プチXP体験記

2002年6月21日

(株)永和システムマネジメント
田中　真弓

目次

- XPの概要
- 実践したプロジェクトの概要
- 実践したプラクティスの紹介
- 実践の様子
- 反省と感想
- XPに期待すること
- XPの課題①②

Copyright 2002 Mayumi TANAKA/Kenji HIRANABE, All Rights Reserved.

XPの概要

計画ｹﾞｰﾑ(The Planning Game)	ﾋﾞｼﾞﾈｽ優先度と技術的見積もりにより次回ﾘﾘｰｽ、ｲﾃﾚｰｼｮﾝの範囲を早急に決めます。 顧客はｽﾄｰﾘｶｰﾄﾞと呼ばれる、ｼｽﾃﾑの動作ｼﾅﾘｵを書きます。開発側は、ｽﾄｰﾘの見積もりを行います。顧客はｽﾄｰﾘのｺｽﾄを踏まえて、優先度を決めてｽｹｼﾞｭｰﾙに落とします。
小さなﾘﾘｰｽ(Small Releases)	新ﾊﾞｰｼﾞｮﾝを非常に短いｻｲｸﾙ(通常2～3ヶ月)でﾘﾘｰｽします。小さな単位で頻繁に顧客にﾘﾘｰｽし、実際に使用してもらうことにより短い時間でﾌｨｰﾄﾞﾊﾞｯｸを得ることができます。
ﾒﾀﾌｧ(Metaphor)	ｼｽﾃﾑの動きを示すﾒﾀﾌｧを顧客と開発で共有します。ﾒﾀﾌｧはｼｽﾃﾑがどのように動くかを説明した誰にでもわかるｼﾝﾌﾟﾙな例えです。ﾁｰﾑはこのﾒﾀﾌｧを使って互いのｺﾐｭﾆｹｰｼｮﾝをよりｽﾑｰｽﾞにします。
ｼﾝﾌﾟﾙな設計(Simple Design)	XPではできるだけｼﾝﾌﾟﾙに設計します。明日を考えた先行・前払いのﾃﾞｻﾞｲﾝではなく今の要求を満たす最もｼﾝﾌﾟﾙな設計です。余分な複雑さはXPでは“悪”です。ｼﾝﾌﾟﾙﾃﾞｻﾞｲﾝはﾘﾌｧｸﾀﾘﾝｸﾞを常に行うことで保たれます。
ﾃｽﾃｨﾝｸﾞ(Testing)	ﾌﾟﾛｸﾞﾗﾏは継続的にﾕﾆｯﾄﾃｽﾄを書きます。顧客は受入ﾃｽﾄを書きます。 XPではﾃｽﾄを非常に重視します。ﾃｽﾃｨﾝｸﾞはﾌﾟﾛｸﾞﾗﾏにﾘﾌｧｸﾀﾘﾝｸﾞを行う勇気を与えます。
ﾘﾌｧｸﾀﾘﾝｸﾞ(Refactoring)	ｼｽﾃﾑの動作を変えることなくシステムを再設計します。ﾕﾆｯﾄﾃｽﾄと受入ﾃｽﾄを準備することで、今行ったﾘﾌｧｸﾀﾘﾝｸﾞがｼｽﾃﾑを壊していないかをすぐに確かめることができます。
ﾍﾟｱﾌﾟﾛｸﾞﾗﾐﾝｸﾞ(Pair Programming)	ｺｰﾃﾞｨﾝｸﾞは常に2人がﾍﾟｱになって1台のｺﾝﾋﾟｭｰﾀに向かって行います。常にﾍﾟｱでﾌﾟﾛｸﾞﾗﾐﾝｸﾞを行うことにより、ｺｰﾄﾞは常にﾚﾋﾞｭｰされることになり、単純ﾐｽが減り品質が上がります。また、ﾍﾟｱを交代することでｼｽﾃﾑに関する知識がﾁｰﾑに伝播します。
共同所有権(Collective Ownership)	XPではｺｰﾄﾞの担当者を決めません。すべてのｺｰﾄﾞは全員で共有し、誰でもｺｰﾄﾞを自由に変更することができます。これも知識をﾁｰﾑに伝播させる効果があります。
継続的ｲﾝﾃｸﾞﾚｰｼｮﾝ(Continuous Integration)	1日に何回もｲﾝﾃｸﾞﾚｰｼｮﾝ(結合)を行います。ﾕﾆｯﾄﾃｽﾄが完了したﾀｽｸは、すぐにｼｽﾃﾑに結合され、全ﾕﾆｯﾄﾃｽﾄ、受入ﾃｽﾄが行われて、ﾘﾎﾟｼﾞﾄﾘに登録されます。
週40時間(40-Hour Week)	週40時間以上の仕事は良い結果を生みません。適度に疲れて帰宅し、次の朝はﾌﾚｯｼｭな気持ちと体調で仕事に向かいましょう。
ｵﾝｻｲﾄ顧客(On-Site Customer)	顧客をﾌﾙﾀｲﾑでﾁｰﾑに加えます。日本では難しいと思われるﾌﾟﾗｸﾃｨｽの 1つです。ｵﾝｻｲﾄ顧客の役割は、ｽﾄｰﾘを書くこと、受入ﾃｽﾄを書くこと、それに日々の質問に答えることです。
ｺｰﾃﾞｨﾝｸﾞ標準(Coding Standards)	全ﾌﾟﾛｸﾞﾗﾏがｺｰﾃﾞｨﾝｸﾞ標準に従います。ｺｰﾃﾞｨﾝｸﾞ標準はﾍﾟｱﾌﾟﾛｸﾞﾗﾐﾝｸﾞを円滑にし、またｺｰﾄﾞの共同 所有にも貢献します。

プロジェクトのかたち

- プロジェクトの概要
  - プロジェクト名・・・SSS(Simple Schedule Service)
  - 3週間でワンリリースの実験的プロジェクト
    - とりあえずXPをやってみたい！！
    - Webサービスを勉強したい！！
- スケジュール

第1リリース		
第1イテレーション	第2イテレーション	第3イテレーション
3/7 ～ 3/13 (=5営業日)	3/14 ～ 3/15 (=2営業日)	3/18 ～ 3/29 (=9営業日)

Copyright 2002 Mayumi TANAKA/Kenji HIRANABE, All Rights Reserved.

開発環境

OS	Linux 2.2.18 (Turbolinux)
データベース	PostgreSQL 7.1.3
言語	Java (一部 C#)
ﾃｽﾃｨﾝｸﾞﾌﾚｰﾑﾜｰｸ	JUnit 3.7
ソース管理	CVS 1.10.6
Webｻｰﾋﾞｽﾌﾚｰﾑﾜｰｸ	GLUE 2.3

メンバー紹介

メンバ	開発経験	Java経験	OO経験	役割
MT	入社4年	1年	1年	ﾌﾟﾛｸﾞﾗﾏ
ST	入社2年	2年	2年	ﾌﾟﾛｸﾞﾗﾏ
KH	経験豊富		11年	ﾏﾈｰｼﾞｬ、ｺｰﾁ、ﾄﾗｯｶ
YO	入社8年			ｺｰﾁ、お助けﾏﾝ

Copyright 2002 Mayumi TANAKA/Kenji HIRANABE, All Rights Reserved.

実践したプラクティス①

- 計画ゲーム
  - 顧客なし、マネージャが仕様の権限を持つ。
  - リリース計画は１回のみ。イテレーション計画は３回行った。
- スタンドアップミーティング
  - 毎朝、チーム全員参加ミーティング(15分以内)を行った。
  - 前日までの作業の問題点や今日やるつもりの作業などを話す。
- メタファ
  - ホテルのモーニングコール。
- ミッションステートメント
  - オリジナルのプラクティスでマネージャが新しく提案したもの。
  - 目標をもってプロジェクトに参加することを期待して採用。

実践したプラクティス②

- 受け入れテスト
  - 今回はマネージャが受け入れテストを作成。
  - Junitを用いて行った。
- スパイクアーキテクチャ
  - 計画ゲームの中でシステムの全体像を考え、全員で共有。
- テストファースト
  - 今回はテストファーストを意識し、徹底的に行った。
- ペアプログラミング
  - 最初から最後まですべてペアで行った。
  - 「十箇条」を「心得」としてプログラミングを行った。
    ※ペアプログラミング十箇条は後述。

Copyright 2002 Mayumi TANAKA/Kenji HIRANABE, All Rights Reserved.

実践したプラクティス③

- コードの共同所有
  - CVSを積極的に導入。
  - 今回はペアが１つなので主にバージョン管理に利用。
- リファクタリング
  - 時間をきちんととってリファクタリングを行った。
- シンプルデザイン
  - 今回は「タスクを満たす、一番の近道をする」と解釈した。
- コーディング標準
  - 今回は特に定めず、「いつものように」コーディングした。

実践したプラクティス④

- 用語辞書
  - 最初のｲﾃﾚｰｼｮﾝで作り、辞書に基づき一貫した名前をつけた。
  - イテレーションが進むにつれて用語辞書も成長していった。

用語辞書例

用語	英語	意味
予定	schedule	予定。入力者が、予定日、内容、通知先をつけてこのSSSサービスに登録する。
モデル	model	データの永続化を実現する機能。SSSではデータベースに対して予定の追加、変更、削除を行う。
・・・	・・・	・・・

Copyright 2002 Mayumi TANAKA/Kenji HIRANABE, All Rights Reserved.

補足：ペアプログラミング十箇条

1. ドライバー、パートナーは５～１０分毎に交代しよう。　ドライバーは引き際が肝心。パートナーの助言が多くなったら交代。
2. やることを紙に項目として書き出そう。終わった項目を横線で消そう。
3. コードより先にテストを書こう。テストをパスさせるための最もシンプルな実装をしよう。
4. パートナーは、ツッコミの要領で助言しよう。 ・もっとシンプルな方法はないか。 ・コードは意図を表現しているか。 ・クラスやメソッド、変数の名前は意図を表しているか。 ・タイプミスはないか。括弧の数は合っているか。 ・テストは先に書いたか。 ・次のテストはどう書こうか。テストし忘れていることはないか。 ・全体から俯瞰してバランスはとれているか。ヘンな方向に突き進んでいないか。 ・コーディング標準にあっているか。
5. パートナーは、じれったくなったら「わたしにやらせて！」といおう。
6. パートナーは、理解できないコードをみたらドライバーに聞こう。「なんでそうなの？」
7. ドライバーは、パートナーの助言にいつでも耳を貸そう。そしてその助言に返事をしよう。
8. ドライバーは、行き詰まったら助けを求めよう。「このメソッド、ちょっとお願いできないかな？」
9. 腹が減ってはプログラミングはできぬ。一緒にお菓子を食べよう。
10. 楽しくやろう。Enjoy Pair Programming！

ストーリーカード例

第１イテレーションタスク

予定を入力する

・日時と予定が入力できる

予定を通知する

・予定期日になったら期日を通知するメールを送信することができる

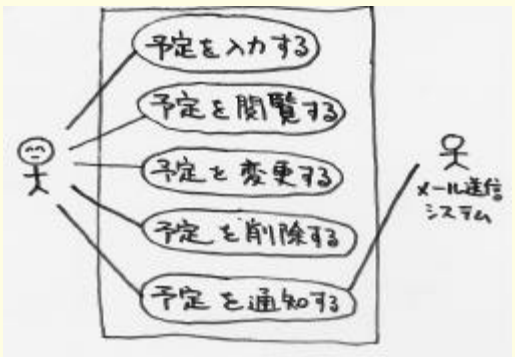
予定を閲覧する

・個人による予定一覧を閲覧することができる
・日付による予定一覧を閲覧することができる

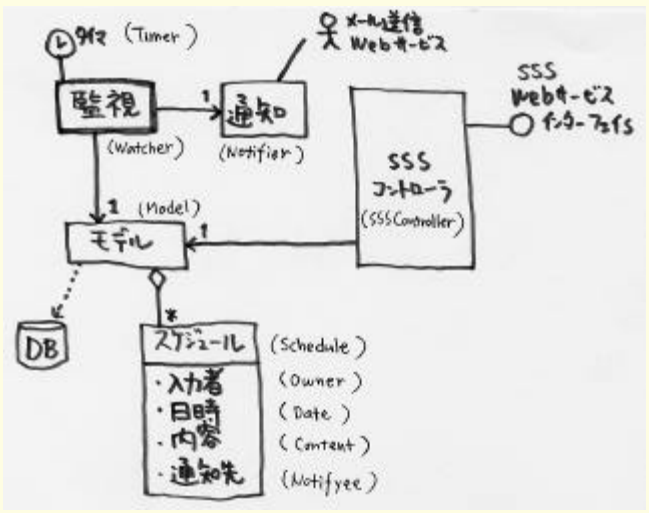
※　コクヨの情報カード「シカー１０W」(壁に貼る)

Copyright 2002 Mayumi TANAKA/Kenji HIRANABE, All Rights Reserved.

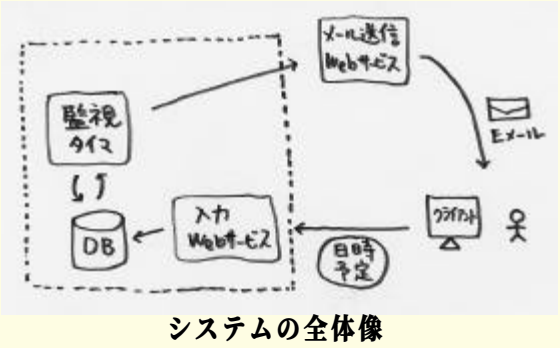
システムの概要



ユースケース図



クラス図



システムの全体像

イテレーションタスク例

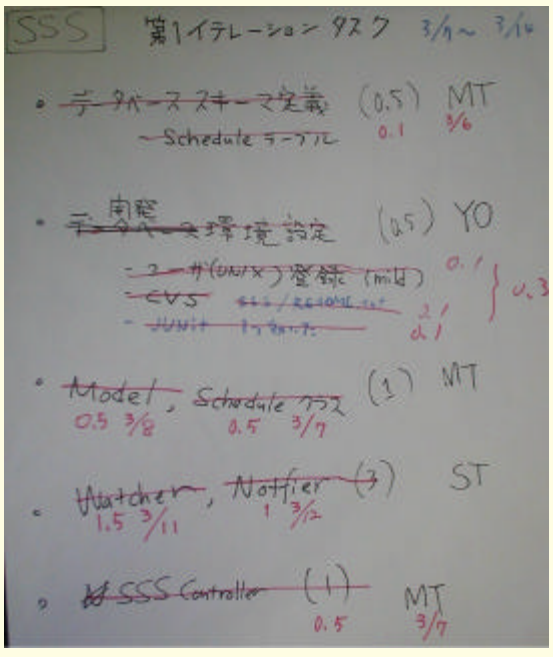
第１イテレーションタスク

- ﾃﾞｰﾀﾍﾞｰｽｽｷｰﾏ定義(0.5)　MT
  - scheduleﾃｰﾌﾞﾙの作成
- 開発環境設定(0.5)　YO
  - UNIXユーザ登録
  - CVS
  - Junit
- Model、Scheduleｸﾗｽ(1.0)　MT
- Watcher、Notifierｸﾗｽ(3.0)　ST
- SSSControllerｸﾗｽ(1.0)　MT

コミット量

- MT・・・2.5
- ST・・・3.0
- YO・・・0.5
- 合計・・・6.0

タスク表実物



Copyright 2002 Mayumi TANAKA/Kenji HIRANABE, All Rights Reserved.

ペアプログラミングの様子

テストファースト

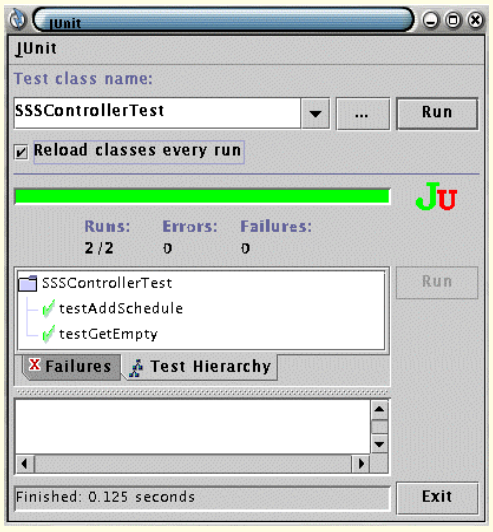
1．テストメソッドを書く

```
public void testGetEmpty() {
    Schedule[] scs = SSSController.getSchedules(“”);
    assertNull(scs);
}
```

2．コンパイル！！ → コンパイルエラー

3．メソッドを実装

```
public Schedule[] getSchedules(String owner) {
    return null;
}
```



4．コンパイル！！
5．テスト実行！！

Copyright 2002 Mayumi TANAKA/Kenji HIRANABE, All Rights Reserved.

反省と感想(XPを終えて)①

- ミッションステートメント
  - MT、STともにそれなりの成果を実感
  - このプラクティスで大切なのは、あとで達成できたかを反省することではなく、ミッションをもって開発に臨むことでタスクへのコミットが意識的にできる部分にある。
- 計画ゲーム
  - PJに関わる人全員で行うため意識のずれがなくてよかった。(MT)
  - タスクを明確にすることでTODOと終了が誰の目からもはっきりわかってよかった。(MT/ST)
  - 自分から進んでタスクをコミットできてよかった。(ST)
  - 外部Webサービスを調べたり、利用できる社内のLinuxマシンを調査したり、最初の計画以前の下準備がかなり必要だった。(KH)

反省と感想(XPを終えて)②

- スタンドアップミーティング
  - 朝、今日何をやるのかが確認できてよかった。(MT)
  - これがないと、ペアの声をかけるﾀｲﾐﾝｸﾞがよくわからない。 (MT)
  - 問題が起きても、みんなの前で共有できるので不安がなかった。(ST)
  - このプラクティスだけは、SSSプロジェクトが終わってもチームで続いている。(KH)
- メタファ
  - 最初の計画ゲーム以降、ほとんど意識しなかった。作るもの自体がシンプルだったせいかもしれない。(MT/ST)
  - 「モーニングコール」というメタファはきれいだが、そこから内部の仕組みを連想するのが難しい。(KH)

Copyright 2002 Mayumi TANAKA/Kenji HIRANABE, All Rights Reserved.

反省と感想(XPを終えて)③

- 受け入れテスト
  - 単体レベルでは考えていなかったテストケースでの障害が発見できて、とても有効だった。(MT)
  - ストーリにはなかったが、hello()というメソッドを作って、通信や環境がらみの疎通を最初に行えたのがよかった。(ST)
  - 今回はWebサービスだったため、受け入れテストが書きやすかった。データベースもうまくテストできた。(KH)
- スパイクアーキテクチャ
  - アーキテクチャ設計にはその人が関わってきたシステムや文化・経験が大きく影響することがわかった。(MT)
  - ストーリを眺めてクラスへブレークダウンする、という作業は経験がいるし、簡単にできるものではない。(KH)

反省と感想(XPを終えて)④

- テストファースト
  - 最初、どのクラスのテストから始めるのかわからなかった(上位から？下位から？)。(MT)
  - Unitテストで結果が目に見えるのは楽しい。すぐにテストをしたいという衝動にかられた。(MT)
  - テストファーストしにくいものの実装の仕方、どのレベルまでテストするのかなどといった点で少し迷うことがあった。(MT)
  - 先に実装したい！という誘惑に負けてしまうことも多い。(MT)
  - 静的なクラスはテストファーストを実行できたが、自律的に動くクラスはテストの仕方がわからず、実装が先になってしまった。(ST)
  - テスト、実装を繰り返しているうちに、何をしていたか頭がこんがらがってしまうことがある。やることを紙に書き出すなどして整理しながら進めるのがよい。(MT/ST)

Copyright 2002 Mayumi TANAKA/Kenji HIRANABE, All Rights Reserved.

反省と感想(XPを終えて)⑤

- ペアプログラミング
  - ツールの操作を実際に見るより早く覚えることができた。(MT)
  - タイプミスは恥ずかしいが、タイプミスによるコンパイルエラーが減った。(MT)
  - お互いの癖などがわかって面白い。(MT)
  - わからないことがその場で質問できるため、聞き忘れたなどということがなくて良いと思う。(MT)
  - ひとりで実装しているときよりも頭が疲れる。(MT)
  - ２人で調べながら進んでいった方が動いた時の喜びが大きい。(ST)
  - 問題にぶつかった時に、解決策がいくつも出し合える点で有効。(ST)
  - 設計やリファクタリングの仕方、ツールの使い方など勉強になることが多かった。(ST)
  - 設計の仕方や考え方、コーディングなど人のコードを見るのはとても勉強になる。(MT/ST)
  - 自分の口臭が気になって、お昼に歯をみがくようになった。(KH)

反省と感想(XPを終えて)⑥

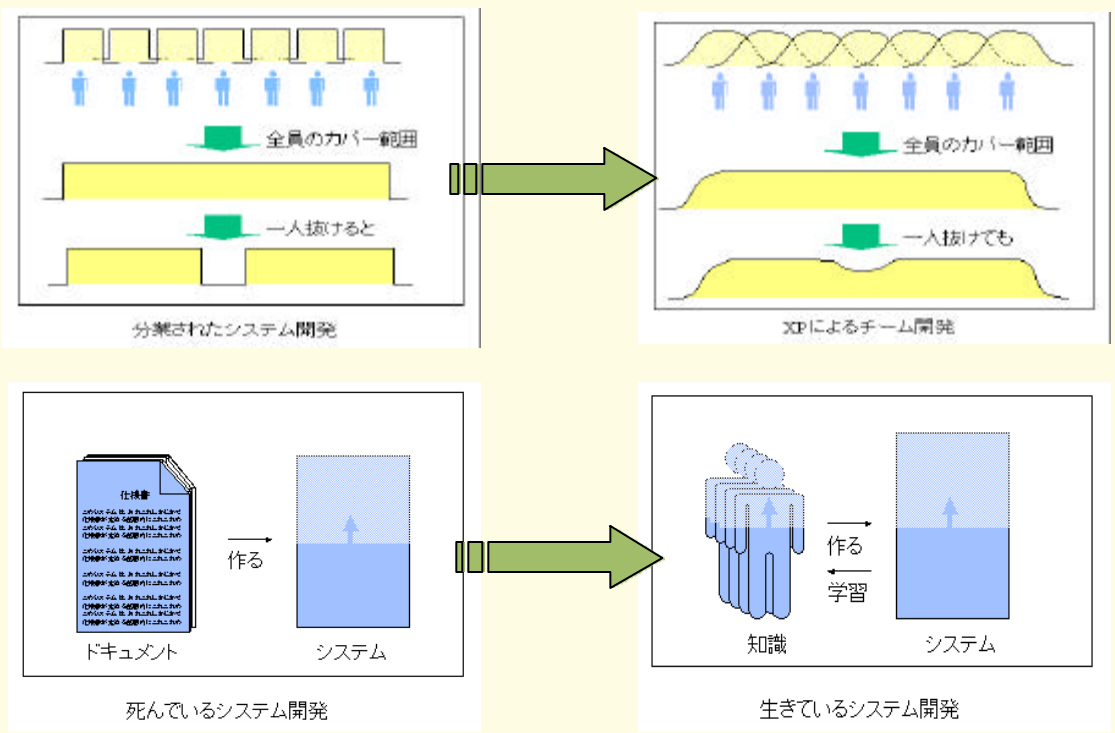
- コードの共同所有
  - CVSで管理するだけでなく、スタンドアップミーティングでも情報を伝え合うことが大切だと思った。(MT)
  - システム全体の知識がチームで共有できる。(MT)
  - 前のバージョンなどをすぐに取り出せてとても便利(MT/ST)
- リファクタリング
  - テストを先に作っているということがﾘﾌｧｸﾀﾘﾝｸﾞへの勇気になる。 (MT)
  - すべてのクラスを一度に流せるクラスを作っておくことは大事。(MT)
  - 大きなリファクタリングをした時にクラス図を見ながら２人できちんと整理してから実装に入ったため大きな混乱や戸惑いを避けることができた。(MT)
  - 自分たちで書いたコードながらほれぼれする出来になった。(MT)
  - １メソッドが短くなって、コードが見やすくなっていくのは快感。(ST)
  - 第３者にもすぐにわかるようなコードを書くことを今後も心がけていきたい。(ST)

Copyright 2002 Mayumi TANAKA/Kenji HIRANABE, All Rights Reserved.

反省と感想(XPを終えて)⑦

- シンプルデザイン
  - 自分がついつい拡張性を考えてしまう癖があることに気が付いた。(MT)
  - 最初は拡張性を考えた設計になっていたが、途中で余計な機能を削除して、結果的にシンプルデザインになった。(ST)
- コーディング標準
  - 今回は２人で実装していたためそれほど必要性を感じなかったが、人数が増えると必要だと思う。(MT)
  - １メソッド１０行以内を目標に頑張った。(ST)
- 用語辞書
  - ｺｰﾃﾞｨﾝｸﾞ標準と同じで人数が多いときにはとても有効。 (MT/ST)
  - 最初にほとんどの用語を決めたため第２ｲﾃﾚｰｼｮﾝ以降はそれほど必要性を感じなかった。(MT/ST)

XPに期待すること



Copyright 2002 Mayumi TANAKA/Kenji HIRANABE, All Rights Reserved.

XPの課題①

XPを採用する際の不安

顧客との契約、
予算などの問題

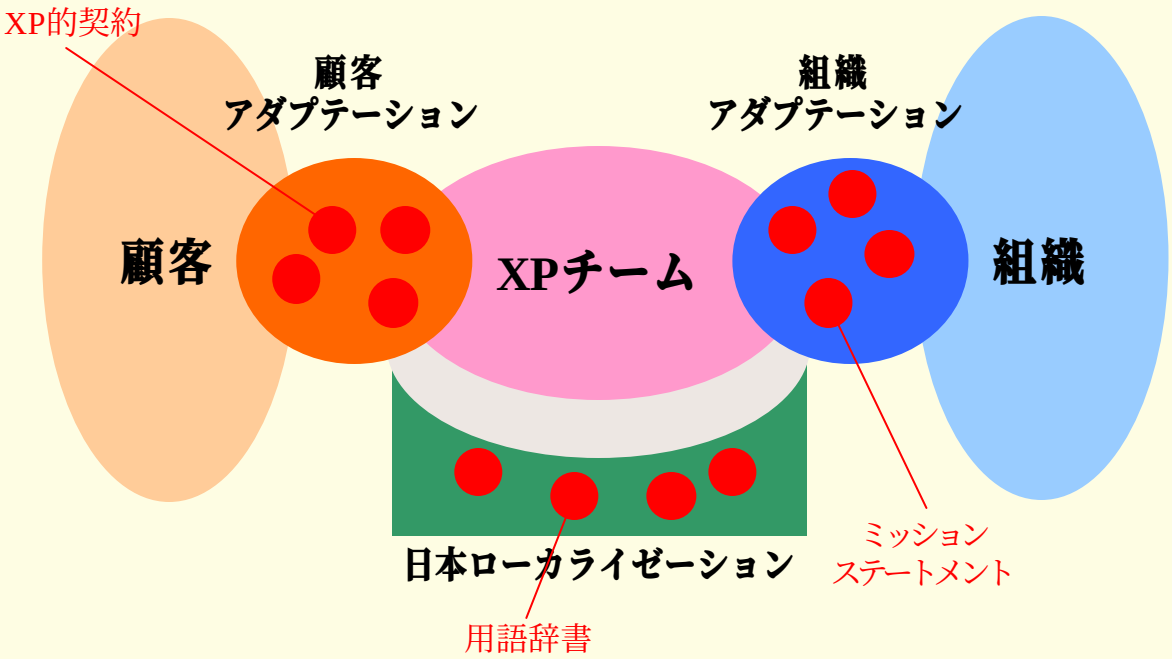
組織としての
評価

日本の習慣や文
化とのマッチング

顧客アダプテーション
組織アダプテーション
日本ローカライゼーション

の必要性

XPの課題②



Copyright 2002 Mayumi TANAKA/Kenji HIRANABE, All Rights Reserved.

契約方式への応用

日本電器㈱　牛尾 剛

2段階プロジェクト

予備イテレーション

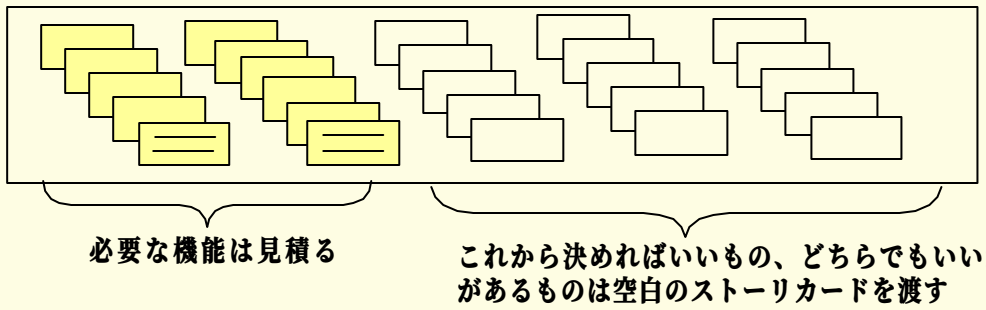
予備イテレーションでリリースしながら
チーム速度の測定と必ず必要な機能要求出しと見積もり

本番イテレーション

△ **見積**

必要な機能は確実に得られ、どうでもいいのはあとで決めれる．必要とした機能でも未実行なら交換可能

プリペード式XP契約手法



必要な機能は見積る

これから決めればいいもの、どちらでもいいが決める必要があるものは空白のストーリカードを渡す

最後に

- 今回紹介したWebサービスはクライアントの参照ソースとともに「Webサービス同好会」で公開しています。
  http://ObjectClub.esm.co.jp/webservice/

Copyright 2002 Mayumi TANAKA/Kenji HIRANABE, All Rights Reserved.

ご清聴ありがとうございました。

http://objectclub.esm.co.jp/

eXtremeProgramming/

Copyright 2002 Mayumi TANAKA/Kenji HIRANABE, All Rights Reserved.