System Creates

XDDPによる品質と生産性の同時達成

JaSST'12 Niigata 基調講演資料

2012年3月16日

株式会社 システムクリエイツ
代表取締役 清 水　吉 男
shimz@nifty.com
URL=http://homepage3.nifty.com/koha_hp
派生開発推進協議会　代表
URL=http://www.xddp.jp

System Creates

agenda

1. 派生開発における品質問題
2. 品質のために生産性を犠牲にした
3. XDDPで派生開発のプロセス改善を
4. 追加機能要求仕様書
5. 変更３点セット
6. XDDPでビジネスに勝つ
7. XDDPから「次」へ

USDM：Universal Specification Describing Manner の略．仕様がモレない表現方法として筆者が開発したものです．
XDDP：eXtreme Derivative Development Process の略．派生開発に対応するように筆者が開発したプロセスです．
PFD　：Process Flow Diagram の略．筆者がDFDをアレンジしてプロセスの設計に使いやすくしたものです．

System Creates

5WCSQ（上海）で “Best Paper Award” を受賞

5WCSQ Best Paper award

第五届世界软件质量大会 中国・上海

5th World Congress for Software Quality 2011, Shanghai, China

Process Improvement using XDDP
- Application of XDDP to the Car Navigation System -

Keiji Kobata, Eiji Nakai, Takahiro Tsuda

［JaSST新潟2012基調講演］

5WCSQ（上海）で“Best Paper Award”を受賞

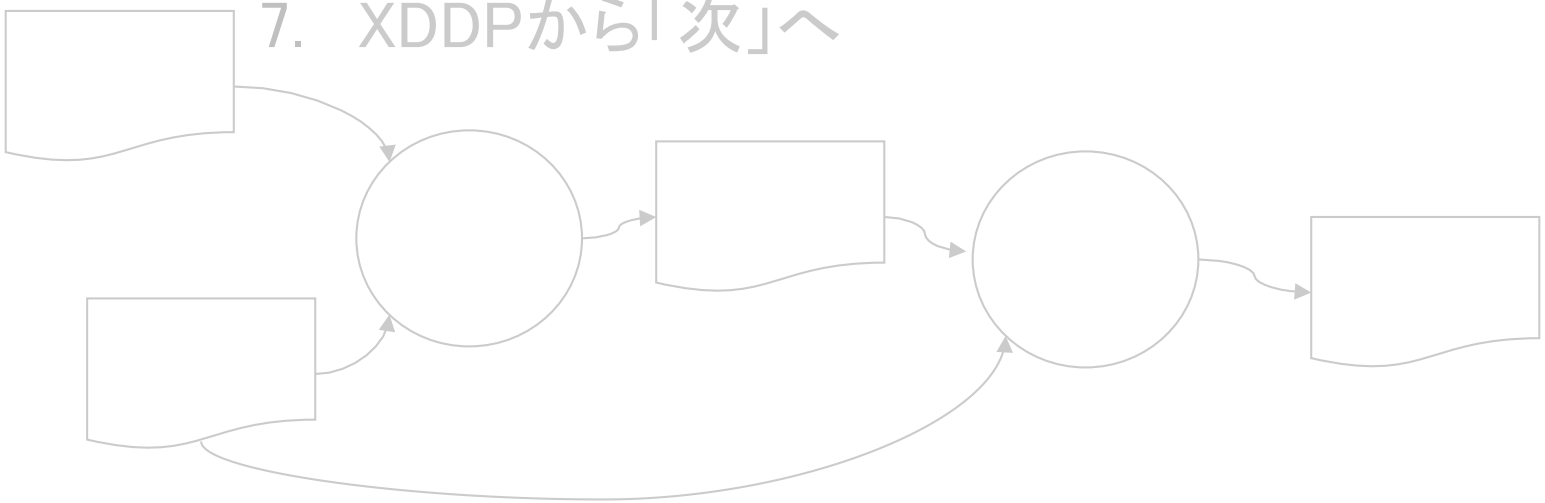
- 11／1～4：上海の復旦で開催された5WCSQ（World Congress for Software Quality：第5回世界ソフトウェア品質会議）で「XDDP」の導入を扱った論文が「Best Paper Award」を受賞

- 論文タイトル

” Process Improvement using XDDP
– Application of XDDP to the Car Navigation System –“
Keiji Kobata, Eiji Nakai, Takahiro Tsuda Denso

- Dr. Patricia McQuaid 氏（ASTQB 会長）が、Keynote 講演で、アリアンロケットの打上げ失敗、パトリオットミサイルの誤射、放射線医療機器の誤照射のトラブルを紹介。これらはすべて派生開発のエラー。

1. 派生開発における品質問題
2. 品質のために生産性を犠牲にした
3. XDDPで派生開発のプロセス改善を
4. 追加機能要求仕様書
5. 変更３点セット
6. XDDPでビジネスに勝つ
7. XDDPから「次」へ



System Creates

「保守開発」の限界

- 「保守開発」の定義
  - 保守開発はJIS（JIS X 0161／ISO14764）で以下のように定義されている

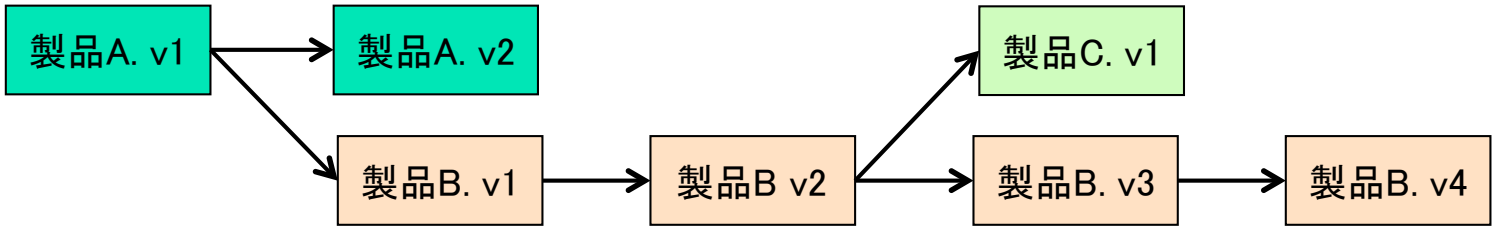
参照　http://www.jisc.go.jp/

保守のタイプ		作業内容
訂正	是正保守	ソフトウェア製品の引き渡し後に発見された問題を訂正するために行う受身の修正.
	予防保守	引き渡し後のソフトウェア製品の潜在的な障害が運用障害になる前に発見し、是正するための修正.
	緊急保守	是正保守実施までシステム運用を確保するための、計画外で一時的な修正.
改良	適応保守	引き渡し後、変化した又は変化している環境において、ソフトウェア製品を使用できるように保ち続けるために実施するソフトウェア製品の修正.
	完全化保守	引き渡し後のソフトウェア製品の潜在的な障害が、故障として現れる前に、検出し訂正するための修正.
	改良保守	新しい要求を満たすための既存のソフトウェア製品への修正.（機能追加を扱う）

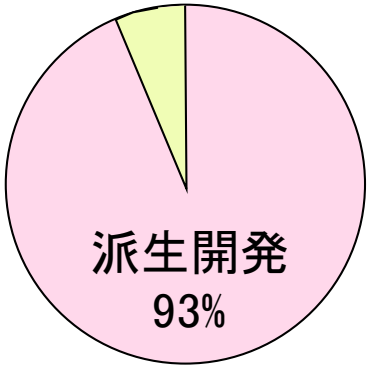
- 従来の「保守」は「是正」と「適応」の世界
- 08年に「改良保守」と「緊急保守」が追加されたが、「是正保守」と同じプロセスで「改良保守」が実施されることによって新たな問題が生じる危険がある

大部分は派生開発

派生開発のイメージ



- 組織によっては、開発案件の90%以上は「派生開発」



- 入社後、一度も新規開発に関わらない可能性大
- 一般に公表されている開発方法は「新規開発向け」
- 組織の標準プロセスも派生開発にマッチしていない

派生開発は「部分理解」の制約を受ける

全体を理解できて
いなかったからナ
（反省会）

現状

- 理解の参考になる成果物はない
- 保守性を無視して作られたソースコード
- これまでの変更で無節操に弄られていてきた
- 読解技術も時間も不足している

全体を理解できる状況にはない

重要

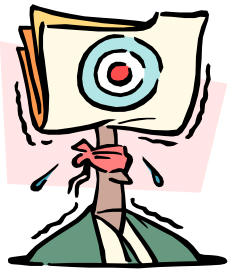
「全体を理解すれば問題は解決する」と思っているかぎり、
理解できなかったときの対応が想定されない

- 派生開発では、「部分理解」の制約の中で変更作業が強いられる
- 担当者の「思い込み」「勘違い」の混入を防ぐことはできない

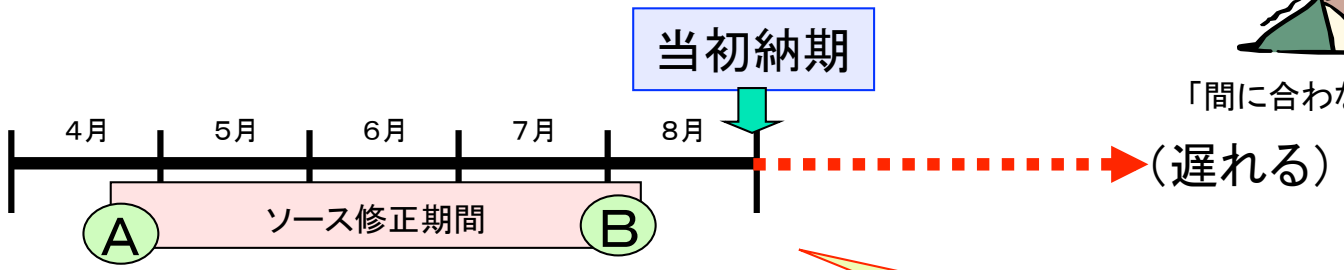
どう、発見するか？

拙速なソースコードの変更が行なわれている

- 開発期間が短いため、“ソースコードの変更を急がなければ間に合わない”という圧力がかかる
- 該当する箇所を見つけ次第に変更してしまう



「間に合わないモンスター」

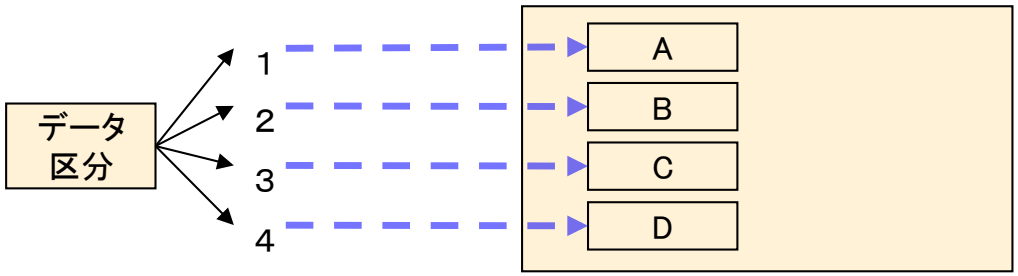


実装工程の生産性が極端に低下

なぜ、こんなことが
起きてしまうのか？

- 結果・・・当初納期から大幅に遅れる
  - この状況に対して「次回はもっと早く変更しないと間に合わない」という

変更の実現方法も1つとは限らない



依頼：区分が2で「C1」の条件のときは表示の位置をB欄からC欄に変更する

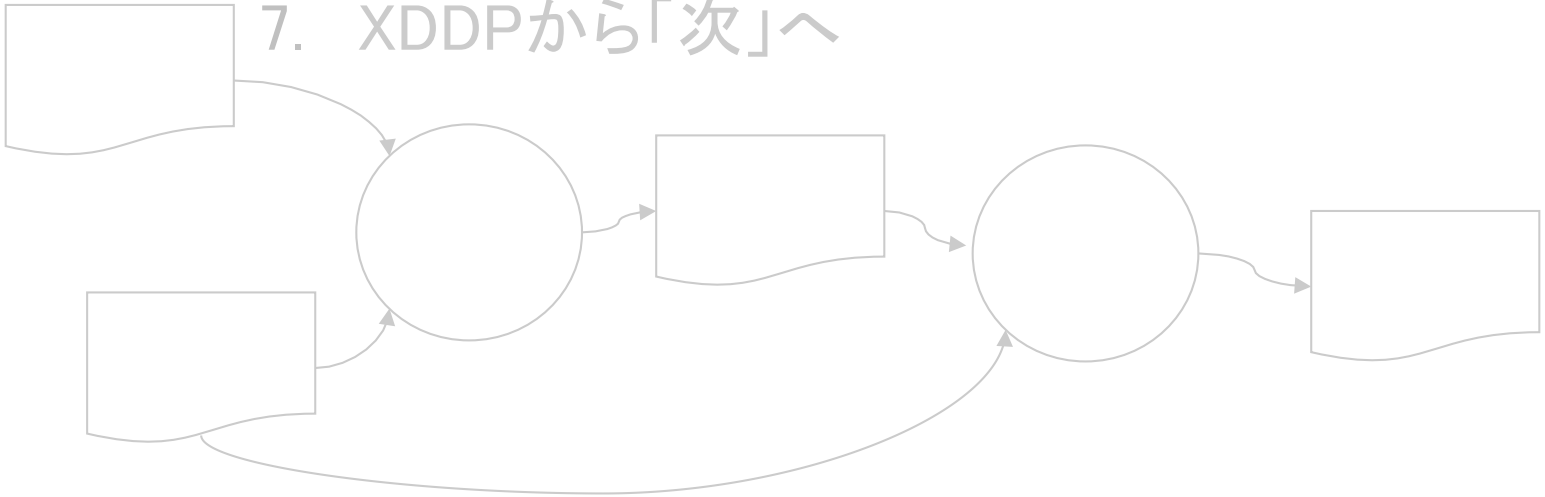
対応
- ①：表示処理のところで条件を判断して表示の場所を変更する
- ②：早い段階でデータの区分を変更してC欄にでるようにする
- ③：データ区分とは別に表示区分のコードを導入して対応する

ほとんど「対応①」が選択される…なぜ？

テストでは、どの対応でも「OK」となる

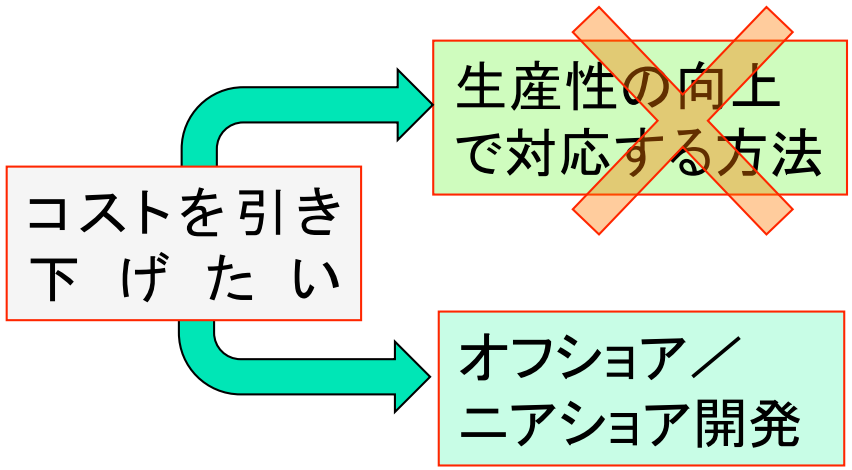
- 変更済みソースコードレビューでこの問題を発見できるか？

1. 派生開発における品質問題
2. 品質のために生産性を犠牲にした
3. XDDPで派生開発のプロセス改善を
4. 追加機能要求仕様書
5. 変更３点セット
6. XDDPでビジネスに勝つ
7. XDDPから「次」へ



少しでも安いところへ発注！

- 基本・・・安いところに発注する
- 誤解・・・「ソフトウェア開発の発注」と「部材の調達」と同じ発想？
  - 「Faster, Cheaper, Worse」＝安かろう、悪かろう



品質と生産性を同時に達成するようなプロセスを設計できない

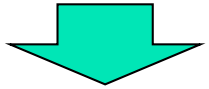
準備もせずに「オフショア開発」を競ったことで混乱

コスト削減効果は？

品質を下げただけ？

なぜ、価格の叩き合いに巻き込まれるのか？

- 理由
  - 品質と生産性において「ダントツ」の状態にない
  - そのため、選択肢が発注側にある
  - 受注側・・・部材の調達との違いをアピールしていない



- 対応策：段階１
  - 生産性を30%以上向上させる
  - 現行のコストで品質と納期を達成する
- 対応策：段階２
  - 選択肢を発注側から獲得する

中国のコストは、
まもなく優位性を
失う

System Creates

自分たちのソフトウェア開発の生産性は？

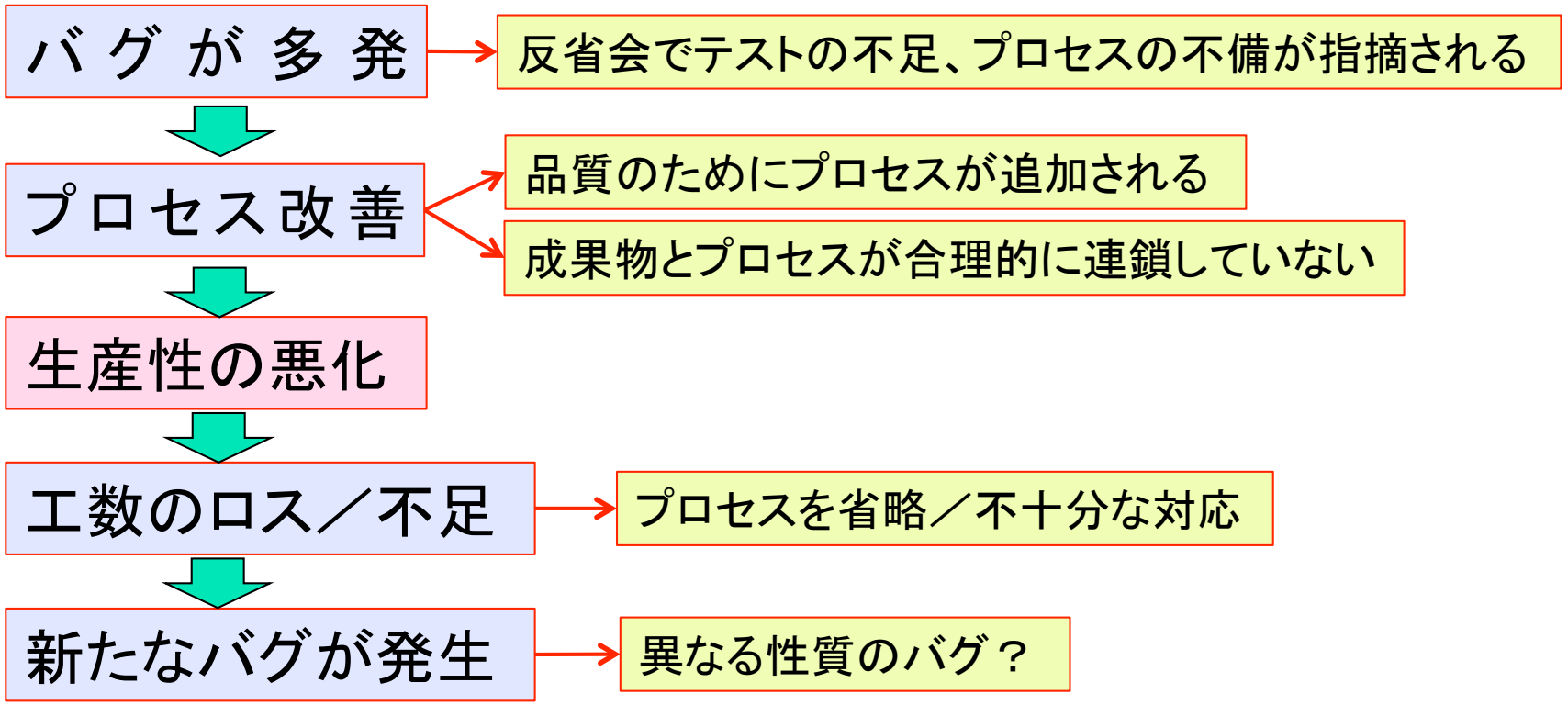
- ソフトウェア開発に関する生産性データは？

- 多くの組織は、品質のデータは収集している
  - 「KLOC」あたりのバグの発生率
  - レビューでの指摘率
  - ベースライン設定後の仕様変更率　など

生産性データはなぜ収集していないのか?

生産性が犠牲に？

- 生産性を犠牲にした「改善」行われていないか？



System Creates

「品質」と「生産性」は相反すると考えられている？

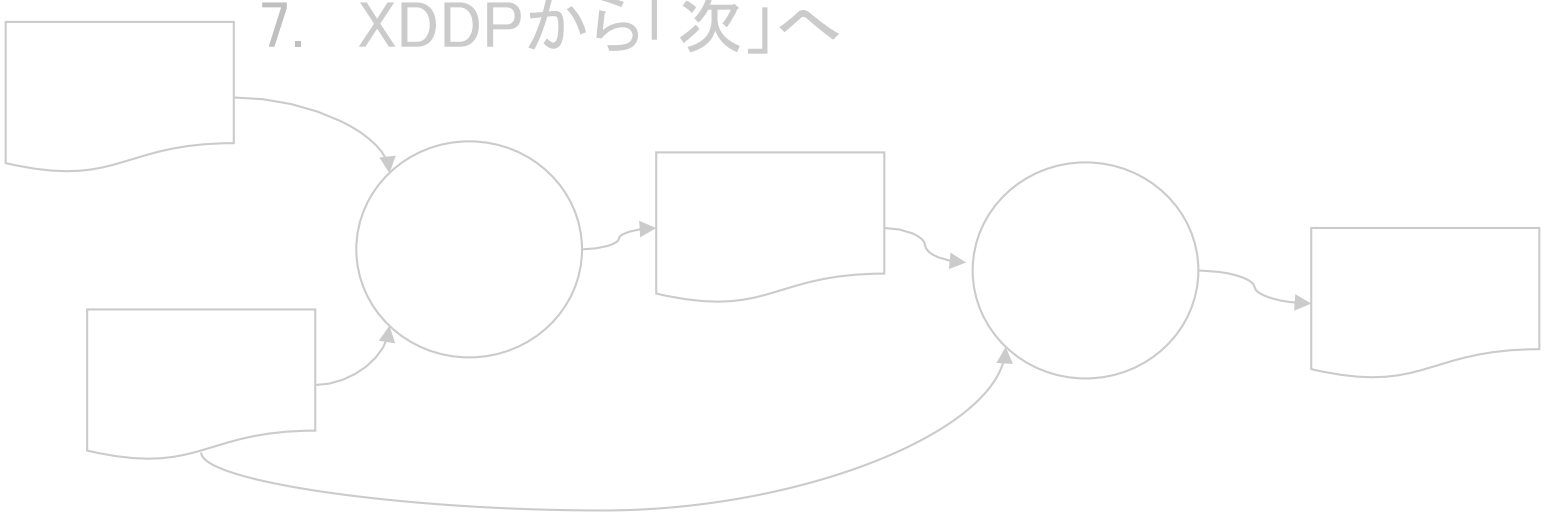
- これまでのプロセス改善の取り組み・・・「品質」が中心
  - 合理性を欠いたプロセスや成果物が変更される
- 「品質」＝「テストの強化」？
  - トータルでの対応になっていない

「品質」のためには「生産性」を犠牲にするのもやむを得ないの？

「プロセス」に対して「非人間的」と揶揄される原因にもなる

- 「生産性」を考慮しないプロセスに合理的はない

1. 派生開発における品質問題
2. 品質のために生産性を犠牲にした
3. XDDPで派生開発のプロセス改善を
4. 追加機能要求仕様書
5. 変更３点セット
6. XDDPでビジネスに勝つ
7. XDDPから「次」へ



「XDDP」の誕生

４０数項目の機能追加と変更を３ヶ月で！

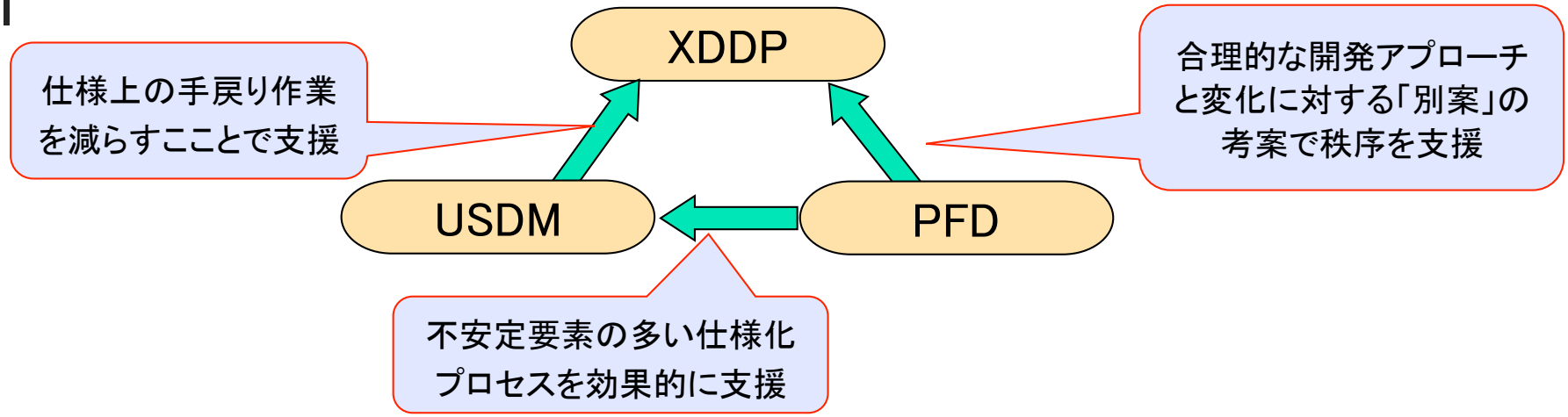
- アメリカの顧客からの要求（1978年）
  - “初めて”づくし
  - 国内の客先には、その製品の仕様を知る人はいない
- １週間で、機能追加と変更に分けたプロセスを設計
- プロセスの合理性はシミュレーションで確認

品質	変更箇所（理解した仕様を含む）を「before/after」で記述し、Faxでレビューを依頼
	「before」は、現状の仕様を私が理解した状態 → レビューで確認
生産性	必要不可欠な最小限の成果物とプロセスの合理的な連鎖で構成 見積りが可能な状態を確保

３ヶ月の納期を達成

XDDPトライアングル

- 「XDDP」は、「USDM」と「PFD」の支援を受けて成り立っている



- 「PFD」
  - ムダのない合理的な開発アプローチを設計し「計画書」に繋げる
  - 途中で生じる変化に対して適切にプロセスと成果物を調整する
- 「USDM」
  - 追加機能の仕様モレを減らしたり、変更箇所（変更仕様）の抽出モレを減らすことで、短納期などの制約の中での開発作業を支援する

System Creates

USDMにおける「要求仕様書」の考え方

- そもそも「要求仕様書」とは、どういう文書なのか？

USDMにおける定義

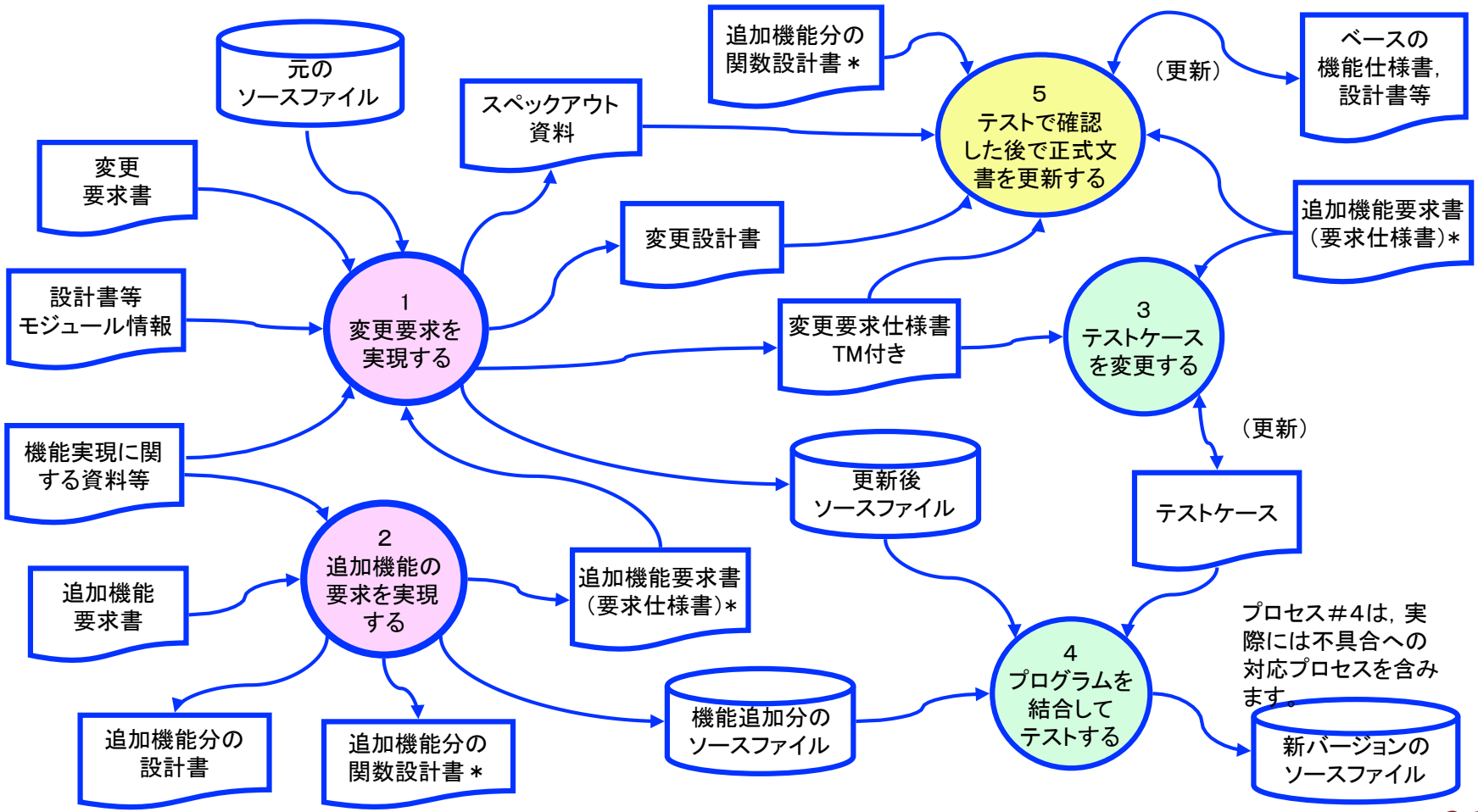
要求仕様書とは、今回のプロジェクトで実現して欲しいこと（Requirements）について、“作ることの関係者”が実現内容についての認識を特定（Specify）できている文書

「USDM」では「Specification」とは関係者が同じものをイメージできる状態のことと捉える

	要求仕様書	機能仕様書（等）
目的	作るためのもの	機能を説明するもの
関係者	計画書で特定されている	特定されていない
表現	関係者がSpecifyできる状態	一般的な記述
納期やコスト	背後に背負っている	意識されない
バージョン管理	今回だけの文書	ソースコードと対でバージョンアップする

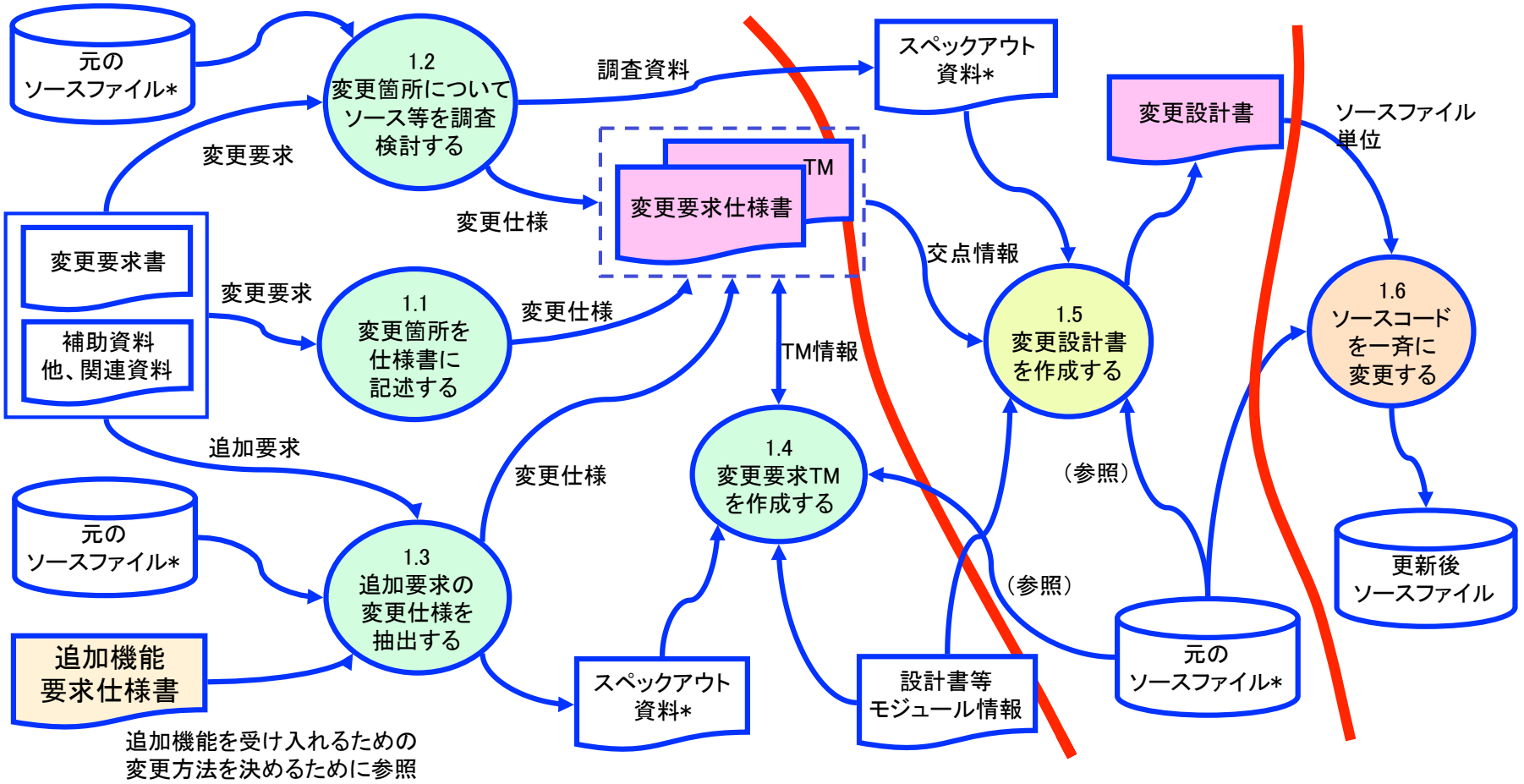
(PFD)
派生開発アプローチ(上位)

- 「追加」と「変更」では「①変更用PFD」「②追加用PFD」として下位のPFDで展開する
- 公式文書は、テスト終了後に「差分」情報によって更新する



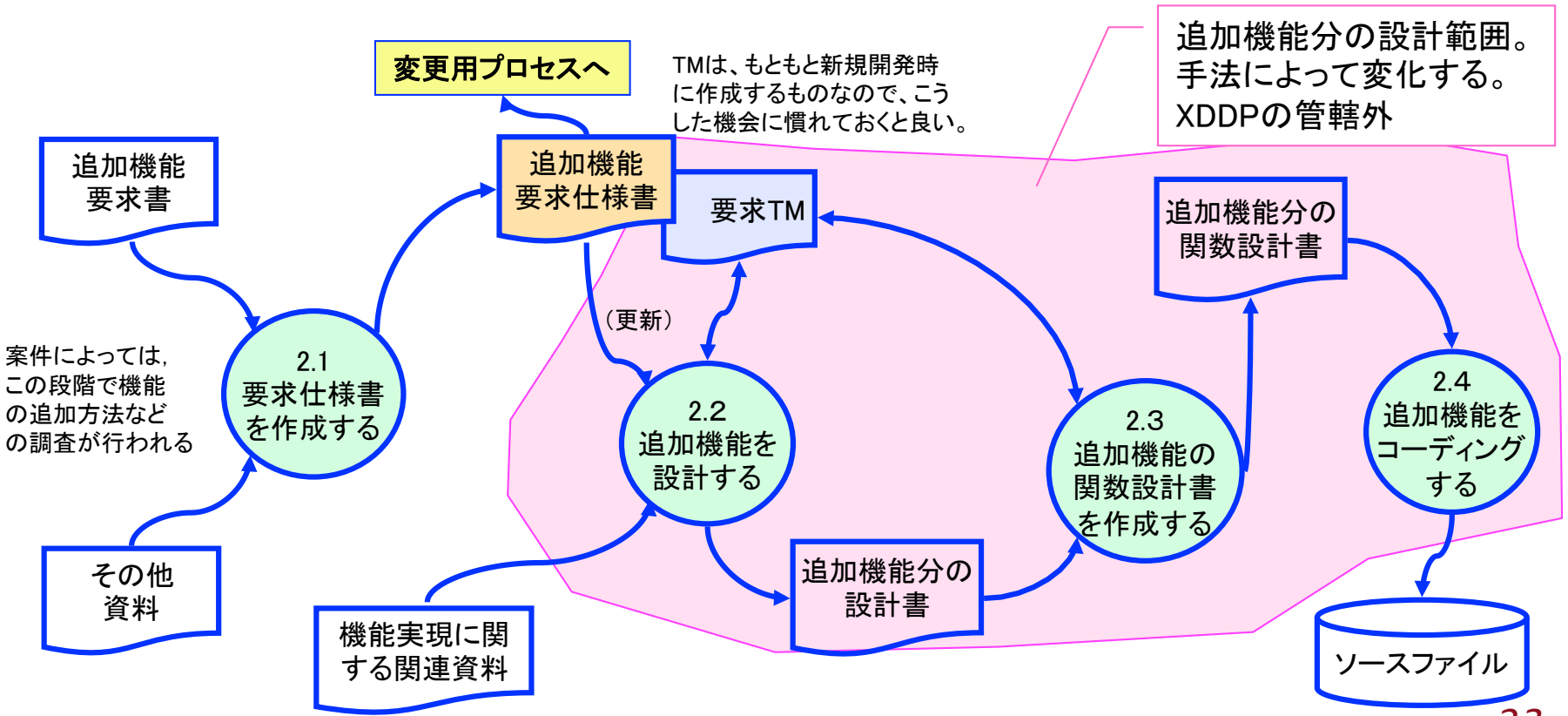
(PFD)
変更プロセス:すべての変更を扱うプロセス

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



(PFD)
機能追加プロセス：機能追加を扱うプロセス

- 採用する設計手法の違いによっていくつかのプロセスは変化する
  - 変更プロセスの「1.3」が終了すれば「2.2」以降に取り掛かる
  - 機能追加がないときは、このプロセスは存在しない

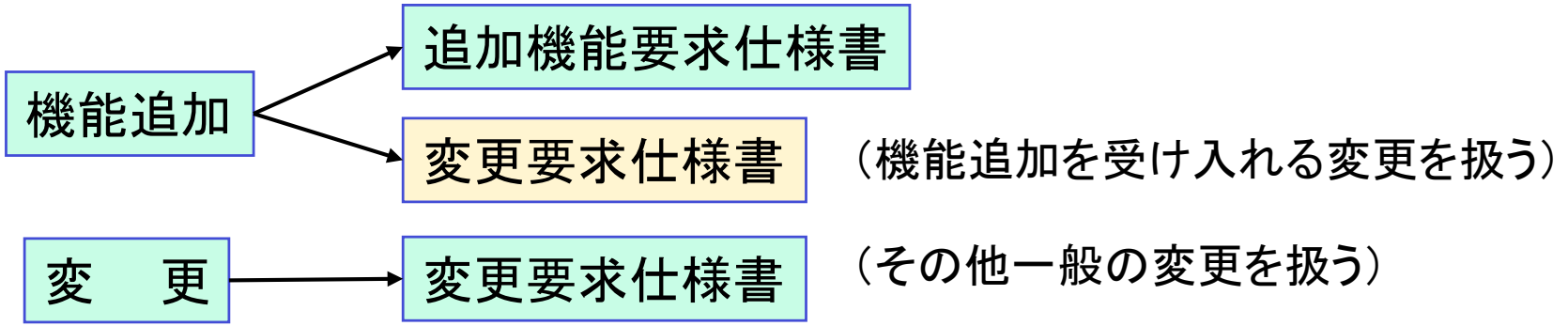


System Creates

派生開発では2種類の要求仕様書が必要

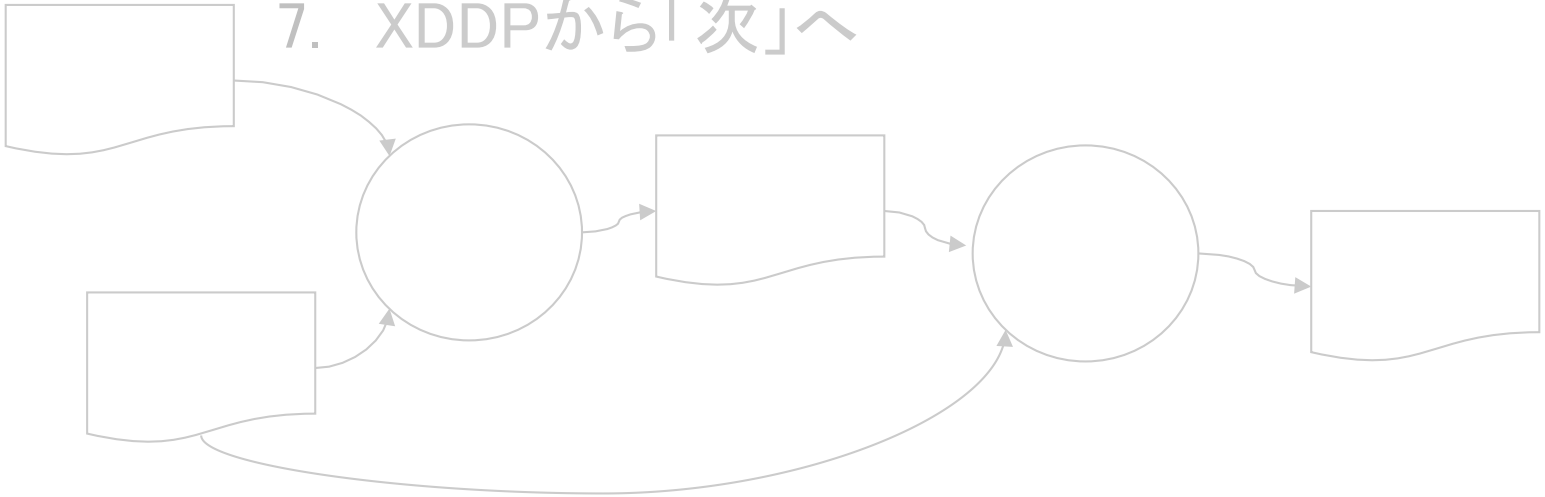
- 要求仕様書が異なる以上、対応するプロセスも異なるべき

	要求	対　応	要求仕様書	対応プロセス
機能レベル	追加	機能追加として扱う	追加分＝追加機能要求仕様書	追加用プロセス
			変更分＝変更要求仕様書	変更用プロセス
	移植	通常は変更として扱う	変更要求仕様書	
	削除	変更として扱う		
仕様レベル	追加	変更として扱う		
	変更			
	削除			



System Creates

1. 派生開発における品質問題
2. 品質のために生産性を犠牲にした
3. XDDPで派生開発のプロセス改善を
4. **追加機能要求仕様書**
5. 変更３点セット
6. XDDPでビジネスに勝つ
7. XDDPから「次」へ

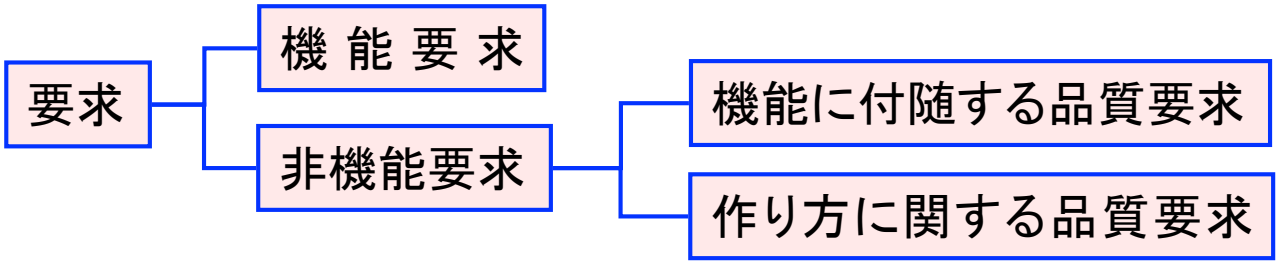


「追加機能要求仕様書」

- 追加機能を扱う要求仕様書
  - 新規開発時の要求仕様書と基本的に同じ構成

今回のプロジェクトで追加実現したいこと（Requirements）について、特定された関係者がその機能の内容を特定した（Specify）ことがまとめられた文書

  - 実現方法は「設計プロセス」で選択され評価される
  - 作るための文書だから「作り方の品質要求」も含まれる



  - 「納期」や「コスト」も背負っており、表現の多様性を許容する必要がある
  - “関係者”は特定されている

（追加機能要求仕様書）

要求は振る舞いの中の「動詞」を表現する

“仕様”は、要求の中の「動詞」および「目的語」に存在する

USDMの基本的な考え方

- “振る舞い”を要求として表現する
- 要求に含まれる「動詞」をすべて表現することを目指す
- 要求には、それを必要とする「理由」をつける

要求	DA07	計測データを受信し、平均値を算出しながらリアルタイムに表示し、異常値が検出されたときは警告を表示する
	理由	計測データは熱によって変動しやすいため、内部の変化を早期に発見したい

- この要求の「赤字」の部分について仕様を展開することになる
  - 受信方法
  - 平均値の算出方法
  - 計測データの表示
  - 異常値の判断
  - 警告の表示

要求にすべての「動詞」が表現されていれば、これらについて仕様を展開すればよい

System Creates

分割・階層化して要求の範囲を狭める

- 範囲の広い要求＝そこに含まれる「動詞」が多い＝＞分割・階層化
  - 上位要求：主要な動詞によって振る舞いの範囲と特徴を表現する
  - 下位要求：そこに含まれるすべての動詞を表現する

要求	MAL01	事前に指定された受信および送信した電子メールをキーワードで検索して、選択した電子メールをメーラーに繋いで再利用したい	
	理由	メールが多くて、関連するメールを 探すのに手間取る	
	要求	MAL01-01	表示された検索グループの中から一つを指定する
		理由	検索グループが複数あるから
	要求	MAL01-02	いくつかのキーワードの入力を受付それらを組み合わせて検索する
		理由	可能性のあるキーワード（複数）で探したい
	要求	MAL01-03	検索結果を表示し、見つかったときは「subject」などの情報を一覧で表示する
		理由	Subjectと日付などから目的のメールを探すことになる
	要求	MAL01-04	一覧の中から選択されたメールを開く
		理由	中身を見ないと分からないから
	要求	MAL01-05	一つのメールを開いた状態でメーラーに繋いで編集できるようにする
		理由	コピーの手間を省いて編集の操作に入りたい

System Creates

仕様の＜グループ＞を先に設定する

- 「動詞」及び「目的語」をから仕様の＜グループ＞を設定する
  - 要求の範囲が適切に制御された状況では、すべての「動詞」が見える
- この後の仕様化の作業では、人を投入することが可能になる

要求	RSV01-03	予約操作を終了したときは、予約を確定させてクレジット処理を行い、登録されているメールアドレスに予約状況を送信する
	理由	顧客に予約が成立したことを示すことと、あとで確認できるようにするため
	説明	クレジット情報の確認はログイン時に行っている
	＜メールアドレスの確認＞	
□ □ □		
	＜予約番号の付与と予約結果の表示＞	
□ □ □		
	＜クレジット処理＞	
□ □ □		
	＜メールの編集と送信＞	
□ □ □		

時系列に整理している

ここまでは「要求」を記述する作業の範囲

System Creates

要求仕様の条件

- 要求に含まれる“具体的”な処理や振る舞いを表現したもの

- 仕様は「要求」から導出される
  - 全ての仕様は、いずれかの要求に属し、すべてソースコードに変換される

- 実現可能性が見える（設計担当者）
  - 前提条件や処理や振る舞いのための「制約」なども含むこと
  - 具体的であるために“仕様間の矛盾”が見える
  - 設計の様子がイメージできることがある

- 検証可能性が見える（テスト担当者）
  - その仕様が実現していることを検証する方法が見える
  - 具体的であることで、関係者の間で違った意味に解釈されない

- 品質要求も例外ではない
  - 品質要求にも、実現可能性と検証可能性は求められる

System Creates

仕様＜グループ＞の中で仕様を抽出する

- 基本的に一つの「動詞」の単位なので、仕様の抽出は容易になる
  - レビューアにとってもレビューしやすい

要求	RSV01-03	予約操作を終了したときは、予約を確定させてクレジット処理を行い、登録されているメールアドレスに予約状況を送信する
	理由	顧客に予約が成立したことを示すことと、あとで確認できるようにするため
	＜メールアドレスの確認＞	
□ □ □	RSV01-03-1	予約結果を送信するメールアドレスを確認する為に、登録されているアドレスを表示する
□ □ □	RSV01-03-2	メールアドレスの変更がある時は、変更を受付ける
□ □ □	RSV01-03-3	メールアドレスが変更された時は、再度入力を求めて確認する
	＜予約番号の付与と予約結果の表示＞	
□ □ □	RSV01-03-10	今回の一連の予約操作に対して、「予約番号」を付与する
□ □ □	RSV01-03-11	「ご予約状況」として以下の情報を表示する
	＜クレジット処理＞	
□ □ □	RSV01-03-15	事前に登録しているクレジットカードの番号を使って、合計金額をクレジット会社に送信する
	＜メールの編集と送信＞	
□ □ □	RSV01-03-20	「ご予約状況」と同じ内容の情報を編集する（図12．3参照）
□ □ □	RSV01-03-21	確認されているメールアドレスに送信する
□ □ □	RSV01-03-22	送信エラーの状態であれば、画面の印刷を促すメッセージを出す

実際には、もっと詳細化が必要

System Creates

仕様変更率“5%以下”を目指す

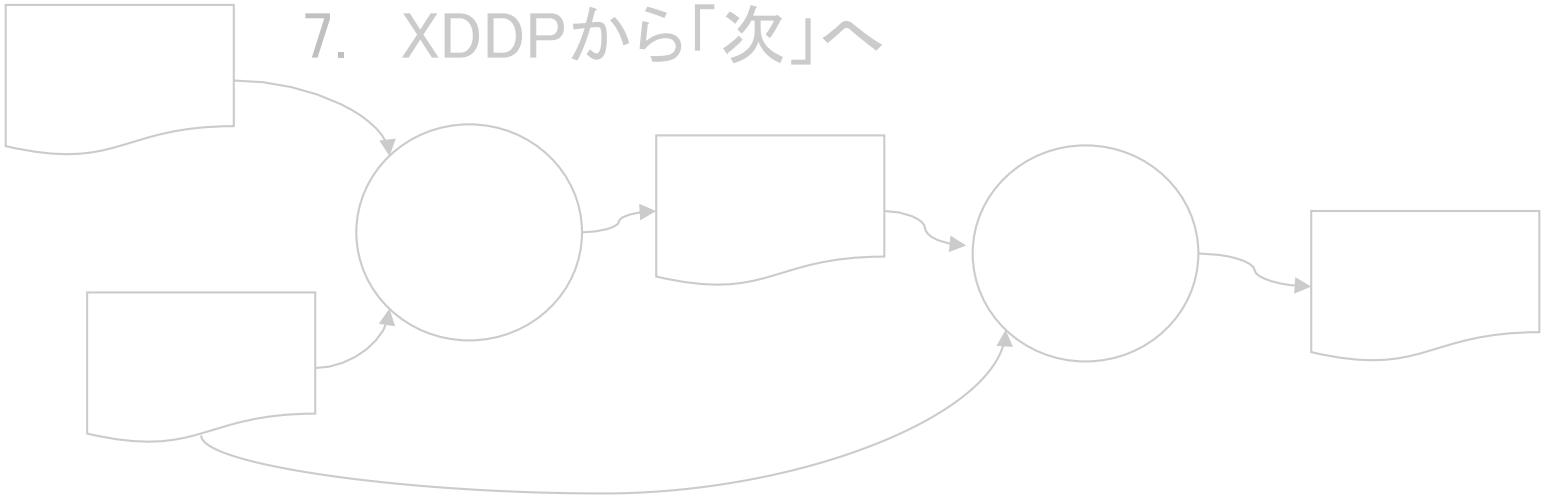
- 仕様の詳細さが適切かどうかを測る方法は？
  - ① 要求仕様のレビューの指摘件数（指摘率）
  - ② 仕様関係のバグの発生率（KLOC単位）
  - ③ 要求仕様のベースライン設定後の仕様変更率
- ベースライン設定後の仕様変更率が5%以下に収まることを目指す

$$\left(\frac{\text{変更仕様数}}{\text{総仕様数}} \times 100\right) \leqq 5\%$$

- （ある程度）母数が多くなれば変更率が下がる
- 5%以下になると要件管理の対応が容易になる

System Creates

1. 派生開発における品質問題
2. 品質のために生産性を犠牲にした
3. XDDPで派生開発のプロセス改善を
4. 追加機能要求仕様書
5. **変更３点セット**
6. XDDPでビジネスに勝つ
7. XDDPから「次」へ



「変更要求仕様書」を導入する

「要求仕様書」の概念から発展したもの

- 「XDDP」では、すべての変更を扱う「要求仕様書」として「変更要求仕様書」を導入する

> 今回の派生開発で変更したいこと（Change Requirements）について、特定された関係者が変更内容まで含めてその内容について特定（Specify）したことがまとめられた文書

- 変更の実現方法まで特定する（Specify）ために、変更仕様としては最終的にソースコード（関数仕様）のレベルで記述する
  - 【理由】・・・変更にははっきりとした「設計プロセス」がない
  - 変更で行われるのは、通常は既に設計されたものを変更するだけであり、新たに設計することはない
  - 具体的に変更する関数名は「TM」がカバーする

（変更要求仕様書）

変更要求の範囲と理由を表現する

- 変更が及ぶ“範囲”を表現し、何をどのように変更したいのかを記述することで、「変更要求」としての役割を果たす
- 「理由」を付けて変更要求の意図を明らかにする
- 変更要求も変更仕様も必ず「before / after」で表現する
  - 「before」は現状を表現し、該当カ所、影響箇所を探すのに有効
  - 「追加する」「削除する」はそれ自体に「before / after」を含んでいる
- 追加機能を受け入れるための変更も扱う

要求	LST.01	商品一覧と在庫確認リストの商品名の横に写真の表示を追加する
	理由	派生品が多くなったことで、商品名を間違えるケースが増えているから

要求	DSP.01	計測したデータのマルチ画面表示で、縦に並べて表示出来るように変更する
	理由	各形測地点での相違を時系列で一目で見えるようにして、同期のずれを早期に発見したい

System Creates

（変更要求仕様書）

「USDM」による変更仕様の表現

- 変更仕様がいろいろな箇所に散在する場合は、＜グループ＞でまとめる
- 変更箇所を見つけながら「TM (Traseability Matrix)」を作成する

				file	file	file	file
要求	CCL30	加入者データに家族データを追加して家族割りサービスを始める					
	理由	同業他社との競争に勝つため					
		＜加入者データの追加＞					
	□ □ □	CCL30-01	主従区分と10個の家族データを追加する 追加する家族データの属性は以下の通り ・加入者数 ・加入者番号		f1()		
		＜加入者データの表示の変更＞					
	□ □ □	CCL30-05	加入者名の横に主従区分の表示を追加する				f3()
	□ □ □	CCL30-06	主従区分＝主の時は、その横に家族の加入者番号を登録数分表示する			f7()	
	□ □ □	CCL30-07	主従区分＝従の時は、主となる加入者番号を表示する			f7()	
		＜計算方法の変更＞					
	□ □ □	CCL30-10	主従区分＝主に繋がる加入者の利用料金をまとめて計算する方法に変更			f8()	
	□ □ □	CCL30-11	加入者数に応じて以下の割引率の適用を追加する				f9()

（追加／変更要求仕様書）
機能追加には2種類の要求仕様書で対応する

- 現状のシステムに新機能を追加するには、少なくとも1カ所以上の変更（仕様変更）が必要になる

追加機能要求仕様書

動画の保存と再生機能の新規に追加される要求仕様

要求	MOV.01	入手した動画の保存と再生を行う
	理由	カメラ付き携帯電話で静止画と動画が撮れるので、動画も一緒に扱いたい.

新規開発の時の要求仕様と同じ要領で作成する

対応させる

変更要求仕様書

動画の保存と再生機能を追加するするために、既存の「仕様」を変更する部分を扱う

要求	MVC.01	入手した動画の保存と再生機能を追加する	
	理由	カメラ付き携帯電話で静止画と動画が撮れるので、動画も一緒に扱う必要が生じた	
	要求	MVC.01	設定操作のところで、動画を扱うための設定メニューを追加する
	要求	MVC.02	機能選択メニューを変更して、これらの機能を選択出来るようにする
	要求	MVC.03	再生時に必要なバッファーを確保するために既存機能のバッファサイズを減らす
	要求	MVC.04	保存時のファイル名の割り付けルールを変更する

それぞれの下位要求の下に変更仕様を抽出する

（3点セットの関係）

TMを介して変更要求仕様書と変更設計書を繋ぐ

- TMは「変更要求仕様」と「変更設計書」の蝶番の役目を果たす
- TMの部分にいろいろな気付きの仕掛けを配置できる

TM（トレーサビリティ・マトリクス）

変更要求仕様書

#	変更要求・仕様		A	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			F5()			F7()		
	5.5	受信時データの区切りにコードを挿入する		F9()						

変更設計書

変更設計書

モジュールDの中で表示の変更方法についてのみ記述する

変更要求仕様を細かく分けることで、「5.1」と「5.4 」の変更方法に関する記述（変更設計書）を分けることができ、その効果として、レビューの精度が上がる。

変更設計書

モジュールDの中でメモリの配置の変更方法についてのみ記述する

変更設計書

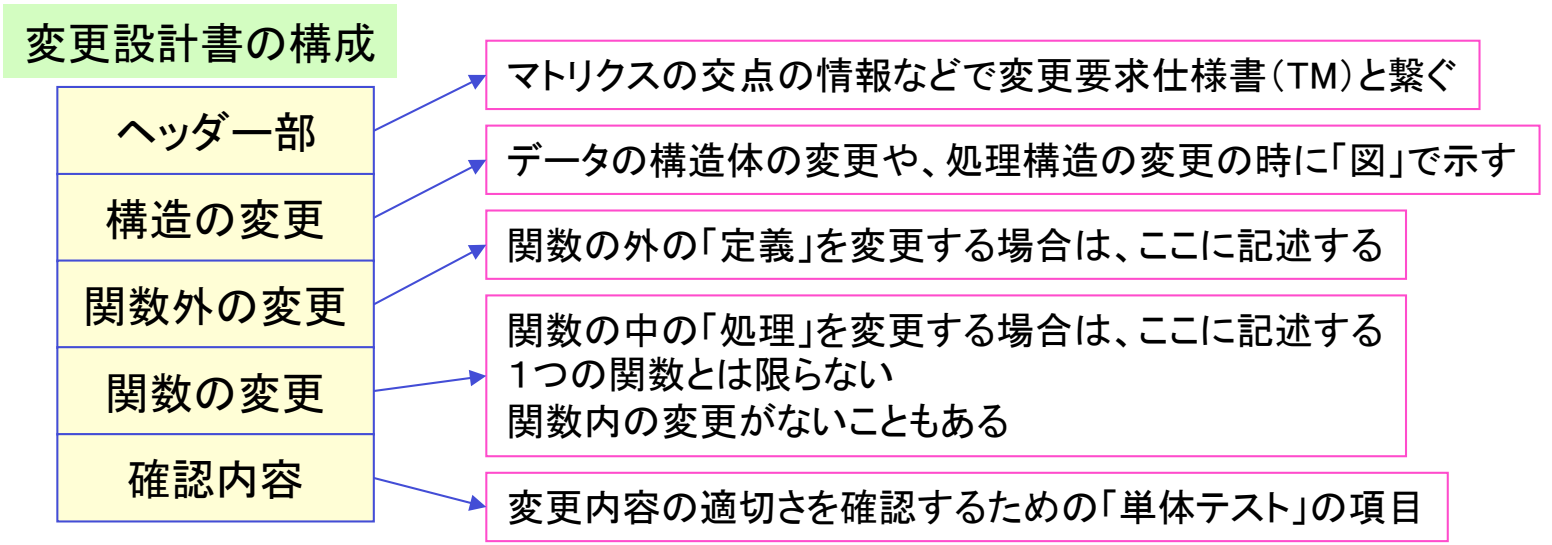
モジュールFの中でメモリの配置の変更方法についてのみ記述する

System Creates

具体的な変更方法は変更設計書で対応する

- TMに「変更あり」が示された単位で、ソースコードレベルの具体的な変更方法を記述する（変更部分だけ）
- 変更に伴うテスト内容（単体テストに対応）もここで記述する
  - ここで最終的に変更方法をレビューする
  - バグの発生時に最初に立ち戻る場所でもある

派生開発においては
バグは変更した箇所
に生じる



System Creates

「3点セット」の成果物の意味

- 「XDDP」の変更プロセスでは3つの成果物を作成する（必須）

成果物	カバー範囲	記述内容	レビュー機会	
変更要求仕様書	What （Why）	何を変更するか？ どのような振る舞いを変更するか なぜ、変更するのか？	○	○
TM（Tracability Matrix）	Where	変更する仕様がどこにあるか？	○	
変更設計書	How	具体的な変更方法を記述する	○	○

- タイミングと視点を変えた効果的なレビューの機会を確保して、「部分理解」の中で生じる担当者の思い込みや勘違いに気付く
- ソースコードを変更する前にレビューでバグを未然に防止する
- 今回の派生開発の変更記録として保存する

一気にソースコードを変更する

- 「XDDP」では、準備された変更設計書を元に、全員が足並みを揃えて一気にソースコードを変更する
  - 事前に変更設計書のレビューを終えている
  - ソースコードの変更に必要な工数は確認されている

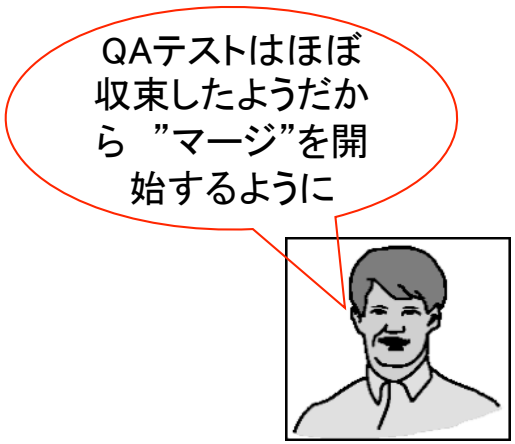
実装プロセス・・・
　単位時間当たりの生産性データを適用できるプロセス

- ソースコードのその箇所を見るのは、通常は「3回目」である
  - 「80～130行／時間」の実装プロセスの生産性を出すこともできる
- ソースコードの変更作業の段階で人を投入することが可能
  - 変更要求仕様の段階で実装工数の予測はできる

ソースコードの変更を「1回」で済ませることで、ソースコードの劣化を防ぐ

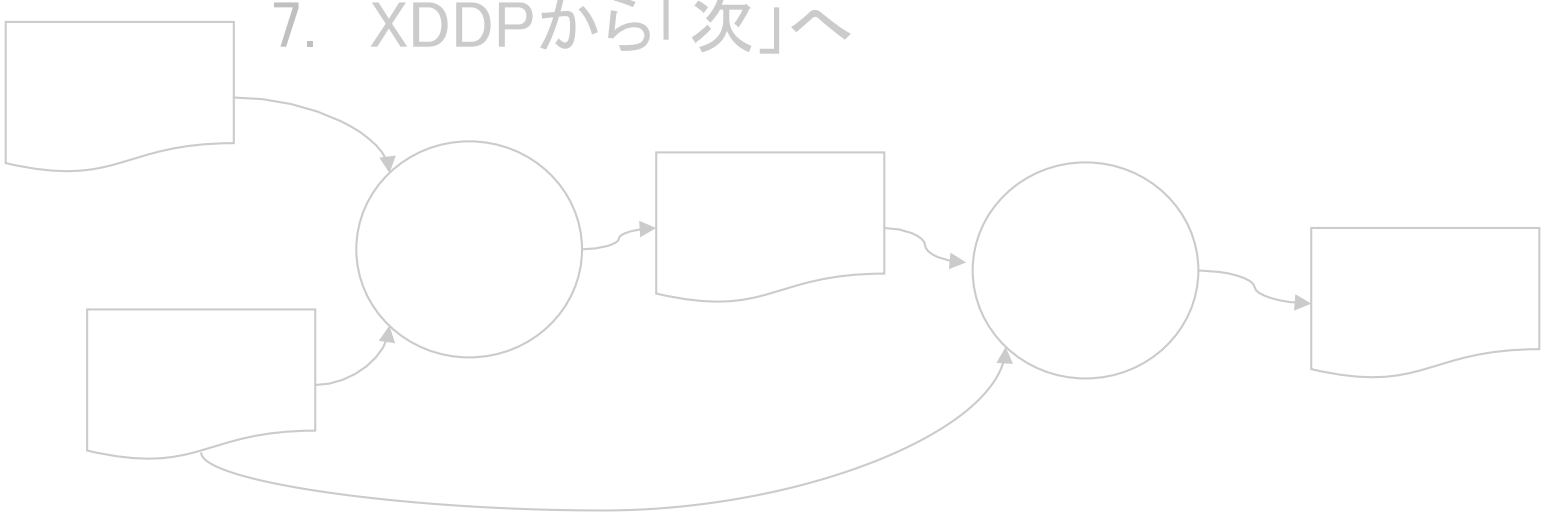
正式文書はテスト後にマージする

- ベースの「公式文書」はテスト終了後（テスト後半）に短期間で更新
  - 「差分」の成果物はすべて事前にレビューによって承認されている
  - さらに、テストによってその変更が正しい（適切な）ことを確認している
  - 構成管理手続きの中で、「差分」情報を元にして公式文書を更新する
- テスト作業の一部と重ねることで文書更新の工数を隠す



公式成果物	変更要求仕様書	変更設計書
機能仕様書	○	
画面操作仕様書	○	
制御仕様書	○	
各段階の設計書（仕様書）	○	
関数仕様書	○	○
関数設計書		○
データ仕様／設計書	○	○

1. 派生開発における品質問題
2. 品質のために生産性を犠牲にした
3. XDDPで派生開発のプロセス改善を
4. 追加機能要求仕様書
5. 変更３点セット
6. XDDPでビジネスに勝つ
7. XDDPから「次」へ



テストでできることは？

元から作り込まれた品質は変わらない

- テスト
  - 川に流れ込んだ「ゴミ」を取り除く行為
- テストの強化
  - ゴミを取り除く網の目を細かくしたり、
  - 網の数を増やすこと

テストでは「水質」は変わらない

「水質」は上流の「森」で決まる

でも、テスト結果から「水質」や「森」がわかる

- 「水質」の問題
  - 応答性に対して不適切なデータ構造
  - 安定化の障害となる複雑な処理構造
  - 可読性が悪く理解の支障になる

変化を加えた時にゴミに変化しやすい性質

品質はプロセスに左右される

- ソフトウェアの開発

変換プロセス ＋ 検証プロセス

- 入力物から出力物に変換する
- 出力物に混入する欠陥を発見し修正する
- レビュー、テスト

合理的な変換プロセス

↓ 成果物とプロセスが合理的連鎖で最終成果物を作り出す

検証プロセスが機能する

↓

品質保証の根底となる

「XDDP」は、この状況を確保している

生産性データから効果が見える（1）

- 2つの生産性の「差」からプロセスの様子がわかる

	対全工程 （一般に認識されている）	対実装工程 （要求仕様の難易度の影響を受けない）
種類	生成行数 全工程の工数	生成行数 実装工程の工数
生産性	2～8行／時間（一般）	80～100行／時間（XDDP）

差	
大	プロセスが整備され事前にレビューされている
小	必要以上に「ながら作業」が行われている可能性大

「3点セット」の成果物があることで
「ながら作業」が大幅に減る

生産性データから効果が見える (2)

- ３点セットの成果物を作ることで、ソースコードの変更作業の生産性が向上
  - 従来方法では、生産性は「5～10行／時間」にまで低下する
  - XDDPのプロセスでは「80～130行／時間」も可能

変更行数	20、000行
人数	5人

注目！

2、3割のソースコードの書き直しが生じていると思われます

	生産性	所要時間	実装時間／個人	
従来方法	5行／H	4000時間	800時間	4ヶ月
XDDP	80行／H	250時間	50時間	1週間余

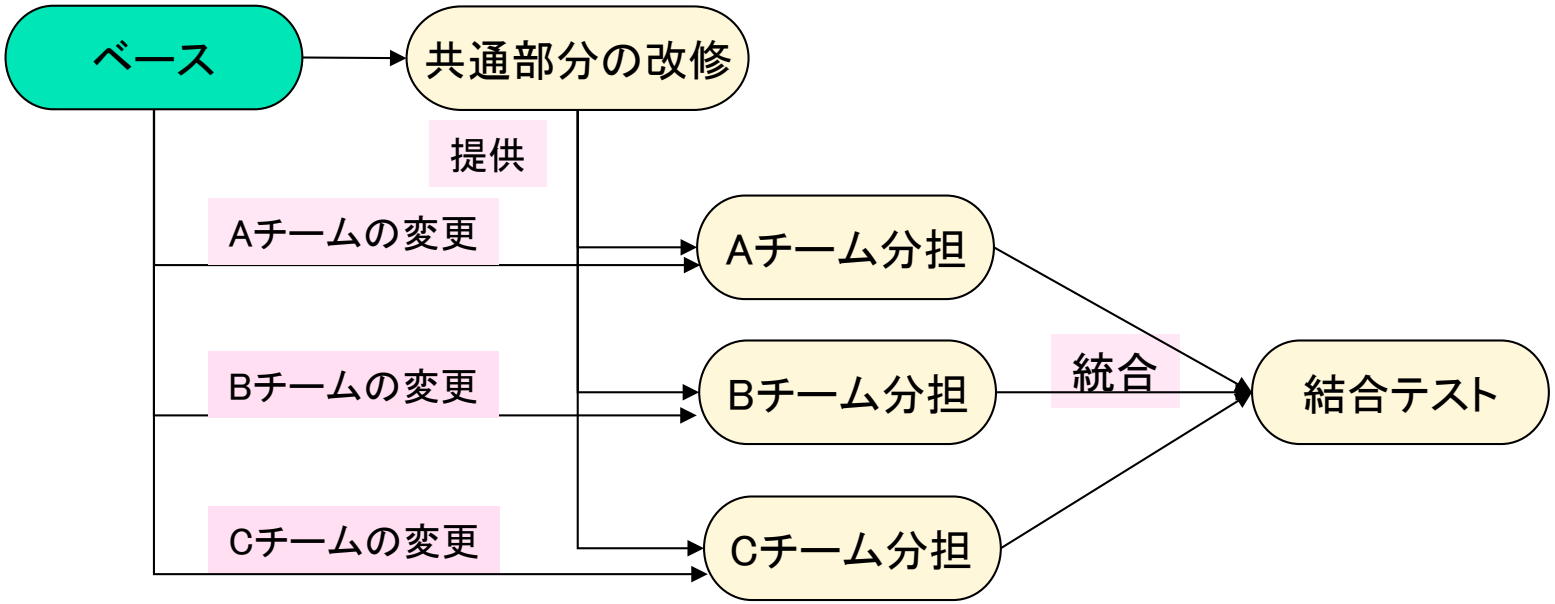
3750時間を使って、250時間でソースコードを変更できる状態を作り上げる.
＝XDDPの「３点セット」の成果物を用意する

実際には，3750時間の8割程度で足りる.
バグが1／10になることでテスト工数大幅減

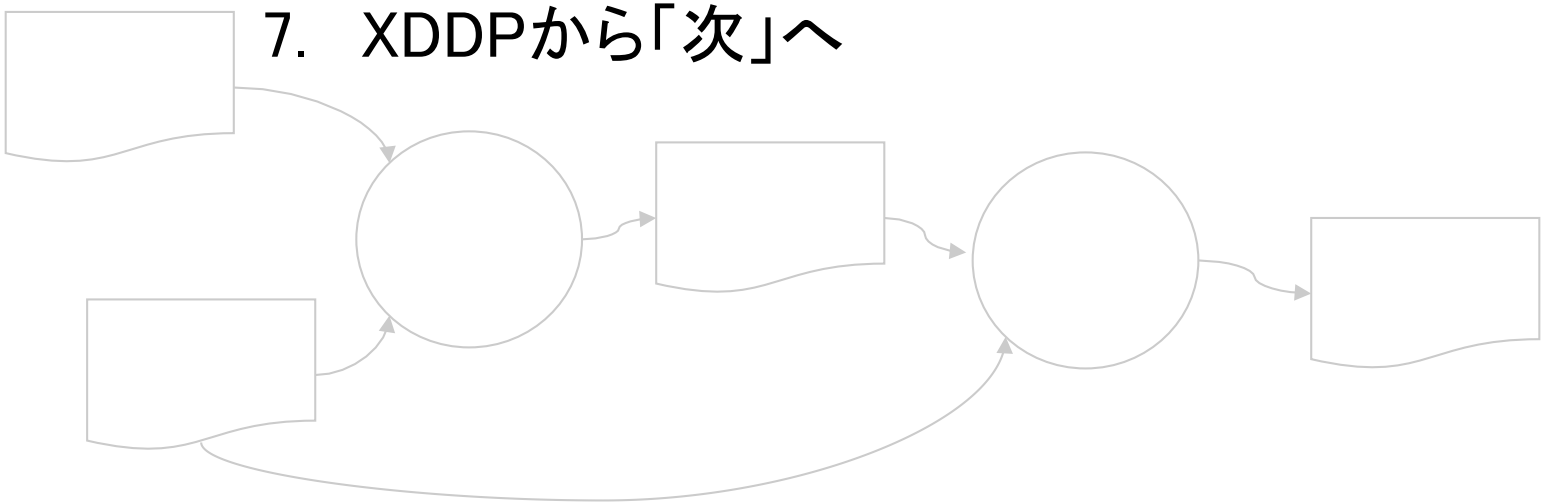
平均で当初予定工数の3割減

複数チームでの開発もスムースに

- 派生開発を複数のチームで分担して開発する際に混乱しない
  - 「3点セット」の成果物で各チームの変更箇所などの情報が交換できる
  - ソースコードの変更をできるだけ遅らせて短期間で仕上げる
- 「五月雨開発」にも効果を発揮する



1. 派生開発における品質問題
2. 品質のために生産性を犠牲にした
3. XDDPで派生開発のプロセス改善を
4. 追加機能要求仕様書
5. 変更３点セット
6. XDDPでビジネスに勝つ
7. XDDPから「次」へ



System Creates

ソフトウェア開発の生産性は大幅に改善できる

生産性・・・3倍、5倍も可能

- 「合理的な開発アプローチ」と「事前のシミュレーション」によって、ソフトウエア開発の生産性は飛躍的に向上する
- 今まで、ソフトウエア開発では生産性を追求してこなかったことで、改善の余地は手つかずで残っている
- 「XDDP」にマッチしたテストのあり方も未開拓の状態

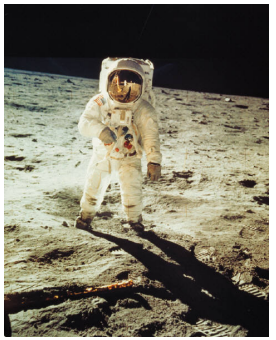
過去の文化と決別する絶好の機会

「XDDP」が、その機会を提供する

オフショア開発から国内回帰へ

「ソフトウェア開発力」を失えば日本の産業の根底から崩れる

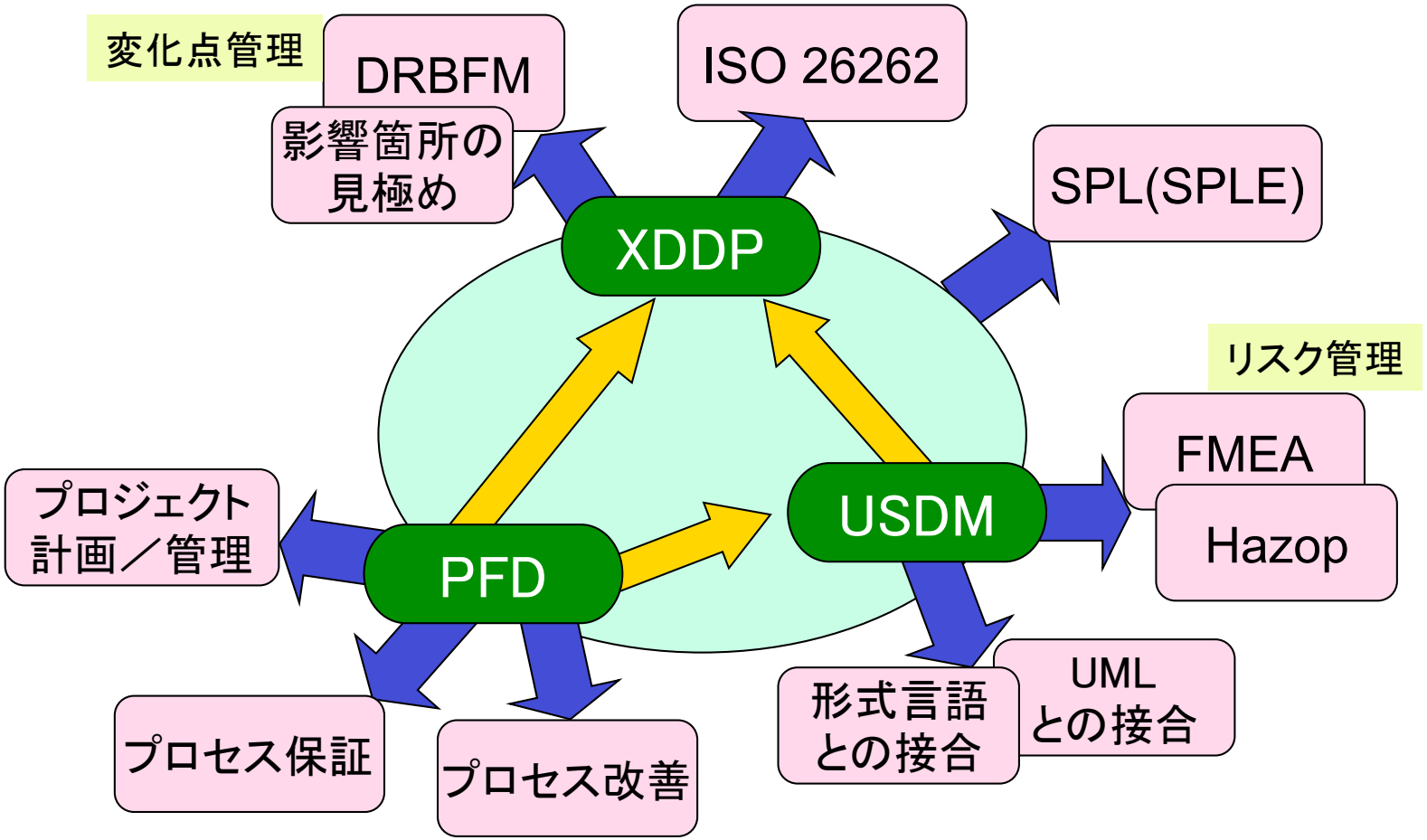
- 今日・・・製品を差別化するのは「ソフトウェア」
- 円高・・・製造部門の海外転出は止められなくても・・・
- ソフトウェア開発は「品質」と「生産性」の向上で国内に留めることは可能
  - 「品質」と「生産性」の大幅な向上による競争力の強化へ
- 「6K職場」と決別し、憧れの職業へ
  - ソフトウェア技術者に対する社会的地位の向上へ



品質と生産性の向上で、これらが可能になることに気付いて欲しい

XDDPトライアングルの展開

- 「XDDP」の3つの技術は、それぞれいろいろな取り組みに展開する



（参考文献）

- 文献（単行本）
  - ①【改訂第2版】要求を仕様化する技術・表現する技術（技術評論社）
  - ②『派生開発』を成功させるプロセス改善の技術と極意（技術評論社）

USDM

XDDP