# JaSST'21 Hokkaido

## Markdown+plantUMLで保守性の高いドキュメンテーションをしよう！

株式会社NTTデータ
技術革新統括本部 システム技術本部 生産技術部
ソフトウェア技術センタ
熊川 一平 イノベータ(ソフトウェアプロセス)

# 熊川 一平

- 株式会社NTTデータ
  - 主にソフトウェアテスト/品質保証のスペシャリストとして、R&Dや現場支援に従事
- 外部活動
  - JaSST Tokyoでは3度ベストスピーカーをいただきました
    - JaSST'14
    - JaSST'19
    - JaSST'21

# 本セッションのねらい

ソフトウェア開発にまつわるドキュメントを
ExcelやPowerpointなどで作って苦労しているあなたに
もっと効率的に気持ちよく作業できることを知ってもらう

# アイスブレイク

# Excel方眼紙あるある言いたい(10min)

- これからZoomのブレイクアウトルーム機能を使って、皆さんを3～4名ぐらいの小グループに分けます。
- 簡単な自己紹介の後、1人ずつ以下の話をしてください。
  - 参加の動機
  - ドキュメンテーションでこれまでに苦労したこと
  - Excel方眼紙への恨み、つらみ
- 実際のワークをする前に気持ちをアゲておきましょう

# 本ワークショップの流れ

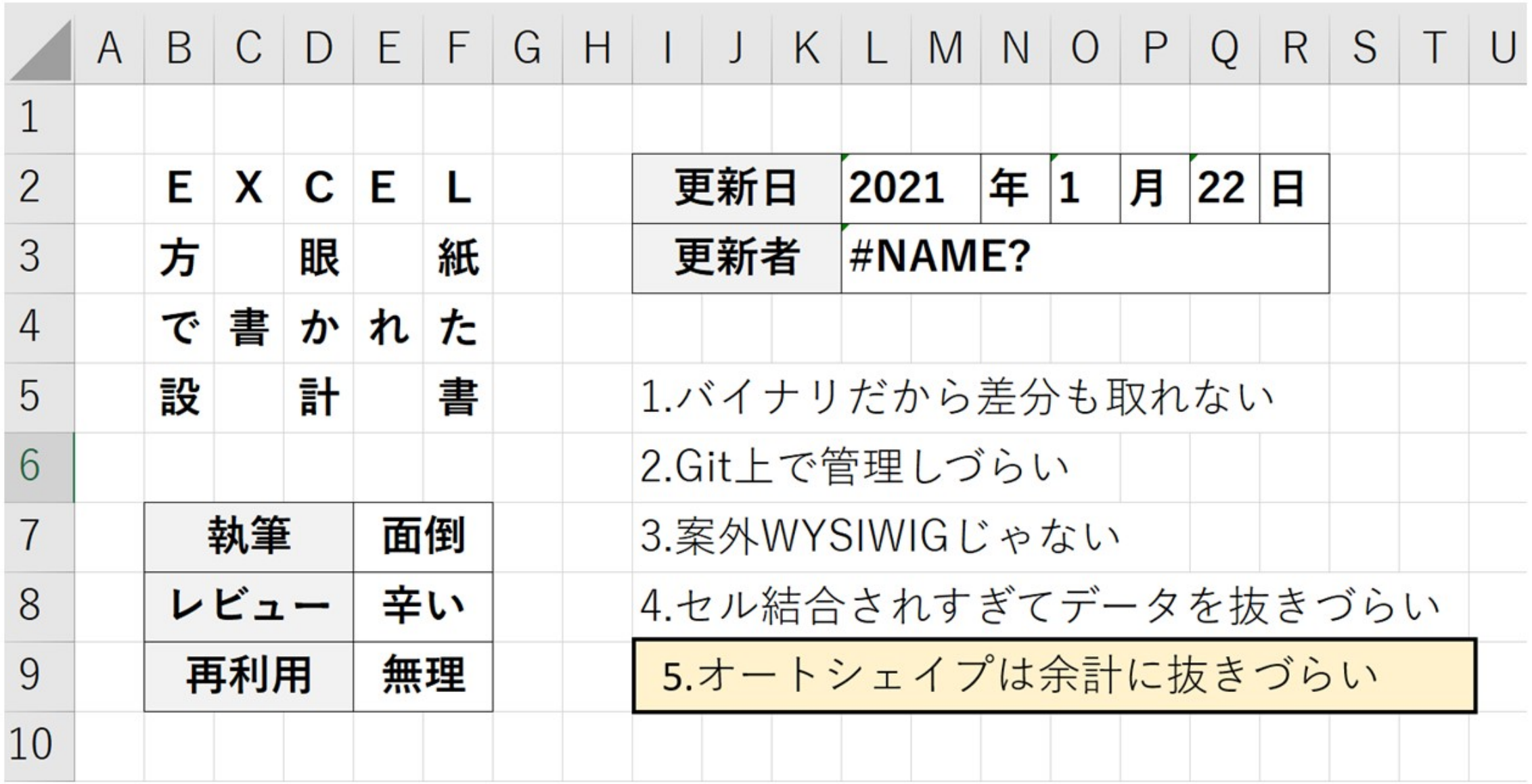
# スケジュール

| 時間 | やること |
| --- | --- |
| 15:15<br>～<br>15:40 | プレインテキストでドキュメンテーションを行う背景やメリットを理解しよう |
| 15:40<br>～<br>17:00 | 実際にドキュメントを書いて、ワークフローがどう変わるか体験してみましょう |
| 17:00<br>～<br>17:15 | FAQと振り返りを通じて学んだことを整理しましょう |

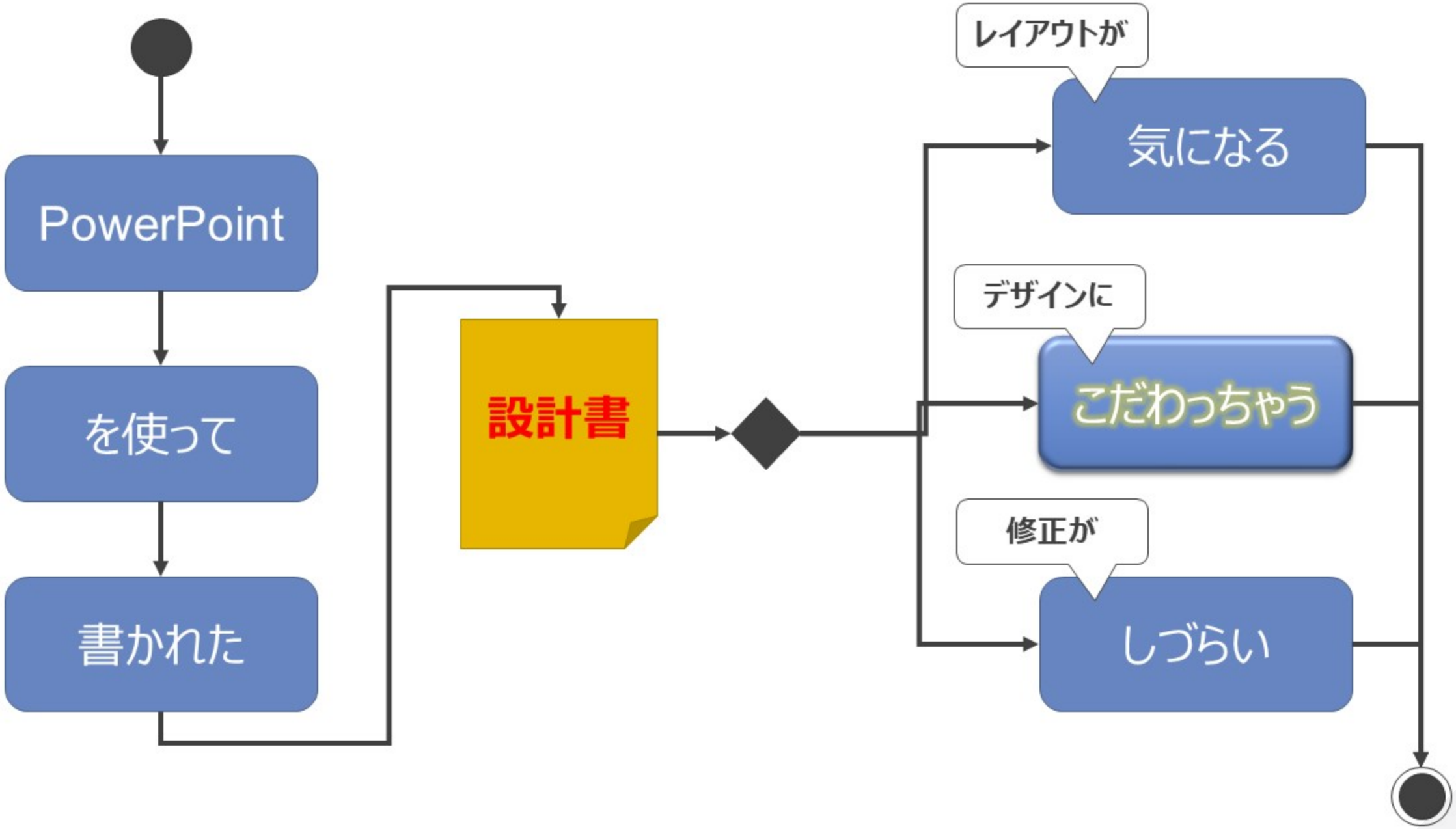
# 参加条件のおさらい

- ワークでは皆さんにドキュメンテーションを実践してもらいます
- 事前の案内に沿って環境構築はできていますよね？
  - vsCode + Markdown preview enhanced + java + graphviz
  - できていなければ、最初のお話の裏でやっておいてください

# Excel方眼紙などでドキュメンテーションする際の問題点




※WYSIWYG : What You See Is What You Get (見たままが得られる)


PowerPoint
を使って
書かれた
設計書
レイアウトが
気になる
デザインに
こだわっちゃう
修正が
しづらい

メンテされない大量の設計書

設計書_20200122版.xlsx.old

図を描くにはキャンバスが狭い

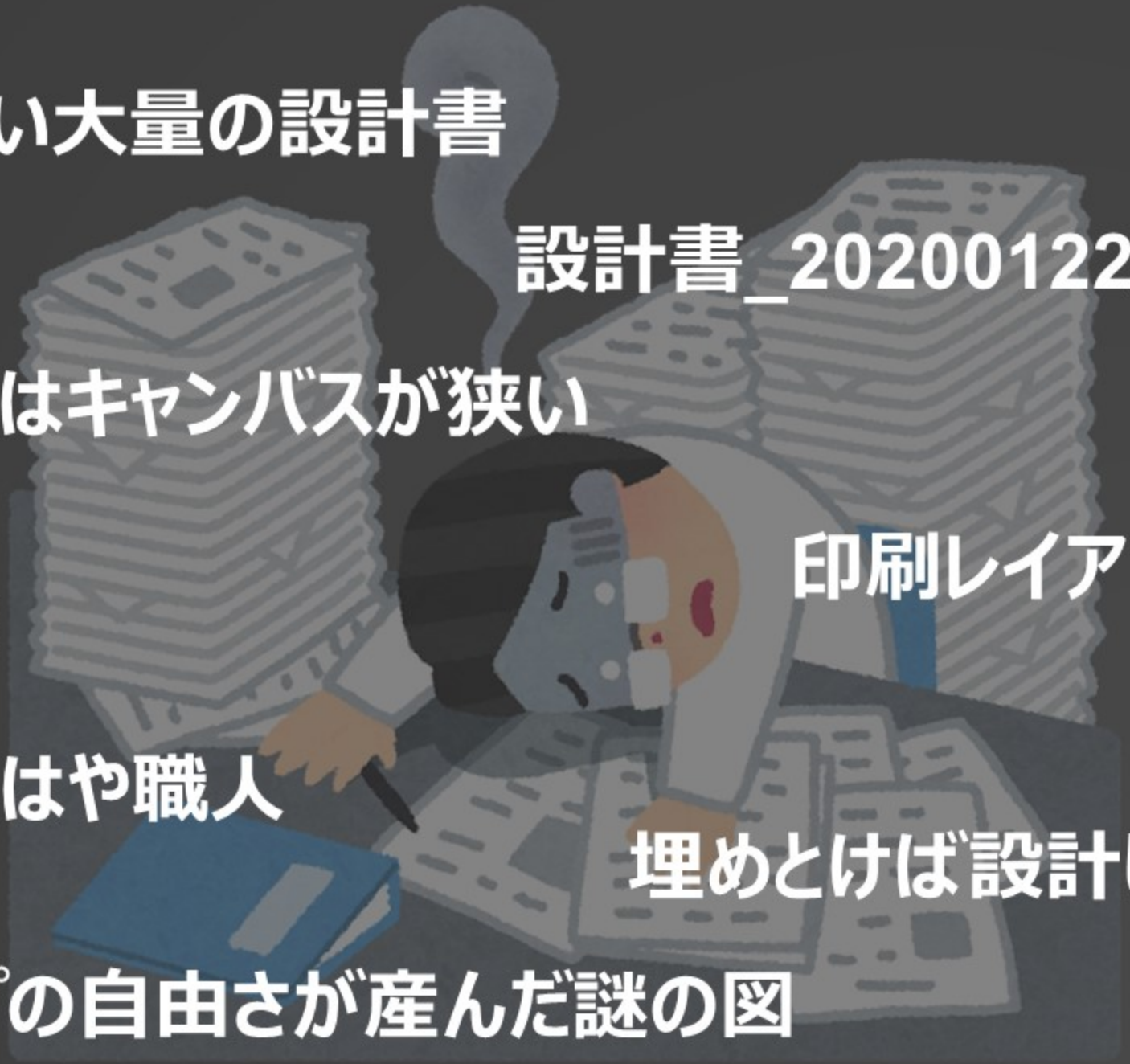
印刷レイアウトの罠

修正の衝突

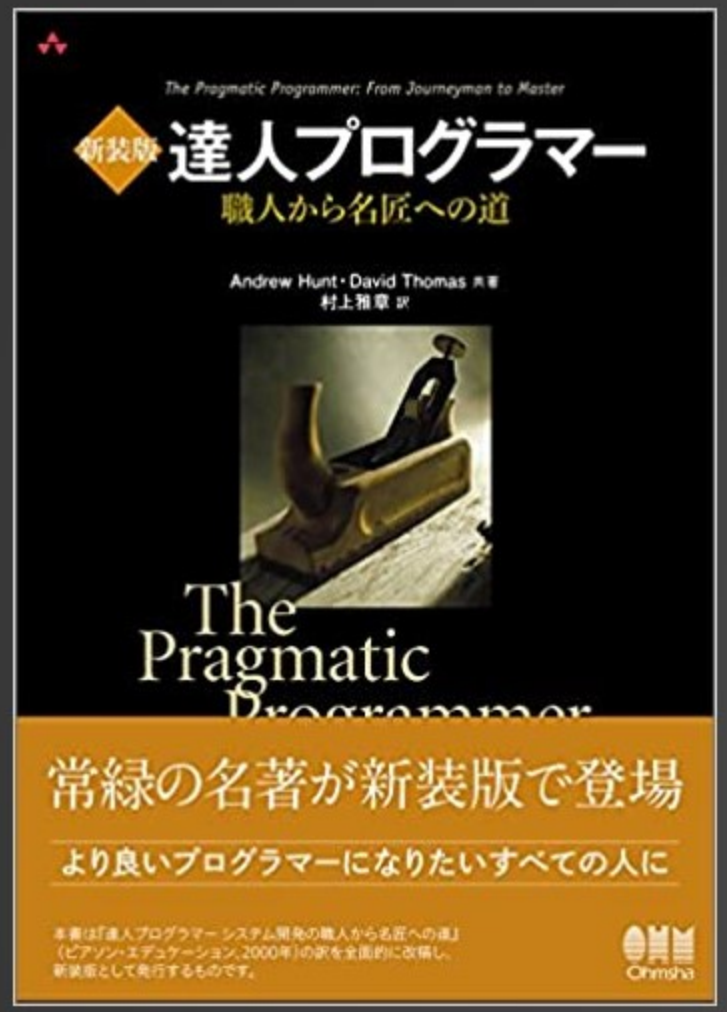
もはや職人

埋めとけば設計した気になる

オートシェイプの自由さが産んだ謎の図



# プレインテキストで
# ドキュメンテーションしよう




# 達人プログラマー 職人から名匠への道

Andrew Hunt (著), David Thomas (著), 村上雅章 (翻訳)

## 第3章 プレイン・テキストの威力

- 達人プログラマーが取り扱う情報は、知識です。その知識を永続的に格納するためのフォーマットで最も適しているものが、プレインテキストです。
- プレイン・テキストのメリットは、透明性が保証される、様々な活用ができる、テストが用意になる。
- ソースコード管理システムは巨大な*UNDO*キー。すべてをソースコード管理システムで管理すること。それがソースコードでなくても。
- 実効可能ドキュメント、ドキュメントからコードを生成する。ドキュメントとコードの２つをメンテナンスすることは*DRY*原則に反する。プレインテキストならば、スクリプト言語によって加工が可能。

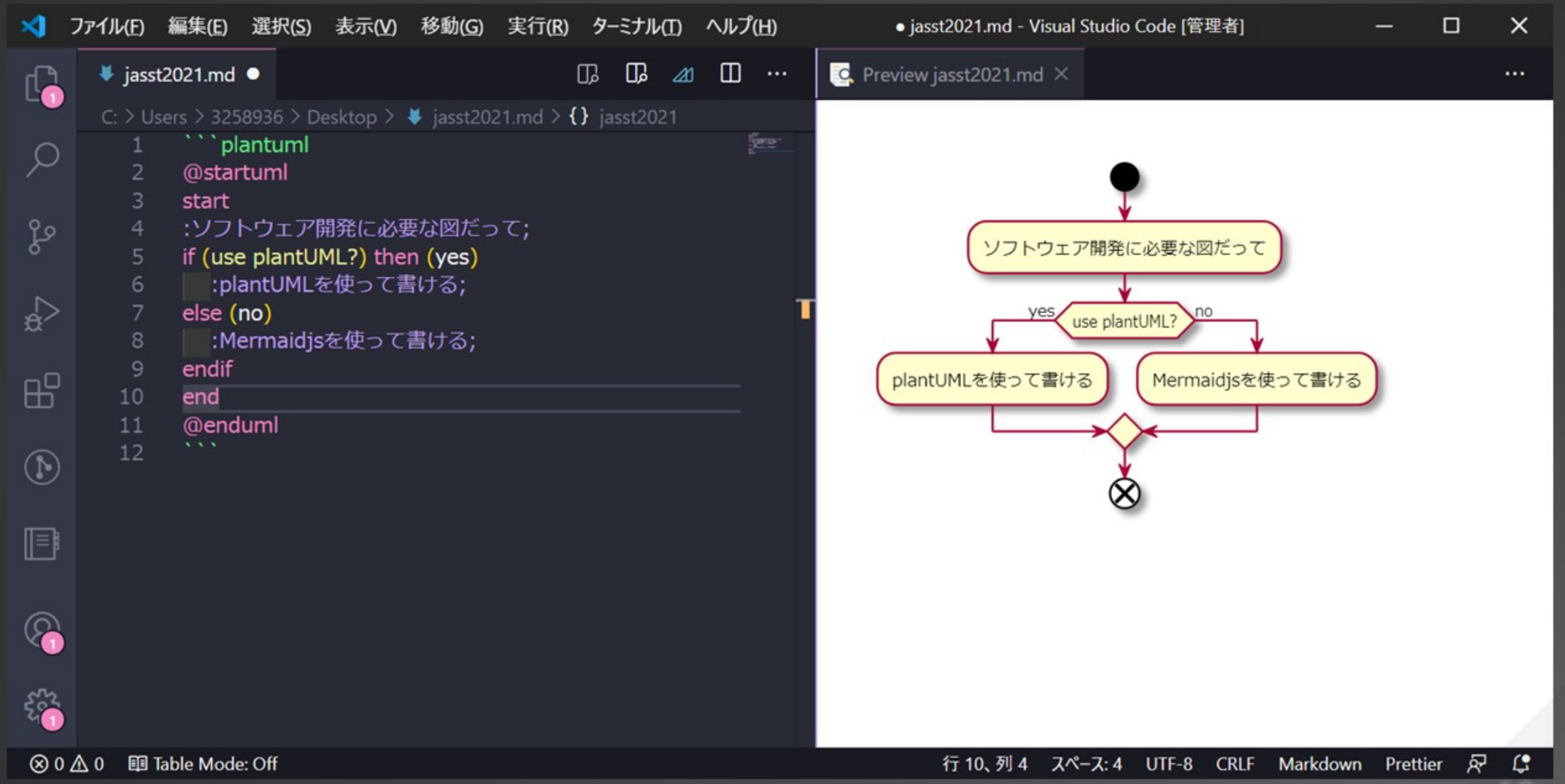
# Markdown

- プレーンテキスト形式の軽量マークアップ言語
- 見出し、段落、改行、箇条書き、太字、引用、表、etc...
  文書作成に必要な装飾は一通りそろっています。
- Githubなど、多くのサービスで利用可能
- 統一仕様としてCommonMarkがあるが、絵文字などの
  追加機能をもった方言が数多く存在する。
  (e.g.GithubFlavoredMarkdown)
- 似たものにAsciiDocなどが存在
- markdownを活用したソリューションも多数存在する。
  実はこのスライドもmarkdownで作られています。


ファイル(F) 編集(E) 選択(S) 表示(V) 移動(G) 実行(R) ターミナル(T) ヘルプ(H)
● jasst2021.md - Visual Studio Code [管理者]
jasst2021.md ●
C: > Users > 3258936 > Desktop > jasst2021.md > # MarkdownやAsciiDocを使えば
1 # MarkdownやAsciiDocを使えば
2
3 **文書執筆に必要な装飾は大体できる**
4
5 * 箇条書きも
6 * 簡単だし
7 * ネストも掘れる
8
9 |表にすることも |可能です |
10 |---|---|
11 |執筆 |わずらわしいことはない |
12 |レビュー |テキスト差分をとって簡単に |
13 |再利用 |テキストだから検索も構造化も簡単 |
14
Preview jasst2021.md
MarkdownやAsciiDocを使えば
文書執筆に必要な装飾は大体できる
箇条書きも
簡単だし
ネストも掘れる
表にすることも
可能です
執筆
わずらわしいことはない
レビュー
テキスト差分をとって簡単に
再利用
テキストだから検索も構造化も簡単
0 0 Table Mode: Off
行 9、列 50 スペース: 4 UTF-8 CRLF Markdown Prettier

# PlantUML

- オープンソースのテキストでUML図形を書けるツール
- テキスト形式で動作も軽量。
- UMLだけでなく様々な図形に対応。
- 描写にはGraphvizを使用。
- .puファイルとして独立して作成する他、Markdownの中に埋め込むことも可能。
- 最近はDiagram as a Codeなどと呼ばれることもあり、周辺ツールなどの開発も活況。



# プレインテキストを使ったドキュメンテーションのメリット

- Gitで管理できる
  - テキストだからソースと同じようにGitで管理できる。
- ワークフローを作れる
  - プルリクエストを通じて必ず承認行為が伴う
- 差分が見れる
  - プルリクエストで差分を見ながらレビューもできる。
- 検索できる
  - 全文検索やGrepで図内のテキストも含めて検索可能。
- 修正が容易
  - キャンバスの大きさやセルの形を意識しなくてよい。

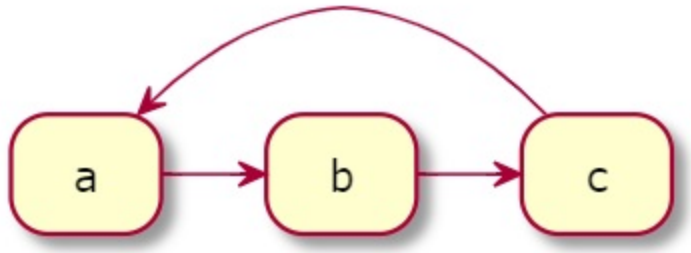
# 実際に手を動かして効果を実感していきましょう

- 書き方はこのあたりを参考に
- Markdownの仕様
  - https://commonmark.org/help/
  - https://shd101wyy.github.io/markdown-preview-enhanced/#/
- plantUMLの仕様や例
  - https://plantuml.com/ja/
  - https://real-world-plantuml.com/

# plantUMLをmarkdownに埋め込むには？

markdownの中で以下のように記述すればよいです。

````
```plantuml
@startuml
hide empty description
state a
state b
state c
 a -> b
 b -> c
 c -> a
@enduml
```
````



# ワークに関する事前説明

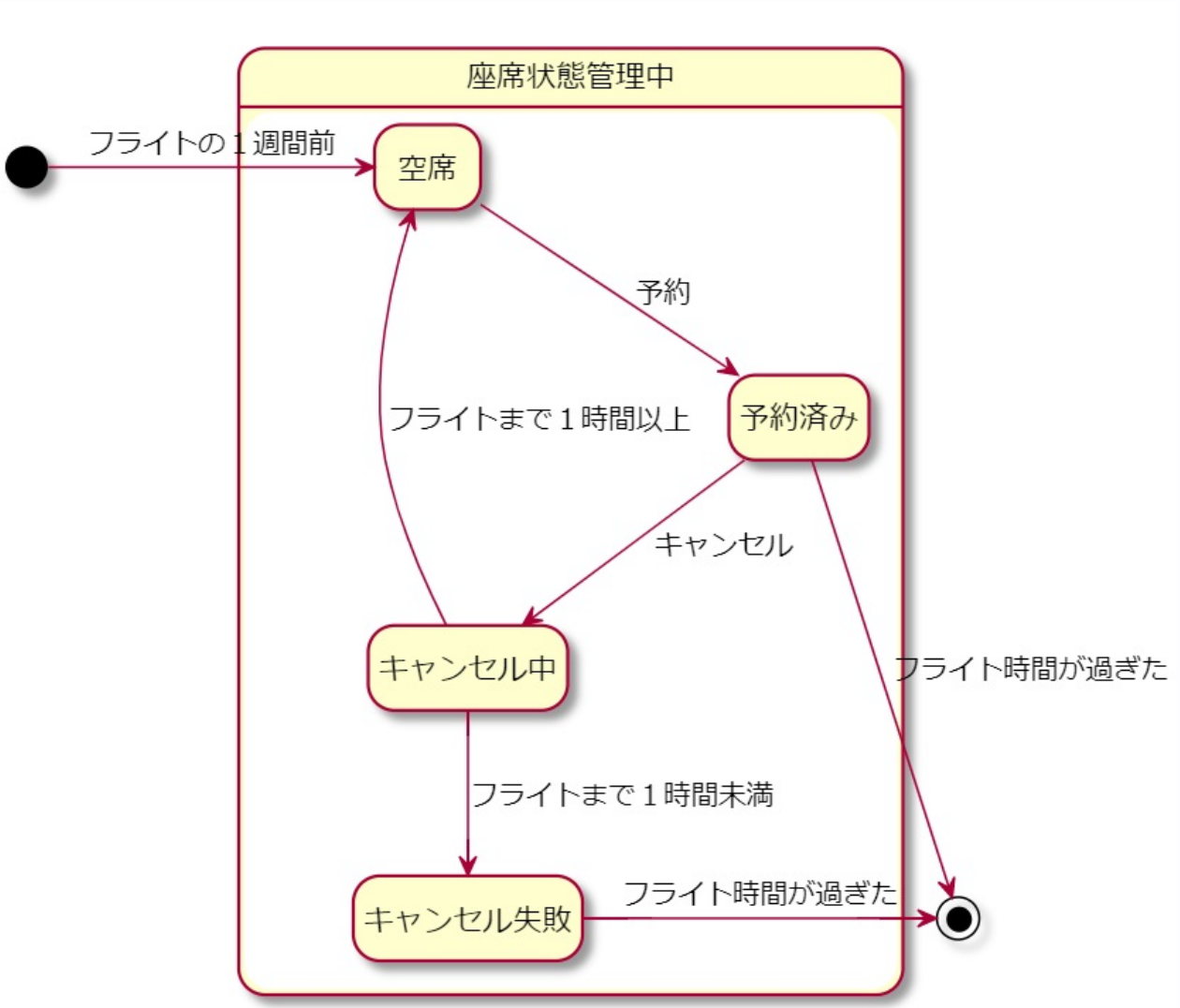
- ワークに必要なファイルなどはdiscordで配布します
- 各ワークでは、まずお題に沿って個々人/グループで取り組みをしてもらいます
- 取り組みの後、何名かの方に感想などをお伺いしつつ、このワークを通じて伝えたかったことを共有します。
- 上記が完了した後、回答例を配布します。

# ワークその１

作業時間 15min
情報共有 5min

# 状態遷移図を書いてみよう

- 「work1.md」には状態遷移に関する仕様について、ディスカッションした記録が書いてあります。
  - 記録を読んで、plantUMLを使って状態遷移図を書いてみましょう。
  - １つ１つの座席の状態がどのように遷移していくかを表しましょう。




# 回答例

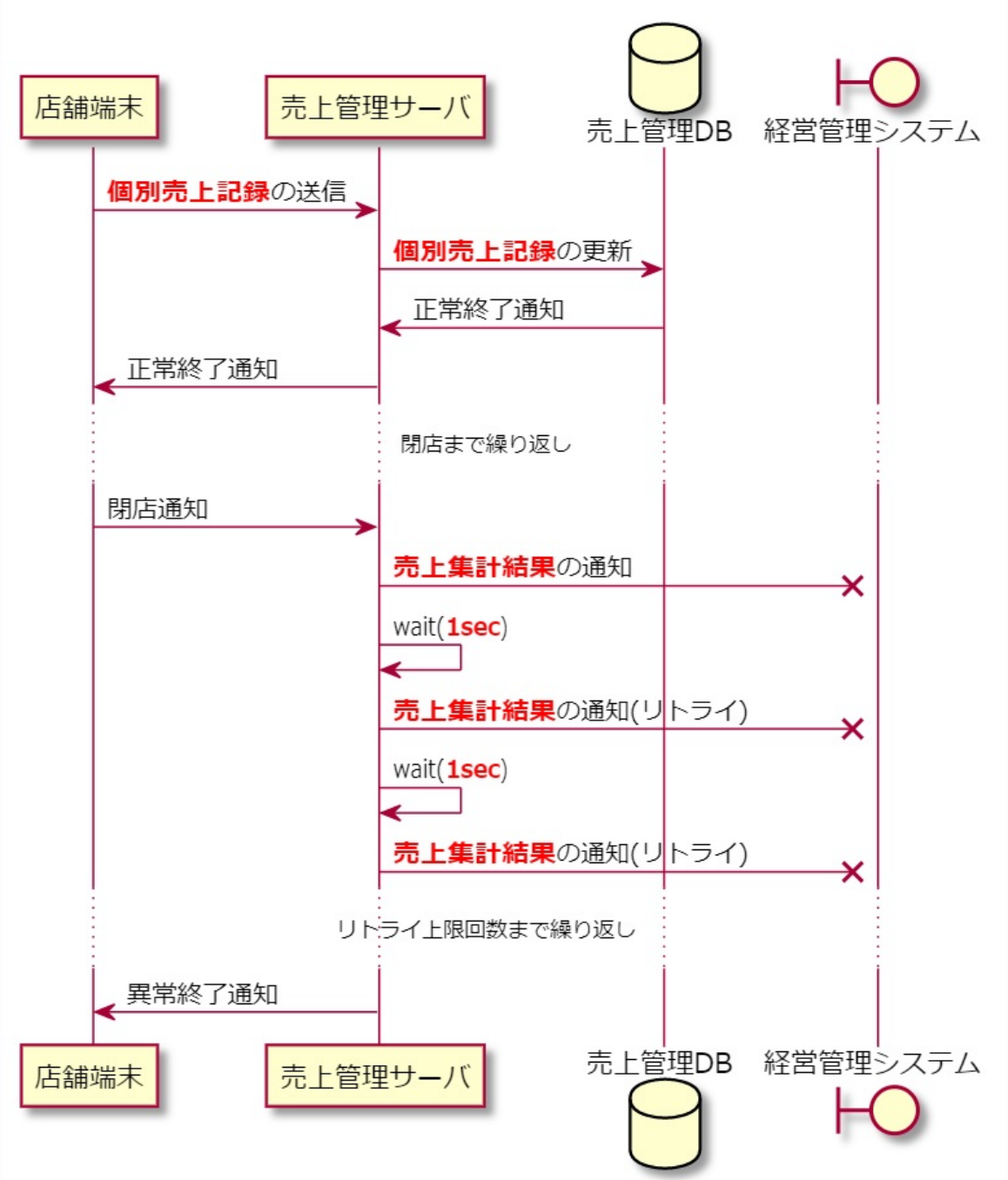
- この通りである必要はない
- あまり見た目にこだわらない

# ワークその２

作業時間 10min
情報共有 5min

# シーケンス図を修正してみよう

- 「work2_uml.md」にとあるアプリケーションの通信仕様が記載されています。
- 「work2.md」には通信シーケンスに関する仕様について、ディスカッションした記録が書いてあります。
- 記録を読んで、シーケンス図を修正してみましょう。




# 回答例

- この通りである必要はない
- 検索・置換を行うとすぐに修正できたはず

# ワークその３

作業時間 10min
情報共有 5min

# ワーク２の結果をレビューしよう

- 皆様を2～3名のブレイクアウトルームに分けます。
- ルーム内の参加者から代表者を1名決めてください。
- 代表者の方は、ワーク2の結果について、修正前後のdiffをとって画面共有しましょう。(念のため修正前ファイルも配布します。)
- 代表者以外の人は、修正が抜け漏れなく実施されているか、diffの結果を見てレビューしてみてください。
- 時間が余ったら、こうしたレビューのスタイルに関するメリットやデメリットについて参加者で話し合ってみましょう。

# ワークその４

作業時間 25min
情報共有 5min

# レビューコメントへの対応をしましょう

- github上に修正作業中の資材があります。
  - URLなどは演習フォルダに配置しております
- どうやら仕様変更が発生していて、plantUMLの修正がされているようですが、仕様の穴が見つかっていて、指摘を受けていますね。
  - 指摘コメントはgithubにログインしないと見れません。
- あなたが引き継いで修正をしてあげてください。
- 実際にpushするところまではしなくてよいです。（ローカルで修正するまででよいです）

# 回答例

別紙(work4_uml_example.md)参照
ちょっと考えてみよう

- これがパワポの狭いキャンバスに書かれていたら？
- Github上でのコメントの管理について

# まとめと振り返り

# プレインテキストを使ったドキュメンテーションのメリット

- Gitで管理できる
  - テキストだからソースと同じようにGitで管理できる。
- ワークフローを作れる
  - プルリクエストを通じて必ず承認行為が伴う
- 差分が見れる
  - プルリクエストで差分を見ながらレビューもできる。
- 検索できる
  - 全文検索やGrepで図内のテキストも含めて検索可能。
- 修正が容易
  - キャンバスの大きさやセルの形を意識しなくてよい。

# 今日学んだこと

- markdownやplantUMLは怖くない
- テキスト化することに多くのメリットがある

# 明日からやること

- とりあえず何でもExcelを開く癖があるならやめる
- 文書も資産なのでちゃんとGitで管理する
- 本ワークショップで得たことを布教する

# Question?