RUP XP

RUP & XP

オブジェクト指向開発プロセスの新構図

株式会社永和システムマネジメント
平鍋　健児
ウルシステムズ株式会社
平澤　章

Agenda

- RUPとXPの概略
- 各視点での両者の比較
- 反論バトルコーナー
- まとめ

RUP

Overview

- Rational Unified Process
- UMLを提案した米Rational社が提唱する開発プロセス
  - Ivar JacobsonのObjectory(OOSE)プロセス, Rational社のコンサルタントのノウハウなど, ソフトウェア開発のBest Practicesを集大成
- 開発プロセス自体がRational社の製品
  - 数百ページに及ぶ膨大なハイパーテキスト
    - ワーカー，アクティビティ，成果物テンプレート，ガイドラインなど

RUP

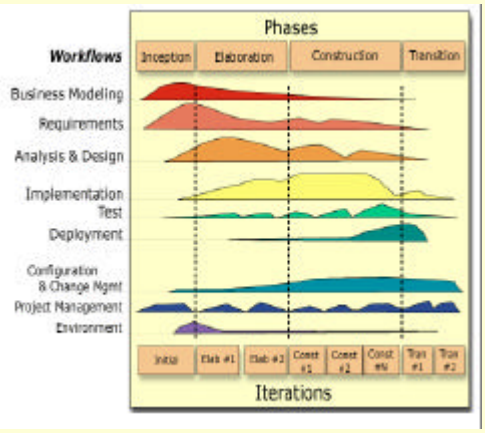
Key Concepts

- Incremental & Iterative（段階的&繰り返し型）
  - 機能追加をしながら繰り返す
- Use Case Driven（ユースケース駆動）
  - ユーザーにとって価値のあるユースケースから実現していく
- Architecture Centric（アーキテクチャ中心）
  - 早い段階でアーキテクチャを構築する
- Configurable Process（カスタマイズ可能）
  - 組織の事情に合わせてプロセス自体を変更してよい

RUP

4 Milestones

- Inception (方向づけ)
  - 全体をざっくり決める
- Elaboration (推敲)
  - 重要な部分を選んで，少しやってみる
- Construction (作成)
  - 全体を作り上げる
- Transition (移行)
  - 完成させる



RUP

Best Practices

- Develop Iteratively (繰り返し開発せよ)
- Manage Requirements (要求を管理せよ)
- Use Component Architectures
  (コンポーネントアーキテクチャを利用せよ)
- Model Visually (ビジュアルにモデリングせよ)
- Verify Quality (品質を検証せよ)
- Control Changes (変更を管理せよ)

XP

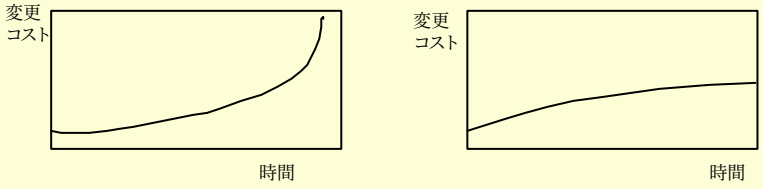
Overview

- E**x**treme **P**rogramming(バーを10まで)
- コーディングとテストに重点
- 前払いの初期設計よりも進化的設計重視
- ライトウェイトな方法論(最近Agileという)
- 明確な「4つの価値」,「14のプラクティス」を示す
- ソフトウェア開発は製造工程ではない.
  対話を基礎に置いた新しいパラダイム.

XP

Embrace Change - 変化ヲ抱擁セヨ

- もし変更の「時間-コストカーブ」を平坦にするには,
  あるいは, 平坦にできたとしたら？
  - 大きな初期設計に前払いをする必要はない
  - 「モデルの修正はコードの修正より安い」は神話になる
  - 「明日のための設計」は明日考える.
  - 本当に必要になるまで, 決定を遅らせる.



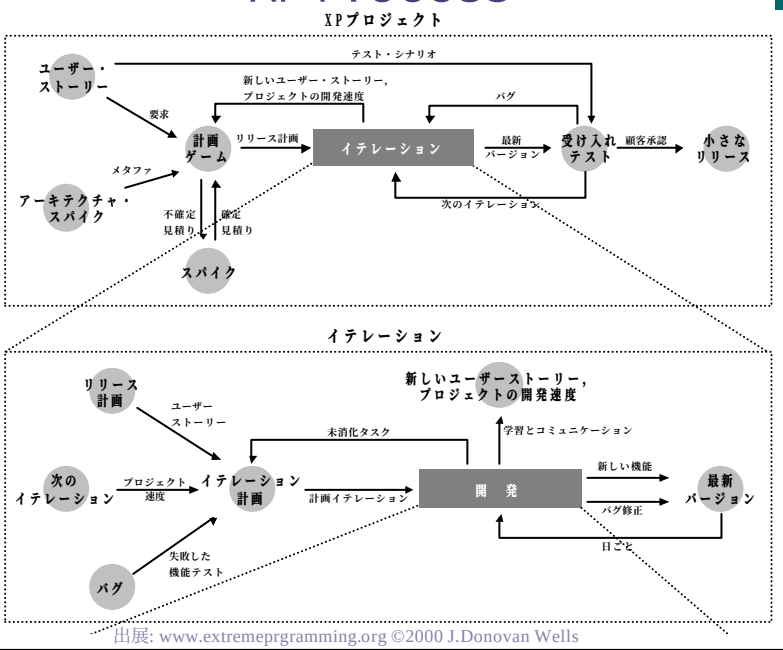
XP

XP Process[1]

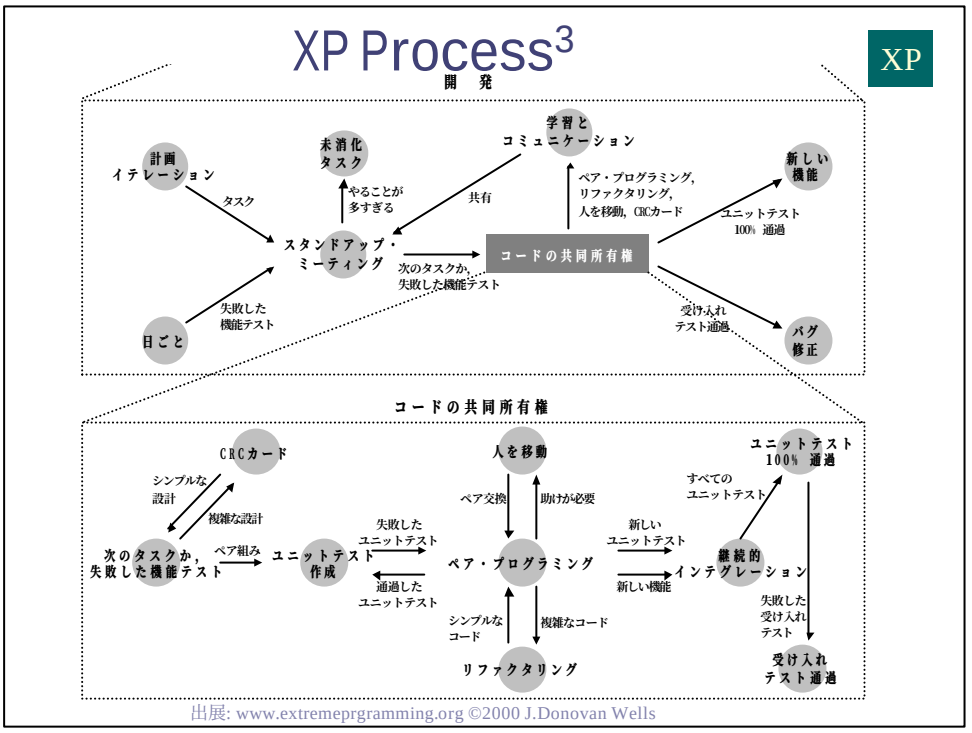
- プロジェクトは
  - 2～3ヶ月のリリースに分割計画される.
- リリースは
  - 2～3週間のイテレーションに分割計画される.
- イテレーションは
  - 1～2日のタスクに分割計画される.
- 1タスクは
  - 数分～1時間のテストケースに分割計画される.

XP

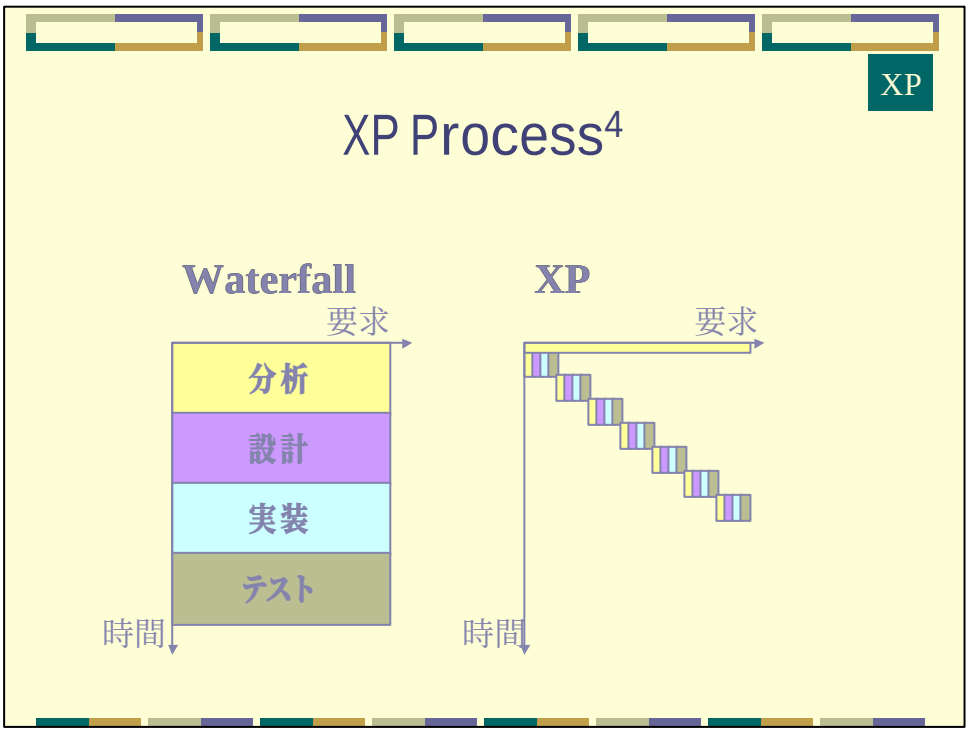
XP Process[2]



出展: www.extremeprgramming.org ©2000 J.Donovan Wells



出展: www.extremeprgramming.org ©2000 J.Donovan Wells



XP

14 Practices

- Planning Game
  - 計画ゲーム
- Small Releases
  - ちいさなリリース
- Metaphor
  - メタファー
- Simple Design
  - シンプルデザイン
- Testing
  - テスティング
- Refactoring
  - リファクタリング
- Open Workspace
  - オープンな作業スペース
- Pair Programming
  - ペアプログラミング
- Collective Ownership
  - 共同所有権
- Continuous Integration
  - 継続的インテグレーション
- 40-Hour Week
  - 週４０時間
- On-site Customer
  - オンサイト顧客
- Coding Standard
  - コーディング標準
- Daily Schema Migration
  - 日ごとのデータベース定義移行

Agenda

- RUPとXPの概略
- 各視点での両者の比較
- 反論バトルコーナー
- まとめ

RUP

Planning

- 全体を４つのフェーズに分割
  - 試行錯誤は前半に，後半は一気呵成
  - マイルストーンで管理する
- フェーズ分割／イテレーションの考え方
  - 基本的には２～３ヶ月
  - イテレーション毎に「リリース可能な」ソフトウェアを作成するが，実際に顧客にリリースするかはプロジェクトで判断

XP

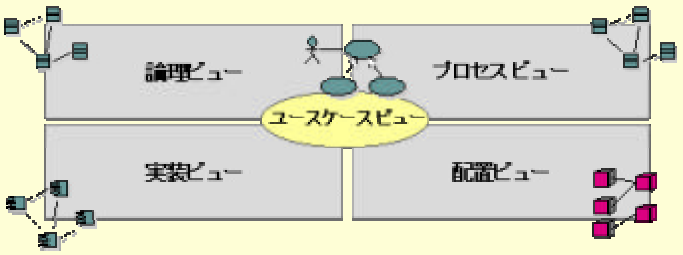
Planning

- 計画ゲーム(ストーリーのスコープ決め)
  - 顧客 – ビジネス価値，プライオリティ
  - 開発 – 見積
- 昨日の天気
- プログラマの見積能力とポイント制
- 2-3ヶ月の小さなリリース(リリース計画)
- 1-3週間で１イテレーション(イテレーション計画)

RUP

Architecture

- Elaborationで重点的に構築
- 最も重要，かつリスクの高いユースケースを選択して実際に動くソフトウェアを作りながら構築する。



XP

Architecture

- ミニマリズム
- メタファーを使って構造，名前群を導く
- 最初のイテレーションで創発する
- 後で変更する勇気

> *“RUP always starts with an architecture. So does XP. Ours is on the small side.*
>
> *RUP allows revisions. So does XP. Ours are on the large side.”*
>
> – Ron Jeffries

RUP

Testing

- 計画→設計→実行→評価のサイクル
  - Plan test -> Design test -> Implement test -> Evaluate test
- テストに関するワーカー
  - テストを書く人　　：Test Designer
  - テストを実行する人　：Tester
- 網羅的な記述
  - 単体テスト，結合テスト，システムテスト
  - 機能テスト，パフォーマンステスト
  - テストの品質評価，網羅率

XP

Testing

- JUnit によるユニットテスト
  - プログラマペアが行う小さいリズム
- 顧客が書く受け入れテスト
  - 何を持って仕様を満たしたとするか.
  - それは，顧客が定義するテストの通過だ.
- テストファースト・デザイン
  - まずテストを書き，それを通過するコードを書く
- テスト熱中症
  - *“Test Infected”* – Erich Gamma

RUP

Design Strategy

- 全体の構造（アーキテクチャ）を早い段階で固める。
  - リファクタリングは必要だが，最小限にとどめる努力を
- 参照モデル
  - バウンダリ／コントロール／エンティティ
- サブシステム分割を強調
  - 大規模システムを意識している？

XP

Design Strategy

- 設計キーワード
  - The Simplest Design that Possibly Work.
  - You’re Not Going to Need It(YAGNI).
  - The second person pays for generality.
  - Once and Only Once(同一コードの繰り返しはダメ)
  - Test First(動かないテストがあってはじめてコードを書く)
- 小さい初期投資．前払いの大きなデザインをせず，リファクタリングを通じた漸進的設計．

> *“In XP, the software life-cycle is:*
> *Analysis, Test, Code, Design. ”*
> – Ralph Johnson

RUP

Expectation Management

- 顧客が何をしたいかは顧客自身にもよくわからない
  - IKIWISI (I know it when I see it.)
- 全体から詳細へ，目に見える物を段階的にリリース
  - Inception　：Stakeholders' needsをまとめる
  - Elaboration　：ユースケースや画面プロトタイプを作る
  - Construction　：イテレーション毎にデモまたはリリース
  - Transition　：システムの完成
- Accept / Control Change *(注：原文にはこういう表現はありません)*
  - 要求はElaborationでほぼ固める。以降の変更、追加分は緊急度と優先度に応じて、取り込んでいく。

XP

Expectation Management

- オンサイト顧客
  - 顧客にフルタイムでプロジェクトに参加してもらう.
- 計画ゲーム
  - 優先順位とスコープ（次のリリースの機能範囲)を顧客が決定
- 小さなリリース
  - ビジネス価値を提供するリリースを２ヶ月以内に行う
- Embrace Change
  - 要求は凍結しない．変更可能.

RUP

Budget and Resource

- Inception
  - 初期の計画案を作成
- Elaboration（の終わり）
  - 詳細なイテレーション計画，要員計画を立てるタイミング
    - 要求はほぼ捉えられている
    - アーキテクチャベースラインが出来上がっている
    - リスクは洗い出され，評価されている
  - しかし，現実のビジネスにおいては，このタイミングでの見積もりでは遅いことも。

XP

Budget and Resource

- ????……
- 「5人で6ヶ月，ベストを尽くしますので，30人月です」という契約形態しかないか？

RUP

Key Technology

- 要求管理
- ビジュアルモデリング(UML)
- 自動テスト
- 構成管理

XP

Key Technology

- リファクタリング
- ユニットテスト
- 継続的インテグレーション

RUP

Tool Support

- ツールによる自動化を推奨
  - 要求管理ツール
  - UMLモデルからのコード生成
  - テストの自動化
  - 構成管理
- 推奨ツール
  - Rational Requisite Pro
  - Rational Rose
  - Rational Test Suite
  - Rational Clear Case

XP

Tool Support

- XP values individuals and interactions over processes and tools…
- とはいっても，xUnitと構成管理(CVS, Envy)はほぼ必須．テストやビルドは徹底して自動化．
- 設計，ブレインストーミングにカードを使う．CRCデザインセッション，ストーリー等．
- UML等はホワイトボートや紙に．コードとテストで表現できたら，捨てる．
- アナログ重視．テスト結果，進捗，重要な図は模造紙に描いて壁に貼る．

RUP

Requirements

- 機能要求
  - ユースケース
  - 概念モデル
- 非機能仕様（補足仕様）
  - システム構成，要素技術
  - 信頼性，パフォーマンス
  - ユーザビリティ，サードパーティコンポーネントの適用　等

XP

Requirements

- 「ストーリー」としてカードに記述
- 受け入れテストとして顧客が定義
- いつでも変更可能

RUP

Management

- マネジメントはプロジェクト管理者の仕事
  - 全体開発計画
  - リスク評価と対策立案
  - 作業の定義とリソースの割り当て
  - 変更要求管理と計画への取り込み
  - 品質管理　等

XP

Management

- タスクは割り当てない．サインアップで取り合う．
- 長い会議を避け，プログラマの時間を尊重する．
- プログラマの物理的，政治的障害をどけて回る．
- 人望のあるコーチ，トラッカ（追跡記録者）が重要．
- お菓子やおもちゃを買い与える．

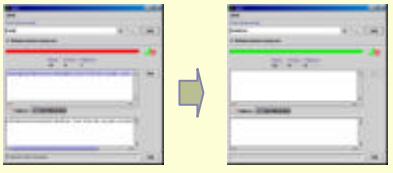
RUP

Motivation

- ????.....
- 独り言
  - まさかXPを聞くまで，モチベーションの重要性を知らなかったわけじゃないでしょ？

XP

Motivation



- 達成感を大切に
  - The Green Bar
  - 統合成功でベルを鳴らす
  - リリースでシャンパンを抜く
- プログラマのリズム
  - Test-Codeサイクル
  - タスクサイクル,イテレーションサイクル
- ペアで困難に立ち向かう

RUP

Communication

- ????.....
- 独り言
  - コミュニケーションは大切ですよ，もちろん。
    そんなこと当たり前じゃないですか…

XP

Communication

- XPの命
- ペアプログラミング
- リリースごとに顧客を交えたパーティ
- お菓子を食べる
- 毎朝のスタンドアップミーティング

Agenda

- RUPとXPの概略
- 各視点での両者の比較
- 反論バトルコーナー
- まとめ

RUP

RUP → XP[1]

- 「オンサイト顧客」なんて本当にできるの？
  - まさか，要求リスクを１人に押しつけようとしているわけでは？
- ビジネスをどう考えてるの？
  - 要員計画や予算はいつコミットするの？
- 最初からいきなりコード？
  - 概念モデリング，業務フロー分析／ユースケース導出なんていう作業をすると２週間ぐらいはあっという間に過ぎてしまうけど。
- Embrace Change と行き当たりばったりのハッキングは紙一重
  - 要求をきちんと確認しない，設計図（UMLモデル）をつくらない，やみくもなコーディング，の免罪符にしないように！

RUP

RUP → XP²

- 何か問題が起きたとき，誰がどう動くの？
  - 役割分担を決めておかないと，「お見合い」しない？
- 「14のプラクティス」ってそんなに大事なの？
  - １人でプログラミングしちゃいけないの？
  - UMLからコード生成しちゃいけないの？
  - XPのプラクティスを実践することを目的にしてませんか？

XP

XP → RUP

- 大きなアーキテクチャを先に作って後でミスに気付いたら？
- Inceptionとかelaborationとか言って議論しているうちに要求が変化しません？
- 成果物として文書を書くことが目的のような錯覚に陥りませんか？
- 理解と知識の共有は文書を書くよりいっしょにやってみるほうが速い，って知ってます？
- RUPのフィードバックは遅すぎてプロジェクトの制御に使えないのでは？
- 若い人の心をつかんでますか？
- やってて楽しいですか？

Agenda

- RUPとXPの概略
- 各視点での両者の比較
- 反論バトルコーナー
- まとめ

RUP

Summary[1]

- フォーカス
  - バランスの良さ
    - ウォーターフォールでもなく，過激なスパイラルでもない
  - いわゆる「開発方法論」の集大成
- 良い点
  - 優れた管理者のノウハウが明文化されている
  - 当たり前のことを，適度な抽象度で表現している
    - 無理して構えなくても，プロジェクトを成功させようとリーダーが工夫をすると，自然とRUPになる。
- 悪い点
  - 情報量が多すぎる
    - 忙しいリーダーに，開発プロセス百科事典を読む暇はない。
  - モチベーションやコミュニケーションに触れない無機質さ

RUP

Summary²

- 完璧なCook bookなんてあり得ない。
  - 百科事典を頭から読むなんてバカげたこと
    - マニュアル通りにやってうまくいくほど，ソフトウェア開発は簡単ではない
- 目的と手段をはき違えるな！
  - RUP（&XP)は手段，プロジェクトの成功が目的
    - 適材適所，いいとこどり，形式にとらわれない姿勢が大切
    - プロジェクトリーダーの経験と勘も大切に

RUP

Summary³

- プロジェクト成功の４つの鍵
  - 方法論
  - ツール
  - マネジメント
  - 要員管理
- 中でもマネジメントは最も重要！
  - プロジェクトリーダーがきちんとしていれば，多くの問題は早期に発見できる

XP

Summary

- フォーカス
  - モチベーション
  - プログラマの本来の能力(性善説)
  - 顧客とのWin-Win
- 良い点
  - チームのモチベーションを高められる
  - 進捗，品質に確信がもてる
- 悪い点
  - ある意味理想論．これでうまくいったら苦労しない．
  - 顧客側がうまく対応できるか．日本の商習慣との兼ね合い．
  - 日本ではドキュメントの問題は避けられない(コードは英語が入るので)

XP

UML/Pattern/RUP/XP