# バージョン管理システムの使い方 (発展篇)

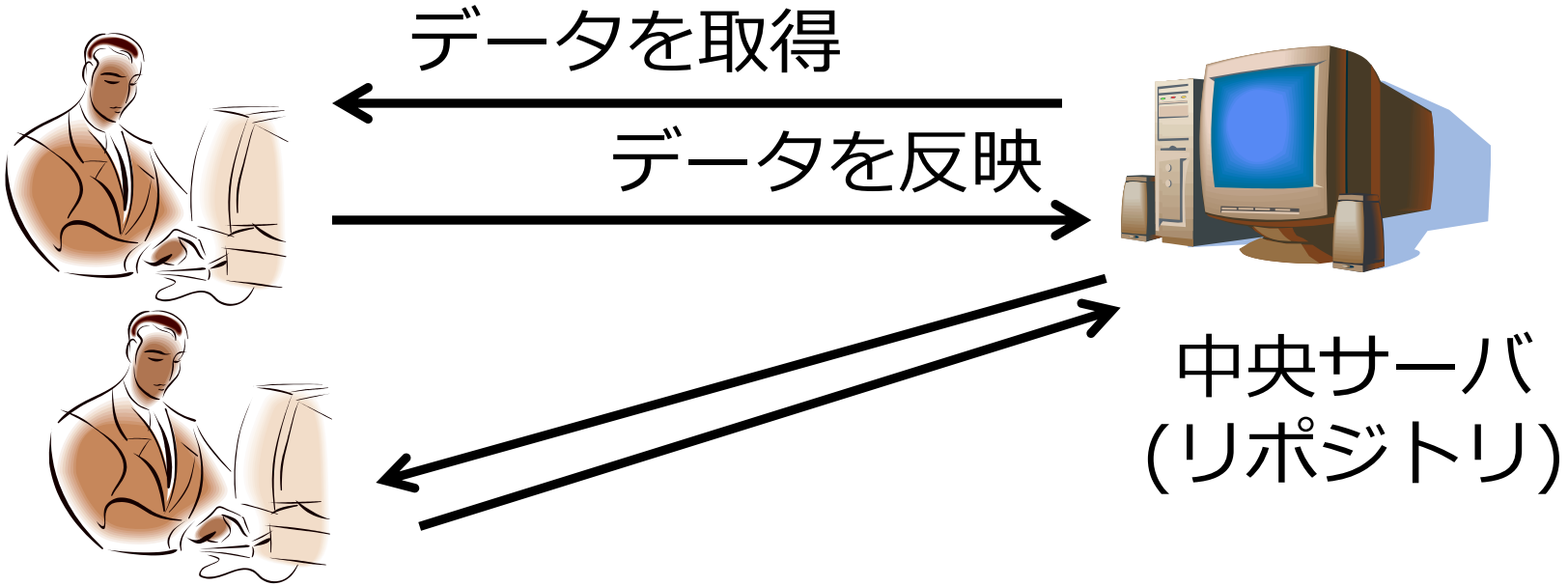
2016/3/25版　今泉 俊幸

# 目次

1. バージョン管理システムの仕組み(再掲)
2. 推奨リポジトリ構成
3. 各ディレクトリの使い分け
4. 段階的なリリースを行う開発例
5. リポジトリ構成例
6. マージとコンフリクト
7. リポジトリにコミットしないファイル
8. まとめ

# バージョン管理システムの仕組み(再掲)

- 中央サーバで変更履歴データを集中管理
- 各ユーザは中央サーバとデータをやり取り
  - 高機能なファイルサーバのようなイメージ



# 推奨リポジトリ構成

- プロジェクトごとにディレクトリを分け、その下にトランク(trunk),ブランチ(branches),タグ(tags)という特別な意味を持つディレクトリを作成する
- ディレクトリ例(BBKプロジェクト)
  - リポジトリルート
    - BBK
      - trunk
        - src
      - branches
      - tags

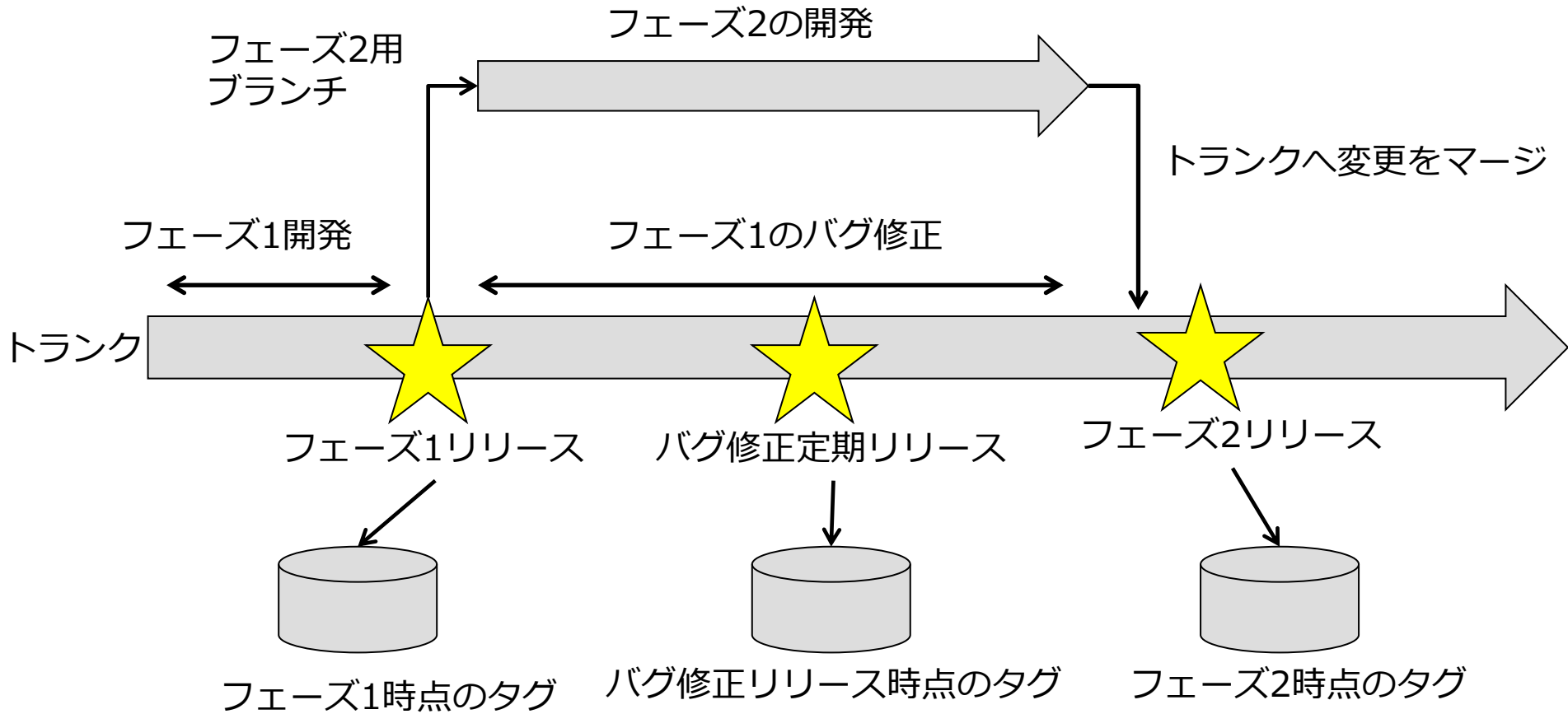
# 各ディレクトリの使い分け

- トランク(幹)
  - 開発の主となる場所
  - 基本はトランクにコミットしていく
- ブランチ(枝)
  - 補助的な開発場所
  - 段階的にリリースする場合などに使う
    - フェーズ1で入力系、フェーズ2で帳票系など
- タグ
  - 区切りのよいタイミングでプロジェクトの状態を保存しておくための編集しない場所
    - 例：フェーズ1のリリース時点のタグ、など

# 段階的なリリースを行う開発例 1/2

- リリース前：全員トランクで作業、トランクにコミットしていく
- フェーズ1リリース
  - フェーズ1リリース時点でタグを作成する
  - 以後、フェーズ1のバグ修正と、フェーズ2の開発の2軸で進める
    - トランクはフェーズ1のバグ修正用とし、バグ修正のみコミット
      - 定期的にリリースし、リリースの度にタグを作成しておく
    - タグ作成と同じタイミングでフェーズ2用のブランチを作成、フェーズ2の内容をコミットしていく
    - 緊急のバグ修正がある場合は直前のリリース時のタグからブランチを作成し、バグ修正をコミット、リリースする
      - トランクで未テストのバグ修正がコミットされていてもOK
- フェーズ2リリース
  - フェーズ2用ブランチからトランクへマージ(merge)する
  - フェーズ2リリース時点のタグを作成

# 段階的なリリースを行う開発例 2/2



# リポジトリ構成例

- リポジトリルート
  - BBK
    - trunk
      - src
    - branches
      - ph2 (フェーズ2用ブランチ)
    - tags
      - 20160301.0 (フェーズ1リリース時点)
      - 20160318.0 (定期バグ修正リリース時点)
      - 20160401.0 (フェーズ2リリース時点)

# マージとコンフリクト 1/3

- マージ機能を使うと、二つのブランチ間でソースコードをうまくまとめてくれる

ブランチ作成前のトランク

```
public class Hoge{
  public void piyo(){
      foo();
  }
}
```

最新のトランク

```
public class Hoge{
  public void piyo(){
      bar();
      foo();
  }
}
```

ブランチ

```
public class Hoge{
  public void piyo(){
      foo();
      baz();
  }
}
```

トランクにマージ

```
public class Hoge{
  public void piyo(){
      bar();
      foo();
      baz();
  }
}
```

# マージとコンフリクト 2/3

- 自動でマージできない場合、コンフリクト(conflict)となり手動でマージが必要となる

ブランチ作成前のトランク

```
public class Hoge{
  public void piyo(){
      foo();
  }
}
```

最新のトランク

```
public class Hoge{
  public void piyo(){
      bar();
      foo();
  }
}
```

ブランチ

```
public class Hoge{
  public void piyo(){
      baz();
      foo();
  }
}
```

- トランクにマージしようとすると、bar();baz();とすべきか、baz();bar();とすべきか自動では判別できない
- コンフリクトが発生する

# マージとコンフリクト 3/3

- コンフリクト発生時の例

```
public class Hoge{
  public void piyo(){
<<<<<<< .working
      bar();
=======
      baz();
>>>>>>> .merge-right.r122
      foo();
  }
}
```

- 上記のようにトランクとブランチの両方の変更が一時的に反映されるので、手動で修正する
  - merge.right.r112の112はブランチ側のコミットのリビジョン番号

# リポジトリにコミットしないファイル

- 自動で生成されるファイルは基本的にはリポジトリにコミットしない
  - 実行ログファイル
  - .classファイル
  - ビルド結果の実行用jarファイル
    - めったに変更されないツールなどであればコミットすることもある
- 自分の環境に依存したファイルはコミットしない
  - C:¥workspace など絶対パスが入った設定ファイルなど

# まとめ

- リポジトリにはトランク(trunk),ブランチ(branches),タグ(tags)という特別な意味を持つディレクトリがある
- 基本的にはトランクにコミットでOKだが、段階的なリリースではブランチ以下で作業することもある
  - プロジェクトリーダーの指示に従い、自分がどのブランチで作業すべきかを間違えないようにすること
- ブランチ間の変更はマージ機能で自動的に反映できるが、自動でできない場合はコンフリクトとなり、手動でマージが必要となる
- ビルド成果物など自動生成されるファイルはコミットしないこと