小さな組織が SAS Programのバージョン管理をするには. （Git, GitLab及びRedmineを用いた プログラム開発環境整備の検討）

吹谷 芳博
株式会社エスアールディ データマネジメント統計解析室

How to manage version control of SAS programs in small organizations; Consideration of environmental improvement for program development (Git, GitLab and Redmine).

Yoshihiro Fukiya
Data Management/Biostatistics, SRD Co., Ltd.

要旨:

SDTMやADaMの標準プログラムの重要性と共にプログラムのバージョン管理についても重要性が高まっている.
今回はGitLabとRedmineを用いたSASプログラム開発環境の構築について紹介する.

キーワード:Git, GitLab, Redmine, バージョン管理

目次

- 目的
- 背景
  - 開発ツールの紹介：Git, GitLab, Redmine
  - 構築ツールの紹介：Docker
- 方法
  - 構築方法
  - 事例
- 結果&考察
- まとめ

目次

- 目的
- 背景
  - 開発ツールの紹介：Git, GitLab, Redmine
  - 構築ツールの紹介：Docker
- 方法
  - 構築方法
  - 事例
- 結果&考察
- まとめ

目的

- プログラム開発の効率化には
  バージョン管理等の開発環境の整備が必要.
- 情報共有による開発チームの効率化も必要.

小さな組織では大規模なシステム導入が厳しい

一つの解決策の提案

1.導入が容易なバージョン管理システム

2.バージョン管理を使ったSASプログラム管理

目次

- 目的
- **背景**
  - **開発ツールの紹介：Git, GitLab, Redmine**
  - **構築ツールの紹介：Docker**
- 方法
  - 構築方法
  - 事例
- 結果&考察
- まとめ

背景(1)

新薬承認申請時のCDISC標準データ形式の
提出が義務化(2016/10/01～)で
プログラム開発の負担が増加.

SASプログラムの再利用化を進める必要あり.
1. システムの導入
2. プログラムの標準化とプロセスの整備

小さな組織では難しい.（コスト/リソース）

背景(2)

プログラムを標準化するときの問題点.

こんな事例ありませんか？？

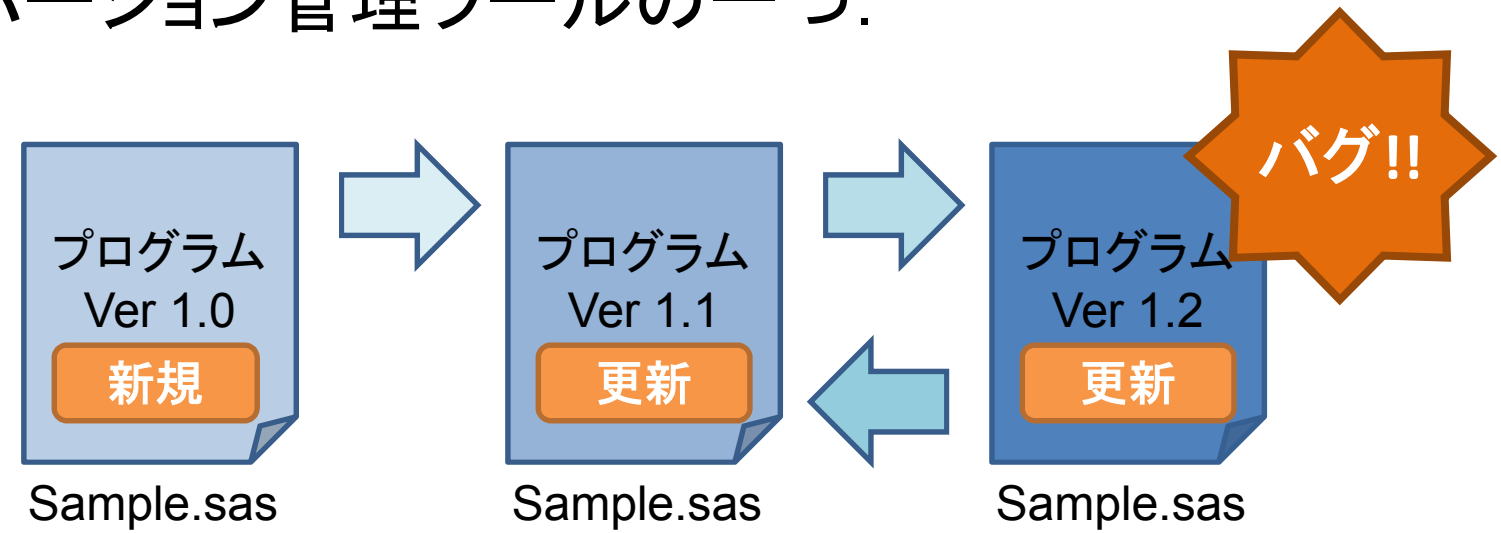
中央(共有フォルダ等)で
SAS macroを管理する場合(SASAUTOSを使用)

過去に
こんなミスも・・・・・

1. 誰かが勝手にUpdateしてしまっていた.
2. バグ情報が共有化されずにそのまま…
3. プログラム名にVersion情報を付けたが使いにくかった.
   etc…

導入(1)開発ツールの紹介:Git

- バージョン管理ツールの一つ.



- 利点:中央(サーバー)とローカルで別々にバージョン管理が可能.

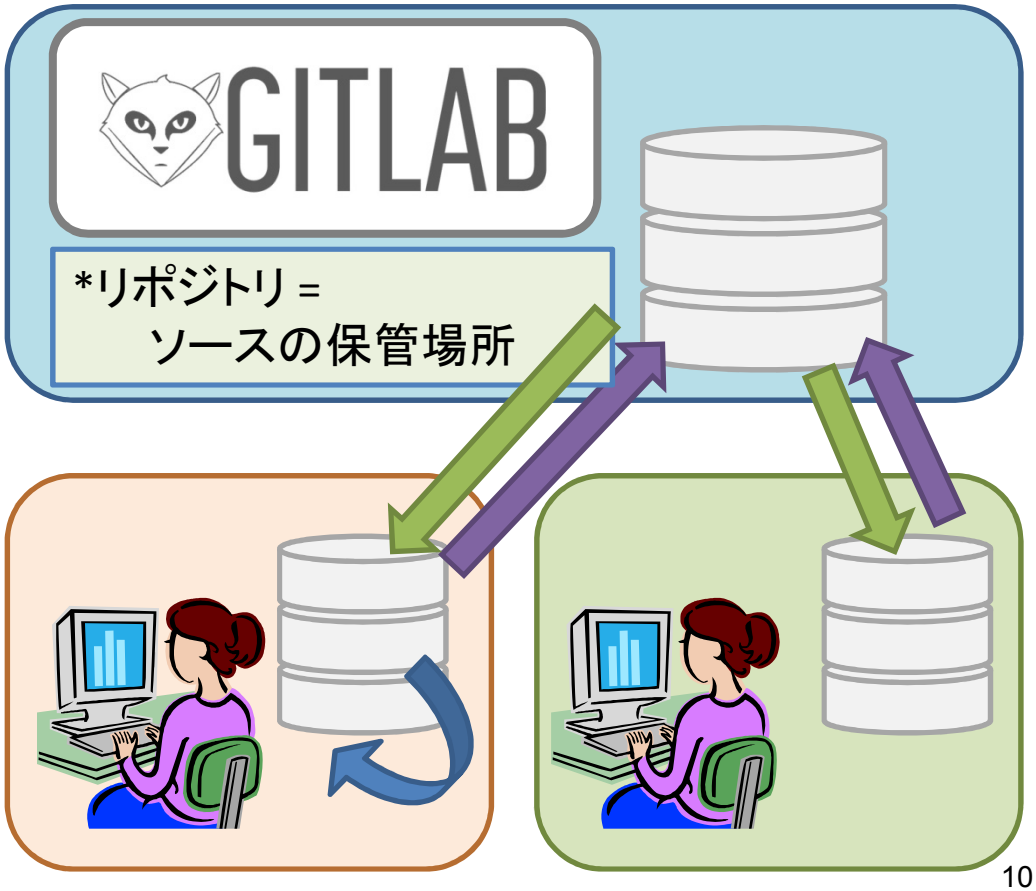
導入(2)開発ツールの紹介:GitLab

- 中央でバージョン管理できるツール

機能
- リポジトリ*管理
- アクセス管理
- ソースの比較
- Wiki

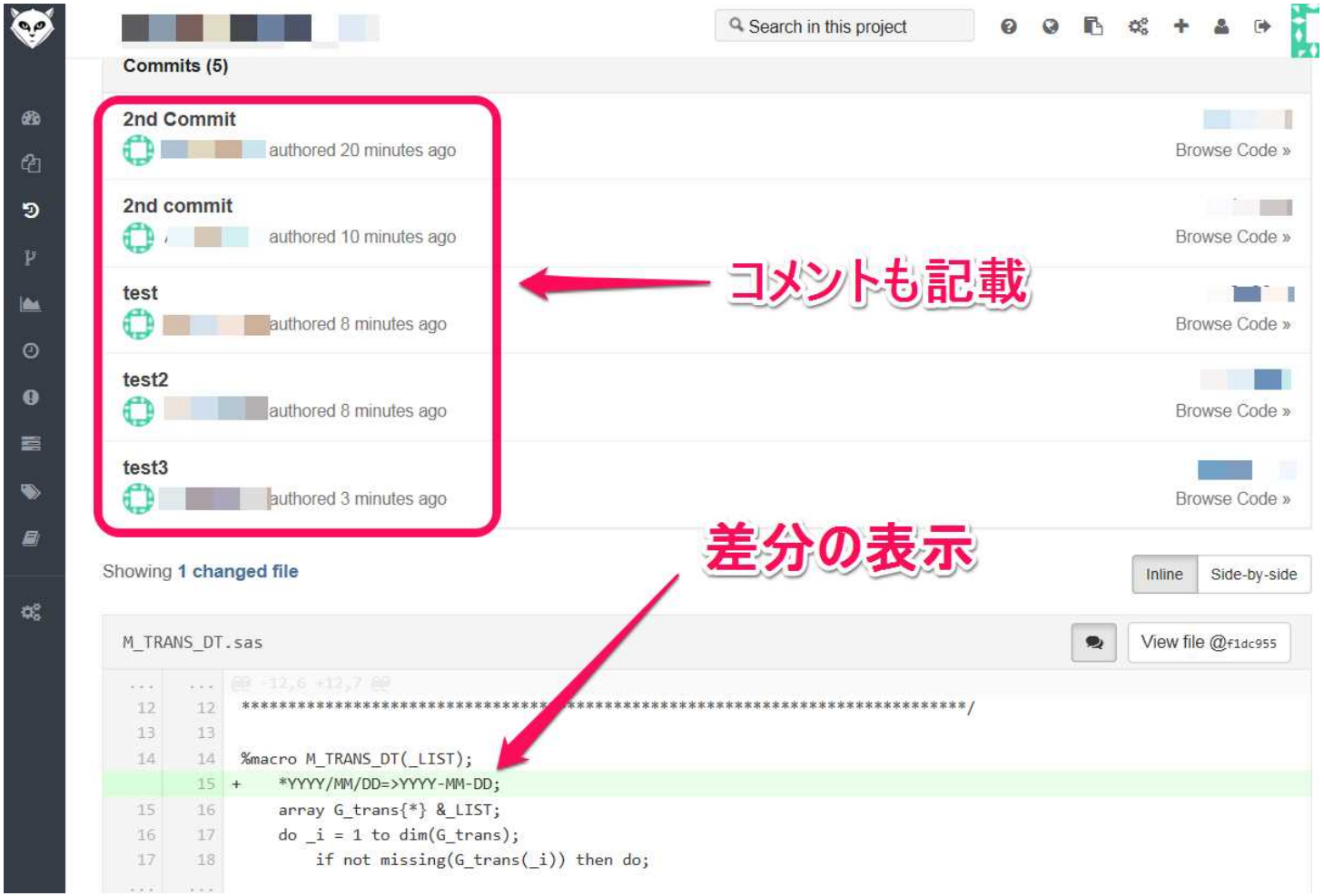
利点
リポジトリ管理が豊富.

欠点
カレンダーやガントチャートなどの可視化が苦手.



導入(2)開発ツールの紹介:GitLab

- 中央でバージョン管理できるツール(参考画面)



導入(3)開発ツールの紹介:Redmine

- タスク管理を可視化し，状況を把握するツール

機能
- タスク管理
- Wiki
- 文書管理
- リポジトリ管理

利点
タスクの状況が見やすい.

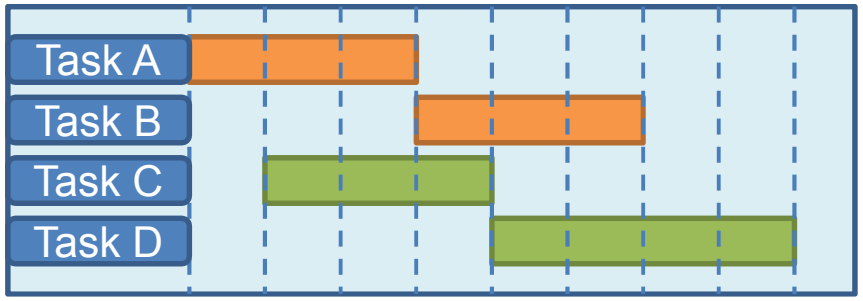
欠点
リポジトリ管理が貧弱.

REDMINE
flexible project management

バグの管理表

トラッカー	ステータス	更新日
バグ	進行中	2015/08/01
バグ	新規	2015/08/06

ガントチャート



導入(4) 構築ツールの紹介:Docker

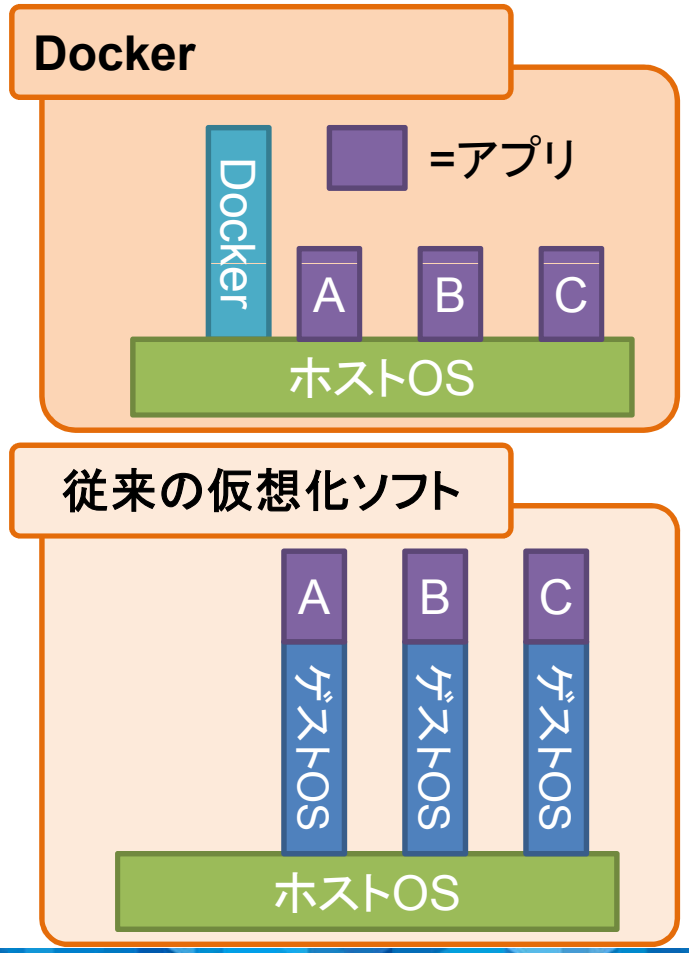
- コンテナ型の仮想化ソフトウェア

仮想化環境の利点

1. ホストPCに依存しにくい.
2. 環境をつくりやすい.

Dockerの利点

1. 軽量化.
2. 再利用しやすい.
3. Docker Hubから選んで使える.



目次

- 目的
- 背景
  - 開発ツールの紹介：Git, GitLab, Redmine
  - 構築ツールの紹介：Docker
- **方法**
  - **構築方法**
  - **事例**
- 結果&考察
- まとめ

モデル

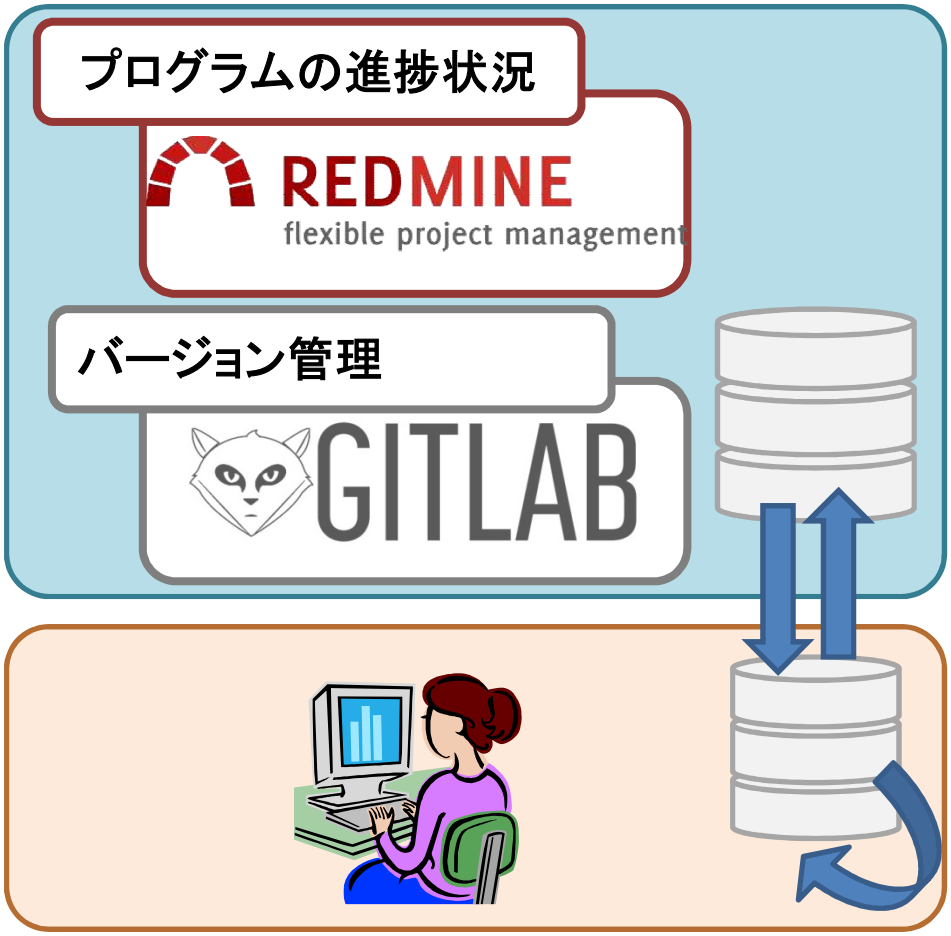
- リポジトリ管理が高機能で情報共有しやすいモデル

連携したことによる利点.

リポジトリ状況が可視化!!
管理も柔軟にできる!!

ただし注意点が・・・.

Gitが柔軟性が高いため,
ワークフローをきっちりしない
と連携しても管理が難しくなる.



構築方法(1) Dockerの導入

使用したPC環境
　　OS: CentOS 7

1. Dockerの導入

```
# yum -y update && yum -y install docker git
# systemctl enable docker
# systemctl start docker
```

※Linux以外の環境の場合
Mac OSXやWindows 7（64bit）or 8.1：
　　Boot2Dockerをインストールでが使用できる.
Windows 7 (32bit)：
　　Vagrantで仮想環境をつくり、その中にDockerをインストール.

構築方法(2) GitLabの導入

1. GitLab用のDBの構築(RDBS: PostgreSQL)

¥=バックスラッシュ

```
# docker run --name=postgresql-gitlab -d ¥
    --env='DB_NAME=gitlabhq_production‘ ¥
    --env='DB_USER=gitlab' --env='DB_PASS=password' ¥
    --volume=/srv/docker/gitlab/postgresql:/var/lib/postgresql ¥
    sameersbn/postgresql:9.4
```

2. GitLab用のDBの構築(NoSQL:Redis)

```
# docker run --name=redis-gitlab -d ¥
    --volume=/srv/docker/gitlab/redis:/var/lib/redis ¥
    sameersbn/redis:latest
```

3. GitLabの構築

```
# docker run --name='gitlab' -d ¥
    --link=postgresql-gitlab:postgresql --link=redis-gitlab:redisio ¥
    --publish=10022:22 --publish=10080:80 ¥
    --env='GITLAB_PORT=10080' --env='GITLAB_SSH_PORT=10022' ¥
    --volume=/srv/docker/gitlab/gitlab:/home/git/data ¥
    sameersbn/gitlab:7.11.4
```

構築方法(3) Redmineの導入

1. Redmine用のDBの構築(RDBS: PostgreSQL)

```
# docker run --name=postgresql-redmine –d ¥
    --env='DB_NAME=redmine_production' ¥
    --env='DB_USER=redmine' --env='DB_PASS=password' ¥
    --volume=/srv/docker/redmine/postgresql:/var/lib/postgresql ¥
    sameersbn/postgresql:9.4
```

¥=バックスラッシュ

2. Redmineの構築

```
# docker run --name=redmine -d ¥
    --link=postgresql-redmine:postgresql --publish=10083:80 ¥
    --env='REDMINE_PORT=10083' ¥
    --volume=/srv/docker/redmine/redmine:/home/redmine/data ¥
    --volumes-from=gitlab ¥
    sameersbn/redmine:3.0.3
```

GitLabと連携させるために必要

Redmine側での設定：Gitのリポジトリの指定.
GitLab側での設定　：Redmine issue trackerの有効.
RedmineとGitLabの両方で設定：Web hooksの設定.

事例（SDTM変換プログラム管理）

運用方法

Templateリポジトリ

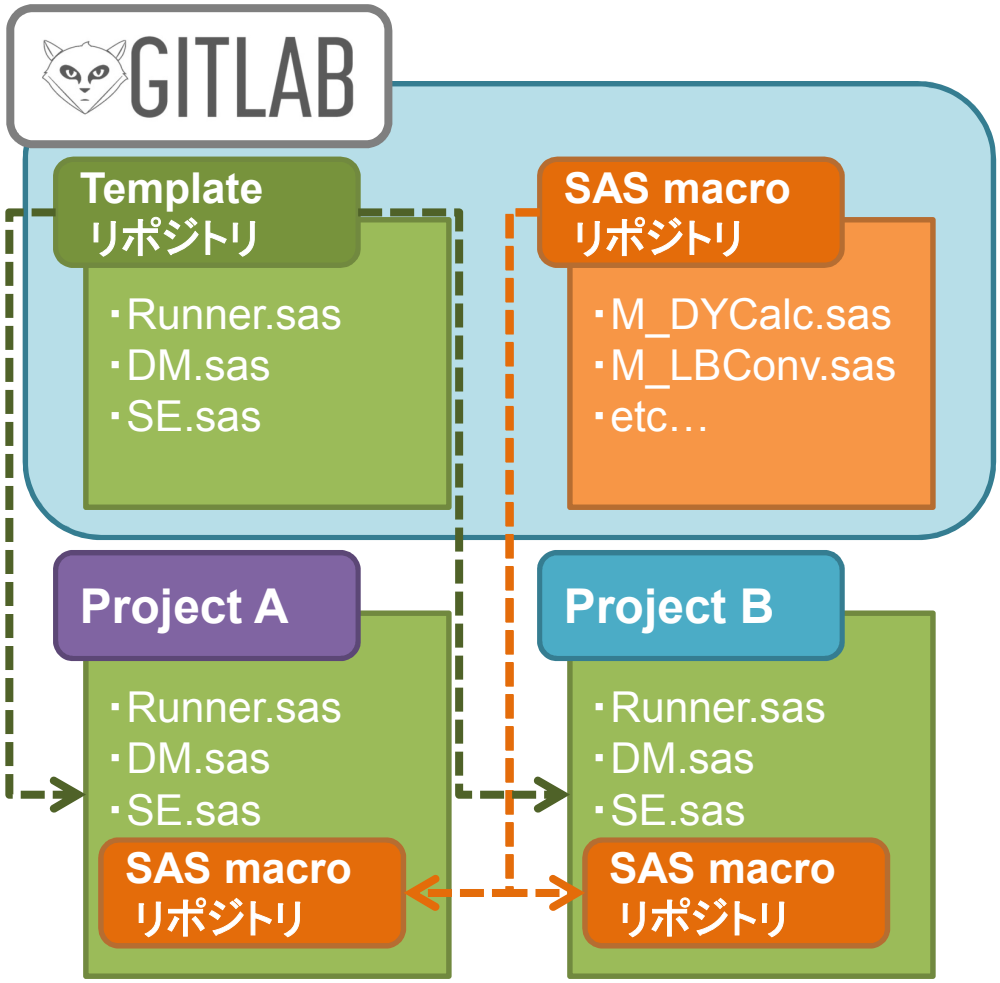
変換プロセスの枠組み.

SAS macro

日数算出や複雑な処理.
git subtreeで読み込む.

各Project

開発中：
ローカルでバージョン管理
Validation直前 or 後
リモートでバージョン管理



目次

- 目的
- 背景
  - 開発ツールの紹介：Git, GitLab, Redmine
  - 構築ツールの紹介：Docker
- 方法
  - 構築方法
  - 事例
- **結果&考察**
- まとめ

結果&考察

- 履歴管理の軽減
  - プログラム上でコメント等の記載不要.
- コミュニケーションロスの減少.
  - バージョン状況の認識.
  - バクの持ち込みの減少.

- 今後の検討：バイナリーファイル管理
  - 文書管理（PDF, Word, Excel）
  - Dataの管理（Dataset, Excel等）

目次

- 目的
- 背景
  - 開発ツールの紹介：Git, GitLab, Redmine
  - 構築ツールの紹介：Docker
- 方法
  - 構築方法
  - 事例
- 結果&考察
- まとめ

まとめ

- バージョン管理ツール導入は容易に可能.
- SAS macroの管理もツールを使えば軽減.

- SASプログラミングだけでなく
  開発環境の整備も重要な効率化の一つ!!
- SAS以外のプログラミングに目を向けて
  開発のアイディアを取り入れるのも重要.

参考

Dockerfileの場所

- Gitlab: https://github.com/sameersbn/docker-gitlab
- Redmine: https://github.com/sameersbn/docker-gitlab

GitLabとRedmine導入方法

- Dockerで使うサーバサイドソフトウェア（エンジニア編）
  http://knowledge.sakura.ad.jp/tech/2339/

GitLabとRedmineの連携方法について

- 少人数チームにおけるプロジェクト管理のベストプラクティス
  http://www.slideshare.net/cakeyoshida/ss-31147447
- RedmineとGitLabの連携。PushでチケットのStatusを変更
  http://se-suganuma.blogspot.jp/2015/05/redminegitlabpushstatus.html