派生開発問題解決セミナー2019

派生開発プロセスの基本と発展

～守りから攻めの派生開発へ～

AFFORDD 派生開発推進協議会

© 2019 派生開発推進協議会

「設計」の重要性

- ソースコードの元となった「考え」＝「設計」が崩れる
  → 処理とデータのまとまりがなくなる
  →ソフトウェアの品質低下に歯止めが利かなくなる
- 変更**要求**仕様書に注視しがちだが、設計の良し悪しは変更作業の効率や品質に大きな影響を与える
  - 流用元の設計品質が良くないと変更作業に負荷がかかるだけでなくモレなどの**新たな不具合の原因**になる
  - さらに、新たに加えた変更の仕方が悪いと、その後の開発は**指数関数的に**効率と品質の低下を招くことになる
- 本セッションでは
  ①派生開発プロセスの基本と事例
  ②守りのスペックアウト
  ③攻めのアーキテクチャ再構築法　　を共有する

© 2019 派生開発推進協議会

時間割

時間		内容	講演者
基本編			
14:00～14:15	15分	AFFORDDの活動とXDDPの成り立ち	南部 妙水
14:15～14:35	20分	PFD、USDM、XDDPの基本	
14:35～15:00	25分	XDDP導入による ソフトウェアプロセスの改善	笹田 朋邦
発展編			
15:05～15:30	25分	スペックアウトによる設計仕様の理解	井貝 智行
15:30～15:55	25分	アーキテクチャ再構築法(簡易分析法)	池田 祐一
15:55～16:00	5分	AFFORDDからのお知らせ	

© 2019 派生開発推進協議会

AFFORDDの活動とXDDPの成り立ち

派生開発カンファレンス2019　プログラム委員長
アンリツエンジニアリング株式会社
南部 妙水

© 2019 派生開発推進協議会

派生開発推進協議会
(AFFORDD)
の活動

© 2019 派生開発推進協議会

設立の背景

- 日本のソフトウェア開発の現状
  - 派生開発によるシステム開発
    - 組込み系、パッケージソフト、制御系ソフト、エンタプライズ系
    - 大規模化したシステムでの派生開発は困難を極めている
  - 現場の技術者
    - 派生開発の混乱に振り回され疲弊している
    - 劣化したコードと格闘しながら、なんとかリリースしている
    - 精神的肉体的ダメージから倒れる技術者も… (社会問題)
  - 企業のビジネス展開
    - どこかで製品やシステムがリリースできなくなる、大幅に遅れる可能性
    - 競争上避けて通れない設計改善の取り組みにも支障をきたしている
    - 国内、世界でのビジネス競争では QCD の同時達成は必至

派生開発の問題を解決しなければ
日本のソフト産業や製造業に明日は見えてこない

© 2019 派生開発推進協議会

設立の主旨

- 派生開発の問題を解決する“XDDP”
  - 派生開発にマッチしたプロセスにより、QCDを大幅に改善できる
  - 手に入れた時間で “次” の準備に取りかかる
    - 設計改善、リファクタリング、新規開発に必要な教育、設計手法の習得 等
- 日本を取り巻く状況
  - 日本経済は飽和状態
    - 雇用、経済の活性化のためにも製品やシステムの輸出が重要
    - グローバル経済の中で成長著しい海外企業との熾烈な競争

QCDの同時達成が不可欠

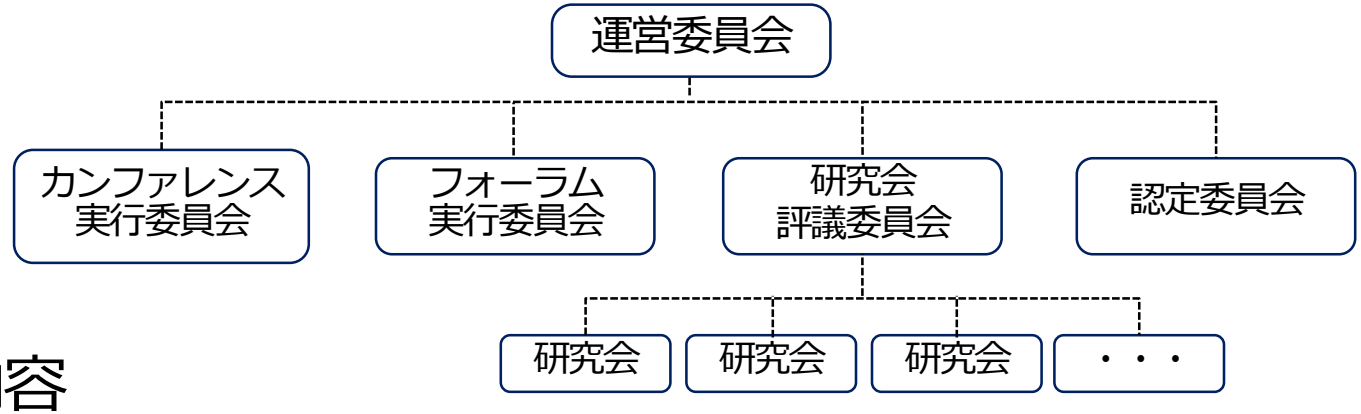
- 設立の主旨
  - 「XDDP」など派生開発に於ける効果的な方法の開発とその普及、促進
  - 協議会活動で得た成果や現場での取り組みのヒントの提供

日本企業の競争力に貢献する

© 2019 派生開発推進協議会

運営組織と活動内容

- 組織



- 活動内容
  - カンファレンス
    - 現場での取り組みの精度を高めたり、これから取り組む人への指針を提供
    - 毎年200名ほどの参加者、現場の切実な思いから質疑が活発
  - フォーラム
    - 派生開発に有効あるいは効果的な考え方や方法の紹介
  - 研究会
    - 詳細は次ページ

© 2019 派生開発推進協議会

研究会テーマ一覧

1	障壁の克服方法
2	「USDM」の入門
3	「XDDP」の入門
4	「XDDP」とテストプロセスとの接続
5	影響箇所の気付き
6	Agile開発との連携
7	ハードの派生開発への適用
8	大規模システムへの効果的対応
9	ビジネス領域での「XDDP」の活用
10	「USDM」とユースケースの連携
11	上位の要件開発技法と「USDM」の連携

12	ソフトウェア品質要求の定義
13	「USDM」のリスク管理への応用
14	SPLと「XDDP」の連携
15	「USDM」の支援ツール
16	「XDDP」の支援ツール
17	「PFD」の支援ツール
18	USDMと形式言語との接合における曖昧表現の克服
19	派生開発におけるスペックアウトの仕方
20	「XDDP」とモデル駆動開発の融合
21	「PFD」によるプロセス設計
22	失敗事例

© 2019 派生開発推進協議会

研究会テーマ一覧

XDDP関連

- 「XDDP」の入門 T03
- 影響箇所の気付き T05
- 派生開発における スペックアウトの仕方 T19
- 障壁の克服方法 T01
- 失敗事例 T22

検証

- 「XDDP」とテストプロセス との接続 T04
- USDMと形式言語との接合に おける曖昧表現の克服 T18

要求関連

- 上位の要件開発技法と 「USDM」の連携 T11
- 「USDM」と ユースケース の連携 T10
- ソフトウェア 品質要求の 定義 T12
- 「USDM」の入門 T02

プロジェクト管理

- USDM」のリスク管理への 応用 T13

開発対象

- ハードウェア開発の 派生開発への適用 T07
- エンタープライズ領域で の「XDDP」の活用 T09

開発手法

- SPLと「XDDP」の連携 T14
- 「XDDP」とモデル 駆動開発の融合 T20
- 大規模システムへの 効果的対応 T08

プロセス関連

- PFDによるプロセス設計 T21
- Agile開発との連携 T06

ツール

- 「XDDP」の支援ツール T15
- 「USDM」の支援ツール T16
- 「PFD」の支援ツール T17

© 2019 派生開発推進協議会

派生開発プロセス XDDPの成り立ち

© 2019 派生開発推進協議会

XDDP誕生のきっかけ

XDDP提唱者・清水吉男氏の体験 (1978年)

- アメリカから変更追加依頼 (納期3ヶ月) を受ける
- 初めてのシステム、初めて見るソースコード
- 今までのやり方では対応できない
  新しい「やり方」を考える必要がある

✕とりあえず、言われた通りコードを直すか。。。

この違いは、仕事に対する
意識・姿勢・こだわり

プロとしての自覚

納期への執着

© 2019 派生開発推進協議会

何を考えたか

- アメリカから変更追加依頼(納期3ヶ月) を受ける
- 初めてのシステム、初めて見るソースコード
- 今までのやり方では対応できない新しい進め方を考える必要がある

① 変更依頼は仕様が示され、**要求が省略**される

② 短納期で**すべての**システム、コードを**理解するのは間に合わない。新規開発のプロセスでは対応できない**

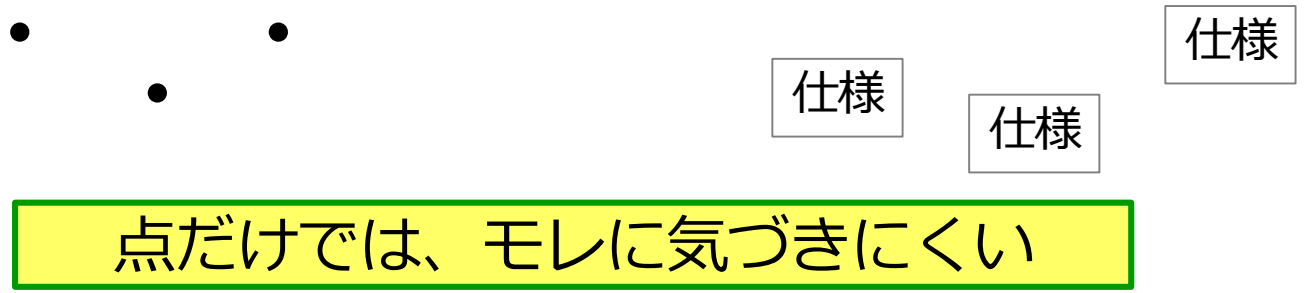
③ 品保、市場での不具合修正は**「バグ管理プロセス」**を実施し、**再発を防止**している

© 2019 派生開発推進協議会

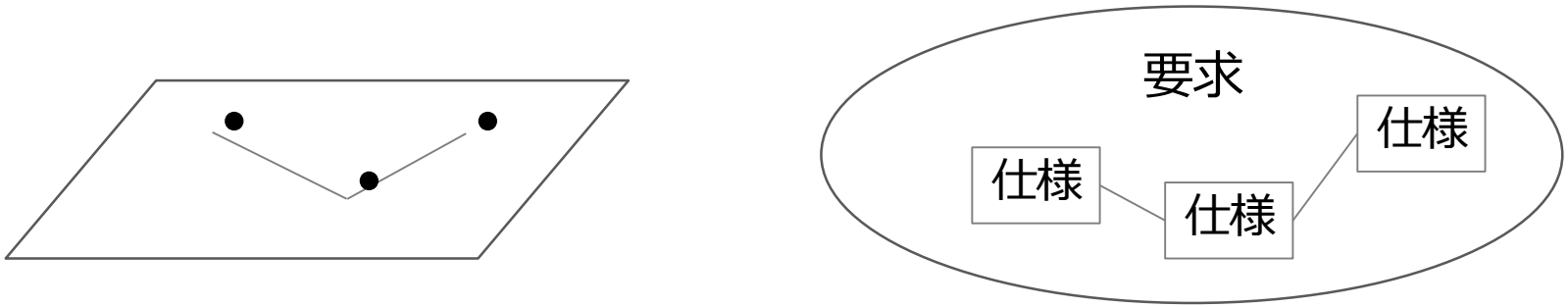
①要求が省略される

- ある箇所を修正することによる**他への影響**がわからない
- **依頼者も気づかない**事象が起こることがある

⇒ 面・線を理解し 点を理解する＝要求を理解し 仕様を理解する



点だけでは、モレに気づきにくい



面・線を理解すれば、点も理解できる

②すべて理解するのは間に合わない

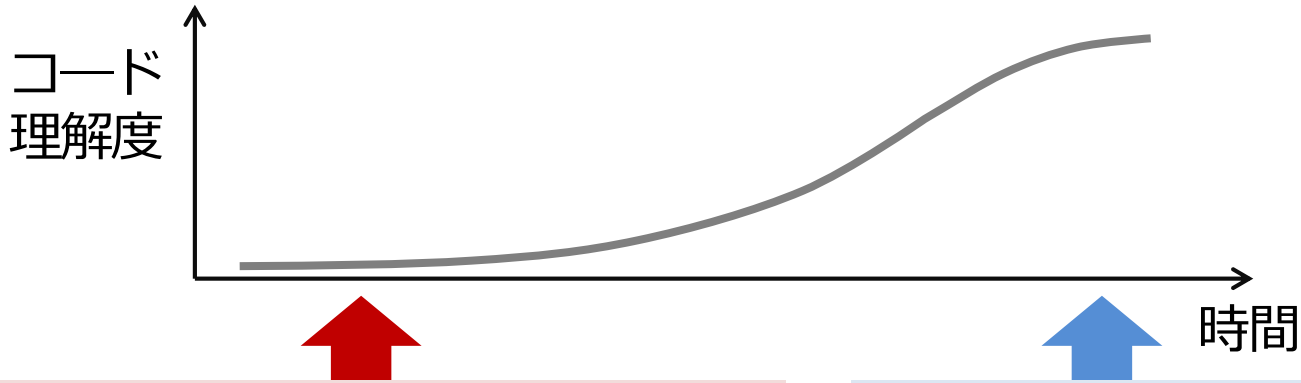
- 新規開発のプロセスでは対応できない

⇒ 変更する内容、変更箇所、機能追加する内容をレビューで確認する (**差分開発**の考え方)

- 限られた時間内でシステム、コードの理解度を高めてから、すなわち、**なるべく遅いタイミングでコードを変更**したほうが良い

なるべく遅いタイミングが良い理由

- **学習曲線** (理解に費やした時間によって、正しい反応を示すことができる) から



ここでコードを変更すると、
・不適切な変更に気づかない
・モレに気づかない
・変更による他への影響に気づかない
・変えたものが正しいと思い込んで、後戻りできない

ここで変更したほうが、
・気づかなかったことが気づく
・より良い変更方法に気づく
・コンフリクトに気づく

当時はFAXで確認し理解を深めた

変更箇所を書き留めておく手段が必要

© 2019 派生開発推進協議会

③バグ管理プロセスを実施している

- バグ管理プロセス
  - 発見 → 担当者割当 → 調査 → 修正 → 確認 → クローズ
  - 調査→修正フェーズにおいて「変更マトリクス」や「変更内容記録」等の**中間成果物で問題がないかレビューで確認**
  - 納期が迫る状況で**焦って十分に考えずに**変更して**失敗する**ことを防止

バグを混入させない仕組みがすでに世の中にある

- 変更依頼や機能追加でも
  「動いているソフトウェアに手を加えるのは同じ」
  ⇒ **変更や機能追加について最初からしっかりしたプロセスがあれば不具合は防げる**

© 2019 派生開発推進協議会

どう対策したか

何を考えたか		どう対策したか
①変更依頼は仕様が示され、要求が省略される	•仕様変更だけでは他への影響がわからない •依頼者も気づかない事象が起こることがある •面・線を理解し、点を理解する	•(変更)要求を引き出す •(変更)要求から(変更)仕様を導き出す ⇒ **USDM**
②短納期で、全てのシステム、コードを理解すると間に合わない	•変更する内容、変更箇所、機能追加する内容をレビューで確認する •遅いタイミングでコードを変更	•調べたこと、変更方法を残す •残したことをレビューで確認すると同時に理解度を上げていく ⇒ **差分開発、学習曲線**
③品保、市場での不具合修正は、「バグ管理プロセス」を実施している	•バグ対策だけでなく、変更や機能追加についてしっかりしたプロセスがあれば不具合は防げる	•変更・機能追加に最適なプロセスを考える •短納期でも対応できる無駄のない最小限の成果物で構成するプロセスにする必要がある ⇒ プロセスの設計 (**PFD**)

© 2019 派生開発推進協議会

対策の結果

納期達成　　不具合ゼロ

どう対策したか
•(変更)要求を引き出す •(変更)要求から(変更)仕様を導き出す ⇒ **USDM**
•調べたこと、変更方法を残す •残したことをレビューで確認すると同時に理解度を上げていく ⇒ **差分開発、学習曲線**
•変更・機能追加に最適なプロセスを考える •短納期でも対応できる無駄のない最小限の成果物で構成するプロセスにする必要がある ⇒ プロセスの設計 (**PFD**)

XDDPの誕生

アイディアを形式化し
10年かけて検証してから
書籍化

© 2019 派生開発推進協議会

ベースとなる知識

- エンジニアリングプロセス
  - –要求仕様定義技法　・・・　弱い人が多い
  - –設計技法
  - –実装技術
  - –テスト技法

これらはソフトウェアエンジニアとして必須のスキル

- サポートプロセス
  - –構成管理、問題解決管理、開発環境整備・・・
  - –プロセス設計技術　・・・　弱い人が多い

© 2019 派生開発推進協議会

ということで、この後

プロセス設計技術　**PFD**

要求仕様定義技法　**USDM**

派生開発プロセス　**XDDP**

の順で「基本」を紹介します。

© 2019 派生開発推進協議会

PFD, USDM, XDDPの基本

派生開発カンファレンス2019　プログラム委員長
アンリツエンジニアリング株式会社
南部 妙水

© 2019 派生開発推進協議会

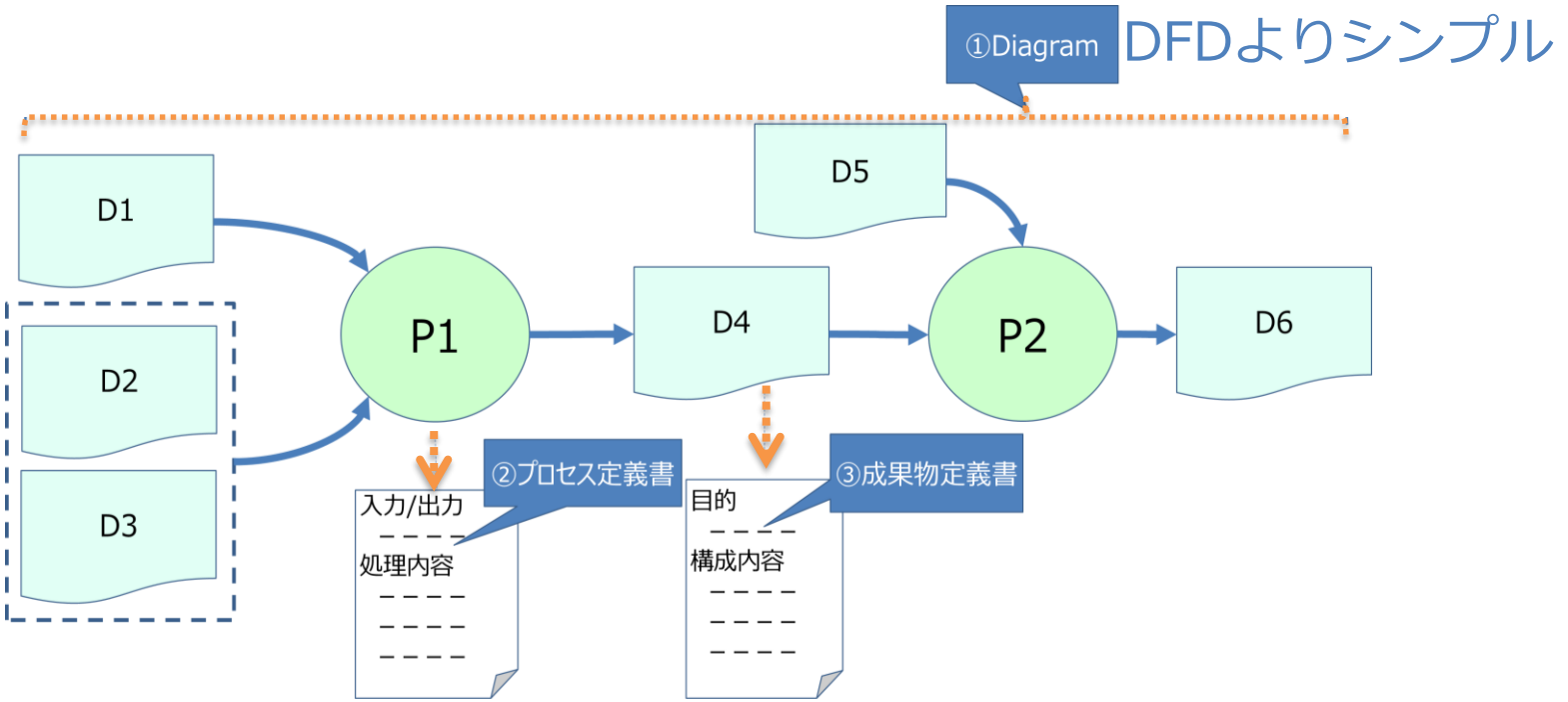
PFD
(Process Flow Diagram)
の基本

書き方のルールは簡単！
使える場面も多彩

© 2019 派生開発推進協議会

PFDとは

- DFDをベースにしたプロセス設計に特化した表現記法
- ①Diagram ②プロセス定義書 ③成果物定義書 の3つで
  プロセス実行に必要な情報を定義し、設計できる技術



プロセスには入力成果物・出力成果物が必ず存在する

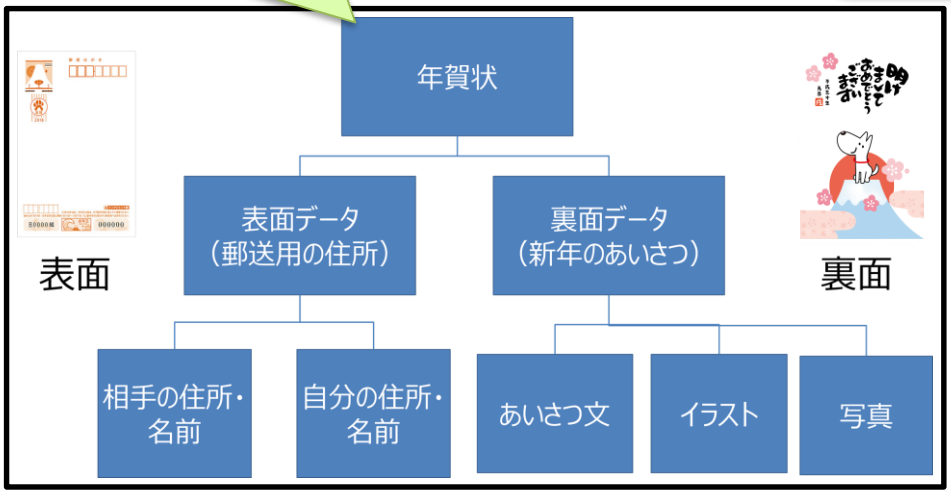
使い方① プロセスの洗い出し

- 実施する順番にプロセスを考えてしまうことが多いですが

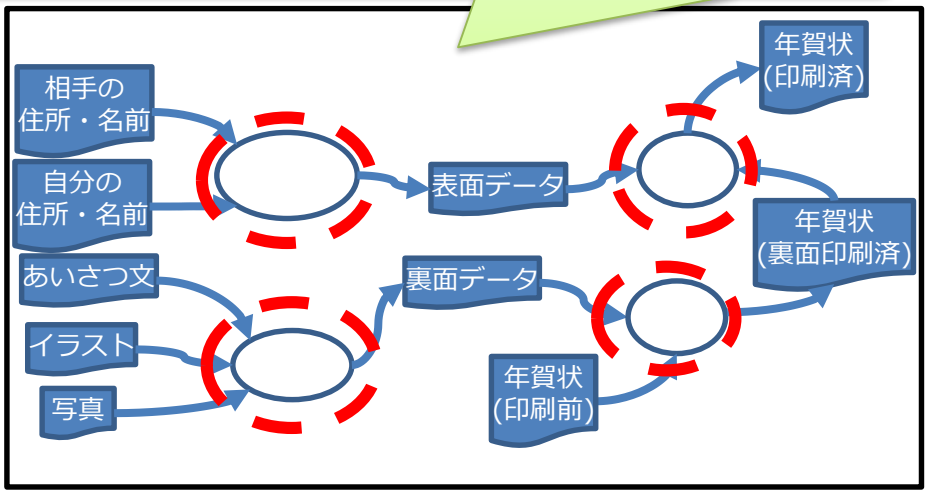
① 最終成果物 (ゴール) から考えて

② 成果物を構成する要素を分解し、中間成果物間のプロセスを検討すると効率的

ステップ１：
最終成果物を中間成果物に分解



ステップ２：
成果物構成を90°回転し
中間成果物の間のプロセスを検討していく



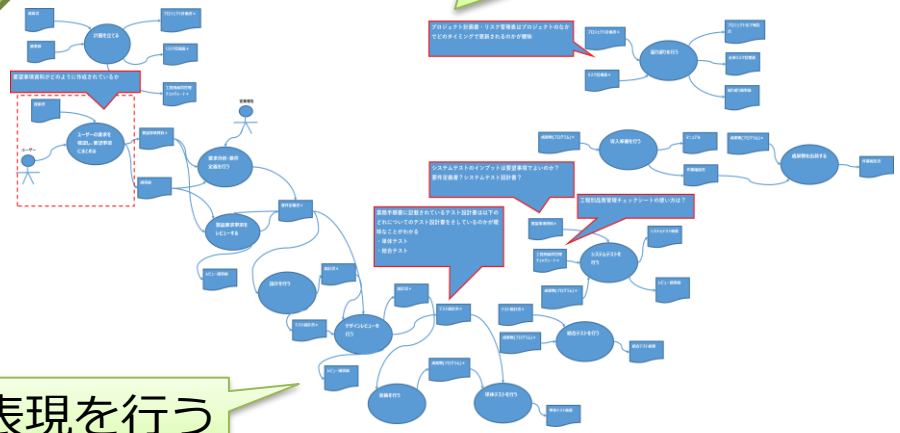
© 2019 派生開発推進協議会

使い方② 既存プロセスの改善

- 組織で定義されている業務手順書等をPFDに書き直すと
- 成果物の入力漏れや、表現されていないプロセスが潜んでいることに気づき、改善の議論ができる

業務手順書		○○事業部	
工程	定義	INPUT	OUTPUT
計画	開発のスコープ及び方針を決定する。また、リスクの洗出しを行う。	提案書 議事録	プロジェクト計画書 リスク管理表 工程別品質管理チェックシート
要求分析・要件定義	ユーザの要求事項を明確に定義する。	要望事項資料 議事録 営業報告	要件定義書
製品要求事項のレビュー	要求事項の問題点を明確にし、必要な処置を行う。(要求事項の可能性、システムの実現性)	要望事項資料 議事録 要件定義書	要件定義書 レビュー議事録
設計	要件を実現する成果物の設計をする。	要件定義書	設計書 テスト設計書

ダイアグラムで繋がりを見える化
→ インプットの漏れや
　成果物・プロセスの曖昧さが見える
→ 改善の議論が行える！



※複雑になった場合は階層化表現を行う

© 2019 派生開発推進協議会

PFDはいつでも使える

- 新規開発では
  - 事前に机上検討して妥当性を確認し、手戻りを防止する
  - 必要な技術要素の洗い出す(プロセス定義書の必要スキル検討)
- 派生開発では
  - 今回の変更点は何かを中心に成果物・プロセスを見直す
  - 変更点による成果物・プロセスへの影響範囲を確認する
- 業務改善では
  - 現在の仕事の非効率部分を見えるようにする
  - 下記の観点で改善を検討する
    - ヤメル　(目的から考えて本当に必要かを見極める)
    - ヘラス・カエル　(工数に見合った効果か？条件改善、方法改善)
    - ウツス　(権限委譲と外部活用)

© 2019 派生開発推進協議会

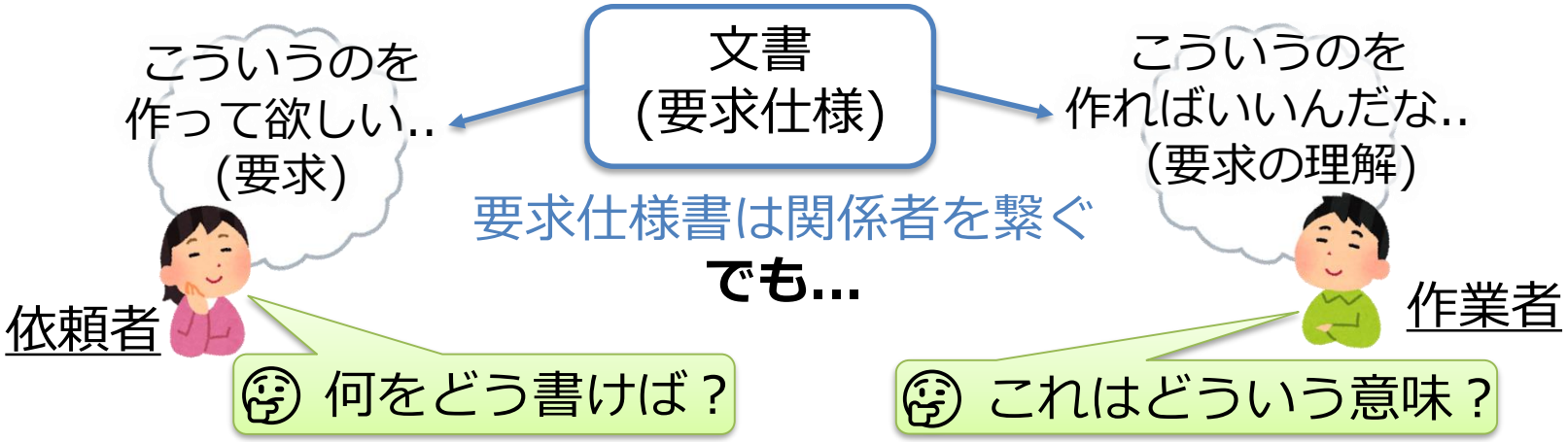
USDM
(Universal Specification Describing Manner)
の基本

沢山の工夫が詰め込まれた
強力なツール

© 2019 派生開発推進協議会

USDMとは

- 要求と仕様を階層化した「作るべきもの」の表現記法
- 仕様を適切に表現することで、要求の**発見**や**洗練**を助ける技法

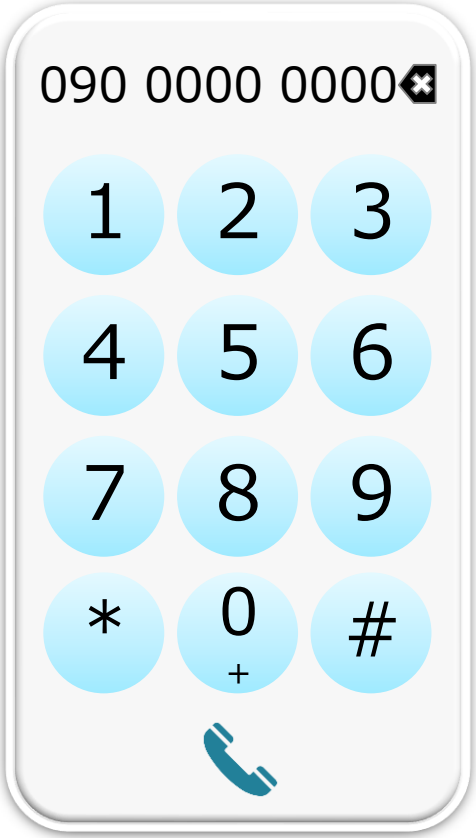


- 書き方を統一することで

共通のゴールをイメージ (Specify:仕様化) する

例:「電話をかける」機能

- 散文的な要求仕様では…



要求仕様	スマホで電話をかける

スマートフォンで電話をかけるための画面を用意し、電話をかけられること。

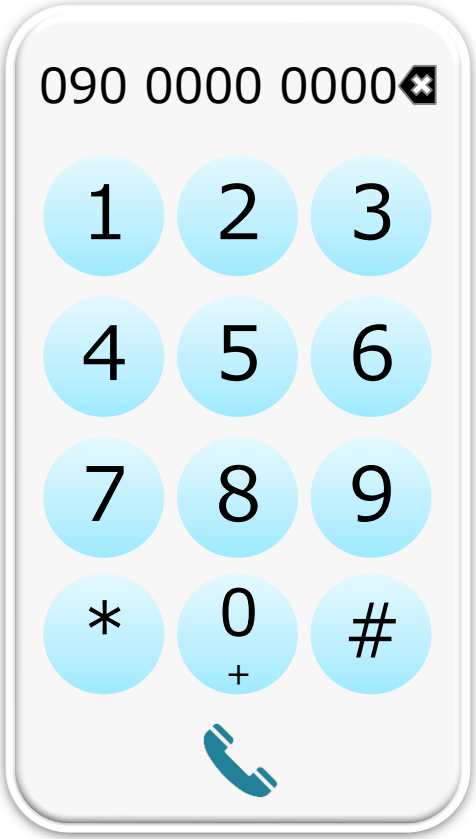
- 電話番号を入力するためのキーパッドを左図のように配置する。
- キーパッドをタッチすると入力内容が上部に表示される。
- 画面下部に受話器アイコンをおき、タッチすると電話をかける。

これで本当に十分？

© 2019 派生開発推進協議会

USDMで書いてみると

- 散文からわかる要求と仕様を書き起こすと



要求	TEL01	電話画面のキーパッドをタッチすると、タッチしたキーパッドの数字が追加され、文字削除ボタンが表示され、電話番号の間にスペースが空き、通話開始ボタンが活性化する。
	理由	使い慣れた固定電話の操作感でタッチ操作しかつ電話番号のかけ間違いを減らすため。
	TEL01-1	左図のように、上部に電話番号表示欄、中央にキーパッド、下部に受話器アイコンを表示する。
	TEL01-7	電話番号が日本の市外局番に対応している場合、市街局番の後にスペースを入れる。
	TEL01-10	入力中の電話番号があれば通話開始ボタンを押せるようにする。

© 2019 派生開発推進協議会

特徴①「要求」と「仕様」に分ける

- 「要求」は「仕様」に紐付く **階層構造** を取る

要求

ユーザやシステムの振る舞いを定義
- イベントから始まる動詞の連鎖
  (XXすると、・を～して、・を～して、・を～する)
- その機能の始まりから終わりをとらえる

仕様

要求を満たすための具体的な条件を定義
- 依頼者と作業者が共通のイメージを持てる粒度
- 合意可能な受け入れ基準
- 仕様をすべて満す＝上位の要求は満たされる

要求	TEL01	電話画面のキーパッドをタッチすると、 キーパッドの数字が追加され、 ボタンが表示され、電話番号の間に が空き、通話開始ボタンが活性化
		固定電話の操作感でタッチ操作し かつ電話番号のかけ間違いを減らすため。
	TEL01-1	右図のように、上部に電話番号表示欄、 、下部に受話器アイコン
		市外局番に対応している 後にスペースを入れる。
	TEL01-10	入力中の電話番号があれば通話開始ボタン を押せるようにする。

© 2019 派生開発推進協議会

特徴②「理由」を書く

- 要求は現状困っている事についての解決策の1つにすぎない
  - 要求は振る舞いに注力するため、その存在理由や背景を書けない
  - なぜこの解決策がベストなのか？を明確にする

要求	TEL01	電話画面のキーパッドをタッチすると、**キーパッドに対応したプッシュ音が鳴り、**タッチしたキーパッドの数字が追加され、文字削除ボタンが表示され、電話番号の間にスペースを空き、通話開始ボタンが活性化する。
	理由	使い慣れた固定電話の操作感でタッチ操作し、かつ電話番号のかけ間違いを減らすため。
	TEL01-1	左図のように、上部に電話番号表示欄、中央にキーパッド、下部に受話器アイコンを表示する。
	TEL01-4	キーパッドタッチ時、キーパッドの番号に対応したトーン信号音を鳴らす。

一般的な電話ならボタンを押したらプッシュ音が聞こえるはず…

- 理由をつかむことで、要求の変化を予測したり
  競争力を高めるための工夫を考えられるようになる

© 2019 派生開発推進協議会

特徴③「グループ」で整理する

- 要求の動詞から、仕様のグループを分割する

明記されていないが
ユーザがタッチ操作できる画面が
初期状態で表示されているはず

グループ化

		話画面のキーパッドを**タッチする**と、キー ドに対応したプッシュ音が**鳴り**、タッチし キーパッドの数字が**追加され**、文字削除ボタ が**表示され**、電話番号の間にスペースが**空き**、 通話開始ボタンが**活性化する**。
	理由	使い慣れた固定電話の操作感でタッチ操作し、かつ 電話番号のかけ間違いを減らすため。
	<初期画面表示>	
	TEL01-1	左図のように、上部に電話番号表示欄、中央にキーパッド、 下部に受話器アイコンを表示する。
	<プッシュ音>	
	TEL01-4	キーパッド 号音を鳴ら 信

初期画面表示として
他にどんな仕様が隠れている？

- 範囲を小さくすることで焦点を絞りモレを防ぐ

※ 要求の扱う範囲が広すぎて、要求の中の動詞が抽出しづらい場合
要求を「上位要求」「下位要求」に階層化して分割する

© 2019 派生開発推進協議会

USDMで書いてみると

- 表現方法を決めることでレビューもしやすい

スマートフォンで電話をかけるための画面を用意電話をかけられること。

- 電話番号を入力するためのキーパッドを左図のように配置す
- キーパッドをタッチすると、入力内容が上部に表示される。
- 画面下部に受話器アイコンをおき、タッチすると電話を

要求	TEL01	電話画面のキーパッドをタッチすると、キーパッドに対応したプッシュ音が鳴り、タッチしたキーパッドの数字が追加され、文字削除ボタンが表示され、電話番号の間にスペースが空き、通話開始ボタンが活性化する。
	理由	使い慣れた固定電話の操作感でタッチ操作し、かつ電話番号のかけ間違いを減らすため。
	<初期画面表示>	
	TEL01-1	左図のように、上部に電話番号表示欄、中央にキーパッド、下部に受話器アイコンを表示する。
	<プッシュ音>	
	TEL01-4	キーパッドタッチ時、キーパッドの番号に対応したトーン信号音を鳴らす。
	<入力中の電話番号を表示>	
	TEL01-7	電話番号が日本の市外局番に対応している場合、市街局番の後にスペースを入れる。
	<通話開始>	
	TEL01-10	入力中の電話番号があれば通話開始ボタンを押せるようにする。

説明しきれないので

グループに分割する基準とか
変更箇所の表現方法(before-after形式)とか

- ＜改訂第2版＞
  要求を仕様化する技術・表現する技術
  (清水吉男、技術評論社)

出版社、Amazonで
電子版もあります

- USDM小冊子基礎編
  (T02「USDM入門」研究会)

Webで無料配布中
手短にまとまっています

要求仕様 USDM

© 2019 派生開発推進協議会

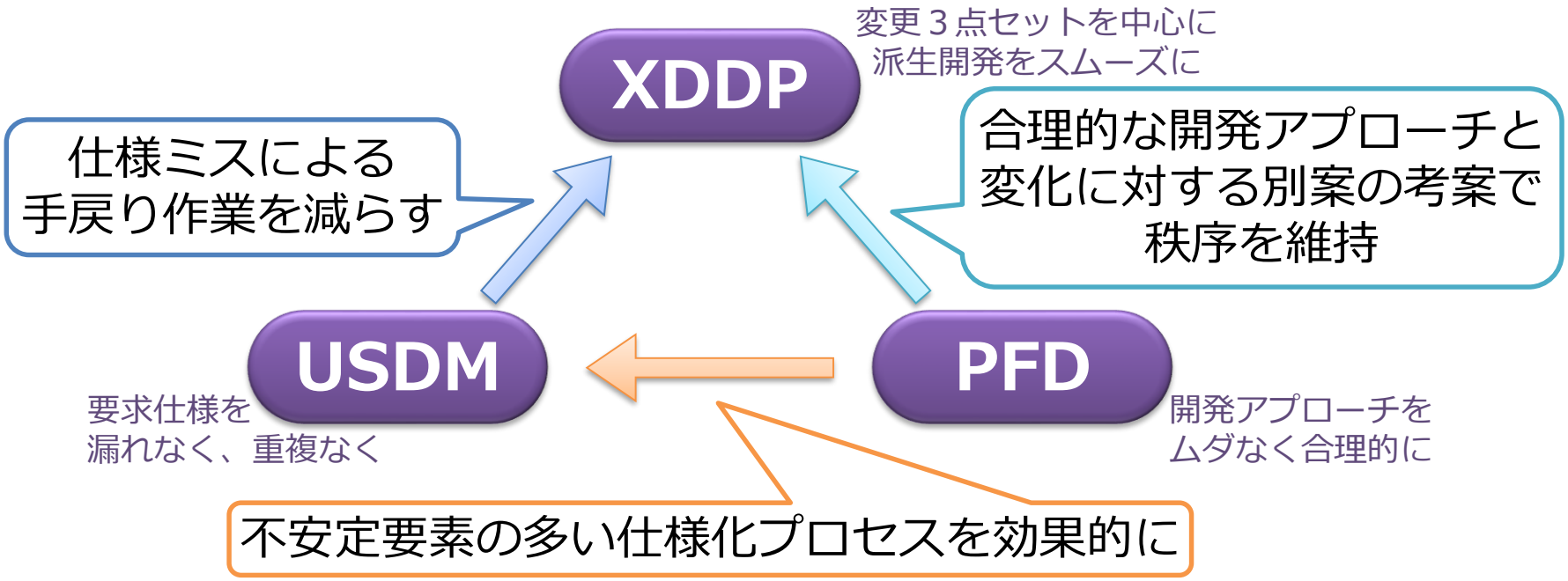
XDDP
(eXtreme Derivative Development Process)
の基本

最適を得るための
シンプルだけど奥深い方法

© 2019 派生開発推進協議会

XDDPとは

- 派生開発に特化した開発プロセス
- 機能追加と変更の2種類のプロセスに分けて品質と生産性を確保し、確実に成功＝期限内に終了する方法



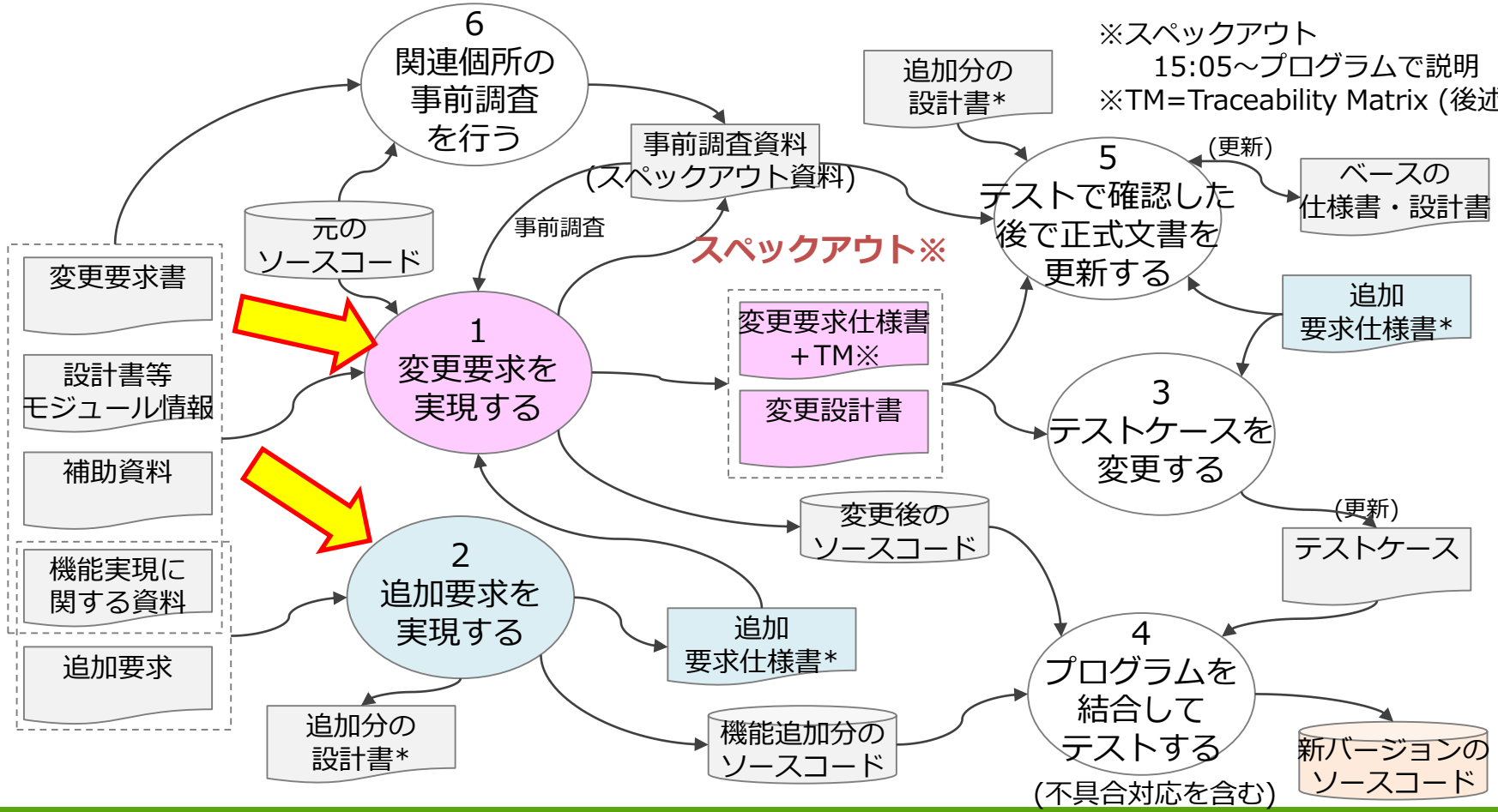
XDDPはUSDMとPFDの上で成り立つ

© 2019 派生開発推進協議会

ポイント①機能追加と変更を分ける

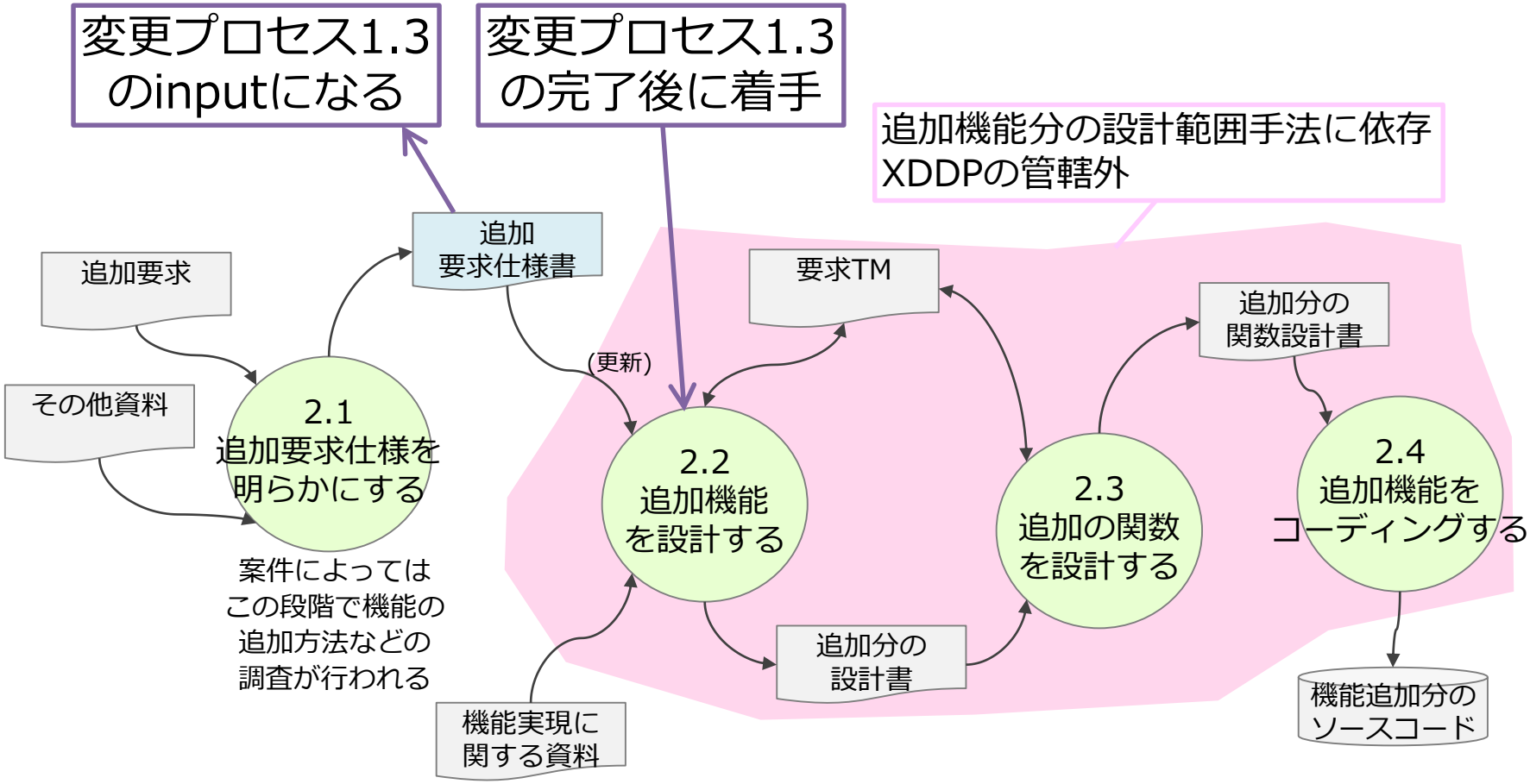
- 追加分は新規開発とほぼ同じ性質を持つ
- 変更は既存コードの制約を強く受ける

➡別々に作業する



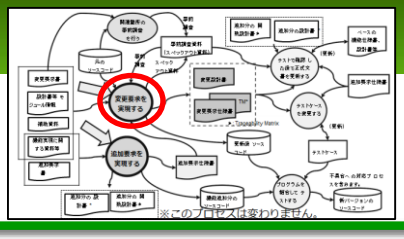
既存資産に機能を追加する

- 2箇所で変更プロセスと連携する以外は独自に開発

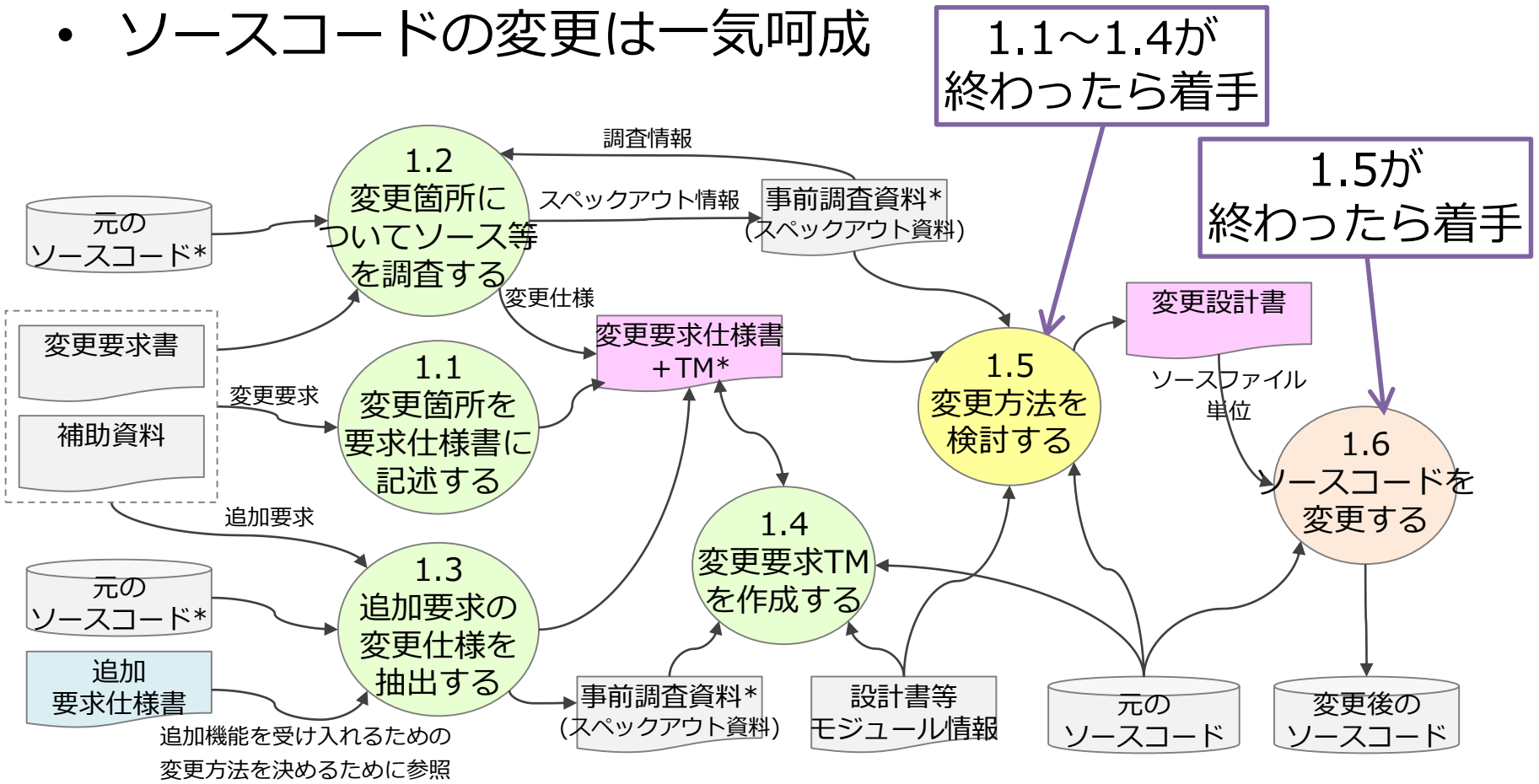


※プロジェクトの状況にあわせて、PFDで「開発プロセス」を設計する

既存資産を変更する



- **変更３点セット**に変更箇所・方法 (before-after) **のみ**を記述
- ソースコードの変更は一気呵成



※プロジェクトの状況にあわせて、PFDで「開発プロセス」を設計する

© 2019 派生開発推進協議会

ポイント②変更3点セット

- 変更内容や方法は「変更3点セット」にまとめて記述する
  - 要求仕様 (What & Why) は 変更要求仕様書 (USDM) に
  - 変更箇所 (Where) は Traceability Matrix に
  - 変更方法 (How) は 変更設計書 に

- 既存文書ではなく 変更3点セットに書くことで
  - 変更前と変更後を明記できる (自然言語は before-after形式 で)
  - 役割にあった表現方法、いつも同じ表現方法

  ⇒ その変更が最善かを検討しやすい

  - 変更要求仕様書では要求と仕様が紐付く
  - TMでは変更しない箇所も見える
  - What・Where・Howが紐付く

  ⇒ モレに気づきやすい

新規開発崩し (既存文書を随時改版) は
「変更後の姿」を書くことになり
変更点や影響箇所が見えにくい

© 2019 派生開発推進協議会

Traceability Matrix (TM)

- 変更仕様とソースコード構造のマトリックス

気づきを誘うため可能な限り**すべて**を表現する
(変更しない箇所、文書、ビルドパラメータ…)

		ファイル名	file1	file2	file3	file4	file5	file6	file7
#		変更要求・仕様							
5		画面に通信記録の表示を追加する							
	5.1	接続状況の表示の大きさを●●から○○に変更する			Fn1				
	5.2	表示用メモリの配置を■■から□□に変更する			Fn1			Fn7	
	5.3	受信時のエラー処理を▲▲から△△に変更する	Fn9		Fn1	Fn3	Fn4	Fn7	

凝集性が悪そう
リファクタリングする？

異なる変更仕様で同じ関数を変更する

変更仕様の粒度が粗いと
変更が多くのモジュールにまたがる

© 2019 派生開発推進協議会

変更設計書

もちろん before-after形式 で

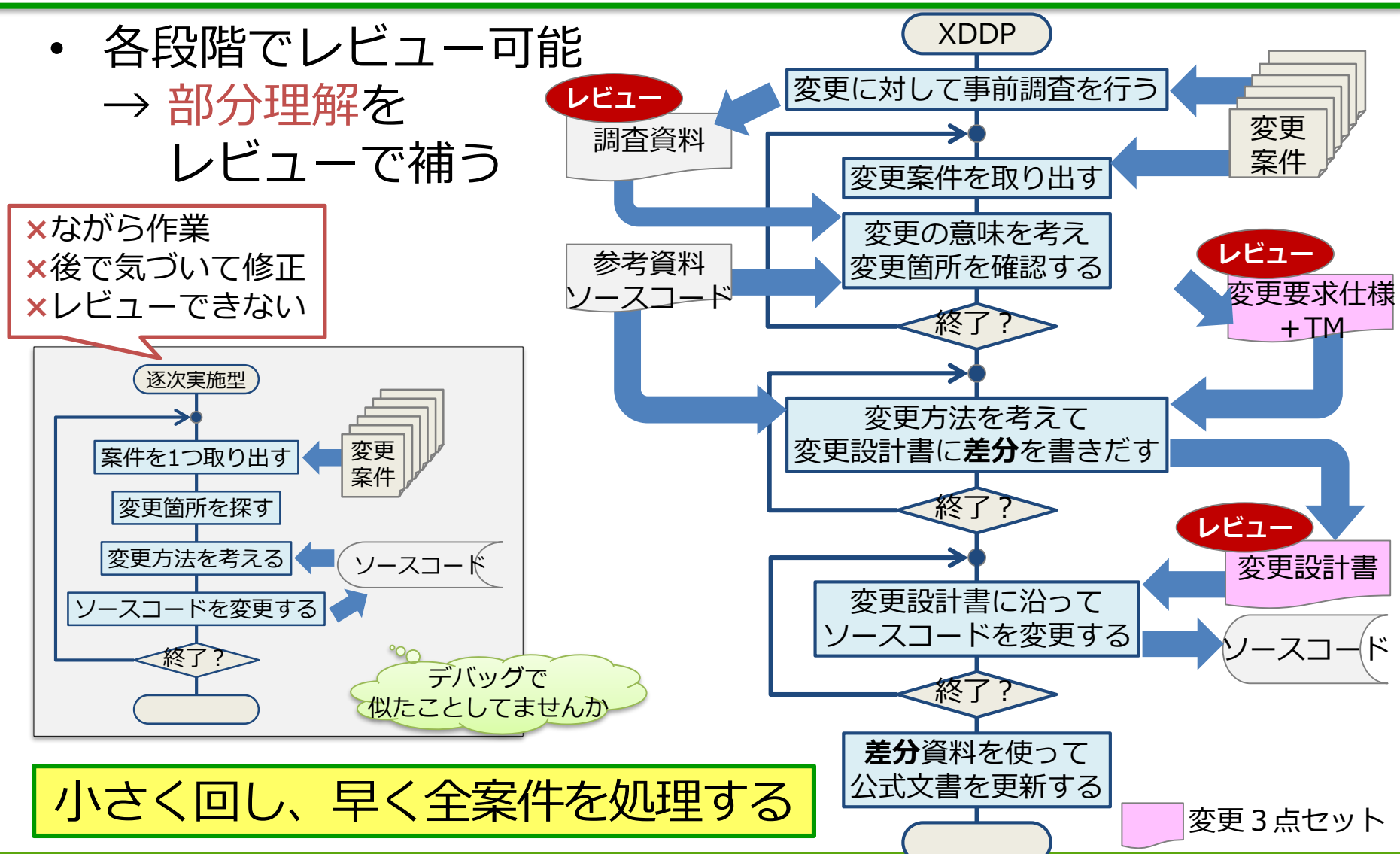
- ソースコードの変更方法を自然言語で定義した文書
  - プログラム言語と異なる表現方法で「設計者の理解」を明文化
    ⇒ 変更仕様やロジック等の理解不足・勘違いをレビューできる
  - TMの交点1つに1変更設計書だが、複数の変更仕様が1箇所に重なる場合はすべてを満たす変更方法を1つの変更設計書に記す
  - すべての変更を変更設計書に記す
    ⇒ ソースコードの変更を1回で済ます＋差分レビューも効率的

ファイル名	file6.cpp	
関数名	void func7()	
変更仕様	5.2	表示用メモリの配置を■■から□□に変更する
	5.3	受信時のエラー処理を▲▲から△△に変更する
変更詳細	①下のメモリ割当表と合致するように、□□のメモリデータを格納する構造体の定義を追加する。	
	②下の■■と□□の対応表に従い、現在のメモリデータ構造体から①のメモリデータ構造体に値をコピーする処理をfunc7の先頭に追加する。	

© 2019 派生開発推進協議会

ポイント③成果物1つずつ確認

- 各段階でレビュー可能
  → 部分理解を
  レビューで補う

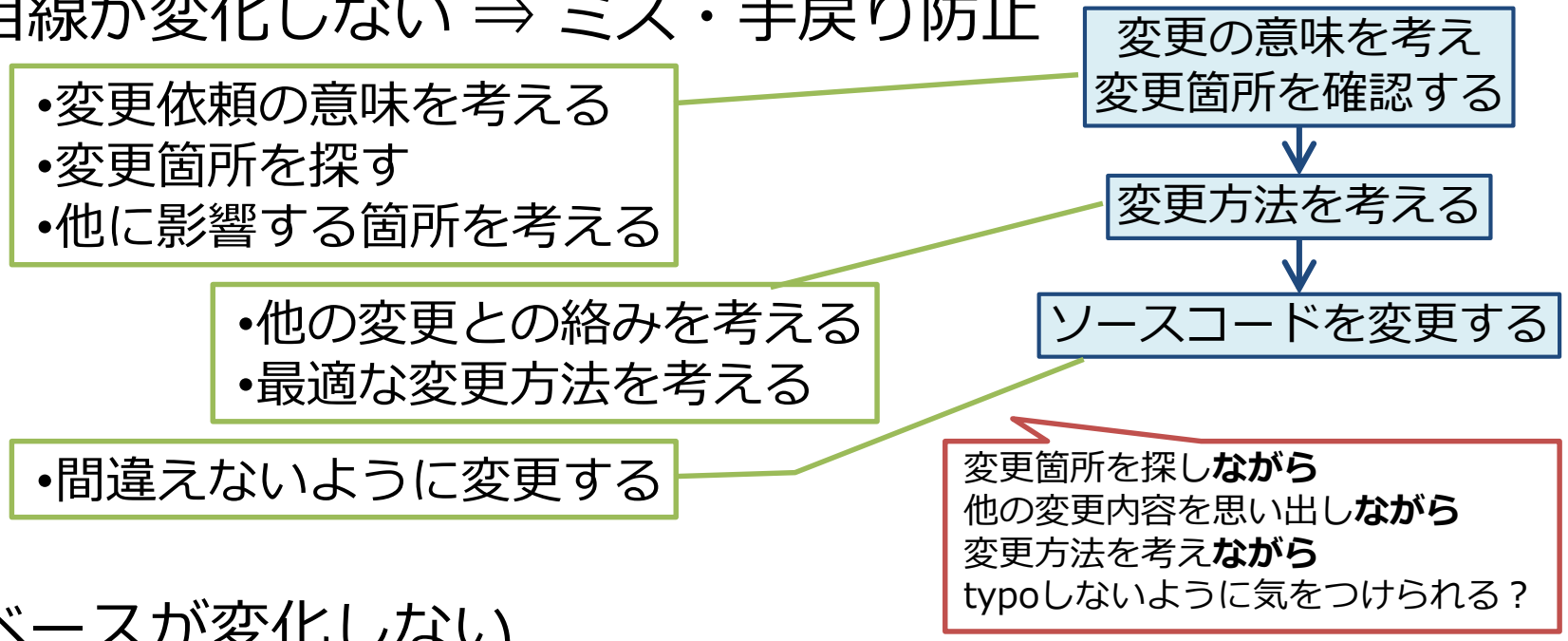


小さく回し、早く全案件を処理する

© 2019 派生開発推進協議会

「要求1つずつ」に比べ

- 目線が変化しない ⇒ ミス・手戻り防止



- ベースが変化しない
  - ソースコード、既存文書が変更されるのは最後
    → どの変更案件を考えるときでも before は同じ
    ⇒ 関係者が同じものを基準に検討・レビューできる
    ⇒ 手を付けた順に関係なく
    最善の方法でコンフリクトを解消できる

© 2019 派生開発推進協議会

XDDPの効果

- 生産性
  - 開発途中で既に変更した箇所を振り返ったり、依頼者に変更の意図を確認したりといった**ながら作業**がなくなるため、開発効率が良い
  - 事前に確認が済むように進めるため80～130行/時間も可能

- 品質
  - ソースコードを変更する前に不具合に気付くことができる
  - すべての変更は「変更設計書」に記述しているので、不具合の原因が特定しやすくなる
  - ソースコードの変更は**1回**で済ますことができ、ソースの劣化を防ぐことができる

ポイントは「性質の異なる行為を混同させない」

© 2019 派生開発推進協議会

ご静聴
ありがとうございました

不具合ゼロで次に備える
時間を手に入れましょう！

© 2019 派生開発推進協議会