

Sphinx事例紹介 ～Sierの場合～

2015/10/31
JUS Sphinxワークショップ@
関西

自己紹介

- Twitter: @kk_Ataka
- GitHub: gosyujin
- Sler勤務
 - Sphinxはプライベートで利用しつつ、仕事にも組み込めないか試行錯誤中



アジェンダ

I. Sphinx導入前

- 導入のための政治活動

2. Sphinx導入中、導入した後

- 導入するにあたっての壁
- 「納品」するには

3. Sphinx導入事例

Sphinx導入前

はじめに

- Sphinx(reST)の知名度調査
 - 知名度0！
- Markdownもあんまり
- Wikiはかるうじて多少知られている

まずは政治から

競合ツールと比較！

→ 競合ツールと比較しての良さを調査

1. Office(Word, Excel)

2. Wiki, Markdown

※ MarkdownはJekyllやHugoなどMarkdownで記述できる静的サイトジェネレータを想定しています

比較1

Office(Word, Excel)

Office 長所 Word, Excel共通

- SI界のスタンダード
- WYSIWYGな操作
 - きめ細かいデザインが可能
 - 図やフローの挿入が容易

Office 長所 特にWord

→ ものすごく複雑な箇条書きが簡単(?)に作れる

1.1. 設計方針

1.2. 開発スケジュール

1.2.1. 要件定義 // => ネストしていく

1.2.1.1. xx機能 // => さらにネスト...

1.2.1.2. yy機能 // => 色々書いて...

1.2.2. 基本設計 // => ネストからの復帰

1.3. 開発体制 // => さらに復帰

Office 長所 特にExcel

- エグい表/テーブルが簡単(?)に作れる
 - 連結、結合たくさんあるマトリクスのようなものとか
 - Excel方眼紙フォーマットで自由自在(泣)
- 値の計算が簡単
 - これは表計算ソフトExcelの独壇場
 - 表計算の用途にExcelを使うのは賛成

Office 短所 Word, Excel共通

- diffが取るのがメンドくさい
 - 最近は取れるっぽい
- 往々にして日付でバージョン管理される事が多い
 - VCSと相性悪い
- 議事録_20140505_2(最新)(xx修正).xls

Office 短所 Word, Excel共通

- 検索しづらい
 - 違うシート／吹出し／非表示とか
- ミリ単位のレイアウト修正を強いられる
 - 大事なのは内容... そうでしょ！
- 職人芸が発揮されるほど重い
 - IGBオーバーのファイル...

比較2

Wiki, Markdown

Wiki, Markdownの長所

→ Officeの短所は解消できている！...と思う

Wiki, Markdownの長所

"diffが取るのがメンドくさい"

- Markdownはプレーンテキストなので簡単
 - バージョン管理もしやすい
- Wikiもだいたい差分表示機能あり
 - diff取りやすい

Wiki, Markdownの長所

"検索しづらい"

- Officeよりは探しやすいと思うのだが...
- ブラウザ、エディタの検索機能とか、Wiki内検索とかを駆使して

Wiki, Markdownの長所

"ミリ単位のレイアウト修正"

→ 出力先(html+cssなど)である程度統一できる

Wiki, Markdownの短所

- Officeでは特に意識していなかったことを考慮する必要あり

Wiki, Markdownの短所

- 記法を覚える必要がある
 - 未経験者に敷居が高い...
 - 図やフローは基本的にタグで挿入
- 「特定部分のみ」のレイアウト修正が面倒
 - 独自の処理を入れる？面倒...

Wiki, Markdownの短所

- しっかり作らないと情報が散らかる
 - いわゆるTips集になってしまふ
- 方言が多い(特にMarkdown)
 - PHP Markdown Extra, GitHub Flavored Markdown etc...
 - パーサが異なると出力結果が変わってしまう
 - 逆に考えると表現方法が増えるため長所になり得る
- 出力はhtmlを想定しているものが多い

そしてSphinx

Sphinxの長所

- Wiki, Markdownの長所と短所はだいたい
Sphinxにも当てはまる
- Sphinxが強いところは...

Sphinxの長所

- 体系的なドキュメント の骨組みを簡単に整えられる
- これだけで導入する価値あり
- 章の組み換え等も簡単
 - Wikiとかでこれを整備するのはちょっとしんどい
 - Office(Word)はちょっと得意かも
- 実際に作ってみると実感がわきづらいと思う

lightweight-markup-language-history 1.0 ドキュメント »

目次

[軽量マークアップ言語の歴史](#)
[TODOリスト](#)
[Indices and tables](#)

次のトピックへ

[1. はじめに](#)

このページ

[ソースコードを表示](#)

クイック検索

モジュール、クラス、または関数名を入力してください

軽量マークアップ言語の歴史

目次

- [1. はじめに](#)
 - [1.1. 本書の目的とゴール](#)
 - [1.2. 注意事項](#)
 - [1.3. マークアップ言語](#)
 - [1.4. 軽量マークアップ言語](#)
 - [1.5. オレオレマークアップ](#)
 - [1.6. 2015年、存在感のある流派](#)
- [2. reStructuredTextの章](#)
 - [2.1. setext](#)
 - [2.2. StructuredText](#)
 - [2.3. reStructuredText](#)
- [3. Markdownの章](#)
 - [3.1. Markdown](#)
 - [3.2. CommonMark](#)
- [4. Wiki\(WikiWikiWeb\)の章](#)
 - [4.1. WikiWikiWeb](#)
 - [4.2. YukiWiki](#)
 - [4.3. MediaWiki](#)
- [5. その他の言語の章](#)
 - [5.1. Textile](#)
 - [5.2. はてな記法](#)
- [6. まとめ](#)
 - [6.1. 軽量マークアップ言語の年表](#)
 - [6.2. 終わりに](#)
 - [6.3. 本書の執筆環境](#)
- [付録: 参考](#)
 - [参考文献](#)
 - [参考サイト](#)
- [付録: 用語集](#)

Sphinxの長所

- 複数ページをまたぐのも得意
 - 索引ページ、用語集ページの作成も簡単
- html以外にも出力形式が豊富
 - text, pdf, epub, etc...

索引ページ

→ こんな感じ

索引

[A](#) | [B](#) | [C](#) | [D](#) | [G](#) | [H](#) | [I](#) | [J](#) | [M](#) | [N](#) | [P](#) | [R](#) | [S](#) | [T](#) | [V](#) | [W](#) | [Y](#) | [記号](#)

A

[Asciidoc](#)

B

[Benjamin Dumke-von der Ehe](#)

C

[CommonMark](#)

D

[David Goodger](#)

[David Greenspan](#)

[Dean Allen](#)

[Doxygen](#)

G

[GFM](#)

[GitHub Flavored Markdown](#)

H

[HTML](#)

用語集ページ

→ こんな感じ

付録: 用語集

- 用語一覧

用語一覧

Asciidoc

[Stuart Rackham](#) によって2002/11/25([AsciiDoc ChangeLog](#))に作成された。Human Readableを目指した言語で、Pro Git 第2版は[Markdown](#) から全面的に[Asciidoc](#)へ書きなおされています。[テクニカルライティングの将来 — GitHub上のAsciidocで技術書Pro Gitを協働執筆 | 開発手法・プロジェクト管理 | POSTD](#)

Benjamin Dumke-von der Ehe

[CommonMark](#) 提唱者の一人。pagedown作者。Stack Overflow所属。

CommonMark

[John MacFarlane](#) を中心に立ち上げした[Markdown](#)共通化プロジェクト。2015年、[Markdown](#)の後継となるべく仕様策定中。

David Goodger

[reST](#) の作者。

David Greenspan

[CommonMark](#) 提唱者の一人。EtherPad、Meteor作者。

Dean Allen

[Textile](#) 作者。

Doxxygen

課題
Doxxygenの一口メモ書く

Sphinx導入中
と
導入した後

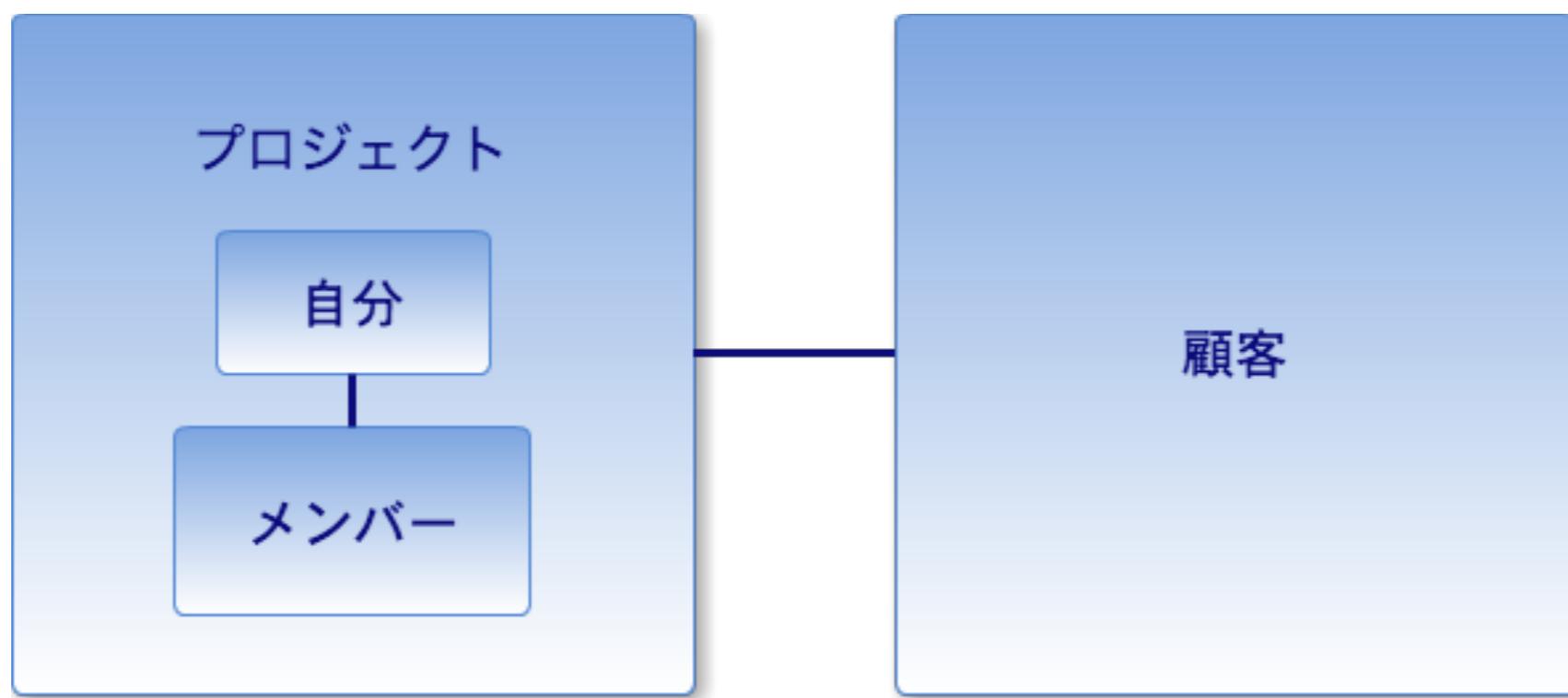
導入するにあたって の壁

I. 対、プロジェクトのメンバー(PM)に対して

→ 布教

2. 対、顧客に対して

→ 納品



壁1. 対PM

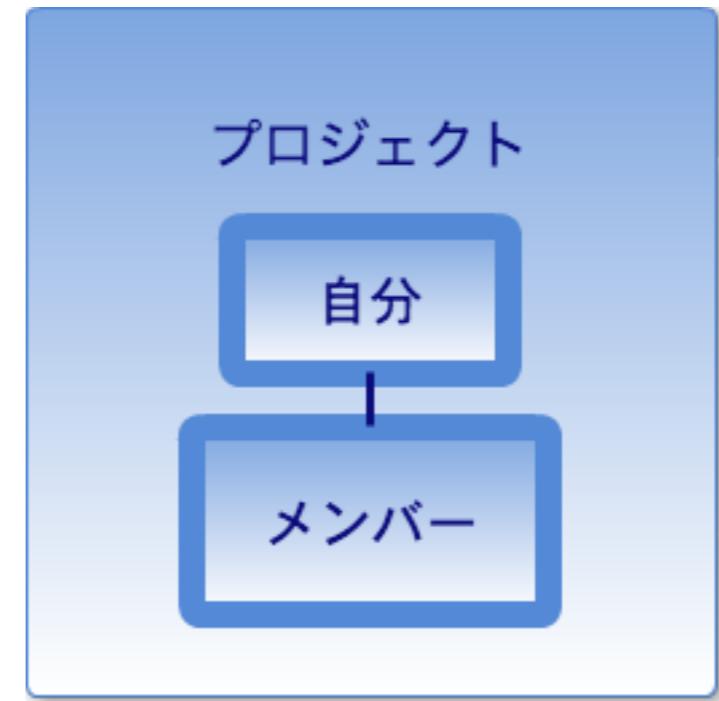
対PM 登場人物

I. 自分

- **Sphinx**を導入したい人、基本的にはなんでもやる

2. プロジェクトのメンバー (PM)

- 導入した**Sphinx**を使ってほしい人



対PM 「自分」の仕事

- 「ドキュメントはreSTで作る」 明確な宣言
 - 一番大事
 - これがうまく周知されてないと負の成果物が生成される...

対PM 「自分」 の仕事

- メンバーのサポート
- 「ドキュメント作成するだけ」 環境を作る
 - 1. `sphinx-quickstart`で下準備
 - 2. ドキュメント 자체のアウトライン作成
 - 3. `doctree`の作成

などなど

対PM 「自分」の仕事

- ビルド環境、デプロイ環境などもお膳立て
 - ビルドはJenkinsなどで拾う
 - デプロイはWebサーバにhtmlファイル配備とかがお手軽
- commitすれば成果物が生成される事を周知

対PM 「メンバー」 の仕事

→ reST記法を覚えてもらう

負担を減らす

対PM 課題

- ローカルPCでのプレビューができない
 - メンバーのPCにSphinxを入れてもらうのは厳しい...
- 確認できるのがcommitした時のみになってしまう
- ローカルで簡単にreSTプレビューできない
 - GitHubとか使えばある程度できるんだけど

対PM 課題

- プロジェクトの風土にあわせたカスタマイズが必要な場合も
 - こういうレイアウトがいい
 - こういうヘッダフッタがほしい
 - 社風にあわせて

壁2. 対顧客

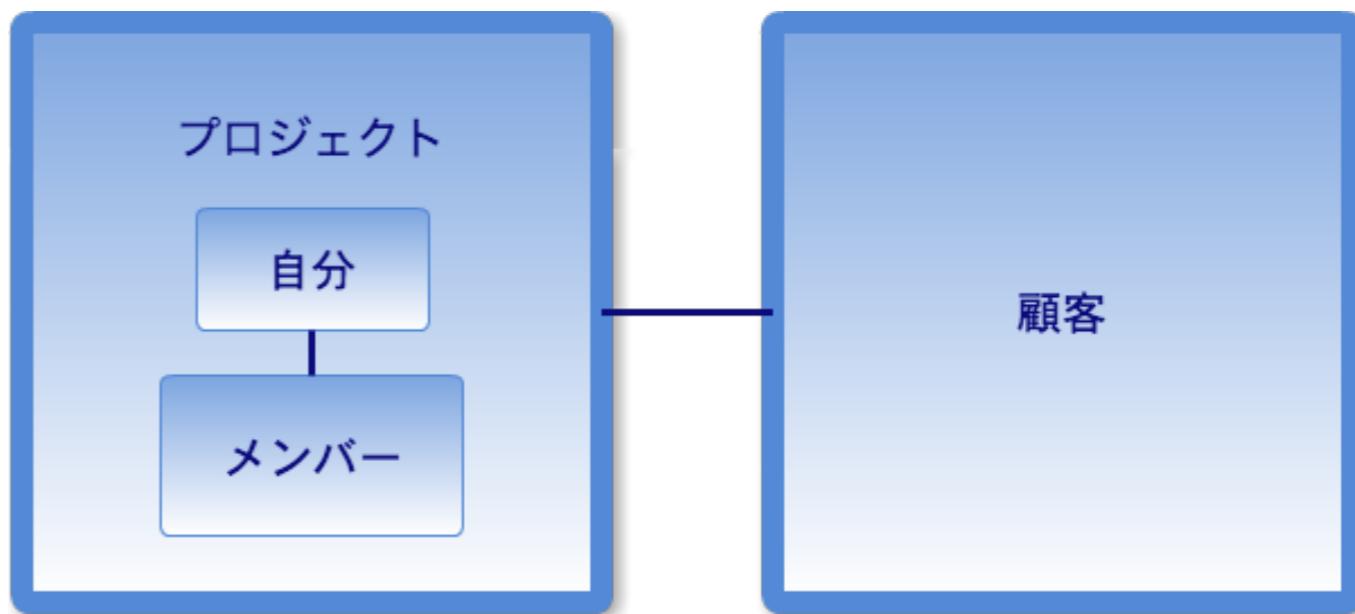
対顧客 登場人物

1. プロジェクト

- **Sphinx**でドキュメント納品する側

2. 顧客

- ドキュメントを納品される側
- 社内の人 or 社外の人
- 歴史的経緯から**Office**で納品される事が多い
- 例外は**Javadoc**とか？



対顧客 「プロジェクト」 側 の仕事

- 顧客に対して宣言&合意を得る
 - 「今回はOfficeじゃない形式で設計書書きますよ」

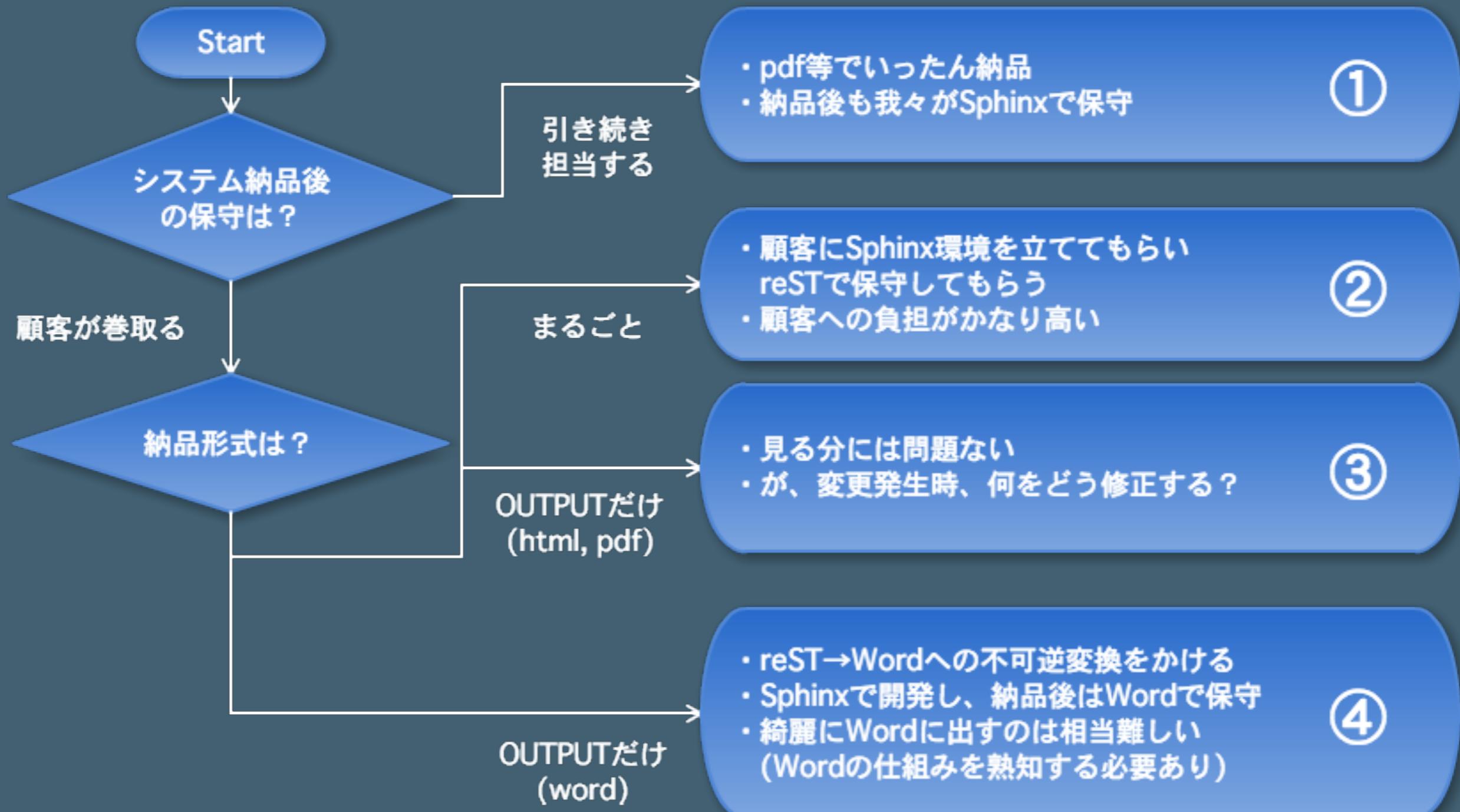
対顧客 「顧客」 側の仕事

→ 特になし？心構えくらい？

対顧客 課題

- 納品形式(次ページ参照)
- 納品後 「誰が」 保守するのか
- 顧客が巻取ってしまう場合、どう保守すればよいのか
- 要解決事項
- これが解決できないと導入できない

「納品」
するには…



自分たちで
保守できないと
厳しい…

Sphinx導入事例

環境

- 3ヶ月位のプロジェクトで **Sphinx** を適用
- ほぼ1人で設計、製造、テストを担当
 - 途中で1人サポートに入ってもらった
- ドキュメント作成環境は **Sphinx + Git(VCS) + Jenkins(Build)**
- ターゲットは「社内資料」
- 顧客へ納品しない資料

導入した感想

1. **Git**でバージョン管理、テキストなので差分管理も簡単！
2. 内容に集中できる！
3. お目当ての章が見やすい！探しやすい！
4. **Output**は、意外と営業の人を受けが良かつた！
→ 「これなんてツール？あとで教えて」

まとめ

1. 「Sphinxで書いていく！」 空気を作るのが難しい
 - reSTで文章を書いてもらうのも難しい
 - 「仲間」を作ろう！
2. 「納品」するには課題もある
 - お客様も巻き込もう！