第 6 分科会（A チーム）

派生開発に XDDP を導入する際の障壁とその解消に向けたアプローチ

Solutions for barriers against introducing XDDP into derivative software development

主査		足立 久美（株式会社デンソー）
副主査		奈良 隆正（NARA コンサルティング）
アドバイザ		清水 吉男（システムクリエイツ）
研究員	リーダ	長友 優治（株式会社ベリサーブ）
	サブリーダ	中間 義人（富士ゼロックス株式会社）
	サブリーダ	南部 妙水（アンリツエンジニアリング株式会社）
		秋山 友秀（キヤノンソフトウェア株式会社）
		市川 哲也（株式会社アドバンテスト）
		鵜飼 智徳（株式会社日立システムアンドサービス）
		石塚 弘機（三菱電機コントロールソフトウェア株式会社）
		中沢 光介（株式会社クレスコ）

概要

近年，XDDP（eXtreme Derivative Development Process）が派生開発の QCD（Quality, Cost and Delivery）を向上させる有効なプロセスであることが認識されるようになってきた。しかしながら，既存の開発プロセスを持つ組織に XDDP を導入する場合には，プロセスの差異から少なからず困難が生じる。

この論文では，我々が実際の組織に対して XDDP の導入を試みた際に生じた「障壁」とその対策を紹介する。さらに，一部の対策についてはその効果と考察についても言及する。

Abstract

Recently, it has come to be recognized that XDDP (eXtreme Derivative Development Process) is an effective way to improve QCD (Quality, Cost and Delivery) of derivative software development. However, when we apply XDDP to organizations with an existing software development process, some difficulty are caused because of differences in processes.

In this paper, we describe solutions for the barriers against introducing XDDP into our derivative software development. In addition, we describe the result and the consideration of some of solutions.

1. はじめに

組み込み系・エンタープライズ系の区別なく，多くのソフトウェア開発において，既存のソフトウェアに機能レベルでの追加・流用・変更を加えるという「派生開発」が行われている。その派生開発の多くは納期遅延や市場トラブルといった問題を抱えており，そこに携わる人々も効果的な打開策を打てず疲弊している状況である。

この状況を打破する方策として，我々は XDDP（[1]，[2]，付録 A を参照）を選択した。XDDP は本分科会のアドバイザである清水吉男氏が開発した派生開発向けのプロセスであり，派生開発の実態にあったプロセスで問題を解決するアプローチである。XDDP の特長は，すべての派生開発に適用が可能であること，プロセスの成果物が具体的で取り組みやすいことである。そこで本論文の研究員はそれぞれが所属・関係する開発現場に XDDP を適用したいと考えた。

1.1. 研究の背景と目的

実際に XDDP の導入を試みた際に，さまざまな障壁が存在し，それらを解決しなければ XDDP を開発現場に適用することができないことがわかった。これらの障壁は多かれ少なかれ，XDDP を導入しようとしたすべての組織で存在すると我々は考えた。ここでの「障壁」とは，XDDP 導入の効果に対する疑問や導入を妨げる課題を指す。この障壁が形成される原因を見つけ，その克服の方

策を見出すことを研究テーマとした。

本論文での読者は，XDDP について書籍（[1]，[2]など）を読み，その内容の大半を理解した上で，自分が直接関係している派生開発に XDDP を適用したいと考えている XDDP 推進者と想定した。

2. 障壁の分析と対策

本論文の研究員は，派生開発の現場が抱えている問題を解決するため，開発部門の実担当者・リーダ・責任者，品質保証部門の担当者などに対して XDDP を紹介した。この結果，XDDP の導入に好意的な意見もあったが，さまざまな障壁が挙がり，XDDP の導入が見送られた。

2.1. 障壁の分析

我々は，1 つでも多くの障壁を解消するため，導入経験から得られた障壁を収集した。その収集した結果を付録 B 表 B-1 に示す。

次に，収集した障壁について特徴や要因に着目して整理，分類した結果，表 2-1 に示すように，XDDP の特徴が生んだ誤解，一般的なプロセス改善において生じる問題，また変化への漠然とした不安や根強い抵抗感があるということがわかった。以下，各分類について説明する。

表 2-1　障壁の分類

分類		名称	定義
A		心理障壁	プロセス改善を受容できない心理
	1	XDDP への不信	XDDP 特有の作業や成果物に対する疑念や不信
	2	最初の 1 歩への不安	新しい取り組みが失敗することへの不安
	3	変化への抵抗	現状から何かが変わること自体に対する抵抗
B		プロセス障壁	組織の標準プロセスと XDDP の差異に対する拒否反応
C		組織障壁	XDDP を導入しにくい開発組織や体制

2.1.1. 分類 A: 心理障壁

この障壁は，プロセス改善を受ける人物が，一見すると合理的に見える理由によって XDDP を受け入れたがらない心理を示す。この原因を分析した結果，さらに 3 つのグループに分割した。

1)　分類 A-1: XDDP への不信

この障壁は「本当に XDDP が適しているのか」という不信である。XDDP は特に設計資料やソースコードの改修のやり方が独特であり，この部分に対して不信感が生じやすい。この障壁は主に，問題が発生していることを認識し，何らかの改善を行いたいと考えているが，どのような改善が適しているのかわからない人物に発現する。

2)　分類 A-2: 最初の 1 歩への不安

この障壁は「今までやったことがないから，この取り組みは失敗するのではないか」という不安である。特に「もし失敗したら事態が悪化する」という思い込みによって，この不安は強固になる。この障壁は主に，問題が発生していることを認識しているが，改善の必要性を強く感じていない人物に発現する。

3)　分類 A-3: 変化への抵抗

この障壁は「今までの慣れたやり方を変更されたくない」という抵抗である。プロセス改善効果に関心が低く，現状を変化させるコストやリスクを強く意識しているため，“みんながやって成功している”という状態にならない限り，この抵抗感は消えない。この障壁は主に，問題が発生していることを認識していない人物に発現する。

2.1.2. 分類 B: プロセス障壁

この障壁は，組織内に標準プロセスが存在しており，「標準プロセスとして定義されているプラクティスはすべて記述通りに実施しなければならない」という誤ったプロセス遵守の考え方によって発現する。主に，CMM/CMMI の認定を受けている組織，プロジェクトに適用するプロセスをテーラリングせず標準プロセスに固定している組織に多く見られる。

2.1.3. 分類 C: 組織障壁

この障壁は，プロセス改善など，ルールの変更を受け入れにくい組織体制で発現する。主に，複数の会社で開発しているプロジェクトや，複数の国・文化圏の人員で構成されているプロジェクトに多く見られる。

2.2. 対策

各組織で明らかになった障壁を解消するために，分類ごとに対策方法を検討し，その結果を表 2-2 にまとめた。なお，対策方法として，組織的権限を行使して強制的に XDDP を導入するといった，本論文の研究員および想定読者の多くにとって現実的でないアプローチは検討の対象外とした。

表 2-2　障壁への対策

分類		名称	対策
A		心理障壁	－
	1	XDDP への不信	XDDP の何に不信を感じているのかを特定し，具体的な成功事例を用いて説明を行う。
	2	最初の 1 歩への不安	実作業がイメージできるレベルで説明する。例えば以下の準備を行う。 ・教育プログラムを作成する。 ・準備期間や説明会を設ける。 ・ガイドラインやテンプレートを作成する。 ・サンプルを用意する。
	3	変化への抵抗	改善の必要性を説明するか，他の組織での改善実績で効果を示すことで納得してもらう。特に，過去の失敗経験がある人物の場合は，強力な XDDP 推進者となる可能性があるので，積極的に個別対応を行うことを推奨する。例えば以下の方法を取る。 ・カウンセリング，ファシリテーションなどを行う。 ・XDDP 適用の実績を組織内で積み上げる（それまでこの障壁は放置する)。
B		プロセス障壁	XDDP の特徴を活かすように，現状の社内標準プロセスをテーラリングする。 ただし，“標準プロセスが新規開発向けに定義されている”ことに留意して，「新規開発崩し」とならないようテーラリングする必要がある。
C		組織障壁	組織障壁の解消には多くの関係者の合意と，継続的な働きかけが必要である。また，その対策の内容も組織特有の事情に合わせたものにしなければならず，一般化することは困難である。これらの理由から，本論文での対策は見送ることとする。

特に，分類 A の心理障壁は XDDP に限らず，どのような手法を新規に導入する場合でも発生する。我々は XDDP 導入の際に特有の心理障壁として，分類 A-1 の XDDP への不信に着目することにし，XDDP の特徴（付録 A を参照）を踏まえた対策について更なる検討を行った。この検討結果を表 2-3 に示す。

表 2-3　「A-1: XDDP への不信」に対する対策の分析

XDDP の特徴	障壁の正体	対策
“変更”と“追加”を異なるプロセスで扱う	プロセスが 2 つに分離することによって，作業量が 2 倍になるような誤解	XDDP の作業手順と現状の作業手順とを比較し，実質的に作業量の増大に直結しないことを説明する。

表 2-3 「A-1: XDDP への不信」に対する対策の分析 [続き]

XDDP の特徴	障壁の正体	対策
差分だけを取り扱う成果物 「3 点セット」(*)	既存プロセスにはない文書を作成することから，作業量が増加するという印象	3 点セットの作成は，現在のプロセスで行っている調査や設計メモ作成の作業量と大きく変わらないことを説明する。
	既存文書と異なるフォーマットに変更するコストがかかるので，それに見合う効果に対する疑問	3 点セットに変更の情報が記述されることで必要な情報が含まれること，レビュー効果が向上することと，手戻り作業が減少することを説明する。
ソースコードの改修は一斉に まとめて行う	ソースコードの変更に着手するまでの工期が長くなるので，納期遅延の不安	従来の方法と比較し，ソースコードの生産性が向上することで，全体の生産性が変わらないことを説明する。

(*)「変更要求書」「トレーサビリティ・マトリクス」「変更設計書」の 3 つの成果物のことを指す。

3. 対策の実施結果

我々は 2.2 で検討した対策のうち，分類 A-1（XDDP への不信）および分類 B（プロセス障壁）に対する 2 つの対策を試行した。以下にその結果を示す。

3.1. 「A-1: XDDP への不信」– シミュレーションによる 3 点セットの効果の検証

分類 A-1: XDDP への不信の中で，「XDDP で作成する 3 点セットの効果を疑う人物を対象にした対策」の 1 つとして，まだ XDDP を導入していないプロジェクト（以下，対象プロジェクトと称する）について，XDDP を適用して 3 点セットを作成したと仮定したシミュレーションを実行した。

3.1.1. 実施内容

シミュレーションの対象プロジェクトの概要を付録 C 表 C-1 に，対象プロジェクトの品質データを図 C-1，図 C-2，図 C-3，表 C-2 に示す。さらに，対象プロジェクトの品質データから対象プロジェクトが抱えている課題とその原因を分析し QCD の観点でまとめた結果を表 3-1 に示す。

表 3-1 より，対象プロジェクトの問題は設計工程（成果物の品質を保証するレビュー）で，変更部分の品質を十分に確保できなかったところにあることがわかる。

表 3-1 対象プロジェクトの問題分析

観点	データからわかること	分析結果
品質	不具合の原因箇所を「追加部分」と「変更部分」に分類した結果，変更部分による不具合が全体の 7 割を占めた。 不具合の 5 割が変更仕様の不備（漏れ・間違い）に起因する不具合であった。	変更部分の品質が全体の品質低下を招いている。 特に変更部分の設計工程で品質を確保できていない。
	機能設計レビューでは，指摘の 6 割が「曖昧な記述の確認」と「記述ミスの指摘」である。 詳細設計レビューでは，「ソースコードレベルの保守性・視認性に関する指摘」と「曖昧な記述の確認」しか行われていない。	設計工程の成果物に曖昧な記述が多く，レビューアが設計者の意図を理解できていない。このため，設計の不具合を検出できず，設計の妥当性を保証するレビューになっていない。
コスト	不具合修正で同一関数のソースコードを修正した回数は，最大で 3 回となっている。 製品として使用された LOC は，記述した総 LOC の 6 割である。	テスト工程に持ち込んだソースコードの品質が悪い上，不具合修正も一度で終わっていないために，プログラミング工程とテスト工程で多くのムダが発生している。
納期	プログラミング工程の工期が計画の約 2 倍になっており，工期計画値とほぼ同じ工期を不具合修正に要している。	不具合修正の多さがプロジェクトの遅延要因の一つである。

XDDP は 3 点セットを用いることで設計工程の品質を確保できるとしていることから，この対象プロジェクトの品質データと，シミュレーションによって得られたデータを比較することで，3 点セットの効果を示す，以下 2 点について検証を行うことができると考えた。

- 現状の設計工程の成果物よりも，多くの設計不備を防ぐことができる（不具合防止率）
- 設計工程により多くの時間をかけても，全体工数に影響を与えない（工数への影響）

なお，時間制約によって対象プロジェクトの変更要求のすべてをシミュレーションすることが難しかったため，対象プロジェクトに要求された 1 つの変更要求を作業対象とし，残りの要求は同様の傾向があるものとした。3 点セットの効果の検証方法を表 3-2 に示す。

表 3-2　3 点セットの効果の検証方法

検証#	検証目的	検証方法	検証対象
1	3 点セットを使用して設計品質を保証するレビューが実施可能かを検証する	以下の条件を満たす設計者をレビューアとして，3 点セットを使用し書面レビュー（第 3 者レビュー）を行う。 ・ 対象製品および類似品の製品知識を持っておらず，対象プロジェクトの内容も知らない ・ 3 点セットを作成した設計者とは別の設計者である	不具合防止率
2	不具合防止率の推定による 3 点セットの不具合防止効果を検証する	対象プロジェクトで不具合の原因となった仕様不備から，3 点セットで発見可能なものをカウントし，下式により不具合防止率を算出する。 不具合防止率（%） = 設計工程で検出すべき不具合の件数 ÷ 不具合総件数 × 100 発見可能な不具合の件数は，作成した 3 点セットを対象プロジェクトの担当者に提示し，担当者が「発見可能」と判断したものをカウントする。 なお，対象プロジェクトの担当者は XDDP を理解していないため，事前に XDDP および 3 点セットの説明を行う。	不具合防止率
3	3 点セット導入による工数の変化を検証する	“対象プロジェクトの総工数”に，“3 点セットを作成するために増加する工数”と，“3 点セットを導入した場合に低減できた不具合修正工数”を加味し，総工数の変化を推測する。	工数への影響度

3.1.2. 実施結果

シミュレーションで得られたデータと 3 点セットの効果の検証結果を以下にまとめる。

まず，検証 1 の設計品質を保証するレビューが実施可能であるかの検証として，第 3 者レビューを実施した結果を表 3-3 に示す。今回の書面レビューで挙がった設計書の問題は，記述不備（検討されていたが設計書には反映されていなかったもの）のみであったが，もし検討漏れをしていた場合，製品の品質に影響を与えかねないものばかりであり，曖昧な記述を確認する質問や指摘はなかった。

このことから，製品知識を持たないレビューアが事前説明を受けなくても，一般的なレビュー観点・手法で設計品質を保証するレビューができたと判断する。

表 3-3　第 3 者によるレビュー結果

項目	実績値
レビューアの人数	1 人
工数	0.5 H
指摘件数	・ 処理タイミングについての指摘×1 件 ・ 条件網羅の漏れについて指摘×1 件 ・ 処理時間に対する考慮の有無を確認×1 件

次に，検証 2 の不具合防止効果の検証として，表 3-2 に定義した不具合防止率を算出した結果を表 3-4 に示す。表 3-4 には「3 点セットを使用した効果」の他に，「補助資料作成による効果」と「XDDP のプロセス実施による効果」も加えてある。

表 3-4　不具合予防率

効果		不具合予防率
3 点セットの効果	設計資料として 3 点セットを作成した場合，仕様不備（変更仕様の抽出漏れ，変更箇所の誤り，上流工程から下流工程へ移行した際の漏れなど）の件数を削減できる	約 60%
補助資料の効果	3 点セットに加え，適切な補助資料（パラメータと関数の関係を示す資料など）が存在すれば，仕様不備の件数を削減できる	約 25%
変更プロセス自身の効果	3 点セットを作成した上，変更設計書とソースコードの差分を比較するレビューを実施することで，実装ミスの件数を削減できる	約 5%
合計		約 90%

最後に，検証 3 の工数変化の検証として，表 3-2 に定義した“3 点セットを作成するために増加する工数”は 3 点セットの作成に要した時間から，表 3-5 に示すように設計工数では約 6% 増，総工数としては約 3%増となると推定した。

表 3-5　工数に対する影響度

項目	比較対象	工数	増加原因の考察
変更要求仕様書	機能設計文書	設計工数約 3%増	既存文書の散文的記述ではおろそかになっていた「何をどう変えるのか」の明確化（Before/After 表現），手順・条件の明確化（分割基準に従ったグループ分け）を行う時間が増える。
トレーサビリティ・マトリックス	詳細設計文書	設計工数約 1%増	既存文書では変更箇所しかリストアップされていないため，変更しない箇所も含めたすべての派生元のモジュール名を列挙する時間が増える。
変更設計書	詳細設計文書	設計工数約 2%増	既存文書のソースコードコピーへの注釈ではおろそかになっていた「どこをどう変えるのか」の明確化（Before/After の変更内容を日本語の文章で表現）を行う時間が増える。
合計	設計工数約 6%増（対象プロジェクトの総工数に対しては約 3%増）		

※「Before/After 表現」および「分割基準」は USDM（[3]，[4]）の一部として定義されている。
※ 設計工数は総工数の約半分である。

上記検証 1～3 の結果，“3 点セットを導入した場合に低減できた不具合修正工数”は，“テストで発見した不具合修正のための工数（修正結果の確認工数を含む）”が設計品質の向上により減少した分として推定できる。検証 2 の結果から，対象プロジェクトで検出された変更仕様の不備を原因とする不具合の 60%が予防できることがわかった。予防できた不具合すべての修正工数が，付録 C 表 C-3 にある最低工数であったとしても，対象プロジェクトの総工数に対して約 3%の工数が低減できることになり，設計工数の増加と等しい。これより，3 点セット導入によるプロジェクトの総工数への影響はほとんどないといえる。

3.1.3. 結果の考察

3 点セットによる効果を実在のプロジェクトでシミュレートした結果，設計の品質向上と，総工数の維持または減少が得られることがわかった。また，不具合予防率の算出過程で，対象プロジェクトの担当者に 3 点セットの効果を納得してもらうことができた。このことより，分類 A-1 の XDDP への不信に含まれる 3 点セットの効果への疑念という障壁は解消できたと結論付ける。

なお，今回のシミュレーションでは補助資料や変更プロセスとの併用によって得られる効果や，3 点セットの効果として期待できる以下の項目（参考文献[1]～[4]）について検討していない。これらの効果については今後の課題とする。

- 見積もり精度の向上
- 生産性の向上
- 並行開発が可能

3.2. 「B:プロセス障壁」– ウォーターフォールプロセスのテーラリング

分類 B: プロセス障壁については，本論文の研究員の 1 人が所属する組織の標準プロセスを対象にテーラリングする対策を実行した。

ここでは，テーラリングの目標として，

・ 標準プロセスにおいて，XDDP と同等のプロセスが定義できること

・ テーラリングしたプロセスについて，同組織の品質部門に使用の許可を得ること

の 2 点を設定した。

3.2.1. 実施内容

対策を実施した組織には複数のモデルに基づく標準プロセスが用意されているが，いずれの標準プロセスも新規開発向けに定義されている。また，対象読者は標準プロセスを変更できない場合が多い。この中からウォーターフォール・モデルの標準プロセス（以下，対象プロセスと称する）を選択しテーラリングを行った。

テーラリング方法として，対象プロセスと XDDP の対応付けを実施した。対象プロセスの定義を付録 D 表 D-1 に示す。また，対象プロセスと比較するために，付録 A の図 A-1，図 A-2 の PFD（Process Flow Diagram）を表に変換したものを，付録 D の表 D-2，表 D-3 に示す。

テーラリングは表 3-6 に示す手順と，表 3-7 に示す方針に従って実施する。実施結果は表 D-4 となる。

表 3-6 テーラリング手順

手順	すること
1	標準プロセスを調査し，表 D-4 の「対象プロセス」列に入力する。
2	XDDP に対応するものがない対象プロセスの工程・成果物を見つけ，表 D-4 の“マッチングさせた XDDP”列に「なし」と入力する。 例）「見積書」の行，「総合テスト仕様書」の行，「単体テスト」の行
3	対象プロセスと XDDP でほぼ同等の工程・成果物を見つけ，“マッチングさせた XDDP”の“工程”または“成果物”列に番号と文書名を入力する。 例）「要件定義」の行に「1.1」，「ソースコード」の行に「更新後のソースファイル」
4	3 でマッチングした工程との前後関係から決まる工程を“マッチングさせた XDDP”の“工程”列に入力し，その成果物をマッチングする。 例）XDDP のプロセス 1.5 は 1.6 の前工程なので，「モジュール設計」の行に「1.5」を入力する。その成果物である「変更設計書」を「モジュール設計書」に対応付ける。
5	残った項目について，工程の前後関係・成果物の意味を考慮してマッチングする。 例）「機能設計」の行に「1.2～1.4」，「機能設計書」の行に「変更要求仕様書」「変更要求 TM」「スペックアウト資料」
6	重複やムダ・漏れがないか最終確認を行う。 例）構造設計を省略，開発完了工程の成果物に「製品の正式文書」を追加

表 3-7 テーラリング方針

方針 1	XDDP の変更プロセスを実現するためのテーラリングとする。 以下の項目はテーラリング範囲外とし，これらは従来通り社内標準プロセスに従う。 ・新規開発プロジェクト ・派生開発の“追加”部分（XDDP の追加プロセス対象） ・開発業務と直接関係のない項目（文書管理，構成管理等）
方針 2	変更プロセスと追加プロセスは機能設計工程から分離し，結合テスト工程で合流する。 プロセスの分離に合わせて成果物も分離する。

3.2.2. 実施結果

対象プロセスのテーラリングした結果，XDDP で定義される工程と成果物は，すべてが対象プロセスに収まった（付録 D 表 D-4）。また，表 D-4 を元にテーラリングしたプロセスと追加プロセスの内容をまとめた資料を作り，テーラリングを行った研究員の所属する組織の品質部門にレビューを依頼した。レビューの結果，このプロセスは組織内のプロジェクトに十分適用可能であり，実際に適用する許可を得ることができた。

このテーラリング作業は，対象プロセスと XDDP の両方を理解していた研究員 1 名が行い，品質部門に提示した資料の作成も含めて約 1 日で終えることができた。

3.2.3. 結果の考察

上記の実施結果により，ウォーターフォール・モデルの組織標準プロセスを対象に，XDDP と同等の開発プロセスとなるようテーラリングすることは可能であると判断した。また，品質部門から実施許可を得たことで，該当組織における分類 B のプロセス障壁は解消できたと結論付ける。

なお，今回はテーラリングしたプロセスをプロジェクトに適用するまでには至っていない。プロジェクトに適用することで，このプロセスの問題や新たな障壁が発見されることが考えられるが，それは今後の課題とする。

4. まとめ

我々が実施した XDDP 導入の障壁に対する対策は，いずれも障壁を解消することができた。このことより我々は，障壁を分析し適切な対策を実行することで，XDDP の導入を容易にすることが可能であることがわかった。

しかし，いずれの対策も“どのような障壁にも効く万能薬”ではないので，実際の障壁を詳しく分析し，それぞれの要因にあった対策を実行すべきである。

我々はさまざまな障壁と出会い，解消するために取り組んできたが，現時点では，研究員の所属する組織のいずれにおいても XDDP を導入するには至っていない。今後は実際にプロジェクトに XDDP を導入し，今回の障壁に対する対策について検証し，また，今回見送った残課題についても検討していきたい。

本論文に記載した方向性や考え方が読者の障壁解消へのヒントになると我々は信じる。そして，同様の障壁を解決しなければならない「XDDP 推進者」の助けになることを切に願う。

5. 参考文献

[1] 清水吉男: 「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，2007

[2] 清水吉男: 失敗しない派生開発（Software People vol. 8），技術評論社，2006

[3] 清水吉男: [入門 + 実践] 要求を仕様化する技術・表現する技術 —仕様が書けていますか？，技術評論社，2005

[4] 清水吉男: 要求の仕様化入門（Software People vol. 4），技術評論社，2004

付録 A　派生開発の問題と XDDP の特徴

本論文の前提知識である派生開発の問題と XDDP の特徴について概説する。なお，詳細は清水氏の著書（[1]，[2]）に記述がある。

派生開発においてトラブルを引き起こす要因として，

- 短納期，低コストでの開発
- 要求が曖昧なままでの開発

がよく知られているが，以下のことは意外に正しく認識されていないため，表 A-1 に示すような事象が発生し，納期遅延や品質低下のトラブルを招いている。

- 要求には“機能変更の要求”と“機能追加の要求”の 2 つがあり，それぞれを分けて対応する必要がある
- 追加の要求の対応には，必ず受け入れるソースコードの変更が発生する

表 A-1　機能変更の際によく起きる事象

事象	説明
新規開発崩し	“機能変更”プロセスに，新規開発向けに定義されたプロセスを適用する。
構成管理の崩壊	変更内容をレビューする適切な成果物を作成せず，製品の正式文書をいきなり変更する。
部分理解の罠にはまる	派生元のソースコードを部分的にしか理解していない状態をカバーする適切な対応を行わない。
間に合わないモンスターの襲来	短納期の圧力に負け，闇雲にソースコードを変更する。

XDDP はこの“変更”の特徴にマッチしたプロセスであり表 A-2 に示す特徴を持つ。

表 A-2　XDDP の特徴

特徴	説明
“変更”と“追加”を異なるプロセスで扱う	“変更”，“追加”の特徴にあったプロセスを適用し，新規開発向けプロセスでは対応できない“変更”に必要なプロセスを実行する。特に，要求定義の書き方を明確にする。 →【新規開発崩し】，【構成管理の崩壊】を防ぐ
差分だけを取り扱う成果物「3 点セット」	“変更”のプロセスでは以下の 3 つの成果物で差分を取り扱う。 ・　「変更要求仕様書」で何を変更するのか明確にする。 ・　「トレーサビリティ・マトリックス（TM）」でどこを変更するのか明確にする。 ・　「変更設計書」でソースコードをどのように変更するのか明確にする。 これにより変更範囲や内容が明確になり，漏れや思い込みを発見しやすくなる。 また，派生元の設計やソースコードを調査した場合には「調査資料」や「スペックアウト資料」という成果物を作成し管理する。 →【構成管理の崩壊】，【部分理解の罠にはまる】を防ぐ
ソースコードの改修は一斉にまとめて行う	3 点セットの作成とそのレビューが完了するまでソースコードを変更しない（メモをコメントとして残すことも禁止）。これによりソースコードの改悪を防止し，手戻りによるムダなプログラミング工数を作らない。結果としてバグの件数も低減するためテスト工数も削減する。 →【間に合わないモンスターの襲来】を防ぐ

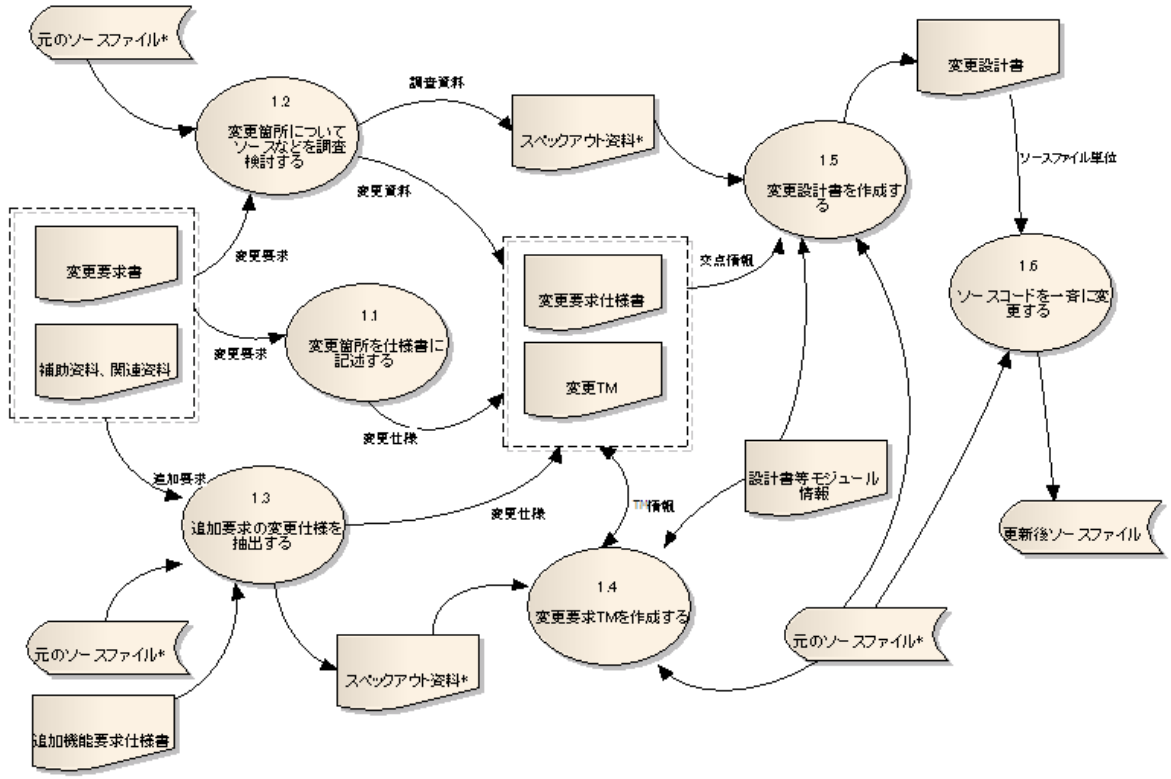


図 A-1　XDDP の変更プロセス

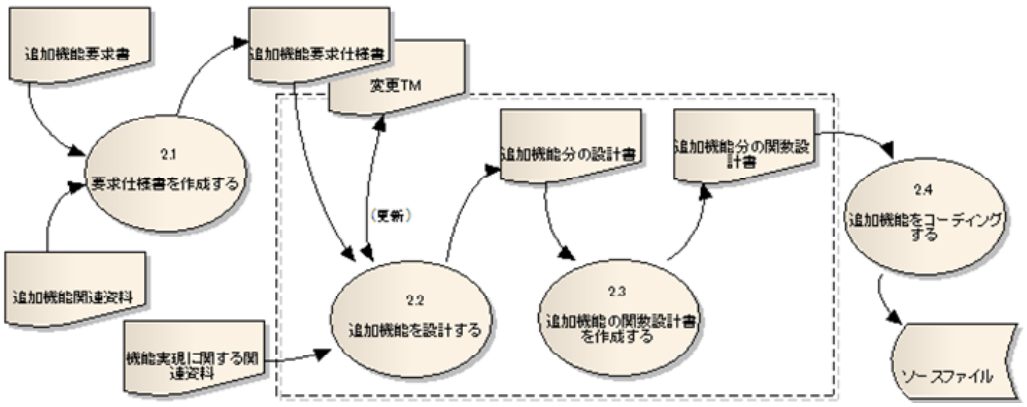


図 A-2　XDDP の追加プロセス

付録 B　2.1 障壁の分析で使用したデータ

表 B-1　収集された障壁のリスト

障壁	分類
・「レビューで不具合検出するというが，そのためにはレビューアの知識・スキルが必要である。レビューで不具合検出できない現状では，XDDP にしてもプロジェクトは失敗するだろう」 ・「作成された成果物の妥当性を確認する方法がわからない」 ・「書く内容が増えることで，プロジェクトの総工数が増えるのではないか」 ・「ドキュメントの書き方を変えたら，不慣れな分，余計に工数がかかる」 ・「作る文書が増えるのはいやだ」 ・「実装開始が遅れるため納期に間に合うか不安。リソース不足のため，増員は困難」 ・「仕様に曖昧な部分が多すぎるため，プロセスどおりに動かない。スケジュールの変更が頻発する」 ・「“詳細仕様は後日”という顧客相手では，仕様が決まっているところからソースコードを変更しないと間に合わない」 ・「開発規模が小さいと XDDP の適用ができないのではないか」	A-1
・「現場がちゃんとやってくれるか不安。指導できるかも不安」 ・「初めて適用するときは，理解不足や作業ロスが生じるのでは？」 ・「現在ソースコードしかない。既存の設計資料等がない状況で XDDP を適用するのは不安」	A-2
・「作業人月が契約金額になる。プロジェクトの失敗はむしろ売り上げ UP。営業担当としては改善してほしくない」 ・「テスト屋が開発プロジェクトの改善なんてできない。範疇外とあきらめるべき」 ・「良いならよそで採用しているはず。つまり，よそがやってないなら良くないはず」 ・「失敗したら誰が責任を取るのか」 ・「(自組織内で）成功した事例があるならやる」 ・「もう大変な思いはしたくないから，失敗するかもしれないことはやりたくない」 ・「改善の必要を感じない。今のままでも何とかなっている」 ・「顧客から文句が出ていないのだから，今のままで十分問題ない」	A-3
・「組織の標準プロセスから逸脱したら品質担当者から怒られる」 ・「組織の標準プロセスに認定されていないプロセスの使用は認められない。しかし，新しいプロセスを定義することは（すぐには）無理」 ・「顧客は CMM 認定プロセスで開発することを求めているはずだから,他のプロセスは使えない」	B
・「開発プロセスに直接口出しをできない立場で客先に入ることが多く,提案しても取り合ってもらえない」 ・「開発規模・成員が大きい。海外を含めた複数拠点にまたがって開発を行っている。一度にすべての作業についてプロセスを変えると大混乱になる」	C
・「実装を伴った技術検証が必要な場合，XDDP を適用できるのか」	なし

付録 C　シミュレーションによる 3 点セットの効果の検証で使用したデータ

表 C-1　対象プロジェクトの概要

属性		値
開発規模 [KLOC]	追加部分	1
	変更部分	0.5
	既存部分	12
開発担当者	人数	2 人
	開発経験	約 10 年
	システム理解度	高い
開発プロセス	モデル	ウォーターフォール・モデル
	成果物	要求仕様書，機能仕様書，詳細設計書（変更箇所については，元ソースコードの抜粋と修正イメージを記述），ソースコード，テスト仕様書

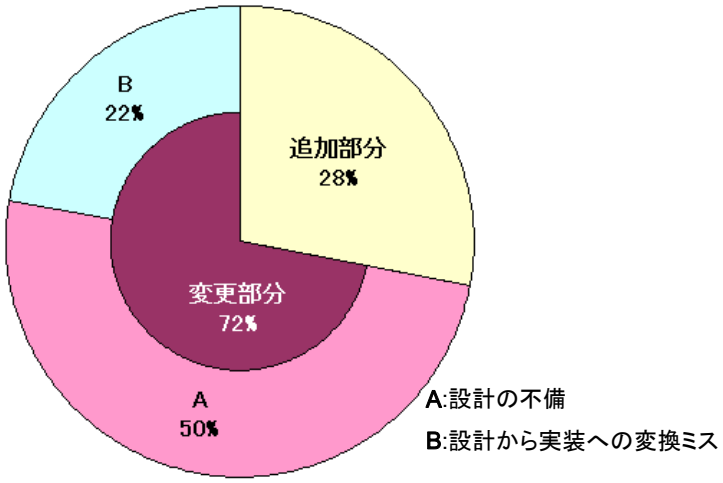
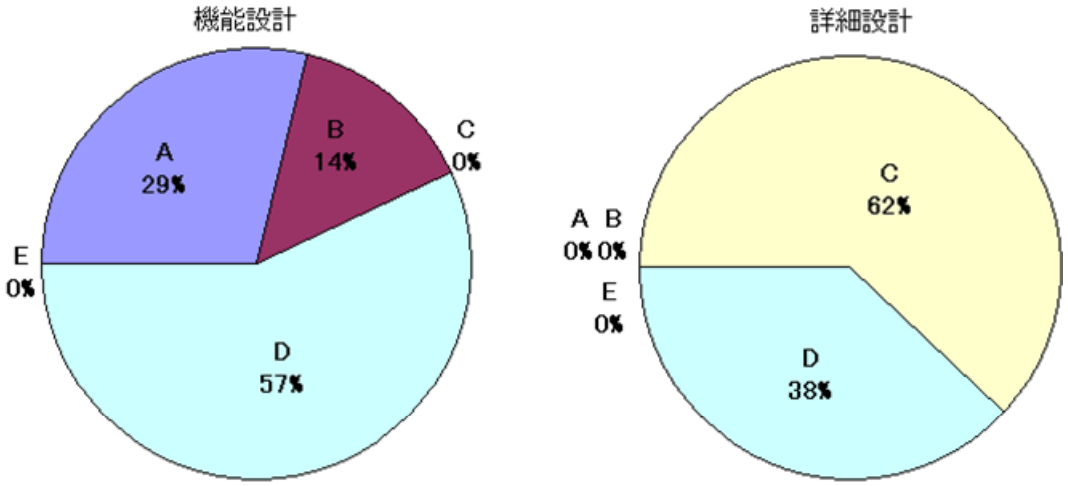


図 C-1　テストで検出された不具合の発生箇所と変更部分の不具合原因



A:要求に対して担当者の間違った認識/漏れに対する指摘
B:不適切な箇所を追加/変更しようとしていることに対する指摘
C:ソースコードレベルでの不適切な方法で追加/変更しようとしている指摘
D:記述内容が曖昧なことによる仕様の確認
E:用語や図の記述ミス

図 C-2　設計レビューの指摘事項

表 C-2　プログラミング工程のデータ

項目	数値	備考
プログラミング工数	約 100 時間	追加と変更部分の合計
プログラミング工期の割合	約 30%	全工期に対しての割合
実績値÷計画値（工期）	約 2	全工期では約 1.5
不具合修正が占める割合（工期）	約 50%	テストで発見された不具合のためソースコードを変更する作業もプログラミング工程としている
ソース規模実績	1.5KLOC	追加:1KLOC，変更:0.5KLOC
コード生産性	15 行/時間	追加と変更部分の合計
工程内のソース変更回数	８回	不具合などでソースコードを変更・削除した回数
工程内でのソース変更量	0.5KLOC	ソース規模実績には含まれない，不具合修正などで変更・削除したコード量 追加:0.2KLOC，変更:0.3KLOC
変更部分のファイル数	22 ファイル	
変更部分のファイル最大更新回数	4 回	同一関数を修正した最大回数 3 回（いずれも不具合修正のため）
追加部分のファイル数	16 ファイル	
追加部分のファイル最大更新回数	7 回	

表 C-3　不具合修正工数の見積もり（1 件あたり）

項目	工数	備考
調査時間	数分～	ばらつきが多く，意味のある平均値を出せない
不具合報告書作成	約 0.5H	どのような不具合かを記述
修正方法の検討	数分～	ばらつきが多く，意味のある平均値を出せない
設計書修正	約 0.5H	
修正した設計書のレビュー	約 0.5H	
ソースコード修正	数分～	ばらつきが多く，意味のある平均値を出せない
構成管理ツール更新	約 0.5H	
再テスト	約 0.5H	
不具合報告書更新	約 0.5H	調査後に作成した不具合報告書にどのような修正と確認を行ったかを記述
合計	最低でも約 3H	

付録 D　3.2 ウォーターフォールプロセスのテーラリングで使用したデータ

表 D-1　対象プロセスの定義

工程	工程内の作業	主な成果物
要件定義	要件定義と見積もり，そのレビュー	要件定義書，見積書
プロジェクト計画	プロセス定義，SQA 計画 策定，日程定義	設計計画書，プロセス定義書，開発日程計画書
機能設計	機能設計，総合テスト設計	機能設計書，総合テスト仕様書，各文書のレビュー記録票
構造設計	構造設計，結合テスト設計	構造設計書，結合テスト仕様書，各文書のレビュー記録票
モジュール設計	モジュール設計，単体テスト設計	モジュール設計書，単体テスト仕様書，各文書のレビュー記録票
プログラミング	プログラミング，机上デバッグ	ソースコード，ソースコードレビュー記録票，解析ツール分析結果
単体テスト	単体テスト，デバッグ	単体テスト成績書，バグレポート
結合テスト	結合テスト，デバッグ	結合テスト成績書，バグレポート
総合テスト	総合テスト，デバッグ	総合テスト成績書，バグレポート
開発完了	法令対応（該非判定等），完成品評価，製品化作業	該非判定書，完成品評価報告書

※ 機密情報保持のため，情報の抜粋および抽象化を行ったもの

表 D-2　XDDP の変更プロセス

工程	工程内の作業	成果物
1.1	**変更箇所を仕様書に記述する** 機能仕様書などの文書から変更箇所（仕様）が特定され，変更仕様書の該当する箇所に記述する。 変更仕様項目数を見積もり，項目数からソースコードの変更行数と作業に必要な工数を見積もる（仮説見積もり）。	変更要求仕様書
1.2	**変更箇所についてソースなどを調査する** それぞれの変更に対してソースコードを調査して変更箇所を探し，発見した箇所を変更仕様として変更仕様書の該当箇所に記述する。その際の調査資料をスペックアウト資料として残す。	変更要求仕様書， スペックアウト資料
1.3	**追加要求の変更仕様を抽出する** 追加機能を受け入れるための変更箇所（仕様）を探し，それを変更仕様書の該当箇所に記述する。	変更要求仕様書， スペックアウト資料
1.4	**変更要求 TM を作成する** すべての変更仕様に対して具体的にソースコードとの対応付けを行う。	変更要求 TM
1.5	**変更設計書を作成する** 変更仕様のレビューを終えた後，改めて具体的な変更方法（差分）を記述する。	変更設計書
1.6	**ソースコードを一斉に変更する** 変更設計書のレビューを終えた後，ソースコードを一斉に変更する。	更新ソースファイル

表 D-3　XDDP の追加プロセス

工程	工程内の作業	成果物
2.1	**要求仕様書を作成する** 基本的に新規開発と同じように，その機能に必要な仕様と品質要求に関する要求を網羅的に記述する。	追加機能要求仕様書
2.2	**追加機能を設計する** 追加機能の要求仕様を受けて，それぞれの機能ごとに要求仕様を実現するための設計を行う。新規開発との差は，「システム設計」というステージがないこと。	追加機能分の設計書， 変更要求 TM
2.3	**追加機能の関数設計書を作成する** 事前に設計された処理構造を元にして，追加機能の分だけの関数の仕様書と設計書を作成する。	追加機能分の関数設計書
2.4	**追加機能をコーディングする** 関数レベルの仕様・設計書のレビューを終えた後，ソースコードへの変換作業を行う。	ソースファイル

表 D-4　テーラリングの実施（変更プロセス）

対象プロセス		マッチングさせた XDDP		テーラリング後の成果物	対象プロセスの変更内容
工程	成果物	工程	成果物		
要件定義		1.1			要件定義として変更要求仕様書を作成し，その中で仮説見積もりを行う。
	要件定義書 （フォーマット未規定）		変更要求仕様書	要件定義書 （変更要求仕様書）	変更要求仕様書を作成し，それを要件定義書と位置付ける。この段階の変更要求仕様書には変更要求のみを記述する。
	見積書		なし	見積書	変更要求仕様書に記述した仮説見積もりの値を使用して作成する。
プロジェクト計画		なし			
	設計計画書，プロセス定義書，開発日程計画書		なし	設計計画書，プロセス定義書，開発日程計画書	
機能設計		1.2，1.3，1.4			ここから追加プロセスと分離する。
	機能設計書		変更要求仕様書	機能設計書とその付録である変更要求仕様書	派生元の機能設計書を変更せず用意し，その差分を記述した付録という位置付けで作成する。
			変更要求 TM	変更要求 TM	スペックアウト資料とともに派生元のソースコードから新規に作る。既存 TM が使用できる場合はそれを使用する。機能設計書の付録に位置付ける。
			スペックアウト資料	スペックアウト資料	従来の個人メモから，機能設計書の付録に位置付けを変更する。よって，構成管理可能な形態に変更する。
	総合テスト仕様書， 各文書のレビュー記録票		なし	総合テスト仕様書， 各文書のレビュー記録票	
構造設計		1.2，1.3，1.4			機能設計で 1.2，1.3，1.4 のすべてを行うことで構造設計を省略する。
	構造設計書		なし	なし	機能設計書とその付録である 3 点セットを作成することで構造設計書を省略する。
	結合テスト仕様書， 各文書のレビュー記録票		なし	結合テスト仕様書， 各文書のレビュー記録票	

表 D-4　テーラリングの実施（変更プロセス） [続き]

対象プロセス		マッチングさせた XDDP		テーラリング後の成果物	対象プロセスの変更内容
工程	成果物	工程	成果物		
モジュール設計		1.5			
	モジュール設計書		変更設計書	モジュール設計書とその付録である変更設計書	派生元のモジュール設計書を変更せず用意し，その差分を記述した付録という位置付けで作成する。
	単体テスト仕様書, 各文書のレビュー記録票		なし	単体テスト仕様書, 各文書のレビュー記録票	
プログラミング		1.6			
	ソースコード		更新後のソースファイル	ソースコード	
	ソースコードレビュー記録票，解析ツール分析結果		なし	ソースコードレビュー記録票，解析ツール分析結果	
単体テスト		なし			
	単体テスト成績書, バグレポート		なし	単体テスト成績書, バグレポート	
結合テスト		なし			ここで追加プロセスと合流する
	結合テスト成績書, バグレポート		なし	結合テスト成績書, バグレポート	
総合テスト		なし			
	総合テスト成績書, バグレポート		なし	総合テスト成績書, バグレポート	
開発完了		なし			
	なし		製品の正式文書	各設計書，各テスト仕様書，各文書のレビュー記録票	
	該非判定書, 完成品評価報告書		なし	該非判定書，出荷検査表	