派生開発問題解決セミナー2018

派生開発プロセスの基本と発展

AFFORDD 派生開発推進協議会

講演概要

- 昨今、派生開発プロセス（XDDP）を知らない人、なんとなく聞いたことがある人、避けている人、実践している人など様々
- 意外と誤解している人も多い
- 「変更3点セット」のイメージが先行し、本質を知らずに「書くこと」が目的になっている人もいる
- 今一度「派生開発プロセス（XDDP）の基本」を共有し、
- その後「派生開発プロセスの発展型」としてAFFORDDの研究会活動の内容を紹介する。

派生開発プロセスの基本と発展　Agenda

時　間	内　容
	＜基本＞
14:00 ～ 14:20	AFFORDDの活動とXDDPの成り立ち
	PFD（Process Flow Diagram）の基本
14:20 ～ 14:45	USDM（Universal Specification Describing Manner）の基本
14:45 ～ 15:10	XDDP（eXtreme Derivative Development Process）の基本
	＜発展＞
15:10 ～ 15:35	大規模システムおよびモデル駆動開発への発展
15:35 ～ 15:50	Agileへの発展
15:50 ～ 16:00	Q&A

AFFORDDの活動とXDDPの成り立ち

派生開発推進協議会　運営委員
株式会社エクスモーション
斎藤　賢一

Agenda

- 派生開発推進協議会（AFFORDD）の活動
  - 背景
  - 研究会活動
- 派生開発プロセス XDDPの成り立ち
  - XDDP誕生のきっかけ
  - その時、何を考えたか
  - どう対策したか

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

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

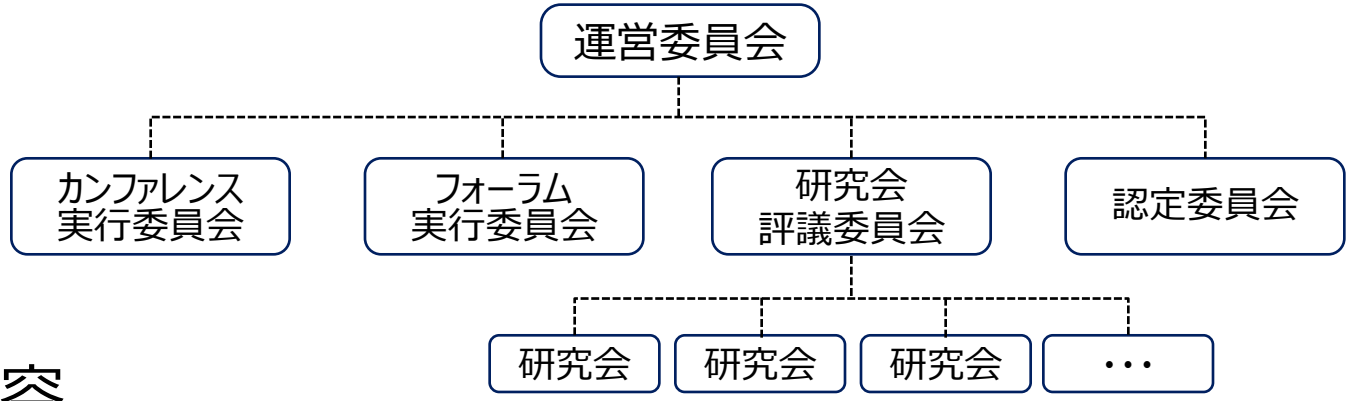
QCDの同時達成が不可欠

- 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

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

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

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

派生開発プロセス XDDPの成り立ち

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

何を考えたか

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

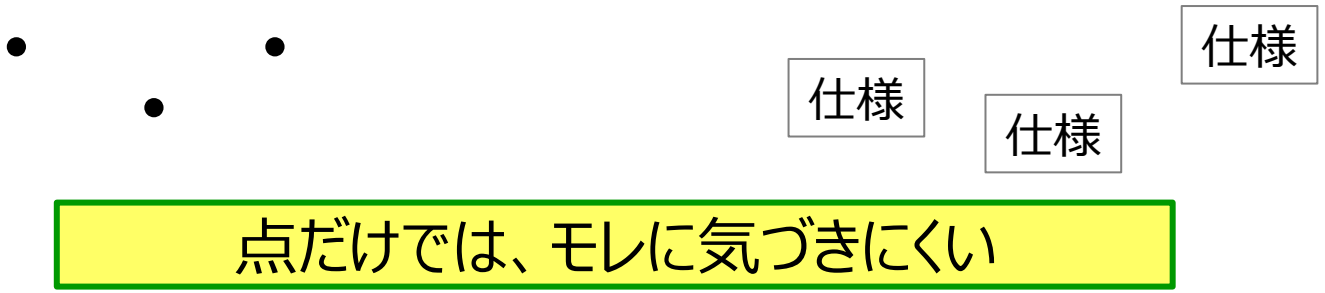
②短納期で、**すべての**システム、コードを**理解すると間に合わない。新規開発のプロセスでは対応できない**

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

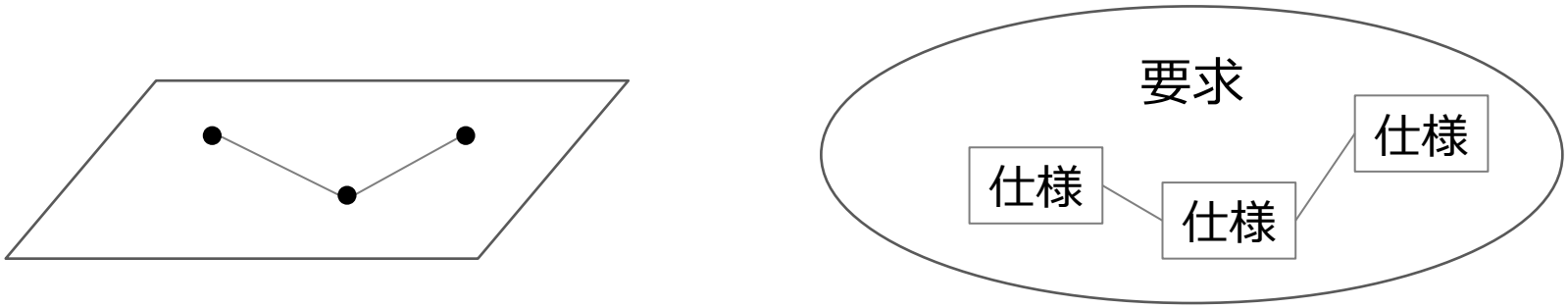
①変更依頼は仕様が示され、要求が省略される

- 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

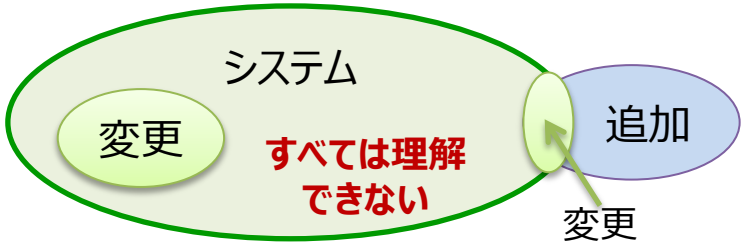


面・線を理解すれば、点も理解できる

②短納期で、すべてのシステム、コードを理解すると間に合わない

- 新規開発のプロセスでは対応できない

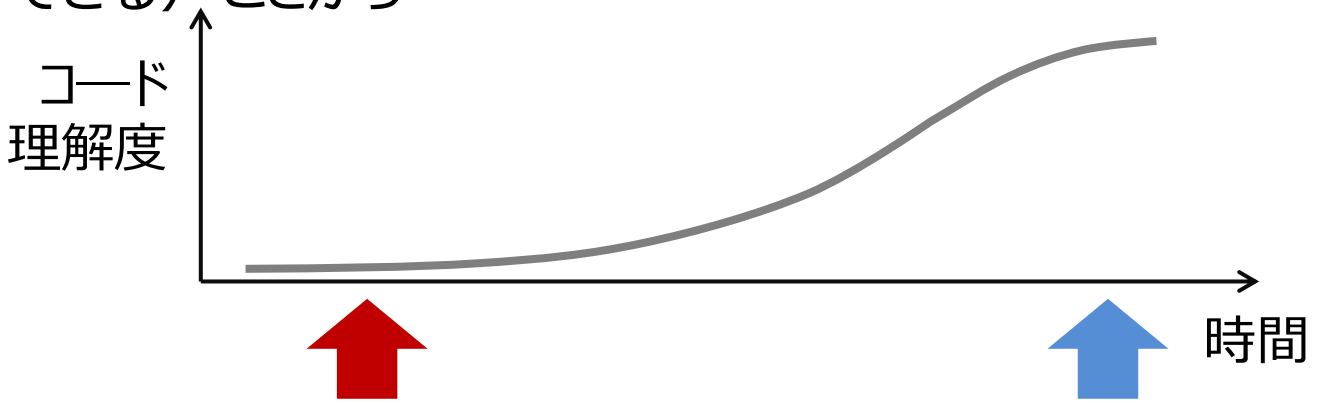
  ⇒ 変更する内容、変更箇所、機能追加する内容をレビューで確認する（**差分開発**の考え方）



- 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でコードを変更したほうが良い理由

- **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）ことから



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

変更箇所を書き留めておく手段が必要

当時はFAXで確認し、理解を深めた

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」などの**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」

  であれば、バグ対策だけでなく、**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**

どう対策したか

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**）

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

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は必須のスキル

- 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

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。