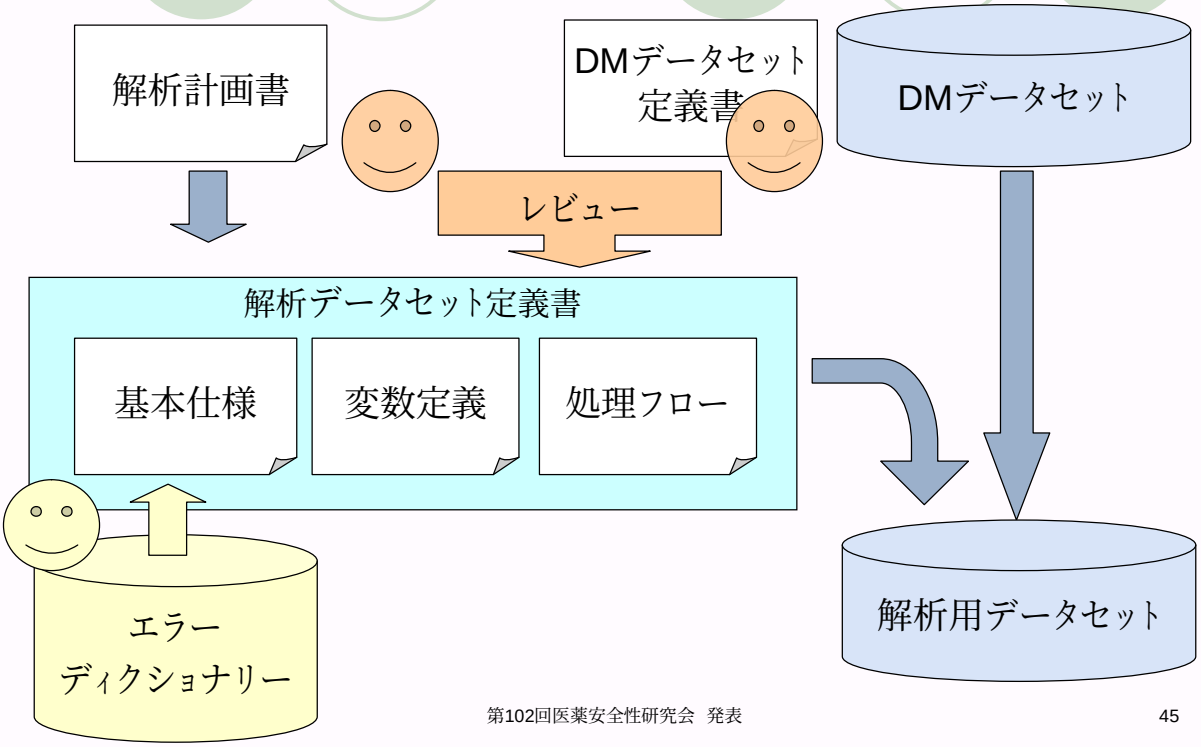
解析用データセット定義書：処理フロー



解析用データセット定義書：処理フローの例



解析用データセット作成のフロー



統計解析のバリデーションにおける最近の進歩

- ダブルプログラミングの実施が標準化
- 標準化、モジュール化の推進
- 文書管理システムなどの充実
- 解析用データセットの作成
- CDISCなど標準化、電子化への努力

5.まとめ

- ブラックボックステストの検証手段としてのダブルプログラミングは、バリデーションには不十分である
  - 独立照合エラー判定システム
  - エラーディクショナリー

- バリデーションへのキー
  - 設計を完全にするために
    レビュー→フィードバックを徹底すること･記録すること
  - 設計者･レビューアーのレベルを確保すること

ご静聴ありがとうございました