XDDPの基本

2018年11月16日
派生開発推進協議会(AFFORDD)
T03「XDDP入門」研究会

アジェンダ

1. XDDP(eXtreme Derivative Development Process)とは
2. 従来の開発
   - 逐次実施型、新規開発崩し
3. XDDPのプロセス
   - 変更プロセス、追加プロセス
4. XDDPの成果物
   - 変更要求仕様書、変更トレサビリティマトリックス、変更設計書、追加要求仕様書、追加設計書、スペックアウト
5. まとめ
   - XDDPトライアングル、XDDPの特徴、XDDPの利点
   - 「変更３点セット」の理由、気をつけること

1. XDDP(eXtreme Derivative Development Process)とは

- 「**XDDP**」とは「派生開発に特化した開発プロセス」で、確実に成功(期限内に終了)する方法
- 「**XDDP**」の誕生
  - アメリカの顧客からの要求 (1978年)
  - 「保守のプロセスでは不可能」と判断。１週間でプロセスを設計・シミュレーションで確認
  - 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
  - 品質を保ちつつ「３ヶ月の納期」を達成
- 「**XDDP**」を用いると、次のような効果がある

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

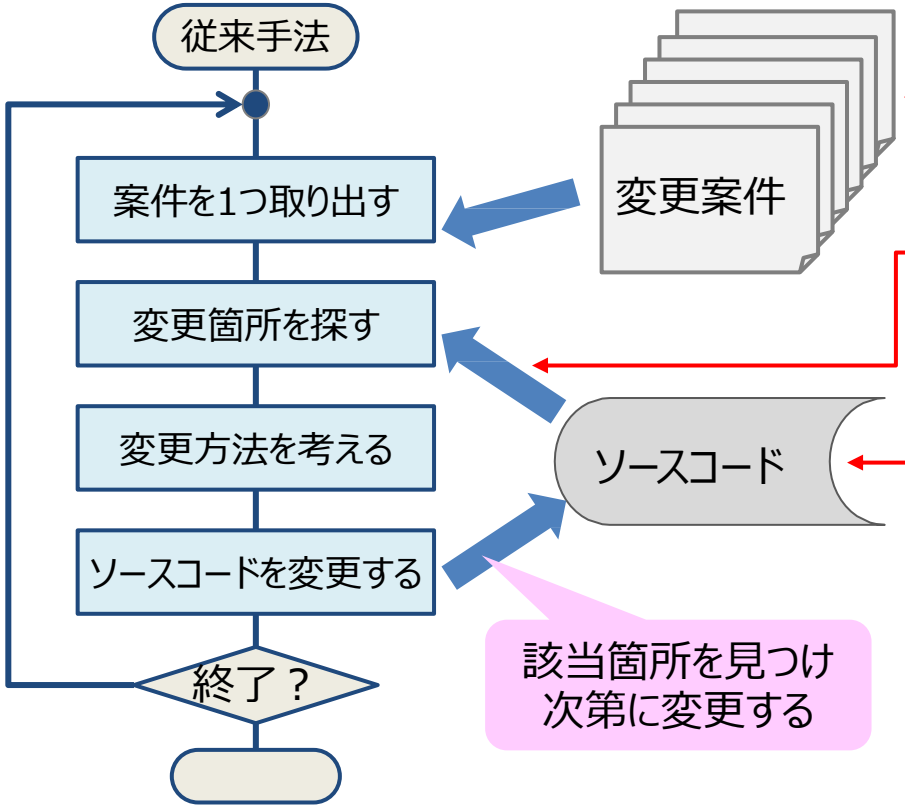
・ソースコードを変更する前に不具合に気付くことができる
・すべての変更は「変更設計書」に記述しているので、不具合の原因が特定しやすくなる
・ソースコードの変更は「１回」で済ますことができ、ソースの劣化を防ぐことができる

2. 従来の開発

2.従来の開発

公式文書が残されていない場合・・・逐次実施型

- 要求を１つ1つ逐次にソースコードを変更するパターン



変更箇所が重なる場合もある

ソースコードを見ながらの
「ながら作業」が増える

同一箇所に対して、複数の変更内容が干渉して問題を起こすこともある

・**ながら**作業による**確認不足**
・変更時に**適切なドキュメントがない**ため、**レビューでフォローできない**

・ミス、モレ、重複、干渉を後から気づき、コードを修正する → **手戻り、コード品質低下**
・変更による他への影響に気づくのが遅れる → **手戻り**
・不具合の修正で新たな不具合が出る → **手戻り**

2.従来の開発

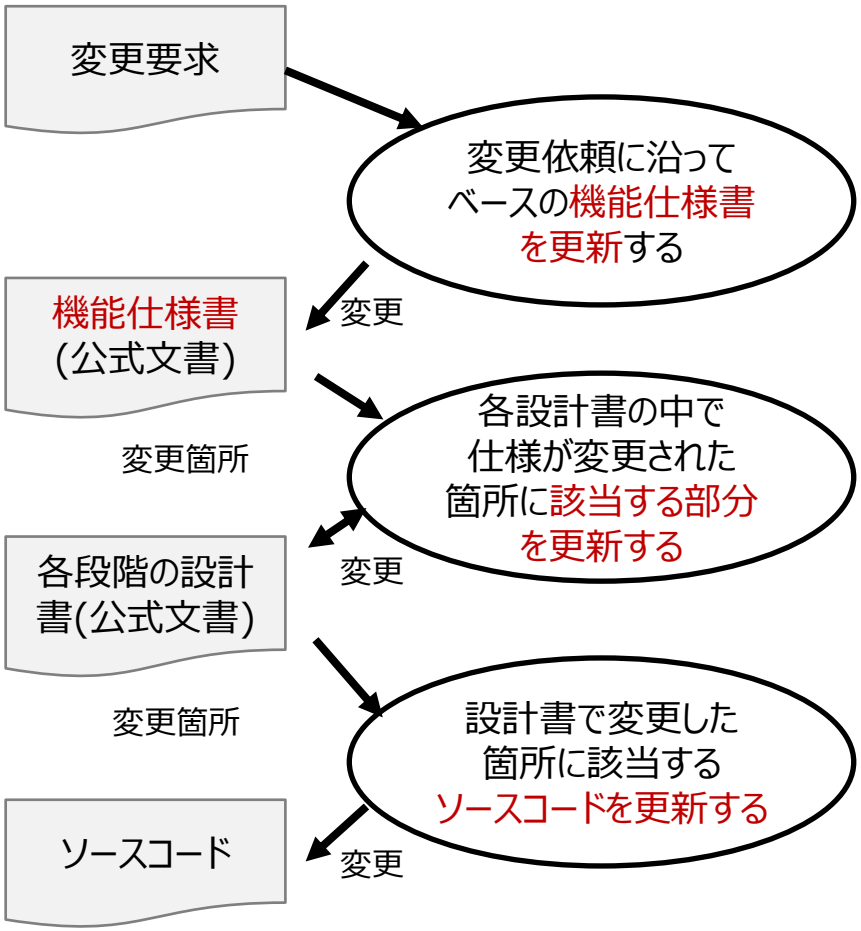
公式文書が残されている場合・・・新規開発崩し

- 新規開発のプロセスに則り、公式文書に変更内容を盛り込み、ソースコードを変更(完成)するパターン

■ 公式文書には、必ずしも変更に耐え切れるようには書かれてない場合がある

■ 公式文書に**逐次変更が織り込まれる**ことで、**他への影響が見えなくなる**

■ 公式文書には、変更後のイメージで書かれてしまうことが多く、**変更箇所や変更方法を十分、検討しないまま**、ソースコードを変更してしまう

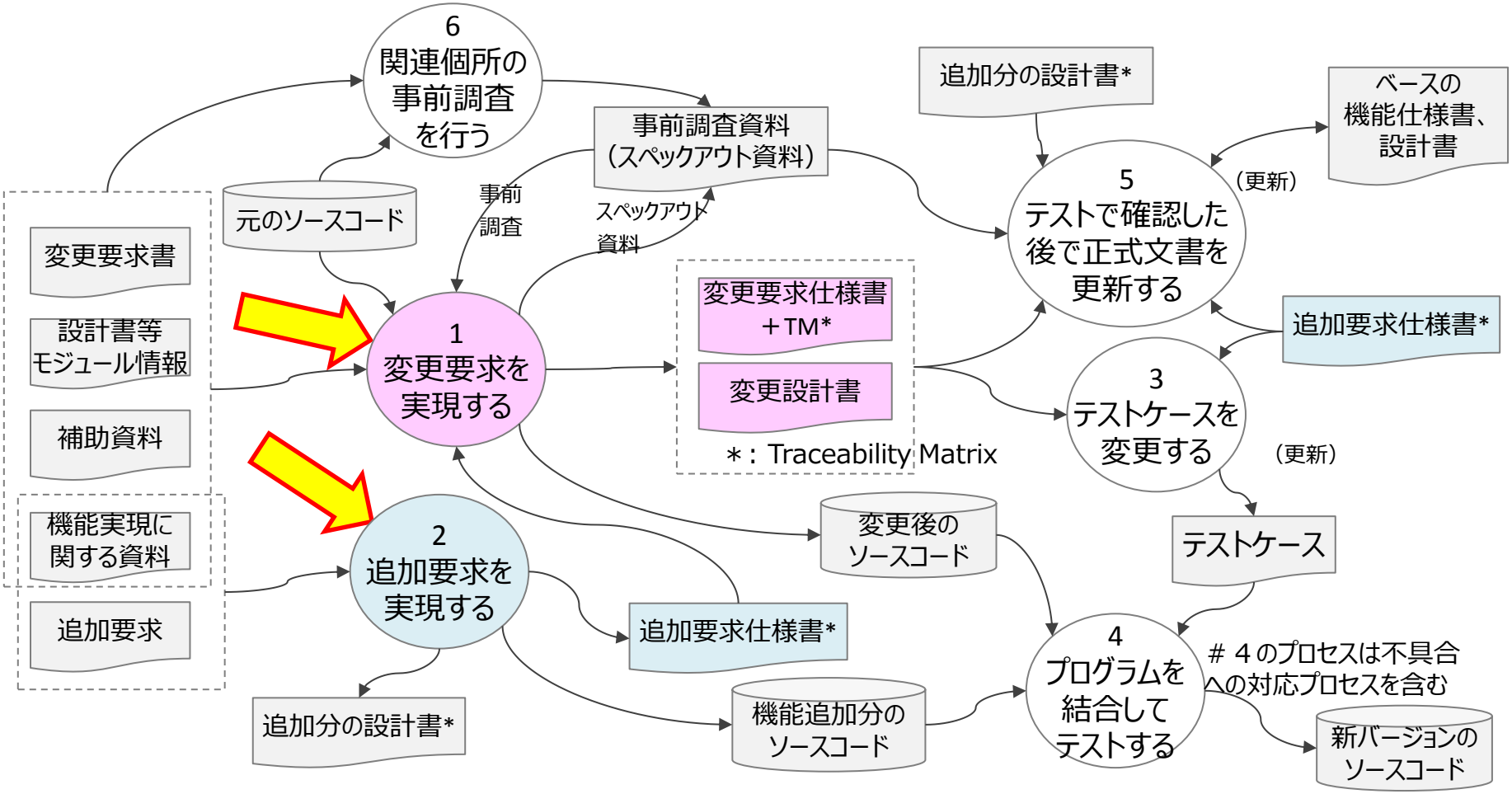


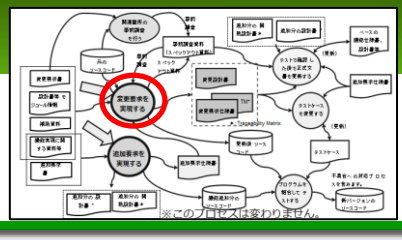
3. XDDPのプロセス

3.XDDPのプロセス

XDDPでは２種類のアプローチを並行させる

■ XDDPでは「変更」と「追加」を別々のプロセスで対応する
【理由】プロセスが異なるから

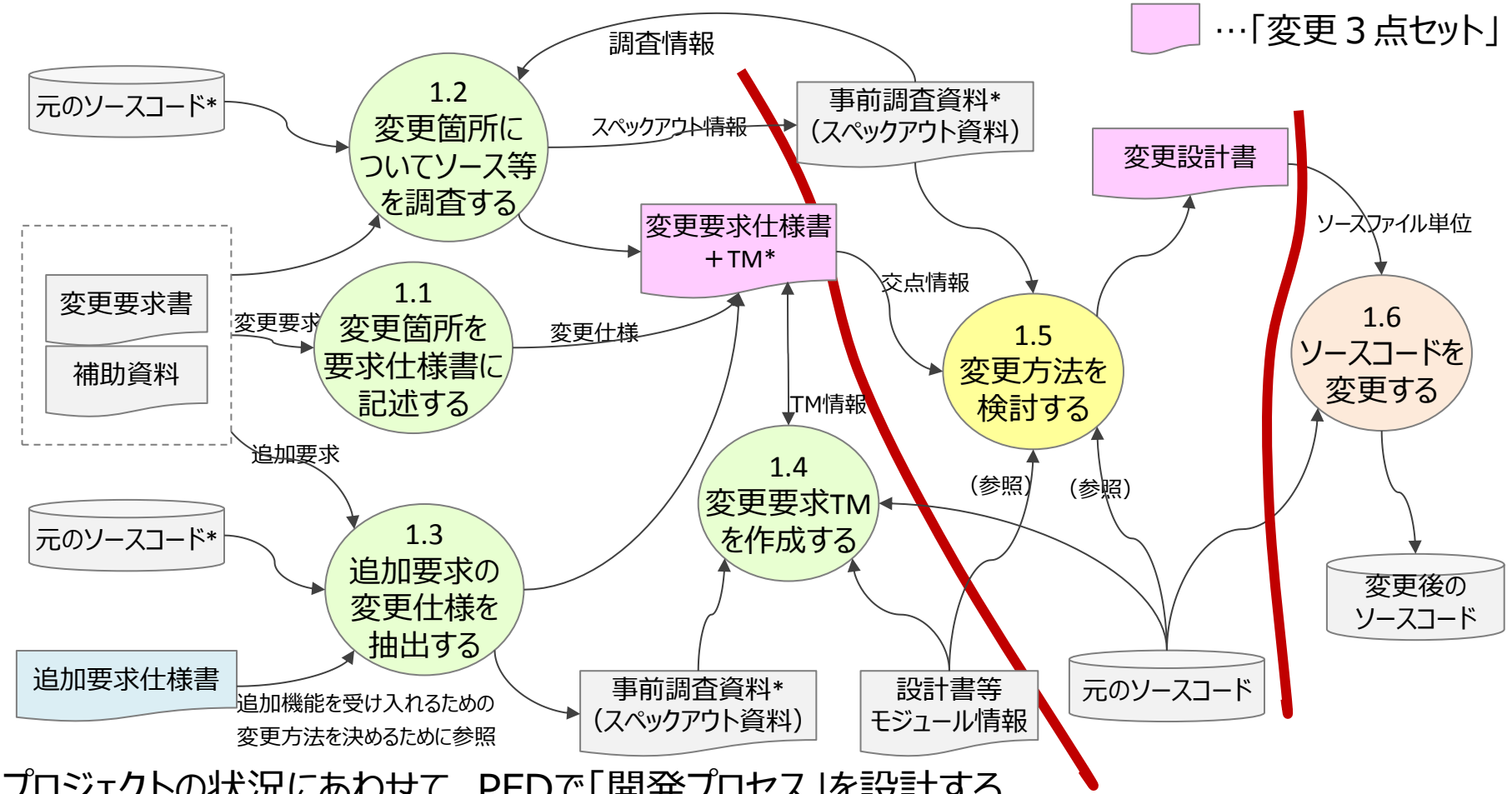




3.XDDPのプロセス

変更プロセス：変更を扱うプロセス

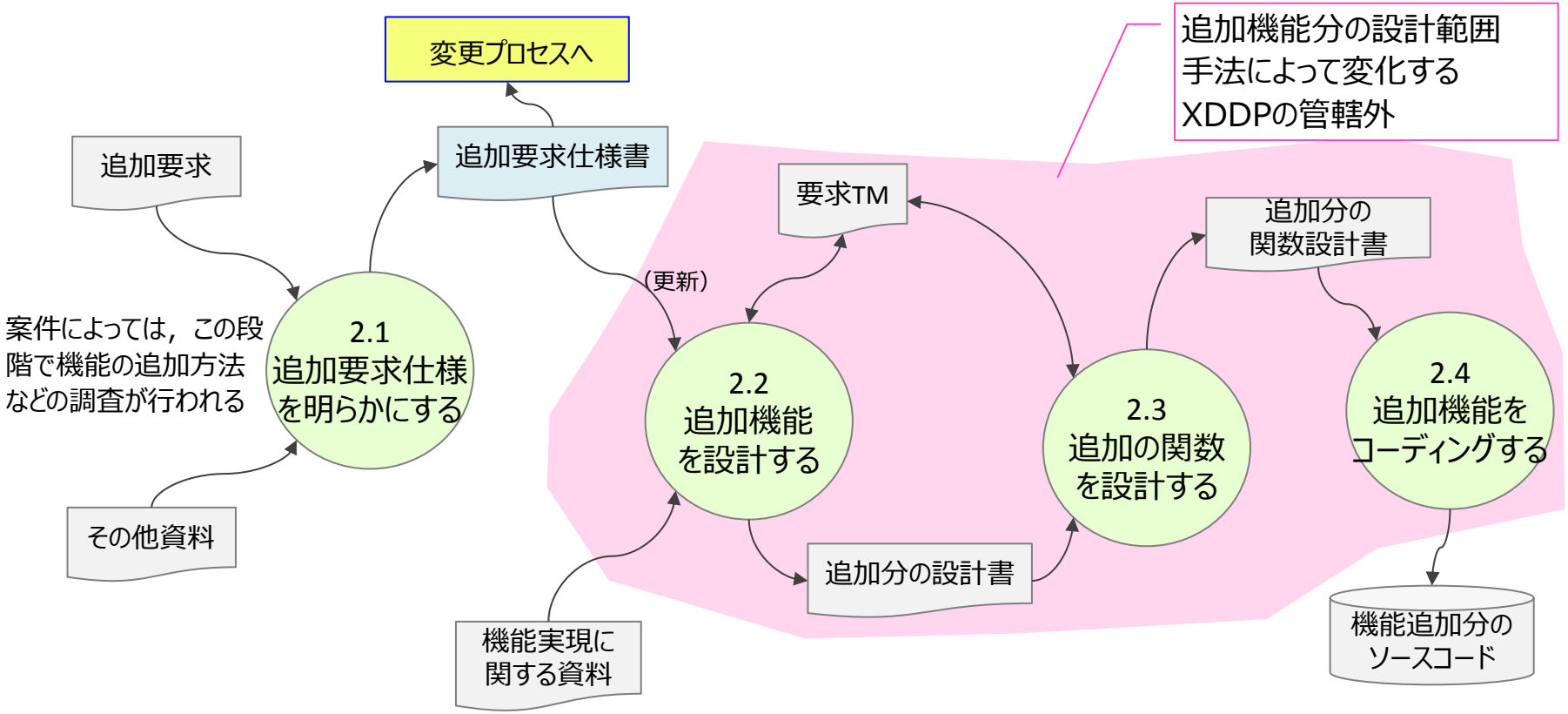
■ 変更プロセスでは、「変更３点セット」の成果物に変更箇所や変更方法を記述し、すべて揃った後に一斉にソースコードを変更する



※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

3.XDDPのプロセス
追加プロセス：既存資産に機能を追加する

- 基本的には新規開発のプロセスの同じ
- 変更プロセスの「追加要求の変更仕様を抽出する」が終了したときに「追加機能を設計する」以降に取り掛かる

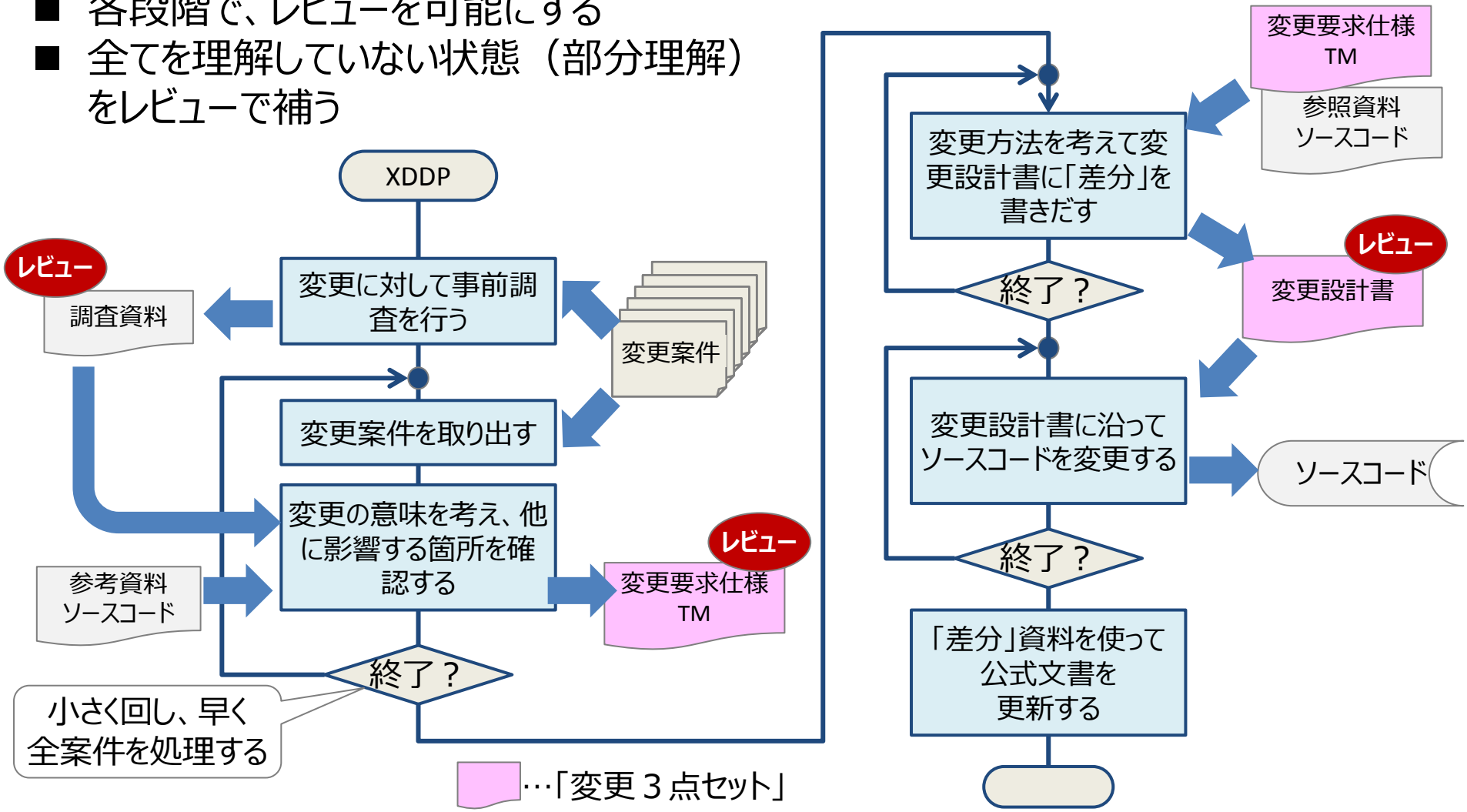


※プロジェクトの状況にあわせて、PFDで「開発プロセス」を設計する

3.XDDPのプロセス

変更プロセスは成果物に対して小さく回す

- 各段階で、レビューを可能にする
- 全てを理解していない状態（部分理解）をレビューで補う

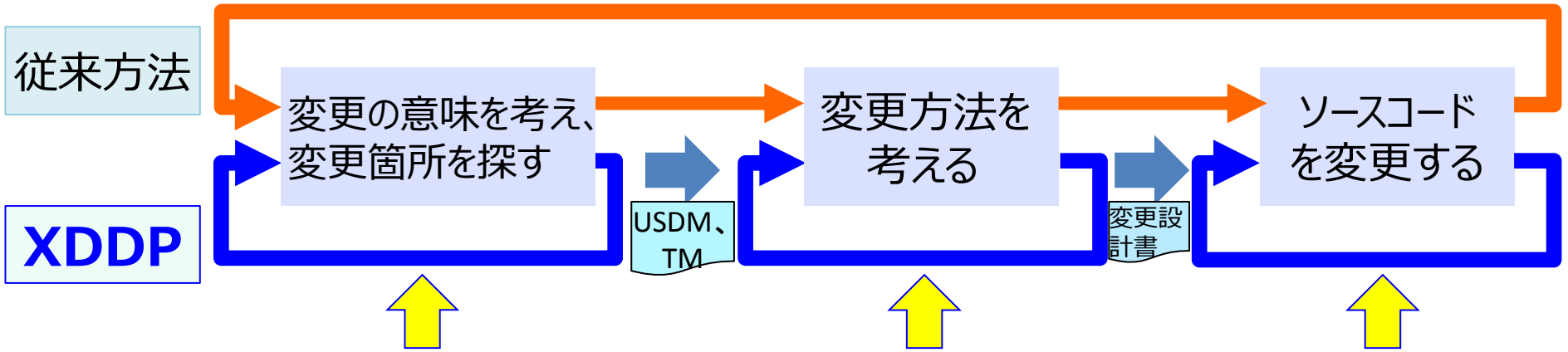


3.XDDPのプロセス

XDDPは性質の異なる行為を混同させない

従来の方法では、変更１件ごとに「目線」が変化する

あの時の仕様と変更ってどうだった？あの時の修正、こうしておけばよかった。。。



変更依頼の意味を考えながら変更箇所を探し、他に影響する箇所に意識を巡らせる

他の変更との絡みに配慮しながら最適な変更方法を考える

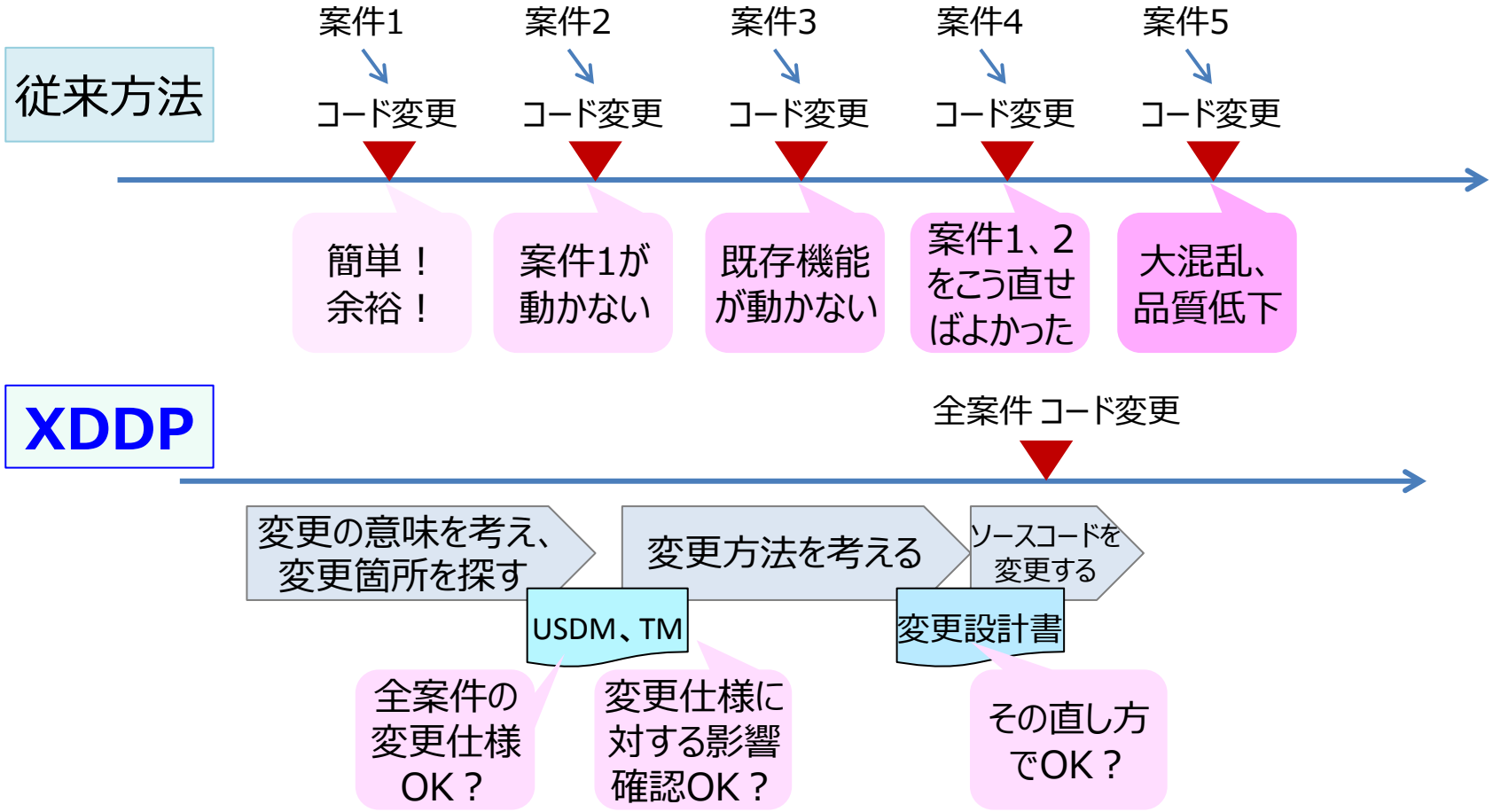
細かな「事実」に注意しながら間違えないように変更する

性質の異なる行為を混同させないようにする仕掛けが
XDDPに組み込まれている

3.XDDPのプロセス

XDDPは性質の異なる行為を混同させない

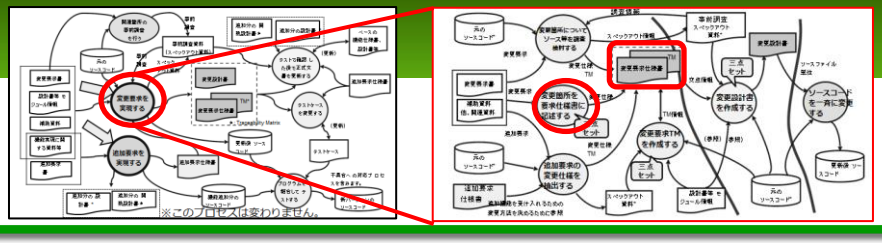
■ ソースコードを修正するタイミングの違い



4. XDDPの成果物

4.XDDPの成果物
変更要求仕様書

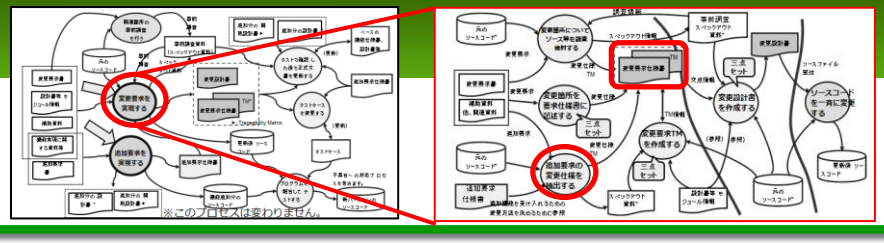
【変更３点セット】



■ USDMで変更要求仕様書を作成する

変更要求	REQ01	目覚しの設定できる時刻を1つから5つに増やす
	理由	ユーザーへのアンケートにより要望が多かったから
	＜時刻の設定画面の変更＞	
	□□□	時刻の設定の項目を1つから5つにする。
	＜設定時刻の保存の変更＞	
	□□□	設定した時刻を保存するメモリを1つから5つの配列にする。
	＜目覚し時刻になったときの変更＞	
	□□□	目覚し時刻の判断の処理で、保存したメモリ1つから5つの配列を見るよう変更する。

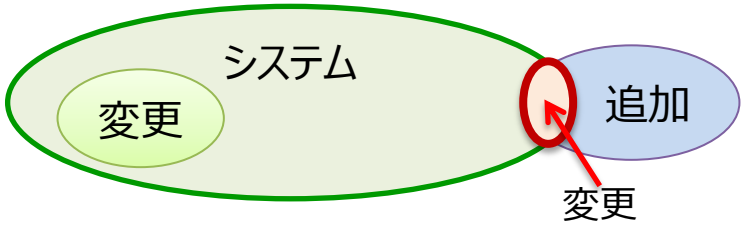
・変更したいことを、関係者が特定(Specify)するための文書
・USDMで動詞より仕様を導出し記述することで、「漏れ」や「重複」を防ぐことができる
・「before/after」で表現することで修正箇所や影響範囲を特定しやすくする
・「適切な要求の細分化」や「仕様からの適切な要求の導出」も可能



4.XDDPの成果物

追加に伴う変更要求仕様書

■ 追加がある場合、追加によるシステムの変更を変更要求仕様書に記述

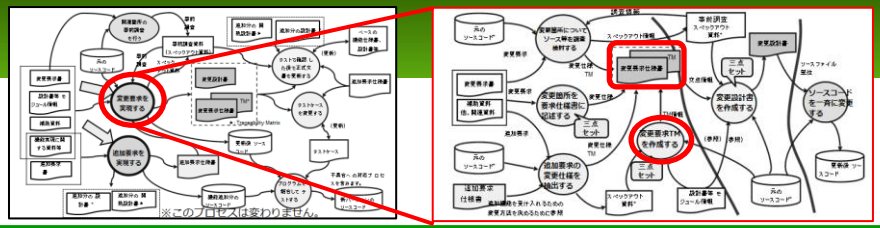


変更要求	目覚しのメロディを、リストから選択できるようにする機能を追加する。	
	理由	ユーザーへのアンケートにより要望が多かったから
	＜目覚しメロディ選択画面の追加＞	
	□□□	目覚し時刻を設定する画面で、スペースのある右下に「♪」アイコンを設ける
	□□□	「♪」アイコンが押されたら、時刻設定画面の右側にメロディ選択画面を追加する。
	＜目覚し時刻になったときの変更＞	
	□□□	目覚し時刻になったかの判断の処理で、固定のメロディだったのを、選択したメロディを鳴らすようにする。

・追加機能が既存システムで問題なく作動するかどうか確認できる
・今まで動いていた部分に影響はないか、確認できる

4.XDDPの成果物
変更トレサビリティマトリックス

【変更３点セット】



■ 変更要求仕様書を**「行」**、機能名ソース名を**「列」**にマトリックスを作成

・“気づき”を誘うために、構成を安定させ、可能な限り「全て」を表現する
・文書やビルドパラメータ等も扱う

#	変更要求・仕様	B	C	D	E	F	G	H
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・・に変更する			F1				
5.4	表示用メモリの配置を・・に変える			F1		F7		
5.5	受信時のエラー処理を変更する	F9			F3	F4		

異なる変更仕様で同じ関数を変更する状況が見える

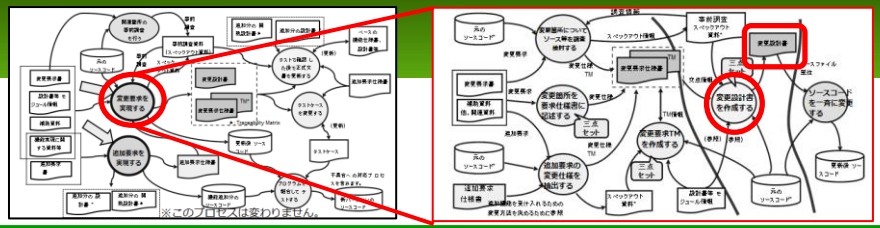
変更情報（差分）のみ

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ことが見える

・変更要求仕様を適切に細分化することで、変更設計書が扱うテーマが単一化する
・変更箇所の関連性から気づき(修正箇所・影響箇所)を誘導する効果もあり、バグを未然に防ぐ効果もある
・変更TMを参照することによって凝集性や複雑性が一目で分かり、リファクタリングをしたほうがいいか判断できるようになる

4.XDDPの成果物
変更設計書

【変更３点セット】

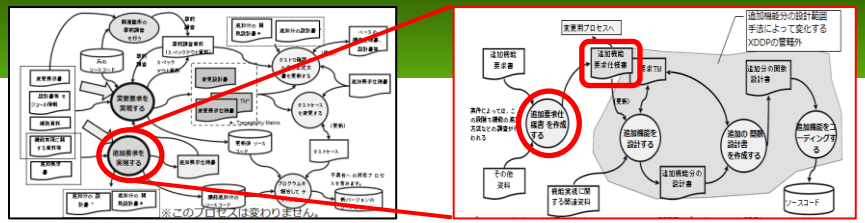


■ 変更要求仕様書を元に変更設計書を作成する

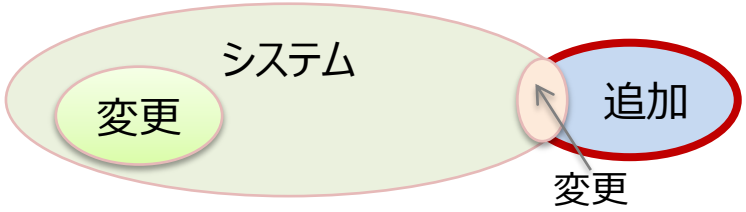
変更設計書		
機能名	目覚し時刻設定	
ファイル名	set.cpp	
関数名	funcS1()	
変更内容	時刻の設定の項目を1つから5つにする。	
変更詳細	①設定項目としてテキストボックスを5つにする。	
	②テキストボックスは縦に配置する。	

- 具体的な変更方法を、自然言語で変更設計書に記述する
  → 自然言語で記述できることは、変更方法を正しく理解できていることを意味する
- 複数の変更が一箇所に重なる場合、全てを満たす変更方法を検討する
  後で最良の変更方法が見つかった場合、変更設計書に取り入れる
- 変更設計書でまとめることによって、ソースコードを一括で変更が出来るようになる

4.XDDPの成果物
追加要求仕様書(※追加がある場合)

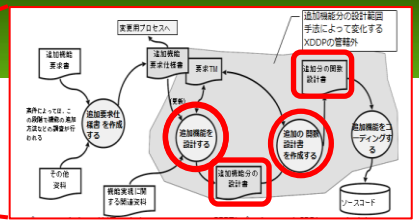


- 追加機能については、USDMで追加要求仕様書を作成する
- 完全新規とは異なり、既存のソフトウェアのアーキテクチャに対応するように記述する



要求	目覚しのメロディを、リストから選択できるようにする。	
	理由	ユーザーへのアンケートにより要望が多かったから
	＜メロディ選択画面＞	
	□□□	メロディリストを表示するメロディ選択画面を設ける
	□□□	1つだけメロディを選択できるようにする。
	＜メロディ選択情報の保存＞	
	□□□	選択したメロディ情報をメモリに保存する。

・追加の要求仕様を、関係者が特定(Specify)するための文書
・要求を振る舞い（動詞で）表現し、動詞に対する仕様を導出することで「漏れ」や「重複」を防ぐことができる
・「適切な要求の細分化」や「仕様からの適切な要求の導出」が可能



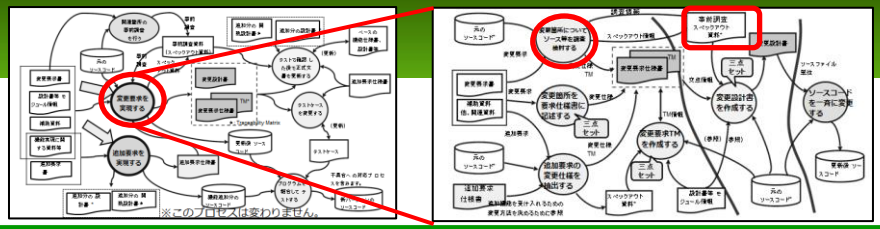
4.XDDPの成果物
追加設計書(※追加がある場合)

■ 追加要求仕様書をもとに追加設計書を作成する
※組織の持っている設計プロセスを適用することが前提

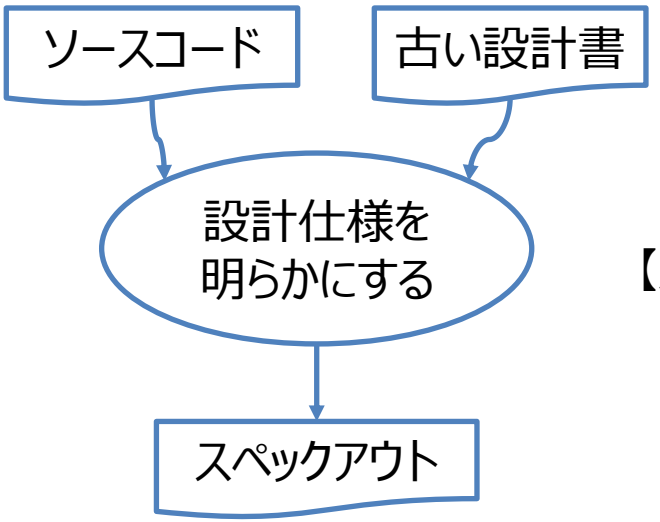
追加設計書		
機能名	目覚しメロディ設定	
ファイル名	mel.cpp	
関数名	funcM1()	
内容	メロディリストを表示するメロディ選択画面を設ける	
詳細	①メロディ情報を取得する。	
	②取得したメロディ情報をもとにメロディ選択画面のメロディリストを編集する。	
	③メロディ選択画面を表示する。	

・既存のソフトウェアのアーキテクチャに対応するように設計する
・追加要求仕様書を元に、組織の既存設計手法で設計書を作成する
・既存の共通関数やグローバル変数(状態)を使用する場合は、他に影響がないように、ここで検討する

4.XDDPの成果物
スペックアウト(※状況に応じて実施)



■ 公式文書がなくソースコードしかない場合、有識者がいない場合に行う事前調査



・プログラムの構造
・処理のタイミング
・データのアクセス

を明確にする

【スペックアウト資料(例)】
※組織が持っている設計手法で用いられている成果物が前提
[構造図] [クラス図] [シーケンス図]
[PAD] [ER図] [状態遷移図] [データ構造]
[機能ソースコード対応表] [CRUD図]

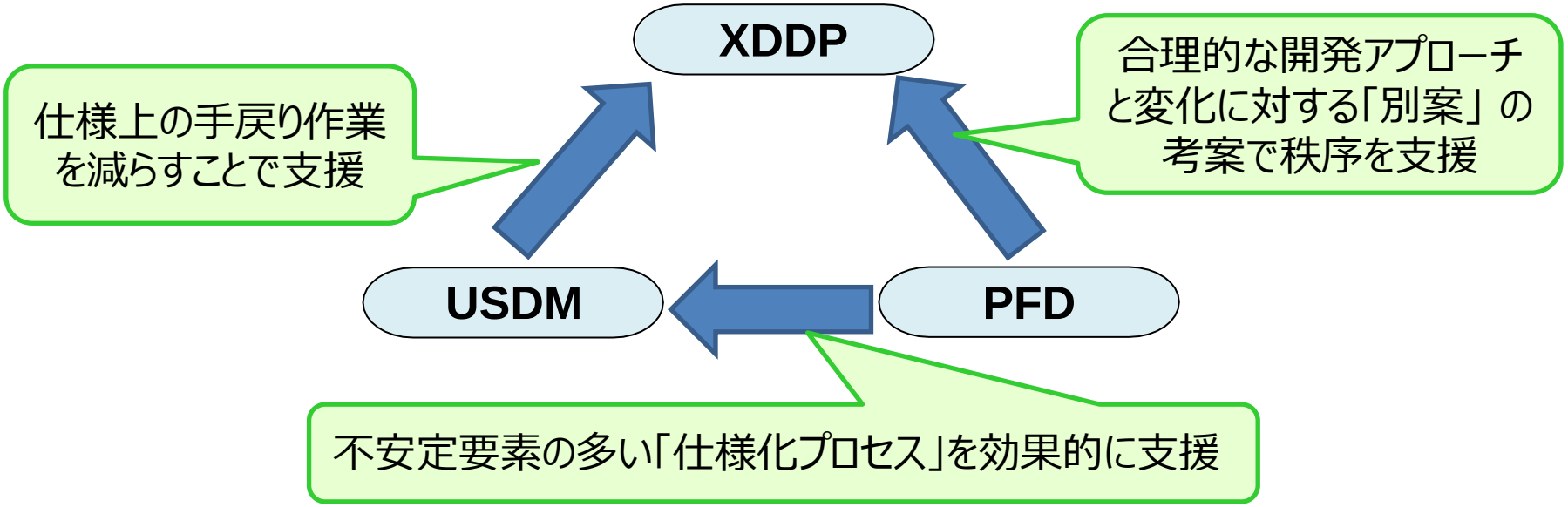
・「部分理解」の中で生じる担当者 の思い込みや勘違いをカバーする
・変更箇所および影響範囲の特定、変更要求仕様書の作成を支援

5. まとめ

5.まとめ

XDDPトライアングル

■「XDDP」は「USDM」と「PFD」の支援の上で成り立っている



PFD	ムダのない合理的な開発アプローチを設計するための技術
USDM	要求仕様を「漏れなく、重複なく」書くための技術
XDDP	派生開発のプロセスを組み立てる技術 「変更３点セット」を中心に派生開発をスムースに推進する

5.まとめ

- XDDPの特徴
  - 「変更と追加のプロセスを分離する」、「成果物に対して小さく回す。性質の異なる行為を混同させない」という特徴を持ったXDDPは、派生開発を行ううえでの最適な開発プロセス
- XDDPの利点
  - 開発途中で変更箇所を振り返る、依頼者に確認するなどの「ながら作業」がなくなる
  - ソースコードを変更する前に不具合を発見することができる
  - すべての変更は「変更設計書」に記述しているため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る
- 「変更3点セット」の理由
  1. 変更箇所を探す
     - 変更依頼の意味を考え、他に影響する箇所に意識を巡らせる **➔変更要求仕様書、TM**
  2. 変更方法を考える
     - 他の変更との絡みに配慮しながら、最適な変更方法を考える **➔変更設計書、TM**
  3. ソースコードを変更する
     - 細かな「事実」に注意 しながら間違えないように変更するソースコードの変更を遅らせる
- 気を付けること
  - 効果を見るため、メトリクスを取ること。特にソースコードの生産性
  - 不具合ゼロで手に入れた時間は、次の開発に向けて準備に使う